USING IMPROVED MACHINE LEARNING AND STATISTICAL ASSESSMENTS TO MITIGATE BIAS IN ANOMALY DETECTION SYSTEMS

by

Gerard Shu Fuhnwi

A dissertation submitted in partial fulfillment
of the requirements for the degree

of

Doctor of Philosophy

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

August 2025

DEDICATION

I dedicate this dissertation to my mother and twin sister, who passed away, my uncle, my wife, daughter, and son, and my siblings.

ACKNOWLEDGEMENTS

I want to express my deepest gratitude to my advisor, Dr. Clemente Izurieta, for his unwavering support, motivation, and invaluable guidance throughout this journey. His mentorship has been instrumental in helping me achieve this significant milestone. I sincerely appreciate his constructive feedback and insights, which greatly contributed to the completion of this dissertation. I also appreciate Dr. Matthew Revelle, Dr. Stacey Hancock, and Dr. Bradley Whitaker for being my valuable committee members and for their continued support throughout the program.

A special thanks to Dr. John Paxton for providing me with the teaching assistantship opportunity throughout my program.

I am deeply grateful to Montana State University (MSU) for granting me an incredible educational experience and enabling me to fulfill my Ph.D. aspirations at such a prestigious institution. Lastly, I sincerely thank the Gianforte School of Computing and the Graduate School for providing me with the resources and assistance necessary for the successful completion of this work.

TABLE OF CONTENTS

TABLE OF CONTENTS – CONTINUED

TABLE OF CONTENTS – CONTINUED

TABLE OF CONTENTS – CONTINUED

LIST OF TABLES

Table Page

LIST OF TABLES – CONTINUED

Table Page

LIST OF FIGURES

Figure Page

ABSTRACT

Anomaly detection systems are essential in domains like cybersecurity, marketing, fraud detection, and healthcare. However, bias in these systems can result in unfair outcomes, reduced accuracy, and compromised decisions. Traditional machine learning models often struggle with biased training data, inconsistent feature selection, and imbalanced class distributions, leading to unreliable performance.

Bias mitigation is a growing area of research aimed at identifying, quantifying, and reducing bias in machine learning systems to ensure fairness, accuracy, and robustness. This work addresses bias in anomaly detection by integrating improved machine learning techniques with statistical assessments.

First, a hybrid anomaly detection model is introduced for obfuscated malware detection. It combines a deep autoencoder and logistic regression to mitigate representation bias. The autoencoder learns a compact representation of the input, which is then classified using logistic regression. Statistical parity difference (SPD) is employed to assess and reduce bias quantitatively.

Second, to address measurement bias caused by noisy or irrelevant features from network delays, hardware issues, or software faults in intrusion detection systems, a method combining data preprocessing and feature selection is proposed. Statistical hypothesis testing is also performed to evaluate the model's performance against state-of-the-art methods.

Third, to tackle human-induced bias, three strategies are proposed: i) the use of unsupervised learning methods, such as Isolation Forest and one-class SVM, to reduce labeling bias in intrusion detection; ii) a language model-based SMS spam classification system that captures linguistic patterns and contextual nuances, improving fairness in keyword interpretation; iii) a context-enhanced clustering approach for SMS spam detection that reduces subjective labeling, inconsistent interpretations, and the need for domain expertise.

This work contributes to closing existing gaps in bias mitigation for anomaly detection, advancing the intersection of machine learning, statistical analysis, and fairness-aware AI. By addressing multiple types of bias, such as representation, measurement, and human-induced, this research enhances both the performance and fairness of anomaly detection systems

1

INTRODUCTION

In this digital age, cybersecurity, sentiment analysis, and fraud detection have emerged as one of the most critical and urgent challenges facing individuals, organizations, and governments. The need for robust and adaptive security measures becomes increasingly apparent as cyber threats grow in complexity and scale. Anomaly detection systems are vital components among the various tools and technologies employed to defend against these urgent cyber threats. These systems, powered by machine learning algorithms, are designed to detect unusual patterns or behaviors in data [28] that may indicate the presence of malicious activities, such as unauthorized access, data breaches, or malware attacks.

Anomaly detection in cybersecurity involves identifying deviations or outliers from established behavior patterns within IT systems, networks, credit card fraud, or user activities [28]. Traditional approaches often need help to cope with the scale and complexity of modern datasets, particularly in the face of rapidly evolving threats. Machine learning techniques offer promising solutions to this challenge, allowing automated analysis of large-scale data and adaptive learning from various sources [4].

Despite their crucial role in modern cybersecurity and fraud detection, machine learning-driven anomaly detection systems have challenges. One of the most pressing concerns is the potential for bias [82], which can significantly hamper their effectiveness and lead to severe consequences. In this context, bias refers to the systematic errors that can creep into the model due to various factors, such as data bias (e.g., measurement bias, omitted feature bias, sampling bias, and representation bias), human bias (historical bias, label bias, and cognitive bias), and learning bias (algorithmic bias and evaluation bias) [79]. These biases can lead to skewed detection outcomes, potentially resulting in a high rate of false positives or false negatives,

both of which can have profound implications for cybersecurity [51], including potential data breaches and unauthorized access.

For instance, if an anomaly detection system is trained on a dataset where certain types of attacks are overrepresented, the model may become overly sensitive to those attacks while neglecting others. This can result in a high rate of false positives, where benign activities are mistakenly flagged as malicious, or false negatives, where actual threats go undetected. Both scenarios are problematic: false positives can lead to unnecessary investigations and resource allocation, while false negatives can leave a system vulnerable to undetected attacks.

Moreover, the ever-evolving nature of cyber threats and financial fraud exacerbates the issue of bias in machine learning-driven anomaly detection systems. As attackers and defrauders constantly refine their techniques to evade detection, the training data used to construct these systems can quickly become obsolete, further amplifying the problem of bias. This underscores the urgent and ongoing need for continuous updates and enhancements to the models to effectively detect the latest threats, highlighting the ongoing effort required in cybersecurity and fraud detection.

My research seeks to address these challenges by investigating the sources and impacts of bias in machine learning-driven anomaly detection systems and developing strategies to mitigate these biases. The research will begin with a comprehensive analysis of the various factors contributing to bias, such as dataset imbalances, feature selection, and algorithmic constraints, to understand how these elements influence the performance of anomaly detection models and lead to biased outcomes. The study will then focus on quantifying the extent of bias present in existing models and developing metrics and methodologies to better understand the relationship between bias and model performance. Building on these insights, the goal of this research is to introduce novel bias mitigation strategies in machine learning-driven anomaly detection systems. These techniques involve re-weighting training datasets, fair representation methods such as feature selection and feature learning, and enhancing existing

learning algorithms or evaluation metrics to be more resistant to bias. These techniques will be rigorously tested and validated using real-world datasets and scenarios, ensuring that the solutions are theoretically sound but also practical and effective in real-world applications. By leveraging rigorous statistical techniques, the research seeks to enhance the robustness of anomaly detection frameworks, ensuring that they are not only accurate but also fair and unbiased in their operations [79].

By addressing these areas, this research aims to make significant contributions to the field of cybersecurity, particularly in the context of anomaly detection. The ultimate goal is to enhance the reliability and fairness of machine learning-driven anomaly detection systems, ensuring they provide accurate and unbiased detection of cyber threats, thereby creating more secure digital environments.

The remainder of this dissertation is structured as follows. The Background & Related Work chapter 2 provides the necessary foundation for this research by discussing key concepts such as anomalies, anomaly detection, which includes types and methods of evaluation, rigorous statistical assessment, and the bias problem, which involves types, methods of evaluation, and bias mitigation strategies. Additionally, it presents an overview of related research in mitigating bias in machine learning-driven anomaly detection systems. The Research Objective chapter 3 outlines the motivations driving this study, identifying the key challenges and research gaps that necessitate further exploration. It introduces the Goal Questions Metic (GQM) framework, which serves as a structured approach to defining the research objectives and ensuring alignment between research questions, methods, and evaluation strategies. Chapter 4 titled ‘A Hybrid Anomaly Detection Approach for Obfuscated Malware‘ presents a novel hybrid approach that integrates a deep autoencoder with logistic regression to mitigate representation bias in malware detection. This approach leverages deep learning’s feature extraction capabilities alongside logistic regression’s interpretability, resulting in a more balanced and robust detection mechanism. Chapter 5 titled ‘Improving Network

Intrusion Detection Performance: An Empirical Evaluation Using Extreme Gradient Boosting (XGBoost) with Recursive Feature Elimination‘ presents a methodologically rigorous approach to addressing measurement bias in network intrusion detection. By employing XGBoost with Recursive Feature Elimination (RFE), this work enhances feature selection efficiency, ensuring that intrusion detection systems operate with higher accuracy and reduce false positives. Chapter 6 titled ‘An Empirical Internet Protocol Network Intrusion Detection using Isolation Forest and One-Class Support Vector Machines‘ explores the applications of unsupervised learning techniques proposed to combat the human-induced labeling bias problem in intrusion detection. This work demonstrates an effective approach for detecting anomalies in network traffic without relying on biased human-labeled data. The ‘Using Large Language Models to Mitigate Human-Induced Bias in SMS Spam: An Empirical Approach‘ chapter 7 explores reducing human-induced data labeling bias by leveraging in-context learning with prompts. This approach enables LLMs to dynamically adapt to specific language tasks and context, effectively mitigating biases inherent in human-labeled data for more accurate spam detection. In chapter 8 titled ‘Reducing Human-Induced Label Bias in SMS Spam with Context-Enhanced Clustering (CEC),‘ a context-enhanced clustering approach is introduced to address bias in SMS spam detection. By mitigating challenges such as subjective labeling, inconsistent interpretation, and domain expertise limitations, this work improves the fairness and reliability of spam classification models. Chapter 9 highlights the key threats to the validity of this research and emphasizes the importance of interpreting findings with appropriate caution. Finally, the ‘Conclusion and Future Work‘ chapter 10 summarizes the key findings and contributions of this dissertation, outlying potential future research directions to further advance bias mitigation in anomaly detection systems across diverse domains.

BACKGROUND & RELATED WORK

Background

This section introduces relevant concepts in anomaly detection, describes the bias problem, and explains the various techniques used to mitigate bias in anomaly detection systems. It also illustrates the bias challenges and the motivation for using automated methods to help mitigate bias.

Anomalies

As shown in Figure 1, anomalies are data patterns (outcomes, values, or observations) that deviate from the rest of the other observations or outcomes.

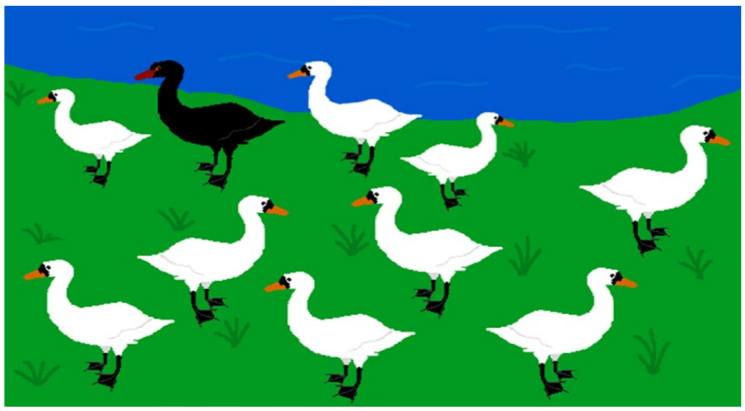

Figure 1: An anomaly is indicated by the black bird

Anomaly Detection in Cybersecurity

Anomaly detection is a proactive technique used in cybersecurity to identify patterns, instances, or events within a dataset that deviate significantly from expected behavior [28]. Anomalies, or outliers, may represent unusual activities, unexpected system behaviors, or potentially malicious actions. Anomaly detection aims to flag such anomalies for further investigation, as they may indicate security breaches, system failures, or other critical events.

Anomaly detection finds applications in various domains of cybersecurity, such as intrusion detection, fraud detection, malware identification, etc.

Types of Anomaly Detection Methods Anomaly detection methods can be classified into the following according to how they detect and model anomalies.

- **Statistical Anomaly Detection:** Statistical anomaly detection methods rely on the statistical properties of the data to identify anomalies. These methods assume that regular data instances follow a known statistical distribution, such as a Gaussian (normal) distribution. Any data instance that significantly deviates from this distribution is considered an anomaly. Standard statistical anomaly detection techniques include Z-Score Analysis [53], Grubbs' Test [38], and Dixon's Q Test [20].

- **Machine Learning-Based Anomaly Detection:** Machine learning is the study of computational methods that learn from the data (by extracting the hidden patterns) and use this information for future predictions [84]. Machine learning techniques are vital in cybersecurity, particularly in anomaly detection tasks. Supervised learning involves training machine learning models on labeled data, where each instance is associated with a known class or category (e.g., normal or abnormal). Unsupervised learning techniques, such as clustering or density estimation, do not require labeled data and aim to identify patterns or anomalies based solely on the input features. Semi-supervised learning methods leverage labeled and unlabeled data for anomaly detection, combining the benefits of both supervised and unsupervised approaches. Some widely used machine learning-based anomaly detection algorithms include K-Nearest Neighbors, Support Vector Machines, Random Forests, Isolation Forests, and Neural Networks.

- **Deep learning Anomaly Detection:** As shown in Figure 2, a fully connected NN is comprised of a series of fully connected layers containing multiple neurons, where each

neuron in a layer l is associated with the neurons in the next layer $l+1$. A simple neural network comprises an input layer, a hidden layer, and an output layer. A neural network is considered deep if it contains multiple hidden layers. A neuron in a neural network is a fundamental computing unit that receives an input, performs a dot product (of input and corresponding weight parameter), applies the activation function, and then forwards the result to the neurons connected in the next layer. Some of the expected activation functions used in DNNs include sigmoid, ReLU, and Tanh [93]. The output layer has either the softmax (if the problem is a multi-class classification problem) or the sigmoid activation function (if the problem is a binary-class classification problem). The output layer produces the probability vector, and the class for which the probability score is higher is treated as a predicted output for the given test instance.

Each layer in a neural network extracts and learns the patterns in the training data, using this knowledge to predict the future. The neurons in each layer are connected with other neurons using a weight parameter that represents the strength of the connection. The weights are learned during training to minimize the loss function (e.g., cross-entropy is used for the classification problem, and mean square error is used for the regression problem). The most popular algorithm to update the weights and train the neural network is the gradient descent with backpropagation [96].

Deep learning anomaly detection leverages deep neural network architectures to automatically learn complex patterns and representations from data, enabling the detection of anomalies in large-scale and high-dimensional datasets. This approach has gained prominence in cybersecurity due to its ability to capture intricate relationships within data and adapt to evolving cyber threats. Incorporating deep learning techniques for anomaly detection can enhance detection accuracy and robustness.

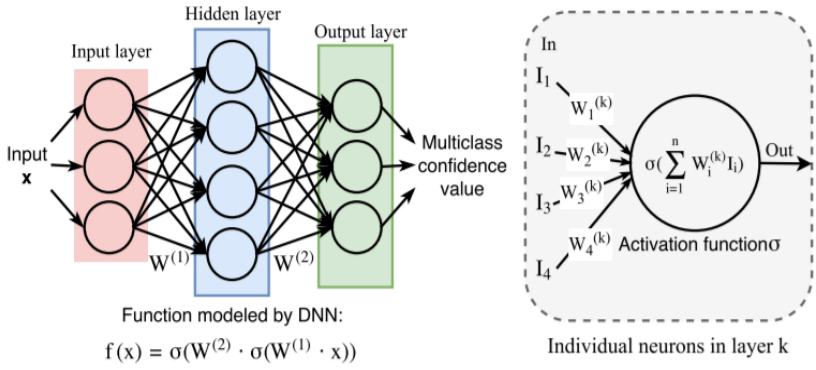

Figure 2: Neural Network Architecture [93]

- **Hybrid Anomaly Detection:** Hybrid anomaly detection techniques combine multiple anomaly detection approaches, including statistical, machine learning-based, feature learning, and feature engineering methods, to improve detection accuracy and robustness. These techniques leverage the strengths of different methods while mitigating their weaknesses. Common hybrid anomaly detection approaches include:

  – *Ensemble Methods:* Ensemble methods combine predictions from multiple anomaly detection models to obtain a more robust and accurate anomaly score. Examples include bagging, boosting, and stacking. Ensemble methods can integrate diverse anomaly detection techniques, such as statistical, machine learning-based, and feature engineering approaches, to perform better than individual methods alone [37].

  – *Feature Selection and Feature Learning:* Feature selection involves the process of selecting a subset of the most relevant and informative features from a dataset's original set of features. On the other hand, feature learning refers to the automatic extraction of features from raw data using machine learning techniques, such as deep learning. Hybrid anomaly detection techniques may combine feature selection or feature learning approaches, such as deep autoencoder networks with traditional machine learning techniques to capture complex patterns and relationships in the

data. By leveraging both handcrafted and learned features, hybrid techniques can improve the symbolic power of anomaly detection models and adapt to diverse data characteristics [55].

– *Meta-Learning:* Meta-learning frameworks learn to adaptively combine multiple anomaly detection algorithms based on the dataset's characteristics or specific application domain. Meta-learning approaches aim to optimize the selection and combination of anomaly detection techniques to maximize detection performance [112]. In hybrid anomaly detection, meta-learning may incorporate feature learning methods to dynamically adjust the feature representation of the data based on its intrinsic properties. Meta-learning with feature learning enables anomaly detection systems to adaptively learn and update feature representations over time, improving their ability to detect evolving cyber threats.

Evaluation Metrics for Anomaly Detection Methods Evaluation metrics play a crucial role in assessing the performance of anomaly detection models. These metrics provide quantitative measures of the model's effectiveness in identifying and distinguishing anomalies from normal instances. Evaluation metrics play a crucial role in assessing the performance of anomaly detection models. These metrics provide quantitative measures of the model's effectiveness in identifying and distinguishing anomalies from normal instances. The evaluation metrics for anomaly detection are described below:

- **True Positive (TP):** Instances correctly classified as anomalies by the detection model.
- **True Negative (TN):** Instances correctly classified as usual by the detection model.
- **False Positive (FP):** Normal instances incorrectly classified as anomalies by the detection model (Type I error).

- **False Negative (FN):** The detection model incorrectly classified abnormal instances as normal (Type II error).

- **Negative predictive value (NPV):** The fraction of negative instances correctly predicted to belong to the negative class out of all predicted negative instances, which can be calculated using the formula:

$$\text{NPV} = \frac{TN}{TN + FN}$$

- **False negative rate (FNR):** The fraction of positive instances incorrectly predicted to belong to the negative class out of all actual positive instances, which can be calculated using the formula:

$$\text{FNR} = \frac{FN}{TP + FN}$$

- **Precision or Positive predictive value (PPV):** The proportion of true positive predictions among all positive predictions, indicating anomaly detection accuracy, which can be calculated using the formula:

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall (Sensitivity or Detection Rate):** The proportion of true positive predictions among all actual positive instances indicates the model's ability to detect anomalies, which can be calculated using the formula:

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

- **F1 Score:** The harmonic mean of precision and recall provides a balanced measure

of model performance that considers false positives and false negatives, which can be calculated using the formula:

$$\text{F}_1 \text{ Score} = \frac{2 * TP}{2 * TP + FN + FP}$$

- **Balanced Accuracy (ACC):** Balanced Accuracy (ACC) is the mean accuracy calculated across both the positive and negative classes [106].

$$ACC = \frac{\frac{correct_{negative}}{examples_{negative}} + \frac{correct_{positive}}{examples_{positive}}}{2}$$

- **Matthews correlation coefficient (MCC):** It is defined as the Pearson product-moment correlation coefficient between actual and predicted abnormal instances [32]. MCC takes into account true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) to provide a balanced measure of the model's performance, particularly in imbalanced datasets. MCC ranges between $[-1, +1]$, where -1 corresponds to the worst overall system performance and 1 corresponds to the best overall system performance. A high MCC score indicates that the binary classifier could correctly predict the majority of the anomalies and the majority of normal instances.

$$MCC = \frac{TN * TP - FN * FP}{\sqrt{(FP + TP)(FN + TP)(TN + FP)(TN + FN)}}$$

- **False Alarm Rate (FAR) or False positive rate (FPR):** It is the fraction of normal instances that are misclassified as anomalies which can be calculated using the formula:

$$FAR = \frac{FP}{FP + TN}$$

- **Receiver Operating Characteristic (ROC) Curve:** The Receiver Operating Characteristic

(ROC) curve is a graphical representation used to evaluate the performance of binary classification models in machine learning. It is created by plotting the ratio between the total number of anomalies detected by the anomaly detection system to the total number of anomalies present in the dataset (detection rate) against the fraction of normal instances that are misclassified as anomalies (False Alarm Rate) at various classification threshold levels. The area under the curve (AUC) of the ROC quantifies the overall performance of the classification model. AUC values range from 0 to 1, with a value of 0.5 representing a random classifier and a value of 1 indicating a perfect classifier. A higher AUC value suggests a better-performing classification model.

- Jaccard Coefficient: The Jaccard Coefficient measures the fraction of true positive point pairs, but after ignoring the true negatives [126]. It is defined as follows:

$$Jaccard = \frac{TP}{TP + FN + FP}$$

Rigorous Statistical Assessment Rigorous statistical assessment involves applying statistical methodologies to evaluate the performance and reliability of anomaly detection models. Statistical techniques such as cross-validation, hypothesis testing, and performance metrics analysis are used to objectively assess the accuracy, precision, recall, and other relevant metrics of anomaly detection models [6]. Rigorous statistical evaluation ensures the validity and generalizability of research findings, enabling informed decision-making in cybersecurity. By rigorously evaluating anomaly detection models, researchers can identify strengths, weaknesses, and areas for improvement, ultimately advancing the state-of-the-art in cybersecurity.

Bias Problem in Anomaly Detection Models For Cybersecurity

Bias in anomaly detection systems refers to systematic errors in a machine learning system that unfairly favor certain outcomes, groups, or patterns. Most anomaly detection systems

and algorithms are data-driven and require data upon which to be trained. Thus, data are tightly coupled to the functionality of these algorithms and anomaly detection systems in cybersecurity. In cases where the training data contains biases, the algorithms trained on them will learn them and reflect them in their predictions. As a result, existing biases in the data can affect the algorithms that use the data, producing biased or discriminatory results. Algorithms can even amplify and perpetuate existing biases in the data. In addition, algorithms can display biased behavior due to specific design choices, even if the data are unbiased. The outcomes of these biased algorithms can then be fed into the data, resulting in discriminatory results or unfair outcomes. For example, imagine an intrusion detection system; if the algorithm is trained to flag anomalies based on specific characteristics such as IP address or location, it may disproportionately flag anomalies from particular regions or countries, leading to biased results. The loop capturing the feedback between biases in data and algorithms that lead to unfair outcomes or discriminatory results is shown in Figure 3

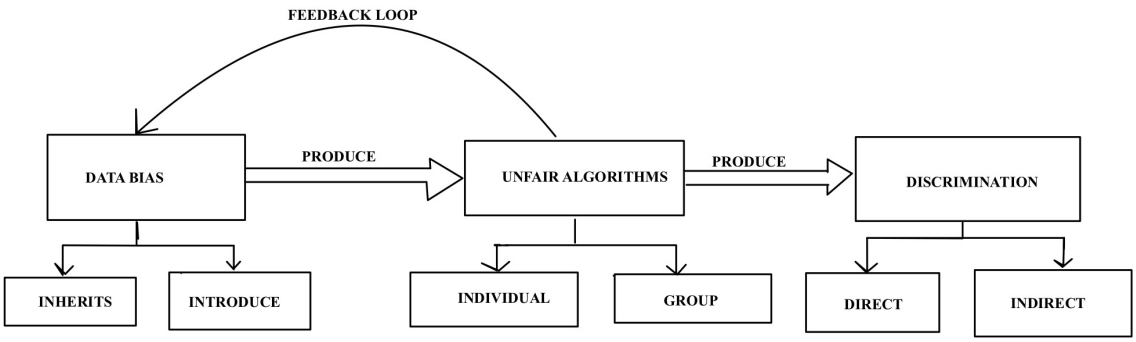

Figure 3: Transformation of bias in discriminatory results feedback loop [82]

Types of Bias in Anomaly detection Systems: Bias can exist in many shapes and forms, some of which can lead to unfair outcomes or discriminatory results in different anomaly detection systems. In reference [82], the author talks about the various sources of bias in

machine learning for cybersecurity, using their categorizations and descriptions to motivate future solutions to each source of bias introduced in the paper. In reference [79], the authors prepare a complete list of different types of bias in machine learning with their corresponding definitions according to data, algorithms, and the user-interaction loop. Here, we will reiterate the most important sources of bias introduced in these two articles, according to Figure 3.

- **Data Bias:** Data bias refers to systematic errors or inaccuracies in the data used to train and test an anomaly detection system. This bias can lead to incorrect or incomplete identification of anomalies, as the system cannot accurately learn and recognize patterns in the data. Data bias can be caused by various factors, such as how the data set is represented and sampled (representation bias), features considered in the data set, and how particular features are measured in the data set (measurement bias) [79].

- **Human Bias:** Many data sources used in training anomaly detection models are user-generated. Any inherent biases in the users might be reflected in the data generated. Human bias can be caused by factors such as the labeling of the data set (label bias), historical prejudice in the data set (historical bias), and when developers or users influence the model algorithms (cognitive bias) [82].

- **Learning Bias:** Learning bias is the inherent assumption of an algorithm when learning data. Algorithms modulate data behavior, and any bias in the algorithm could introduce biases in data behavior [79]. Learning bias can be caused by various factors, such as bias not present in the input data and added purely by the algorithm (algorithmic bias) [14] and inappropriate and disproportionate benchmarks for evaluating applications, such as imbalance class problems (evaluation bias) [108].

Bias Evaluation in Anomaly Detection Systems: Fighting against discrimination and bias has a long history in philosophy, psychology, and, recently, anomaly detection. However, to

achieve fairness and fight bias, we have to look at what fairness means as used in the context of anomaly detection. When detecting anomalies, fairness in anomaly detection ensures that the algorithms, data, and models are unbiased and equitable towards different groups or individuals. Some of the statistical metrics used to measure fairness [116] in anomaly detection systems are:

- **Equalized Odds:** Equalized odds requires that an anomaly detection model provides the same true positive rate (TPR) and false positive rate (FPR) across different groups (e.g., different network traffic sources, user demographics, malware families, or SMS spam-related keywords) [116].

- **Demographic Parity (or Statistical Parity:** Statistical parity ensures that anomaly detection models make positive predictions at the same rate for different groups(e.g., different network traffic sources, user demographics, malware families, or SMS spam-related keywords), regardless of actual ground truth labels [116].

- **Equal Opportunity:** Equal opportunity is a relaxed version of equalized odds, ensuring that an anomaly detection model maintains equal true positive rates (TPR) across different groups (e.g., different network traffic sources, user demographics, malware families, or SMS spam-related keywords) [116].

- **Treatment Equality:** Treatment equality ensures that the ratio of false negatives (FN) to false positives (FP) is the same across different groups (e.g., different network traffic sources, user demographics, malware families, or SMS spam-related keywords) [116].

Bias Mitigation in Anomaly Detection Systems: Bias mitigation refers to reducing or eliminating errors that may exist in the data or models. In reference [79], the authors came up with a list of different methods to mitigate biases in classification-based machine-learning techniques. Also, a survey by Mmaduekwe [82] examines bias and fairness issues with artificial

intelligence-driven cybersecurity applications. Figure 4 is the block diagram showing the taxonomy of bias mitigation in anomaly detection-driven applications, with the rounded rectangles in green and yellow showing my focus direction. In all these survey papers, non-addressed how to mitigate data bias, such as representation bias, measurement bias, sampling bias, etc, human bias, such as labeling bias, and learning bias, such as algorithmic and learning bias in anomaly detection systems for cybersecurity.

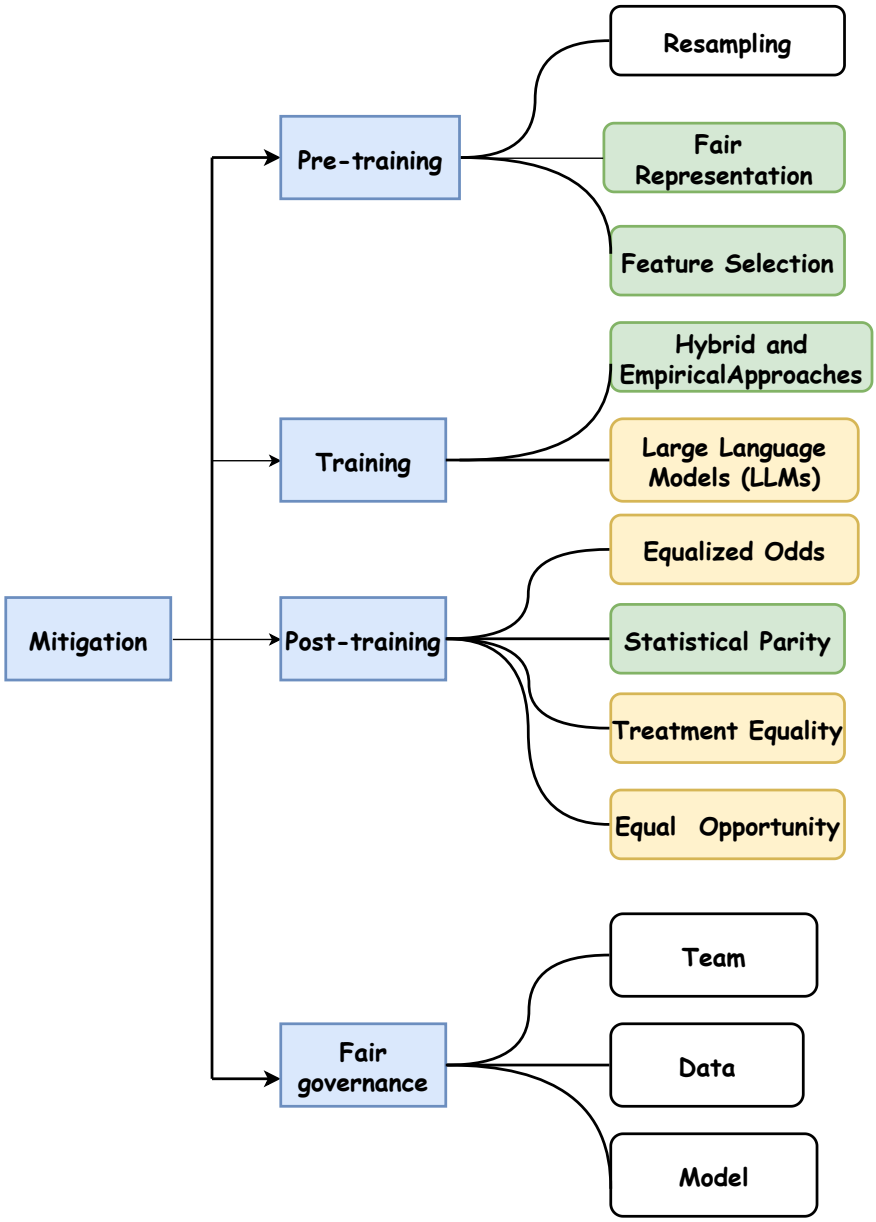

Figure 4: Overview of Taxonomy of bias mitigation[82]

Related Work

Advancements in cybersecurity have spurred research aimed at mitigating bias in anomaly detection techniques and rigorous statistical assessment. Various studies have explored novel approaches to artificial intelligence applications in cybersecurity and cyber defense, such as intrusion detection, malware detection, hybrid anomaly detection methods, and bias in machine learning applications.

Applying machine learning and deep learning techniques to cybersecurity problems has garnered significant attention in recent years. Several studies have contributed to advancing the field by proposing innovative approaches and methodologies.

A review of bias and fairness in artificial intelligence by González-Sendino et. al [52] categorize biases and how they are linked with different stages of an AI model's developement (including the data generation stage). A review by Mmaduekwe [82] examines bias and fairness issues with artificial intelligence-driven cybersecurity applications. Farnaaz and Jabbar [41] introduced a Random Forest modeling approach for Network Intrusion Detection Systems (NIDS). Their research demonstrated the effectiveness of Random Forest in detecting network intrusions, showcasing its potential for real-time threat detection and mitigation. Guyon and Elisseeff [55] provided a comprehensive review of variable and feature selection methods for intrusion detection systems. Their work outlined various techniques for selecting relevant features, contributing to optimizing NIDS performance. Hamed, Dara, and Kremer [57] proposed a novel approach for NIDS based on recursive feature addition and the Bigram technique. Their study introduced innovative feature engineering methods to improve the accuracy and efficiency of intrusion detection systems. Meftah, Rachidi, and Assem [78] focused on network-based intrusion detection using the UNSW-NB15 dataset. Their research contributed to evaluating intrusion detection algorithms on real-world network traffic data, providing valuable insights into the effectiveness of different approaches. Saheed, Arowolo, and

Tosho [98] presented an efficient K-means and genetic algorithm hybrid based on a Support Vector Machine for cyber intrusion detection systems. Their study introduced a novel hybrid approach, aiming to enhance the performance and scalability of intrusion detection systems in complex network environments. Sharma and Yadav [104] developed an optimal intrusion detection system using recursive feature elimination and an ensemble of classifiers. Their research highlighted the importance of feature selection techniques and ensemble learning methods in building robust and effective intrusion detection systems. Elmasry et al. [40] explored evolving deep-learning architectures for intrusion detection. Their work utilized a double PSO metaheuristic to develop deep learning models, contributing to creating more efficient intrusion detection systems. Mohammed and Gbashi [83] proposed an intrusion detection system based on deep learning and recursive feature elimination for the NSL-KDD dataset. Their study showcased the effectiveness of deep learning techniques and feature selection methods in improving intrusion detection accuracy. Zhang et al. [128] presented a real-time and ubiquitous attack detection system based on deep belief networks and support vector machines. Their research demonstrated the feasibility of using deep learning models for network intrusion detection in real-world scenarios. Carrier et al. [27] investigated the detection of obfuscated malware using memory feature engineering. Their study focused on developing techniques to identify stealthy malware through memory analysis, contributing to advancing malware detection methods. Wang et al. [119] proposed a method for detecting stealth software with Strider Ghostbuster, which involved analyzing system call sequences to identify malicious behavior. Their research provided insights into techniques for detecting stealthy software threats. Darabian et al. [33] introduced a deep-learning approach for detecting crypto-mining malware using static and dynamic analysis. Their study highlighted the importance of utilizing deep learning techniques for malware detection, particularly in emerging threats like crypto mining malware.

Current cybersecurity and cyber defense practices are focused on using machine learning-

based techniques, such as anomaly detection, to improve detection rates and reduce false alarms. Therefore, exploring novel practices that could help alleviate bias issues in anomaly detection systems for cybersecurity and cyber defense is a significant challenge. This research direction has the potential to develop fair anomaly detection systems in cyberspace and a fresh perspective on the field of specific applications.

While efforts have been made to select the best machine learning techniques and applications in cybersecurity and cyber defense, bias remains a threat that could limit their application. Further work is needed to develop fair, reliable, and effective systems that can steer the selection of anomaly detection systems, thereby addressing the bias problem. Possible approaches could be using unsupervised machine learning methods like DBSCAN and Isolation Forest to alleviate the human-induced labeling bias to label data for supervised learning, which is fast, or using feature selection methods like recursive feature elimination to solve measurement bias, and so forth.

A promising avenue for future research involves incorporating a feature learning method like deep autoencoders with a traditional machine learning model like logistic regression for malware detection to alleviate the representation bias problem. The results would be an advanced method that accomplishes a fair malware detection system and avoids unfair interference with sensitive features.

RESEARCH OBJECTIVES

Motivation

Cybersecurity threats pose significant risks to individuals, organizations, and nations worldwide, with increasingly sophisticated attacks exploiting vulnerabilities in digital systems. Traditional cybersecurity measures often struggle to keep pace with the evolving threat landscape, necessitating the development of more effective defense mechanisms. Machine learning (ML) techniques offer promise in bolstering cybersecurity by automating the detection of abnormal activities indicative of malicious behavior. However, the practical application of machine learning in cybersecurity is often faced with bias problems limiting its application [79, 82].

My work is motivated by recognizing and addressing the effect of bias, a critical gap in existing research on machine learning-driven anomaly detection for cybersecurity. This gap represents an area for exploration and further investigation because bias limits the usage and efficiency of machine learning applications in cybersecurity. By addressing the sources and effects of bias on machine learning for cybersecurity, I aim to focus on mitigating bias in anomaly detection systems and rigorous statistical assessment within the context of ML-based cybersecurity; the research seeks to enhance fair and efficient anomaly detection techniques. By providing unbiased models and robust statistical evaluations, this research aims to develop more resilient and adaptive cybersecurity defenses, ensuring that the model decisions do not reflect discriminatory behavior toward certain outcomes, spam keywords, or specific network segments. The following motivates me to utilize machine learning (ML) in cybersecurity, mainly through mitigating bias in anomaly detection systems and rigorous statistical assessment.

- **Escalating cyber threats:** With the proliferation of digital technologies and the interconnectedness of systems, cyber threats have become more prevalent and sophisticated.

malicious actors continuously develop new techniques to exploit vulnerabilities, posing significant risks to individuals, organizations, and societies.

- **Ineffectiveness of traditional approaches:** Traditional cybersecurity measures, such as rule-based systems and signature-based detection methods, often need help to keep pace with evolving threats. These approaches rely on predefined rules or patterns and may fail to detect novel or previously unseen attacks, leaving systems vulnerable to exploitation.

- **The power of machine learning:** Machine learning techniques offer a promising solution to enhance cybersecurity defenses. By leveraging algorithms that can learn from data, ML-based approaches enable automated detection of abnormal activities indicative of malicious behavior with insignificant human interference [16]. This adaptability and learning capability make ML well-suited for addressing the dynamic nature of cyber threats.

- **Bias Problem:** Despite the applications of ML-based anomaly detection in cybersecurity, bias still poses a severe threat to these systems, thereby limiting their efficiency and usage, as outlined in section 2.1.3. The scarcity of this topic in many machine learning applications in cybersecurity also shows the research gap in this domain [82].

- **Importance of Statistical Assessment:** Rigorous statistical assessment is crucial to validate the performance and robustness of anomaly detection models. Statistical techniques such as hypothesis testing, cross-validation, and significance analysis provide objective measures to evaluate the efficacy of the model and generalize the findings to different contexts. By applying a rigorous statistical assessment, researchers can establish the credibility and trustworthiness of ML-based cybersecurity solutions.

This research aims to explore and adapt machine learning (ML) in cybersecurity, mainly through mitigating bias in anomaly detection systems and rigorous statistical assessment. It

aims to address the bias problem in machine learning-driven anomaly detection applications in cybersecurity, which can also be extended to other domains.

GQM

The Goal Question Metric (GQM) approach, introduced by Basili et al.[17], is a goal-oriented framework that we utilize to structure and guide this research. As illustrated in Figure 5, the GQM framework consists of three key components: **goals**, **questions**, and **metrics.** First, a set of research goals (RGs) is defined. These goals are then refined into specific research questions (RQs), each designed to address different aspects of the research objectives. Finally, quantifiable metrics are established to measure and evaluate the answers to these research questions effectively. Furthermore, in Table 1 the specific chapters that address each research question are outlined, providing a structured roadmap for how this dissertation contributes to the field.

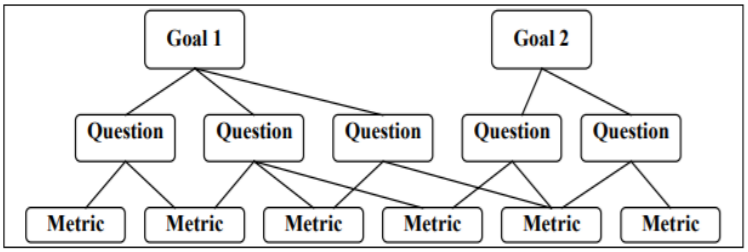

Figure 5: GQM Approach [26]

RG1: My primary research goal is to address the challenge of representation bias in anomaly detection systems. To achieve this, I will implement a fair representation learning approach using a deep autoencoder, followed by logistic regression for malware classification. This approach is designed to improve the automatic extraction of less sensitive and more relevant features while minimizing the dependence on domain experts for feature engineering.

RQ1.1: How can *feature or representation learning* be used to reduce *representation bias* in *malware detection-based* anomaly detection systems, and how does this affect model performance and fairness?

RQ1.2: Is there a significant difference in the model with fair representation and other state-of-the-art approaches?

Research Metrics 1: Detection rate (DR), accuracy, Matthew correlation coefficient (MCC), statistical parity, and Wilcoxon signed rank test.

RG2: To evaluate the effectiveness of *feature selection* to mitigate biases that arise from the way certain features are considered (selection bias) within the datasets of *intrusion-based* anomaly detection systems.

RQ2.1: Can feature selection techniques mitigate selection biases introduced during data collection?

RQ2.2: Is there a significant difference in the model with selected features and other state-of-the-art approaches?

Research Metrics 2: F1-Score, False Alarm Rate, MCC, DR, precision, and ROC AUC.

RG3: To mitigate *human-induced bias* in labeling datasets to train *intrusion detection-based* anomaly detection.

RQ3.1: How accurately can unsupervised methods mitigate human-induced label bias in data sets used to train intrusion detection-based anomaly detection systems?

Research Metrics 3: False alarm rate, F1 score, Area Under the Receiver Operating Characteristic (AUCROC) curve, detection rate (or recall) and precision.

RG4: To mitigate *human-induced label bias* in *SMS spam detection* utilizing the advanced capabilities of *large language models (LLM)* and *contextualized clustering*.

RQ4.1: What are the most and least successful prompt designs for LLMs-based to help reduce human-induced labeling bias for SMS spam datasets?

RQ4.2: How well do LLMs reduce human-induced labeling bias for SMS spam datasets compared to state-of-the-art machine learning models?

RQ4.3: How effective is the context-driven clustering approach in reducing human-induced label bias compared to traditional clustering techniques such as DBSCAN and K-means in SMS spam detection?

RQ4.4: Can a context-driven clustering technique effectively automate the selection of representative samples to generate prompts to fine-tune large-language models in SMS spam detection?

Research Metrics 4: Precision, Recall, Balanced Accuracy (ACC), Statistical Parity Difference (SPD), Equal Opportunity Difference (EOD) and Treatment Equality Difference (TED).

Chapters Addressing Research Questions (RQs)

RQ#	**Paper/Chapter Title**
RQ1.1	A Hybrid Anomaly Detection Approach for Obfuscated Malware
RQ1.2	A Hybrid Anomaly Detection Approach for Obfuscated Malware
RQ2.1	Improving Network Intrusion Detection Performance : An Empirical Evaluation Using Extreme Gradient Boosting (XGBoost) with Recursive Feature Elimination
RQ2.2	Improving Network Intrusion Detection Performance : An Empirical Evaluation Using Extreme Gradient Boosting (XGBoost) with Recursive Feature Elimination
RQ3.1	An Empirical Internet Protocol Network Intrusion Detection using Isolation Forest and One-Class Support Vector Machines

RQ4.1	Using Large Language Models to Mitigate Human-Induced Bias in SMS Spam: An Empirical Approach
RQ4.2	Using Large Language Models to Mitigate Human-Induced Bias in SMS Spam: An Empirical Approach
RQ4.3	Reducing Human-Induced Label Bias in SMS Spam with Context-Enhanced Clustering (CEC)
RQ4.4	Reducing Human-Induced Label Bias in SMS Spam with Context-Enhanced Clustering (CEC)

A HYBRID ANOMALY DETECTION APPROACH FOR OBFUSCATED MALWARE

Contribution of Authors and Co-Authors

Manuscript in Chapter 4

Author: Gerard Shu Fuhnwi

Contributions: Problem identification and proposing solution, running experiment, manuscript writing, creating tables and figures. Primary writer

Co-Author: Dr. Clemente Izurieta and Dr. Matthew Revelle

Contributions: Contribution in manuscript editing/writing, provided feedback, guidance and advice.

Manuscript Information Page

Gerard Shu Fuhnwi, Dr. Matthew Revelle and Dr. Clemente Izurieta

IEEE International Conference on Cyber Security and Resilience (CSR)

Status of Manuscript:
____ Prepared for submission to a peer-reviewed journal
____ Officially submitted to a peer-reviewed journal
____ Accepted by a peer-reviewed journal
_×_ Published in a peer-reviewed journal

IEEE
02 September 2024
10.1109/CSR61664.2024.10679474

Abstract

With the rapid evolution of malicious software, cyber threats have become increasingly sophisticated, employing advanced obfuscation techniques to evade traditional detection methods. This study presents a hybrid anomaly detection approach applied to obfuscated malware. Even though there is a large body of research in this field, existing malware detection techniques have drawbacks, such as requiring large amounts of data, trustworthiness (imprecise results) of algorithms, and advanced obfuscation. There is a need to employ solid and efficient techniques for malware detection to overcome these challenges. This paper proposes a hybrid approach, combining an autoencoder with traditional machine-learning methods to create an efficient malware detection framework. We used the malware memory dataset (MalMemAnalysis-2022) to evaluate this framework. The experimental results show our proposed approach can detect obfuscated malware when a deep autoencoder used for feature learning is combined with logistic regression. It is extremely fast with an Accuracy, Detection Rate (DR), Matthew Correlation Coefficient(MCC), and Statistical Parity Difference (SPD) of 99.97%, 99.98%, 99.93%, and 0.03%, respectively.

Introduction

In an era dominated by interconnected technologies and digital dependence, the cybersecurity landscape faces an ever-evolving and persistent threat—malware. Malware are malicious programs that can potentially compromise computer systems' integrity, confidentiality, and availability. As a pervasive threat, malware manifests itself in diverse forms, each using different tactics, techniques, and procedures (TTPs). Malware can result in consequences ranging from website defacement [2] to the loss of human life [9]. Moreover, the constant evolution of defense evasion techniques in malware continues to require advancement in detection methods.

There are many categories of malware, such as viruses, trojan horses, spyware, ransomware, bots, botnets, worms, and others [27]. While there are many malware categories, there are a number of common tactics and techniques [27] shared between different types of malware. A detection system that utilizes features which indicate the implementation of those tactic and techniques will be capable of detecting malware of various categories. To counteract malware infiltrating systems, gaining access to information, accessing authorized users, and other cybercrimes, the best alternative is to create detection systems capable of detecting and stopping all forms of malware.

Malware is frequently obfuscated to evade signature-based detection methods [15], and to increase the effort required to reverse engineer the malware when it has been collected. Obfuscation can hide the implementation of malicious behaviors from static analysis and require sophisticated detection techniques [45]. These detection methods illustrate the applications of anomaly detection in Machine Learning (ML), which involves identifying data patterns (outcomes, values, or observations) that differ significantly from the rest of the data and are essential for identifying and preventing malware [45]. The basic intuition of these learning systems is figuring out which training data to fit into the machine learning method to make the quickest and most accurate assessment. Anomaly detection techniques perform malware detection by attempting to identify malicious behavior. These anomaly detection techniques use their knowledge of what constitutes normal behavior to decide the maliciousness of a program under inspection. Anomaly detection systems take in a set of features together with large sample sizes to compare and contrast differences. These features can be input in different formats, which is a factor that determines how the anomaly detection system should be used. While some anomaly detection algorithms are focused on minimizing false positives, others are focused on speed, scalability, accuracy, and precision. Therefore, choosing different algorithms corresponding to the objective and input type has an enormous impact on the result of the anomaly detection system. Hybrid anomaly detection is one such technique, which combines

two or more ML models to enhance the overall effectiveness of anomaly detection.

Several approaches to malware detection using hybrid anomaly detection techniques [63] exist. However, in most of the work, complexity, and time consumption are high, fairness of the algorithms are not mentioned and there is no statistical analysis to compare the various models, making them unsuitable for real-world applications. This motivates the development of a fair, fast, efficient, and easy-to-interpret method for obfuscated malware detection using the most relevant features captured through autoencoders.

Main Contributions: The main contributions of this research include [46]:

- An anomaly detection approach for malware analysis, combining a deep autoencoder and logistic regression to increase trustworthiness and transparency in current obfuscated malware detection. To the best of our knowledge, this is the first time this approach has been used in anomaly detection.
- Analysis of the performance of our approach. We perform a comparative analysis of combining a deep autoencoder with logistic regression against state-of-the-art and baseline methods.
- Improvements to the overall anomaly detection speed, fairness, and reduced ML model complexity.
- Improvements in accuracy, detection rates, and MCC when compared to other models and benchmarks in related works with very low statistical parity difference (SPD).

The rest of this paper is organized as follows. Section II discusses related work. Section III proposes a hybrid anomaly detection framework for obfuscated malware including dataset description, data transformation, deep autoencoder, and logistic regression. In Section IV we analyze empirical results, and Section V concludes the paper.

Related Work

Many studies have been conducted to protect devices connected to the Internet from malware intrusion using anomaly detection-based techniques. Research on static anomaly detection, which identifies the program's file structure characteristics under inspection, is used to detect malicious code [6-7]. However, static anomaly detection techniques can not detect obfuscated or hidden malware because of their inability to execute software. Conversely, dynamic anomaly detection techniques can detect malicious code using information gathered from the program's execution, making them capable of detecting obfuscated or hidden malware [8-14]. On the other hand, hybrid anomaly detection techniques can detect malware by combining both static and dynamic anomaly detection techniques [27, 34, 120].

This section highlights previous studies on anomaly detection techniques for malware detection through static, dynamic, and hybrid methods.

Static Anomaly Detection

Li et al.[70] proposed a Fileprint (n-gram) analysis as a means to detect malware. During the training phase, an ML model or set of models is derived to characterize various file types of a system based on their structural (byte) composition. Li et al.'s technique exhibited 94.5% detection rate for PDF files with embedded malware.

Zhang et al.[127] proposed a deep learning-based malware detection framework for classifying eight types of ransomware. To reduce the dimensionality of the opcode sequences used as features, the term frequency-inverse document frequency (TF-IDF) algorithm was applied to filter out N-gram Opcodes with low amount of information according to their TF-IDF values. After the opcode sequence was divided into several patches, a self-attention convolutional neural network (SA-CNN) model was trained using these patches. Finally, to analyze the relationship between opcode sequences and classify the ransomware, a directional self-attention network

(Di-SAN) model was used.

Dynamic Anomaly Detection

Wang and Stolfo [66] proposed a tool called PAYL, which calculates the expected payload for each service (port) on a system by using a centroid model. The detector compares incoming payloads with the centroid model by measuring the Mahalanobis distance [77] between the two. The author's technique had an overall detection rate of approximately 60 % and a false positive rate of 1 % or lower.

Lee and Stolfo [119] proposed using data mining techniques, association rules, and frequent episodes for intrusion detection systems. The association rules and frequent episodes are collectively referred to as rule sets and serve as knowledge of what is normal for the host's services.

Boldt and Carlson [22] proposed a Forensic Tool Kit (FTK) to identify Privacy-Invasive Software (PIS) such as adware and spyware. The basic approach for the baseline consists of initially creating a system free of PIS, that is, a clean system. This tool is used to depict the file system of the target host. Boldt and Carlson discovered that their technique produced false positives and negatives for Ad-Aware.

Hofmeyr et al. [61] proposed a technique to help monitor system call sequences to detect maliciousness by developing profiles that represent the normal behavior of the system's services. The Hamming distance determines how closely the system call sequence resembles another, where processes with large Hamming distances are considered abnormal. The Hofmeyr et al. technique found intrusions that attempted to exploit various UNIX programs like sendmail, lpr, and ftpd.

Sekar et al. [102] proposed a Finite State Automata (FSA) for anomaly detection. Each node in the FSA represents a state (program counter) in the program (static)/process (dynamic) under inspection (PUI), which the algorithm utilizes to learn normal data faster and perform better

detection, where system calls give transitions in the FSA. When a system call is invoked, a new transition is added to the automation, where automations resulting from multiple executions are considered normal.

Sato et al. [100] proposed an n-gram approach and compared it to the FSA approach [102]. Sato et al. evaluated the two approaches on *httpd*, *ftpd*, and *nsfd*, where the FSA was found to have a lower false positive rate when compared to the n-gram approach.

Joen et al. [65] proposed a decreasing additive-increase/multiplicative-decrease (DAIMD) model for detecting new and variant IoT malware. Performing dynamic analysis in a closed-based nested virtual environment, network, process, memory, and virtual file system behaviors were extracted as features and converted into images. These images are used to train a CNN model for detecting IoT malware.

Hybrid Anomaly Detection

Carrier et al. [27] proposed using VolMemLyzer to detect obfuscated malware. Using a memory feature extraction technique called VolMemLyzer, base learners and meta-learners are combined to detect obfuscated malware by stacking them together. By combining naive Bayes, random forest, and decision trees as base learners with logistic regression as the meta-learner, the authors obtained an accuracy of 99.00% and an F_1 score of 99.02%.

Wang et al. [120] proposed a technique called the cross-view difference approach for detecting a type of malware known as ghostware, which is malware that attempts to hide its existence from the operating system's querying utilities. For example, the proposed approach to mitigate hidden files requires comparing results from a user-mode program like *dir* or corresponding API calls to the equivalent data found by inspecting the file system data structures directly.

Darabian et al. [34] proposed a method for detecting cryptomining malware through hybrid analysis. Using opcodes extracted from static analysis and system calls generated

from dynamic analysis, then LSTM, attention-based LSTM (ATT-LSTM), and CNN are used to evaluate their performance for classification by a softmax function.

Overall, these studies demonstrate the effectiveness of anomaly detection techniques and feature extraction for malware detection. Our study builds on these papers by extending the work of Tristan et al. [27] on obfuscated malware detection, and mitigates the limitations mentioned above by combining a deep autoencoder with logistic regression on the obfuscated malware dataset. We demonstrate high accuracy, MCC, and detection rates in less computational time. Our proposed approach also has a low statistical parity difference, indicating how fair, trustworthy, and transparent our approach is.

Proposed Approach

We propose a hybrid anomaly detection approach for malware analysis. The step-by-step process adopted in the proposed methodology is described herein and shown in Figure 10.

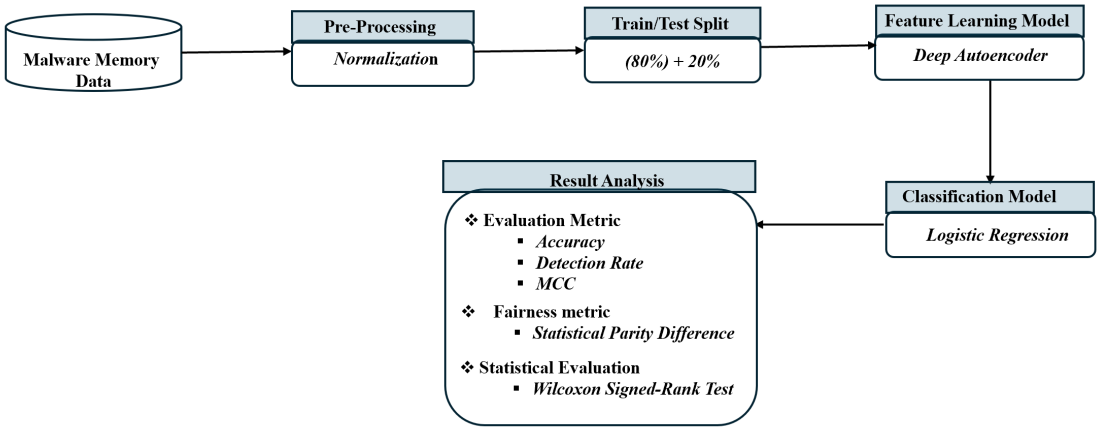

Figure 6: Pipeline of our hybrid approach. Rectangles represent the model's processes, cylinders represent stored data, and the arrows the direction of flow

Malware Memory Data

The malware memory dataset is balanced, comprising 50% malicious and 50% benign memory dumps. Malicious dumps include different malware categories: Ransomware, Spyware, and Trojan Horse. The dataset contains 58,596 records, with 29,298 being benign, 29,298 being malicious, and 55 features. This dataset is designed to test obfuscated malware detection methods through memory [27].

Preprocessing

The main objective of data preprocessing is to enhance the quality of the data, making it easier to work with and enabling more accurate and efficient results for ML algorithms [81].

- **Normalization:** Normalization is a technique that scales numerical variables in a particular range, typically [0, 1] or [-1, 1]. Normalization aims to bring different features into a similar scale, making it easier for ML algorithms to process the data and reduce the risk of certain features dominating the model due to their larger numerical values. Several normalization methods; the most common are Min-Max Scaling and Z-score.

- **Standardization:** In our proposed approach, the Z-score standardization, which scales the value of the entire feature to a standard range of [0,1], is used. In Z-score standardization, the average μ of each numerical feature is calculated and then subtracted from the feature value x. The subtracted average and feature x are then divided by the standard deviation σ as shown in the formula:

$$X_i = \frac{x - \mu}{\sigma}$$

Train/Test Split

The 80/20 split in ML refers to dividing your dataset into two portions: an 80% portion for training your model and a 20% portion for testing its performance. This division is a common

practice to assess how well your model generalizes to new, unseen data. The 80/20 split is a common choice because it balances having enough training data and a sufficiently large test set to evaluate performance.

Feature Learning Model

A feature learning model is a cornerstone in machine learning, designed not only to automatically extract relevant features or representations from raw data but also to uphold principles of fairness. Instead of relying solely on manually engineered features, these models can autonomously unearth hierarchical and meaningful representations from the input data while ensuring fairness across different demographic groups. One example of a feature learning model that integrates fairness considerations is a deep autoencoder, exemplifying the power of learning intricate and informative representations through neural network architectures while promoting equitable outcomes in decision-making processes.

Deep Autoencoder A deep autoencoder, a type of neural network architecture, is a key player in the realm of unsupervised learning. It's comprised of two neural networks: an encoder and a decoder. The deep autoencoder's primary objective is to acquire a compact and informative input data representation. The encoder efficiently compresses the input into a lower-dimensional representation (encoding), and the decoder skillfully reconstructs the original input from this representation [5]. The architecture of a deep autoencoder is as follows:

- **Encoder:** The encoder network transforms the raw input data into a lower-dimensional representation. Each layer of the encoder extracts increasingly abstract and higher-level features.
- **Bottleneck Layer:** This is the layer in the middle of the network where the input is highly compressed. It contains the learned features or encoding of the input data.

- **Decoder:** The decoder network takes the compressed representation from the bottleneck layer and reconstructs the original input data. Like the encoder, each layer of the decoder progressively refines the reconstruction.

Classification Model

For malware analysis, we used logistic regression, a supervised learning algorithm, to classify obfuscated malware into benign and malicious memory dumps.

- **Logistic Regression:** Logistic regression is a widely used algorithm because of its versatility and interpretability. It is beneficial when binary classification, probability estimation, and simplicity are essential. Logistic regression is a statistical technique that helps predict the probability of obfuscated malware belonging to benign and malicious memory dumps.

Result Analysis

The result analysis includes evaluation metrics and assessment of their statistical significance.

Evaluation and Fairness Metrics: Once the logistic regression classifier is trained using the 80% portion of the dataset, it is evaluated using the 20% portion. The classifier's performance is evaluated using accuracy, detection rate (DR), and Matthews Correlation Coefficient (MCC), while fairness is assessed using the statistical parity difference (SPD). These performance metrics are calculated using True Positive (TP), False Negative (FN), False Positive (FP), and True Negative (TN) as shown in the confusion matrix [18] in Table 13.

Table 2: Confusion Matrix: A contingency containing four metrics, True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN).

Memory Dumps		**Predicted Class**	
		Yes	**No**
Actual Class	**Yes**	**TP**	**FN**
	No	**FP**	**TN**

- **Accuracy:** Is defined as the total number of correctly classified memory dumps divided by the total number of memory dumps which can be calculated using the formula:

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

- **Detection Rate:** It is the ratio between the total number of malicious memory dumps detected by the logistic regression classifier to the total number of malicious memory dumps present in the dataset [41] which can be calculated using the formula:

$$DR = \frac{TP}{TP + FN}$$

- **Matthews Correlation Coefficient (MCC):** Is defined as the Pearson product-moment correlation coefficient using the confusion matrix in Table 13 between the actual malicious memory dumps and predicted malicious memory dumps [32] which can be calculated using the formula below.

$$MCC = \frac{TN * TP - FN * FP}{\sqrt{(FP + TP)(FN + TP)(TN + FP)(TN + FN)}}$$

- **Statistical Parity Difference (SPD):** SPD measures the difference in the proportion of malicious memory dumps between different memory dumps. It aims to ensure the model's predictions are balanced across the memory dumps, indicating fairness and

lack of bias in its decision-making process. Smaller values indicate a disparity in the proportion of malicious predictions between the memory dumps, which is calculated using the formula:

$$SPD = \frac{TP}{TP+FN} - \frac{TN}{TN+FP}$$

Statistical Evaluation: We utilize the Wilcoxon signed-rank test to assess the statistical significance of our proposed approach. This non-parametric statistical hypothesis test ranks the differences between two classifiers, disregards the signs, and compares the ranks of the positive and negative differences[36]. It is often used when the assumptions of a paired t-test are violated (such as when the data is not normally distributed). To compare the performance of our proposed approach against logistic regression, we used the Wilcoxon signed-rank test to test whether there is a difference between the performances on the median of accuracy, detection rate, and MCC. The null hypothesis (H_0) for the Wilcoxon signed-rank test states that there is no significant difference between the median performances of the two models. In contrast, the alternative hypothesis (H_1) states a statistically significant difference, unlikely to have occurred by chance, between the median performances of the two models.

Experimental Results

All experiments were performed in Python using default parameters for logistic regression [1]. The deep autoencoder consisted of two encoder levels, a bottleneck, and two decoder levels. All experiments were run on Intel (R) Core(TM) i7-10510U CPU @ 1.80GHz 2.30GHz processor with 16 GB RAM. Logistic regression Training and testing were performed on the 55 features and logistic regression with the features learned from the autoencoder with 100 different random seed values. An 80%/20% training/testing split of the data was used. The proposed approach

[1] Sklearn

was evaluated by finding the median value of the accuracy, detection rate, and MCC metrics. The highlighted row in Table 3 shows the performance of our proposed approach. It is evident from Table 3 that our proposed approach has high accuracy, detection rate, and MCC.

Our proposed approach has a low SPD, which ensures that the model's predictions are fair and equitable across the different memory dumps, promoting trust and transparency.

The Wilcoxon signed-rank test was used to compare the differences between our proposed approach and logistic regression. Table 10 compares our proposed approach against logistic regression on all metrics used. From Table 10, comparing our proposed approach against logistic regression, we can see that the p-values of the test for accuracy, detection rate, and MCC are all significant at $\alpha = 0.05$. We can statistically confirm that our proposed approach outperforms logistic regression.

The highlighted last column in Table 3 shows the approximate run times for a memory dump to be classified as malicious or benign for all samples in the 20% test data. From the table, we can see that the classification time is linear, which demonstrates the scalability of our proposed approach. Our proposed approach has a classification time of 0.00007 milliseconds per sample, significantly lower than all models listed in Table Table 11. Comparison of these related works, as shown in Table 11, shows the relevance of our proposed approach as a fast method to detect hidden or obfuscated malware.

Table 3: Accuracy, Detection Rate, MCC, SPD, and Classification Time. The best results are printed in skyblue.

Model	**Accuracy**	**Detection Rate**	**MCC**	**SPD**	**Classification Time (seconds)**
Logistic Regression	0.9962	0.9966	0.9923	0.0009	1.2665
Proposed Approach	0.9997	0.9998	0.9993	0.0003	0.8541

Table 4: Results of Wilcoxon Signed-Rank Test comparing our proposed approach against Logistic Regression

Comparison	Evaluation metric	Positive Rank	Negative Rank	Rank (Test Statistics)	Hypothesis ($\alpha = 0.05$)	p-value
Proposed Approach vs. Logistic Regression with significant values highlighted in table 3	**Accuracy**	**5050**	**0**	**5050**	**Rejected**	**<0.001**
	Detection Rate	**5050**	**0**	**5050**	**Rejected**	**< 0.001**
	MCC	**5050**	**0**	**5050**	**Rejected**	**< 0.001**

Table 5: Comparison with other obfuscated malware detection methods

Method	Accuracy	Detection Rate	MCC	Speed	Statistical Test	Complexity	SPD
Tristan et al. [27]	High	Not Mentioned	Not Mentioned	Very High	Not Mentioned	Medium	Not Mentioned
Xu et al.[125]	Very High	Not Mentioned	Not Mentioned	Medium	Not Mentioned	Medium	Not Mentioned
Bozkir et al.[24]	High	Not Mentioned	Not Mentioned	Medium	Not Mentioned	High	Not Mentioned
Nissima et al.[89]	High	Not Mentioned	Not Mentioned	Medium	Not Mentioned	Medium	Not Mentioned
Javaheri and Hosseninzadeh [64]	Medium	Not Mentioned	Not Mentioned	Medium	Not Mentioned	High	Not Mentioned
Proposed Approach	Very High	Very High	Very High	Extremely High	Included	Low	Included

Conclusion and Future work

In this paper, a hybrid anomaly detection method combining a deep autoencoder and logistic regression was implemented on an obfuscated malware dataset. We compared our proposed approach with logistic regression using the following metrics: accuracy, detection rate, MCC, and SPD. Because of feature learning, the computational cost is decreased, and our proposed approach has an accuracy of 99.97%, a detection rate of 99.98%, an MCC of 99.93%, and an SPD of 0.03%. We also compared our proposed approach with logistic regression using the Wilcoxon signed-rank test and found that our proposed approach is better or competitive than state-of-the-art and baseline methods. Moreover, our proposed approach is compared with related works on malware detection. The comparison results in Table 11 showed that a deep autoencoder with logistic regression has better performance, low SPD, and requires less classification time.

For future work, we would like to use model interpretation tools such as SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) to gain insights into potential sources of bias.

IMPROVING NETWORK INTRUSION DETECTION PERFORMANCE : AN EMPIRICAL EVALUATION USING EXTREME GRADIENT BOOSTING (XGBOOST) WITH RECURSIVE FEATURE ELIMINATION

Contribution of Authors and Co-Authors

Manuscript in Chapter 5

Author: Gerard Shu Fuhnwi

Contributions: Problem identification and proposing solution, conducting experiment, manuscript writing, creating tables and figures. Primary writer

Co-Author: Dr. Clemente Izurieta and Dr. Matthew Revelle

Contributions: Contribution in manuscript editing/writing, provided feedback, guidance and advice.

Manuscript Information Page

Gerard Shu Fuhnwi, Dr. Matthew Revelle and Dr. Clemente Izurieta

3rd IEEE International Conference on AI in Cybersecurity (ICAIC)

Status of Manuscript:
____ Prepared for submission to a peer-reviewed journal
____ Officially submitted to a peer-reviewed journal
____ Accepted by a peer-reviewed journal
_×_ Published in a peer-reviewed journal

IEEE
16 February 2024

Abstract

In cybersecurity, Network Intrusion Detection Systems (NIDS) are essential for identifying and preventing malicious activity within computer networks. Machine learning algorithms have been widely applied to NIDS due to their ability to identify complex patterns and anomalies in network traffic. Improvements in the performance of an IDS can be measured by increasing the Matthew Correlation Coefficient (MCC), the reduction of False Alarm Rates (FARs), and the maintenance of up-to-date signatures of the latest attacks to maintain confidentiality, integrity, and availability of services. Integrating machine learning with feature selection for IDSs can help eliminate less important features until the optimal subset of features is achieved, thus improving the NIDS.

In this research, we propose an approach for NIDS using XGBoost, a popular gradient boosting algorithm, with Recursive Feature Elimination (RFE) feature selection. We used the NSL-KDD dataset, a benchmark dataset for evaluating NIDS, for training and testing. Our empirical results show that XGBoost with RFE outperforms other popular machine learning algorithms for NIDS on this dataset, achieving the highest MCC for detecting NSL-KDD dataset attacks of type DoS, Probe, U2R, and R2L and very high classification time.

Introduction

With the increasing reliance on technology in our daily lives, cybersecurity has become a crucial aspect of ensuring confidentiality, integrity, and availability of information. Network Intrusion Detection Systems (NIDS) (an application of anomaly detection [45], which in machine learning, is the process of finding data patterns (outcomes, values, or observations) that deviate from the rest of the other observations or outcomes) are essential for identifying and preventing attacks within computer networks. Computer network attacks come in many forms, including Denial of Service (DoS), Probe, User to Root (U2R), and Remote to User (R2L).

Denial of service (DoS) is one of the most common attacks on network resources that make services inaccessible to users [91]. Remote to User (R2L) is another computer attack where attackers send packets to another computer or server over a network without permission. User to Root (U2R) is a third type of attack in which intruders access the network resources as a normal user, and after several attempts, they gain access as a potential root user. Probing is a fourth type of computer attack in which the attackers scan through network devices to check for weaknesses in the network topology and then use them in the future for illegal activities [8]. The most common types of probing attacks are network scans, portsweep, ipsweep, and satan. Traditional rule-based NIDS have proven insufficient in detecting these attacks (DoS, Probe, U2R, and R2L), as they often rely on known attack patterns, so machine learning algorithms have been widely applied to NIDS due to their ability to identify complex patterns and anomalies in network traffic. In recent years, gradient boosting algorithms, such as XGBoost [31], have become increasingly popular in machine learning due to their ability to handle missing values in datasets, the integration of regularization techniques, their optimization for parallel and distributed computing, optimized tree pruning, and a customizable objective function, thereby making it outperform other baseline and state-of-the-art machine learning algorithms [75]. In this paper, we present an approach for NIDS using XGBoost with Recursive Feature Elimination (RFE) for feature selection. RFE is a feature selection technique that recursively eliminates less important features until the optimal subset of features is achieved. The NSL-KDD dataset [111] was used for evaluation, and it contains attacks such as DoS, R2L, U2R, and Probe, a benchmark dataset for evaluating NIDS. Our **goal** can be stated as:

Goal: *Evaluate the effectiveness of XGBoost with RFE in detecting network intrusions and compare its performance to other baseline and state-of-the-art machine learning algorithms.*

Numerous intrusion detection approaches rely on feature selection, but many lack explicit

discussions on complexity and time consumption, often rendering them impractical for real-world applications due to potential inefficiencies. This prompts our proposal for a rapid, efficient, and straightforward intrusion detection solution that leverages Recursive Feature Elimination (RFE) to identify the most impactful features, ensuring both speed and effectiveness in real-world scenarios.

Main Contributions: The main contributions of this research include [47]:

- An intrusion detection framework that uses a two-layer ensemble learning (combining RFE with XGBoost) to improve current intrusion detection solutions.
- An expansion of the dataset's feature set by incorporating over 78 new features in the NSL-KDD for the purpose of testing and evaluating the proposed framework.
- Improvements to the overall speed of detection.
- Evidence of model performance improvements through statistical tests.
- Improvements to detection capabilities on selected attack types when compared to other models and benchmarks in related work.

This paper presents experimental results, showing that XGBoost with RFE outperforms other popular machine learning algorithms for NIDS on the NSL-KDD dataset, achieving the highest MCC for detecting NSL-KDD dataset attacks of type DoS, Probe, U2R, and R2L. This approach also achieved high precision, recall, ROC AUC, F1-score values, and low False Alarm Rate (FAR), indicating its effectiveness in detecting network intrusions. Overall, our empirical results demonstrate the potential of XGBoost with RFE for NIDS and provide valuable insights for future research in this field.

The rest of this paper is organized as follows. Section 7 discusses related work. Section 5 presents our proposed approach, including phases such as data set description, preprocessing,

feature selection, and classification model. The experimental results are presented in Section 7, and Section 5 concludes the paper describing possible future work.

Related Work

Several studies have explored the application of data mining, statistics and machine learning algorithms for network intrusion detection, including gradient boosting algorithms like XGBoost and feature selection techniques like RFE.

Alkasassbeh and Almseidin [8] used three machine learning methods (J48, MLP and Bayes Network classifiers) on the KDD dataset [19] for detecting and classifying attacks such as DoS, R2L, U2R, and Probe, with the J48 classifier achieving the highest accuracy.

In [41], Farnaaz and Jabbar proposed a detection intrusion system using a random forest with feature subset selection method called symmetrical uncertainty. Experimental results were conducted on the NSL-KDD dataset [111]. Empirical results show that the proposed model achieved a low FAR and high recall.

Recently feature selection has been applied to combine machine learning techniques for intrusion detection. A study by Meftah et al.[78] applied RFE and random forests as feature selection methods to the UNSW-NB15 dataset [86], then apply machine learning techniques such as Logistic Regression, Gradient Boost Machine, and Support Vector Machines. Similarly other papers such as; Elmasey et al.[40], Network Harmad et al.[57],Sharma et al.[104], Kunhare et al.[68], Mohammed et al.[83], Saheed et al.[98],Souza et al.[124], Zhang et al.[128], Wang et al.[117], Aljawarneh et al.[7] and Louk et al.[74] both applied feature selection methods together with machine learning techniques for network intrusion detection systems. These papers compare machine learning algorithms with feature selection without the use of statistical methods for choosing their final model and time-consuming issues in their work. To overcome these drawbacks in the models listed in the related works, one can look at the statistical significance tests to compare the performance of machine-learning models and quantify the

likelihood of the samples of performance scores being observed, given the assumption that they were drawn from the same distribution and classification time of each machine learning model. There are also other limitations in the related works; for example, lack of good preprocessing of the NSL-KDD dataset for the categorical features such as protocol_type (3 types), service (70 types), flag (11 types), and label (23 types), a lack of evaluation techniques for these machine learning techniques due to the high level of imbalances in the class labels.

Overall, these studies demonstrate the effectiveness of machine learning and feature selection for network intrusion detection. Our study builds on these papers to extend the work of Farnaaz et al.[41], Souza et al.[124], Louk et al.[74] on intrusion detection and the above-mentioned limitations by applying XGBoost with RFE to the NSL-KDD dataset and achieving a high MCC. By combining XGBoost with RFE, additional evaluation techniques for class imbalances, classification time of each attack, and statistical inference, we could identify the essential features of our model and achieve high performance in detecting network intrusions.

Proposed Approach: XGBoost with RFE

The proposed approach to test the effectiveness of XGBoost with RFE in a network intrusion detection system has seven steps described herein and shown in Fig. 10.

XGBoost with RFE

Step 1: Read NSL-KDD dataset.

Step 2: Preprocess data.

Step 3: Apply recursive feature elimination.

Step 4: Give XGBoost the select features for training.

Step 5: Find the attack type and classes.

Step 6: Evaluate the model performance using detection rate, false alarm, Matthews correlation coefficient, F_1 score, the area under the curve of the receiver operating curve, and precision on

the KDDTest dataset.

Step 7: Statistical Evaluation

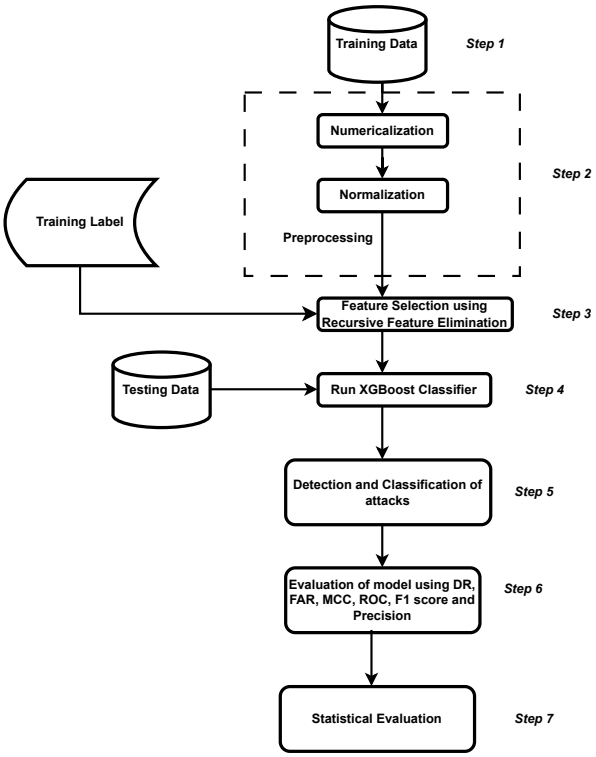

Figure 7: Flow chart of proposed model approach where the rectangles represent the model's processes, cylinders represent stored data, the trapezoid represents the stored training labels (Normal, DoS, Probe, R2L, and U2L), and the arrows the direction of flow.

Dataset Description (step 1)

The NSL-KDD dataset is an improved version of the KDD99 dataset, in which a large amount of data redundancy has been removed [109]. This dataset has the same attributes as the KDD99 having 41 features that are labeled into either normal or one of four attack categories such as DoS, Probe, U2R, and R2. The NSL-KDD dataset repository has two files; KDDTrain.txt and KDDTest.txt. Training is performed on the KDDTrain data, which has 23 attack types, and testing is performed on KDDTest data with 38 (15 additional) attack types, all grouped into the four attack categories. Table 12 shows the attack types, attack categories (classes), and number

of data points per category in the NSL-KDD train and test datasets. The NSL-KDD dataset has 125973 data points in the training dataset and 22544 in the testing dataset.

Our proposed approach is also capable of detecting the 15 unknown attacks in the test dataset.

Table 6: The attack category(class), the number of records in the NSL-KDD training and testing datasets, and a subset of examples for each attack category (attack types)

Attack Class	**No. of records**		**Attack Types**
	Training	**Testing**	
Normal	67, 343	9,711	Normal traffic data
DoS	45,927	7, 460	Worm, Land, Smurf, Udpstorm, Teardrop, Pod, Mailbomb, Neptune, Process table, Apache2, Back
Probe	11, 656	2, 421	Ipsweep, Nmap, Satan Portsweep, Mscan, Saint
R2L	995	2, 885	WarezClient, Worm, SnmpGetAttack, WarezMaster, Imap, SnmpGuess, Named, MultiHop, Phf, SPy, Sendmail, Ftp_Write, Xsnoop, Xlock, Guess_Password
U2R	52	67	Buffer_Overflow, SQLattack, Rootkit, Perl, Xterm, LoadModule, Ps, Httptuneel

Data Preprocessing (step 2)

The primary goal of data preprocessing is to improve the quality of the data, making it easier to work with and enabling more accurate and efficient results for machine learning algorithms [81].

Numericalization Numericalization converts non-numeric data, such as text or categorical variables, into a numerical format that can be used as input for machine learning algorithms, statistical analysis, or other computational methods. Machine learning algorithms

and many statistical methods primarily work with numerical data, so it is essential to represent non-numeric information in a format that these algorithms can understand and process. It is important to note that although the labels of a category are now represented by numbers, the data in it of itself remains categorical and the numbers do not represent ordinal, interval or ratio scales. The NSL-KDD dataset has 41 features, where each feature represents an attack type as described in Section 5 and attack category (class feature). These features are both numeric (38 features) and categorical (3) features. The categorical features are protocol_type (3 types), service (70 types), and flag (11 types) that need to be converted to numeric features. Label encoding is then used to assign each unique type in a categorical variable a distinct integer value. After applying label encoding, one-hot encoding is used in creating binary (0 or 1) features for each unique type in a categorical variable. The attack category (class) feature is also labeled with a numeric type, starting with Normal labeled as 0, DoS labeled as 1, Probe labeled as 2, R2L labeled as 3, and U2R labeled as 4. We further converted the different attack categories to bit form and used 10000 for Normal, 01000 for DoS, 00100 for Probe, 00010 for R2L, and 00001 for R2L. After numericalization, we are left with 122 features and one categorical class feature.

Normalization (step 2) Normalization is a technique used to scale numerical features in a dataset to a standard range, typically [0, 1] or [-1, 1]. Normalization aims to bring different features onto a similar scale, making it easier for machine learning algorithms to process the data and reducing the risk of certain features dominating the model due to their larger numerical values. There are several methods for normalization, two of the most common being Min-Max Scaling and Z-score Standardization. In our proposed approach, the Z-score standardization, which scales the value of the entire feature, is used. In Z-score standardization, the average μ of each numerical feature is calculated and then subtracted from the feature value x. The subtracted average and feature x are then divided by the standard deviation σ as shown

in the formula:

$$X_i = \frac{x - \mu}{\sigma}$$

Feature Selection (step 3)

This is a machine learning process that involves selecting a subset of the most relevant and informative features from a dataset's original set of features. The goal of feature selection is to improve machine learning model performance, interpretability, and generalization of machine learning models by reducing noise, overfitting, and computational complexity. There are three main categories of feature selection techniques; Filter methods, Wrapper methods, and Embedded methods [97]. After numericalization, the NSL-KDD has 122 features, which are not all required for the network intrusion detection system. Essential features are selected using a wrapper feature selection method called recursive feature elimination, which considers the interaction between features and their contribution to the specific model being used. Using recursive feature elimination, the most relevant features in the dataset are identified iteratively, training the classifier model and eliminating the least important features.

After applying recursive feature elimination, 45 out of 122 features are selected for each of the four attack categories. It is important to note that a feature can be selected in more than one attack category.

Extreme Gradient Boosting (XGBoost) (step 4)

Extreme Gradient Boosting [30], or XGBoost, is a scalable, high-performance implementation of the gradient boosting algorithm, a popular ensemble learning method. Gradient boosting combines weak learners, typically decision trees, iteratively to create a robust model that minimizes prediction error. XGBoost has several advantages, including speed, parallelization, regularization, handling of missing data, customizable loss functions, ability to handle

imbalance classes in the data and built-in cross-validation. The selected features for the different attacks are then applied to the XGBoost classifier. During this process, the XGBoost classifier learns to distinguish between normal traffic and various types of attacks based on the features extracted from the NSL-KDD train and because of the above mentioned advantages, XGBoost with RFE can improve the detection time, MCC and FAR.

Detection and Classification (step 5)

Detection and classification of attacks in network intrusion detection systems involve identifying the attack categories on the NSL-KDD Test using the trained XGBoost model.

Model Evaluation (step 6)

Once the XGBoost classifier is trained using the NSL-KDDTrain dataset, it is evaluated using NSL-KDDTest dataset. The performance of the XGBoost classifier is evaluated using detection rate (DR), false alarm rate (FAR), Matthews correlation coefficient (MCC), F_1 score, ROC AUC, and precision metrics. These performance metrics are calculated using True Positive (TP), False Negative (FN), False Positive (FP) and True Negative (TN) as shown in the confusion matrix [18] in Table 13.

- **Detection Rate (DR):** It is the ratio between the total number of attacks detected by the NIDS to the total number attacks present in the dataset [41] which can be calculated using the formula:

$$DR = \frac{TP}{TP + FN}$$

- **Precision:** This measures the fraction of examples predicted as attacks that turned out to be attacks which can be calculated using the formula:

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **F_1 Score:** It is the harmonic mean of the fraction of examples predicted as attacks that turned out to be attacks (precision). It can also be described as the ratio between the total number of attacks detected by the NIDS to the total number of attacks present in the dataset (detection rate) which can be calculated using the formula:

$$\text{F}_1 \text{ Score} = \frac{2 * TP}{2 * TP + FN + FP}$$

- **False Alarm Rate (FAR):** It is the fraction of non attacks that are misclassified as attacks which can be calculated using the formula:

$$FAR = \frac{FP}{FP + TN}$$

- **Matthews correlation coefficient (MCC):** It is defined as the Pearson product-moment correlation coefficient using the confusion matrix in Table 13 between the actual attacks and predicted attacks [32] which can be calculated using the formula below. MCC ranges between $[-1, +1]$, where -1 corresponds to the worst overall system performance and 1 corresponds to the best overall system performance. A high MCC score indicates that the binary classifier was able to correctly predict the majority of the attacks and the majority of non-attacks.

$$MCC = \frac{TN * TP - FN * FP}{\sqrt{(FP + TP)(FN + TP)(TN + FP)(TN + FN)}}$$

- **Receiver Operating Characteristic (ROC) Curve:** The Receiver Operating Characteristic (ROC) curve is a graphical representation used to evaluate the performance of binary

classification models in machine learning. It is created by plotting the ratio between the total number of attacks detected by the NIDS to the total number of attacks present in the dataset (detection rate) against the fraction of non-attacks that are misclassified as attacks (False Alarm Rate) at various classification threshold levels. The area under the curve (AUC) of the ROC quantifies the overall performance of the classification model. AUC values range from 0 to 1, with a value of 0.5 representing a random classifier and a value of 1 indicating a perfect classifier. A higher AUC value suggests a better-performing classification model.

A good NIDS should have high detection rates, precision, MCC, ROC AUC, F_1 score but low FAR.

Table 7: Confusion Matrix: A contingency containing four metrics, True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN).

Attack		**Predicted Class**	
		Yes	**No**
Actual class	**Yes**	**TP**	**FN**
	No	**FP**	**TN**

Statistical Evaluation (step 7)

We use the two-sample t-test to determine the statistical significance of our proposed approach. The two-sample t-test, also known as the independent samples t-test or unpaired t-test, is a statistical hypothesis test used to compare the means of two independent groups to determine if there is a significant difference between them. The test assumes that the data is normally distributed and the variances of the two groups are equal (although modifications are available if this assumption does not hold). To compare the performance of our proposed approach against the Random Forest and Decision Tree approaches, we used a two-sample t-test to test whether there is a significant difference between the mean performances of DR,

FAR, MCC, F_1 score, ROC AUC, and precision. The null hypothesis (H_0) for the two-sample t-test states that there is no significant difference between the mean performances of the two models. In contrast, the alternative hypothesis (H_1) states a significant difference, unlikely to have occurred by chance, between the mean performances of the two models.

Experimental Results

All experiments were performed in Python using default parameters for XGBoost library, Random Forest (Sklearn) and Decision Trees (Sklearn) using Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz 2.30GHz processor with 16gb RAM. Training and testing of XGBoost, Random Forest and Decision Tree algorithm were performed on 122 features and the XGBoost model on the 45 selected features from the NSL-KDD dataset. Ten StratifiedKFold cross-validation was adopted during the training and testing of the classifiers to check if the classifiers were not overfitting on both the training and testing. However, the evaluation of the proposed approach was done only on the test dataset by finding the average and standard deviation of the DR, F_1 score, ROC AUC, precision, FAR, and MCC metrics. The performance of our proposed approach is shown in last row of Table 9. It is evident from Table 9 that our proposed approach have high MCC, precision, ROC AUC, and low FAR.

To compare the differences of our proposed approach (XGBoost with RFE) against Random Forest and Decision Tree, we executed a two-sample t-test (checks for normality and equal variance assumptions were valid) [23]. Table 10 compares XGBoost with RFE) against the Random Forest and Decision Tree algorithms on various metrics and attack types with significant values depicted in sky blue. From Table 10, comparing XGBoost with RFE against the Random Forest, we can see that the p-values of the test for MCC (for the U2R attack type) and FAR (for the R2L attack type) are significant at $\alpha = 0.05$. Similarly, comparing XGBoost with RFE against the Decision Tree, we can see that the p-values of the test for MCC (U2R attack type), FAR (R2L attack type), ROCAUC (for the U2R attack type), and precision (U2R attack type) are

significant at $\alpha = 0.05$. Also, from Table 10 comparing XGBoost with RFE against the XGBoost, we can see that the p-value of the test for FAR (R2L attack type) is significant at $\alpha = 0.05$. We can statistically confirm that our proposed approach (XGBoost with RFE) outperforms Random Forest (on MCC and FAR) and Decision Tree (on MCC, FAR, ROC AUC, and precision).

The last row of Table 8 shows the approximate run time for the intrusion to be classified as DoS, Probe, R2L, and U2R. From this, it can be identified that the classification time displayed is linear, which proves our proposed approach's scalability. Overall, the proposed approach has a classification time of 0.009 milliseconds for Dos, 0.020 milliseconds for Probe, 0.024 milliseconds for R2L, and 0.655 milliseconds for U2R, which all the models in the related works fail to mention. It is worth noting that the run time of Random Forest and Decision Tree looks better than our proposed approach because there are inherently parallel algorithms compared to XGBoost, which is a sequential algorithm, and since all the run times in 8 are approximately linear, XGBoost with REF is significantly excellent at classifying Dos, Probe, R2L, and U2R.

Hence, from the results shown in our experiments, we can conclude that XGBoost with RFE for NIDS is valuable in cybersecurity.

Discussion of Results

In this paper, we investigated the effectiveness of using XGBoost with recursive feature elimination (RFE) for network intrusion detection. Our primary goal was to empirically determine whether combining XGBoost and RFE could yield better detection rates than other methods while reducing false positives and negatives. We employed a comprehensive methodology, analyzing a diverse network traffic dataset, and evaluated our model using several performance metrics, variability of the performance metrics (standard deviation), and classification time of the attack types. Our results indicate that the XGBoost model with RFE achieved a high level of performance in terms of precision, DR, F1 score, MCC, and ROC AUC. This performance indicates that the model can accurately detect network intrusions while maintaining a low

false alarms rate (FAR), classification time and low variability (standard deviation) for all the performance metrics. Our approach improved detection capabilities on selected attack types when compared to other models and benchmarks in related work. The RFE process identified several key features highly relevant to network intrusion detection. These features align with our expectations and prior research [83], [124] and [117], confirming the importance of specific traffic characteristics in detecting malicious activities. The combination of XGBoost and RFE not only improved the model's performance but also improves the detection speed and reduces the complexity of the model by eliminating redundant and irrelevant features as compared to related works that focus on network intrusion detection as shown in Table 11.

The practical implications of our findings are significant for the field of network intrusion detection. The improved detection rates and speed offered by our approach can help security practitioners identify and respond to cyber threats more effectively. Additionally, reducing false positives and negatives can minimize the operational overhead of manually investigating false alarms. Furthermore, our approach demonstrates potential scalability and adaptability for different network environments and evolving cybersecurity threats.

Despite these promising results, our study has some limitations and threats to validity:

1. The dataset does not fully capture the diversity of network traffic patterns and evolving attack techniques. Future work could benefit from more recent and diverse datasets to validate our approach. This is a threat to the external validity of our work.

2. The XGBoost algorithm, while highly effective, may be prone to overfitting and is often difficult to interpret. Alternative methods, such as partial dependency plots (PDPs), SHAP (SHapley Additive explanation), and LIME (Local Additive Interpretable model-agnostic explanations), could be investigated to address these issues.

Table 8: Approximate Classification Time on NSL-KDD Test.

Classifier	Attack type	Time (Seconds)
XGBoost (122 Features)	DoS	170.3
	Probe	109.9
	R2L	149.1
	U2R	85.3
Random Forest (122 Features)	DoS	57.1
	Probe	36.4
	R2L	44.1
	U2R	37.4
Decision Tree (122 Features)	DoS	8.4
	Probe	3.8
	R2L	6.0
	U2R	3.3
XGBoost (45 Features)	**DoS**	**68.1**
	Probe	**49.6**
	R2L	**68.8**
	U2R	**43.9**

Table 9: Performance Measure together with standard deviation (STD) on NSL-KDD Test.

Classifier	Attack type	DR ± STD	F_1 Score ± STD	ROC AUC ± STD	Precision ± STD	FAR ± STD	MCC ± STD
XGBoost (122 Features)	DoS	0.99745 ± 0.00423	0.99745 ± 0.00294	0.99997 ± 0.00007	0.99812 ± 0.00215	0.00144± 0.00002	0.99609±0.00008
	Probe	0.99463 ± 0.00617	0.99547 ± 0.00481	0.99988± 0.00021	0.99633 ± 0.00426	0.00123± 0.00002	0.99096±0.00002
	R2L	0.97792 ± 0.01218	0.97861 ± 0.01006	0.99914 ± 0.00079	0.97938 ± 0.01088	0.00916± 0.00002	0.95723±0.00001
	U2R	0.90217± 0.13172	0.92916± 0.09870	0.99931 ± 0.00150	0.96790 ± 0.10148	0.00041±0.00004	0.86540± 0.00004
Random Forest (122 Features)	DoS	0.99705 ± 0.00335	0.99785 ± 0.00206	0.99989 ± 0.00054	0.99866 ± 0.00240	0.00103± 0.00003	0.99621±0.00009
	Probe	0.99365 ± 0.00588	0.99495 ± 0.00441	0.99986± 0.00026	0.99628 ± 0.00419	0.00113± 0.00001	0.98992±0.00002
	R2L	0.97310 ± 0.01232	0.97419 ± 0.01065	0.99855 ± 0.00096	0.97532 ± 0.00974	0.01081± 0.00002	0.94841±0.00002
	U2R	0.88064± 0.13087	0.90988± 0.09733	0.99943 ± 0.00162	0.95453 ± 0.11931	0.00062±0.00004	0.82417± 0.00002
Decision Tree (122 Features)	DoS	0.99705 ± 0.00506	0.99585 ± 0.00368	0.99647 ± 0.00328	0.99466 ± 0.00505	0.00412± 0.00002	0.99266 ± 0.00002
	Probe	0.99257 ± 0.00617	0.99303± 0.00645	0.99257± 0.00617	0.99349 ± 0.00695	0.00247± 0.00002	0.98606± 0.00003
	R2L	0.96970± 0.01318	0.97052 ± 0.01466	0.97325 ± 0.01259	0.97139 ± 0.01720	0.01277± 0.00001	0.94101±0.00003
	U2R	0.90979± 0.09227	0.89307± 0.07719	0.90979 ± 0.09227	0.88295 ± 0.11079	0.00185±0.00001	0.78490± 0.00003
XGBoost with RFE (45 Features.)	DoS	0.99732 ± 0.00398	0.99725 ± 0.00243	0.99997 ± 0.00008	0.99719 ± 0.00279	0.00216± 0.00002	0.99514±0.00002
	Probe	0.99443± 0.00594	0.99535± 0.00493	0.99985± 0.00022	0.99628 ± 0.00496	0.00123±0.00003	0.99070±0.00003
	R2L	0.97606± 0.00899	0.97670 ± 0.00711	0.99893 ± 0.00130	0.97738 ± 0.00768	0.00009± 0.00007	0.95341±0.00002
	U2R	0.89633± 0.12839	0.93216± 0.09260	0.99875 ± 0.00358	0.98095 ± 0.07407	0.00021± 0.00003	0.87232±0.00003

Table 10: Results of two-sample t-test comparing our proposed approach (XGBoost with RFE) against Decision Tree, Random Forest, and XGBoost

Comparison	**Evaluation metric (Attack Type)**	**Test Statistic**	**Hypothesis** ($\alpha = 0.05$)	**p-value**
XGBoost with RFE vs. Decision Tree with significant values highlighted in table 9	**MCC (U2R)**	**18.93**	**Rejected**	**<0.001**
	FAR (R2L)	**134.65**	**Rejected**	**< 0.001**
	ROC AUC (U2R)	**28.42**	**Rejected**	**< 0.001**
	Precision (U2R)	**2.3**	**Rejected**	**0.031941**
XGBoost with RFE vs. Random Forest with significant values highlighted in table 9	**MCC (U2R)**	**3.72**	**Rejected**	**0.00154**
	FAR (R2L)	**37.99**	**Rejected**	**< 0.001**
XGBoost with RFE vs. XGBoost with significant values highlighted in table 9	**FAR (R2L)**	**393.98**	**Rejected**	**< 0.001**

Table 11: Comparison With Other Network Intrusion Detection Methods.

Method	Attack type	FAR	ROC AUC	MCC	Speed	Statistical Test	Complexity
Farnaaz et al.[41]	DoS	Low	Not Mentioned	High	Not Mentioned	Not Mentioned	Medium
	Probe	Low	Not Mentioned	High	Not Mentioned	Not Mentioned	Medium
	R2L	Low	Not Mentioned	High	Not Mentioned	Not Mentioned	Medium
	U2R	Low	Not Mentioned	High	Not Mentioned	Not Mentioned	Medium
Souza et al.[124]	DoS	Not Mentioned	High	Not Mentioned	Not Mentioned	Not Mentioned	Medium
	Probe	Not Mentioned	High	Not Mentioned	Not Mentioned	Not Mentioned	Medium
	R2L	Not Mentioned	High	Not Mentioned	Not Mentioned	Not Mentioned	Medium
	U2R	Not Mentioned	High	Not Mentioned	Not Mentioned	Not Mentioned	Medium
Louk et al.[74]	DoS	Not Mentioned	Not Mentioned	High	Not Mentioned	Included	Not Mentioned
	Probe	Not Mentioned	Not Mentioned	High	Not mentioned	Included	Not Mentioned
	R2L	Not Mentioned	Not Mentioned	High	Not Mentioned	Included	Not Mentioned
	U2R	Not Mentioned	Not Mentioned	High	Not Mentioned	Included	Not Mentioned
Proposed Approach	DoS	Very Low	High	High	Very High	Included	Medium
	Probe	Very Low	High	High	Very High	Included	Medium
	R2L	Very Low	High	High	Very High	Included	Medium
	U2R	Very Low	High	High	Very High	Included	Medium

Conclusion and Future work

This paper presents XGBoost with RFE to detect four types of attacks DoS, Probe, U2R, and R2L. We adopted ten stratified k-fold cross-validations. Our proposed approach is evaluated using the NSL-KDD dataset. We compared our proposed method with XGBoost, Random Forest, and Decision Tree using the following metrics: DR, F_1 score, ROC AUC, precision, FAR, and MCC. Because of the usage of feature selection, the computational cost decreases, and our experimental results indicate that our proposed approach increases the DR, F1 score, ROC AUC, precision, MCC, and decreases FAR, classification time, variability within all the performance metrics for all of attacks. We equally compared our proposed method against Random Forest and Decision Tree for selected attack types using a two-sample t-test, and found that our proposed approach (with fewer features) is promising. Moreover, this model is compared with related works that focus on network intrusion detection. The comparison results in Table 11 showed that XGBoost with RFE has better performance, and less classification time.

For future work, we will experiment with deep learning approaches like GANs and autoencoders since they are capable of handling data of higher dimensions and to address computationally expensive recursive feature elimination.

AN EMPIRICAL INTERNET PROTOCOL NETWORK INTRUSION DETECTION USING ISOLATION FOREST AND ONE-CLASS SUPPORT VECTOR MACHINES

Contribution of Authors and Co-Authors

Manuscript in Chapter 6

Author: Gerard Shu Fuhnwi

Contributions: Problem identification and proposing solution, running experiment, manuscript writing, creating tables and figures. Primary writer

Co-Author: Victoria Adedoyin and Janet O. Agbaje

Contributions: Contribution in manuscript editing/writing, provided feedback, guidance and advice.

Manuscript Information Page

Gerard Shu Fuhnwi, Victoria Adedoyin and Janet O. Agbaje

International Journal of Advanced Computer Science and Applications (IJACSA)

Status of Manuscript:
____ Prepared for submission to a peer-reviewed journal
____ Officially submitted to a peer-reviewed journal
____ Accepted by a peer-reviewed journal
__×__ Published in a peer-reviewed journal

IJACSA
January 2023
10.14569/IJACSA.2023.0140801

Abstract

With the increasing reliance on web-based applications and services, network intrusion detection has become a critical aspect of maintaining the security and integrity of computer networks. This study empirically investigates internet protocol network intrusion detection using two machine learning techniques: Isolation Forest (IF) and One-Class Support Vector Machines (OC-SVM), combined with ANOVA F-test feature selection. This paper presents an empirical study comparing the effectiveness of two machine learning algorithms, Isolation Forest (IF) and One-Class Support Vector Machines (OC-SVM), with ANOVA F-test feature selection in detecting network intrusions using web services. The study used the NSL-KDD dataset, encompassing hypertext transfer protocol (HTTP), simple mail transfer protocol (SMTP), and file transfer protocol (FTP) web services attacks and normal traffic patterns, to comprehensively evaluate the algorithms. The performance of the algorithms is evaluated based on several metrics, such as the F1-score, detection rate (recall), precision, false alarm rate (FAR), and Area Under the Receiver Operating Characteristic (AUCROC) curve. Additionally, the study investigates the impact of different hyper-parameters on the performance of both algorithms. Our empirical results demonstrate that while both IF and OC-SVM exhibit high efficacy in detecting network intrusion attacks using web services of type HTTP, SMTP, and FTP, the One-Class Support Vector Machines outperform the Isolation Forest in terms of F1-score (SMTP), detection rate(HTTP, SMTP, and FTP), AUCROC, and a consistent low false alarm rate (HTTP). We used the t-test to determine that OCSVM statistically outperforms IF on DR and FAR.

Introduction

Network Intrusion can be referred to as an unauthorized penetration of a computer in an establishment or an address in one's assigned domain [121]. The nature and types of network

intrusion have evolved over the years and become more rampant in recent years [29].

An intrusion can be passive or active. In passive intrusion, the penetration is gained stealthily and without detection, while in active intrusion, changes to network resources are affected. Intrusion can either come from an insider or an outsider. By insider, we mean an employee, customer, or business partner. Outsider means someone not connected to the organization. Network intrusions can occur in different ways. Some announce their presence by defacing the website, while others are malicious, with the goal of siphoning off data until it's discovered. Some redirect users who are unaware of their website through cracking passwords or mimicking your website [121]. Sometimes, intruders absorb network resources intended for other uses or users, which can lead to a denial of service [85]. These unauthorized penetrations on the digital network are imperil on many occasions the security of networks and their data [50].

Network security breaches are rapidly increasing and result in a significant amount of loss to organizations, and often leads to a loss of confidence in them from their unaware customers that have fallen victims. The IBM report shows that the average cost of a data breach has risen 12 percent over the past five years to 3.92 million dollars per incident on average [21]. This is more than the cost of a breach caused by a system glitch or human error.

Many researchers have carried out research and projects on network intrusion detection [54, 67, 113]. Wang and Battiti identified intrusions in computer networks with principal component analysis [118]. Liao and Vemuri used a k-nearest neighbour classifier for intrusion detection [71]. Gaffney and Ulvila evaluated intrusion detectors using a decision theory approach [115]. But this area still longs for more work as a result of the rapid rise in network intrusion. Therefore, we need to design an efficient algorithm that can successfully defend against network intrusions in an ever-evolving threat landscape. To achieve proactive security control, organizations must put in place a good network security infrastructure and leverage the potential of machine learning, which has the capability of automatically and continuously

detecting network intrusions. This will help block intruders and prevent them from achieving their goals. The remainder of this paper is organized as follows: In Section 2, we briefly review some related work in anomaly detection based Network Intrusion Detection. Section 3 gives a description of the algorithms used in this paper. Section 4 analyzes the empirical evaluation, where we review the data sets used, evaluation metrics description, results, and result discussion. Section 5 covers the conclusion.

Related Work

Liu and Ting [72] focused on using an Isolation Forest to detect anomalies that have many applications in the areas of fraud detection, network intrusion, medical and public health, industrial damage detection, and so on. The goal here is to build a tree-based structure that isolates anomalies rather than profiles anomalies like in the previous methods such as classification-based methods [1], and clustering-based methods [60]. Their proposed method, called Isolation Forest, builds a collection of individual tree structures that recursively partition a given data set, where anomalies are instances with a short path length on the trees. The anomaly score is used to determine instances that are anomalies, and has values between 1 and 0, with a score close to 1 being an anomaly and vice versa. The authors compared their results with other methods for anomaly detection techniques [45] like ORCA, LOF, and RF on real-world data sets with high dimensions and large data sizes using the metric AUC (Area Under the Curve) and run times. [87] proposed a hybrid of SVM and decision trees in classifying attacks of different forms of intrusion in knowledge discovery and data mining 1999 (KDDCUP99) data.

In [99], Sarumi et al. compared SVM and Apriori using Network Security Laboratory Knowledge Discovery and Data Mining (NSL-KDD) data and the University of South Wales NB 2015 (UNSW NB-15) dataset. From their results, they concluded that SVM outperformed Apriori in terms of accuracy, while Apriori showed a better performance in terms of speed.

In [41], Farnaaz and Jabbar proposed a detection intrusion system using random forest.

Experimental results were conducted on the NSL-KDD dataset. Empirical results show that the proposed model achieved a low false alarm rate and a high recall. Similarly, [124], [122], [7], and [57] applied machine learning techniques for network intrusion detection systems.

All the above mentioned papers discuss intrusion detection methods without any statistics to compare their results, attacks using web services, and no user guidance for using the proposed algorithms. To overcome this, one can look at the statistical significance of the various evaluation metrics based on the different machine learning algorithms proposed by them and also change the various parameters in the machine learning algorithms to observe their performance.

This paper aims at comparing the performance of One-class SVM and Isolation Forest machine learning algorithms in network intrusion using a two sample t-test and parameter alternation to provide some guidance on these algorithms usage to new researchers in this field.

METHODS

This section presents the intrusion detection approach used in this paper. These approaches include the ANOVA F-test, the Isolation Forest, the One-Class Support Vector Machines, and the two-sample t-test [44].

ANOVA F-test

The ANOVA F-test, or Analysis of Variance F-test, is a statistical technique used to compare the means of two or more groups to determine whether significant differences exist. It is commonly employed in feature selection or variable ranking tasks, where the goal is to identify the most relevant features or variables for a particular analysis or model.

Applying the ANOVA F-test to a dataset can rank features based on their F-statistic or p-value. Features with high F-statistic values or low p-values are considered more relevant, as they exhibit significant differences between the groups or classes. These relevant features

can then be selected for further analysis or modeling, while less informative features can be discarded to reduce dimensionality and improve computational efficiency. In the case of web network intrusion detection, the ANOVA F-test can be used to identify the most discriminative features that differentiate between normal network traffic and malicious intrusion attempts. By selecting the most significant features, it is possible to improve the performance and efficiency of intrusion detection systems by focusing on the most relevant information and reducing noise or irrelevant variables.

Isolation Forest

Isolation Forest and One-Class Support Vector Machine has been applied in different scenarios. Isolation Forest is an unsupervised learning algorithm for anomaly detection that works on the principle of isolating anomalies, instead of the most common technique of profiling normal points [123]. It is different from other distance and density based algorithms. The underlying assumption for this algorithm is that fewer instances of anomalies result in a smaller number of partitions (shorter path length) and the instances with distinguishable attribute values are more likely to be separated in early partitioning [12]. This implies that data points that have a shorter path length are likely to be anomalies. The necessary input parameters for building Isolation Forest algorithm are the subsampling size,the number of trees, and the height of the tree [12]. The subsampling size was suggested to be smaller for the machine learning algorithm to function faster and yield a better detection result [72]. We can use log to base 2 (number of data points) to get the depth of trees needed, but the path length converge before $t = 100$. [72].

Algorithm 1: $iForest(X,\ t,\ \psi)$

Inputs: X – input data, t – number of trees, ψ – subsampling size
Output: a set of t $iTrees$

1: **Initialize** $Forest$
2: set height limit l = ceiling($\log_2 \psi$)
3: **for** $i = 1$ to t **do**
4: $X' \leftarrow sample(X, \psi)$
5: $Forest \leftarrow Forest \cup iTrees(X', 0, l)$
6: **end for**
7: **return** $Forest$

Figure 8: Algorithm 1

One-Class Support Vector Machines

One-Class Support Vector Machines (OC-SVMs) [76] are a natural extension of SVMs. One-Class SVM is an unsupervised learning technique capable of differentiating test samples from a particular class from other classes. The One-Class SVM works on the basics of minimizing the hypersphere of one class in the training set and then considers every other class not within the hypersphere as anomalies or outliers. In order to identify suspicious observations, an OC-SVM estimates a distribution that encompasses most of the observations and then labels as "suspicious" those that lie far from it with respect to a suitable metric. This model uses different kernel functions or hyperspheres: linear, radial basis, sigmoid, and polynomial.

Two Sample t-test

The two-sample t-test, also known as the independent samples t-test or unpaired t-test, is a statistical hypothesis test used to compare the means of two independent groups to determine if there is a significant difference between them. The test assumes that the data is normally distributed and that the variances of the two groups are equal (although there are modifications

available if this assumption does not hold). In order to compare the performance of IF and OCSVM with the ANOVA F-test, we used a two-sample t-test to test whether there is a significant difference between the mean performances of DR, FAR,F_1 score, AUCROC, and precision. The null hypothesis (H_0) for the two-sample t-test states that there is no significant difference between the mean performances of the two models, while the alternative hypothesis (H_1) states that there is a significant difference, unlikely to have occurred by chance, between the mean performances of the two models.

EMPIRICAL EVALUATION

Data Description

The NSL−KDD dataset is an improved version of the KDD99 dataset, in which a large amount of data redundancy has been removed [109]. This dataset has the same attributes as the KDD99 having 41 features that are labeled as either normal or attacks using different web services (http, smtp, ftp, etc.). The NSL−KDD dataset repository has two files: KDDTrain.txt and KDDTest.txt. Table 12 shows the attack categories using different services and the number of data points per category in the NSL−KDD train and test datasets. The NSL−KDD dataset has 125973 data points in the training dataset and 22544 in the testing dataset.

Table 12: The attack types (class) using different internet protocols (http, smtp and ftp), the number of records in the NSL-KDD training and testing dataset.

Attacks using different Internet Protocol	**No. of records**		**Attack Types (class)**
	Training	**Testing**	
Normal	45,078	7,291	Normal traffic data
HTTP	2,289	1,180	Worm, Land, Smurf, Udpstorm, Teardrop, Pod, Mailbomb, Neptune, Process table, Apache2, Back
SMTP	284	316	Ipsweep, Nmap, Satan Portsweep, Mscan, Saint WarezClient, Worm,
FTP	648	48	SnmpGetAttack, WarezMaster, Imap, SnmpGuess, Named, MultiHop, Phf, SPy, Sendmail, Ftp_Write, Xsnoop, Xlock, Guess_Password

Data Preprocessing

The NSL-KDD dataset has 41 features, each representing an attack type described in Section 4.1 and an attack category (class feature). These features are both numeric (38 features) and categorical (3 features). The categorical features are protocol _type (3 types), service (70 types), and flag (11 types) that need to be converted to numeric features. We want to extract the most popular attacks caused by using different internet protocols (the service feature). These widespread attacks use web services (internet protocols) such as hypertext transfer protocol (HTTP), simple mail transfer protocol (SMTP), and file transfer protocol (FTP). After extracting the various internet protocols, the attack types (class) feature is labeled with a numeric type, starting with Normal, labeled as 0 and 1 for the different attack types.

Using ANOVA F-test feature elimination, the most relevant features with the highest F-statistic values in the dataset are identified, eliminating the least important features. These features are src bytes (number of data bytes transferred from source to destination in a single connection),

dst bytes (number of data bytes transferred from destination to source in a single connection), and duration.

Confusion Matrix

The performance of machine learning techniques can be evaluated using different parameters. These parameters are calculated using True Positive (TP), False Negative (FN), False Positive (FP), and True Negative (TN) as shown in the confusion matrix [18] in Table 13. The following parameters are used to evaluate our proposed approach.

Table 13: Confusion Matrix: A contingency containing four metrics, True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN).

Attack		**Predicted Class**	
		Yes	**No**
Actual class	**Yes**	**TP**	**FN**
	No	**FP**	**TN**

Detection Rate (DR) It is the ratio between the total number of attacks detected by the NIDS and the total number of attacks present in the dataset [41] which can be calculated using the formula:

$$DR = \frac{TP}{TP + FN}$$

Precision: This measures the fraction of examples predicted as attacks that turned out to be attacks, which can be calculated using the formula:

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

F_1 Score: It is the harmonic mean of the fraction of examples predicted as attacks that turned out to be attacks (precision). It can also be described as the ratio between the total number of attacks detected by the NIDS and the total number of attacks present in the dataset (the detection rate) which can be calculated using the formula:

$$F_1\ Score = \frac{2 * TP}{2 * TP + FN + FP}$$

False Alarm Rate (FAR): It is the fraction of non attacks that are misclassified as attacks, which can be calculated using the formula:

$$FAR = \frac{FP}{FP + TN}$$

Receiver Operating Characteristic (ROC) Curve: The Receiver Operating Characteristic (ROC) curve is a graphical representation used to evaluate the performance of binary classification models in machine learning. It is created by plotting the ratio between the total number of attacks detected by the NIDS to the total number of attacks present in the dataset (detection rate) against the fraction of non-attacks that are misclassified as attacks (False Alarm Rate) at various classification threshold levels. The area under the curve (AUC) of the ROC quantifies the overall performance of the classification model. AUC values range from 0 to 1, with a value of 0.5 representing a random classifier and a value of 1 indicating a perfect classifier. A higher AUC value suggests a better-performing classification model.

A good NIDS should have high detection rates, precision, AUCROC, F_1 score but low FAR. Most machine learning algorithms are evaluated using predictive accuracy, but this is not appropriate for network intrusion detection because it mostly involves imbalanced data. In terms of imbalanced data, we mean that the proportion of data points in each class is not approximately equal. The evaluation metrics adopted in this paper for evaluation and

comparison of our models are standard AUC (Area under curve). The area under the receiver operating curve gives an average measure of performance across all possible classification thresholds.

Experimental Results and Discussion

All experiments were performed in Python with alternating parameters for Isolation Forest (Sklearn) and One-class support vector machines (Sklearn) using the Intel(R) Core(TM) i7-10510U CPU at 1.80 GHz and 2.30 GHz processor with 16 GB of RAM. Training and testing of the Isolation Forest took four seconds, while it took 40 seconds to train the One-Class support vector machine model on the three selected features from the NSL−KDD dataset. The experimental results for One-class support vector machines and Isolation Forest on different performance metrics are shown in table 14 and table 15 respectively.

In table 14, the polynomial kernel outperformed the other kernels on HTTP and SMTP subsets with high DR, F_1 Score, AUCROC, Precision, and low FAR. On the other hand, in table 14, the sigmoid kernel performed much better than the different kernels on the FTP subset.

In table 15, isolation forest with 100 estimators on the HTTP subset achieved the highest DR, F_1 Score, AUCROC, Precision, and low FAR. The SMTP and FTP subset performs best on the evaluation metrics with 50 estimators. Generally, both Isolation Forest and One-class support vector machines didn't perform well on the FTP subset, having very high FAR and low DR, F_1 Score, AUCROC, and Precision. It is evident in tables 14 and 15 that the one-class support vector machines outperform Isolation Forest on all subsets, that is, HTTP, SMTP, and FTP having high DR, F_1 Score, and low FAR. Statistical analysis of overall performance on the one-class support vector machines and Isolation Forest results used a two-sample t-test with two-tailed probability to determine if each model's DR and FAR score on the test data yielded statistically significant differences ($p < 0.05$). In the HTTP, SMTP, and FTP, the one-class support machines had a significantly different DR,and FAR score ($p < 0.001$), which showed that our hypothesis

Table 14: One-Class SVM Performance Measure on NSL−KDD Test.

Attacks using different Internet Protocol	**Kernel**	**Gamma**	**DR**	**F_1 Score**	**AUCROC**	**Precision**	**FAR**
HTTP	Linear	0.00005	0.9802	0.9303	0.6308	0.8852	0.0198
	Sigmoid	0.00005	0.9969	0.9386	0.8867	0.6383	0.0031
	Polynomial	0.00005	0.9969	0.9385	0.6378	0.8866	0.0031
SMTP	Linear	0.00005	0.8398	0.7228	0.4468	0.6345	0.1602
	Sigmoid	0.00005	0.9806	0.7953	0.5156	0.6689	0.0194
	Polynomial	0.00005	0.9838	0.7969	0.5172	0.6696	0.0162
FTP	Linear	0.00005	0.3333	0.5000	0.6667	1.0000	0.6667
	Sigmoid	0.00005	0.5208	0.6848	0.7604	1.0000	0.4791
	Polynomial	0.00005	0.3333	0.5000	0.6667	1.0000	0.6667

was accepted.

Table 15: Isolation Forest Performance Measure on NSL−KDD Test.

Attacks using different Internet Protocol	**Estimators**	**maximum samples**	**DR**	**F_1 Score**	**AUCROC**	**Precision**	**FAR**
HTTP	50	256	0.9618	0.9563	0.8402	0.9508	0.0382
	100	256	0.9631	0.9570	0.8409	0.9509	0.0369
	200	256	0.9619	0.9563	0.8399	0.9507	0.0381
SMTP	50	256	0.9725	0.7846	0.6697	0.9725	0.0275
	100	256	0.9709	0.7838	0.6697	0.9709	0.0291
	200	256	0.9693	0.7835	0.6699	0.9693	0.0307
FTP	50	256	0.2917	0.3043	0.6225	0.3182	0.7083
	100	256	0.2083	0.2273	0.5809	0.2500	0.7916
	200	256	0.2708	0.2857	0.6121	0.3023	0.7292

CONCLUSION and Future work

The experiments performed on the NSL-KDD network intrusion data show that One-class support vector machines had the overall best performance in terms of DR and FAR scores over the Isolation Forest, with the best performance obtained by tuning the default parameters in both algorithms. Also, the number of estimators in Isolation Forest is comparable; using 100 and 50 estimators outperformed 200 estimators.

Therefore, One-class support is a good model for network intrusion detection by changing the default parameters in Sklearn. Also, polynomial or sigmoid kernel functions could be the best kernels to choose when using One-Class SVM on network intrusion data. Because of the usage of feature selection, the computational cost decreases (four seconds for Isolation Forest and forty seconds), and our experimental results indicate that our proposed approach increases the DR, F1 score, AUCROC, and precision and decreases FAR for three types of attacks. We equally compared one-class support vector machines and Isolation Forest selected attack types using a two-sample t-test and found that our proposed approach (with fewer features) is promising. For future work, we will experiment with deep learning approaches like GANs and autoencoders since they are capable of handling data of higher dimensions and also evaluate one-class support vector machines and Isolation forest using supervised learning methods like the random forest, Xgboost, and support vector machines.

USING LARGE LANGUAGE MODELS TO MITIGATE HUMAN-INDUCED BIAS IN SMS SPAM: AN EMPIRICAL APPROACH

Contribution of Authors and Co-Authors

Manuscript in Chapter 7

Author: Gerard Shu Fuhnwi

Contributions: Problem identification and proposing solution, conducting experiment, manuscript writing, creating tables and figures. Primary writer

Co-Author: Dr. Matthew Revelle, Dr. Bradley Whitaker and Dr. Clemente Izurieta

Contributions: Contribution in manuscript editing/writing, provided feedback, guidance and advice.

Manuscript Information Page

Gerard Shu Fuhnwi, Dr. Matthew Revelle, Dr. Bradley Whitaker and Dr. Clemente Izurieta

4th IEEE International Conference on AI in Cybersecurity (ICAIC)

Status of Manuscript:
____ Prepared for submission to a peer-reviewed journal
____ Officially submitted to a peer-reviewed journal
____ Accepted by a peer-reviewed journal
_×_ Published in a peer-reviewed journal

IEEE
29 January 2025
10.1109/ICAIC63015.2025.1084863

Abstract

Short Message Service (SMS) is a widely used text messaging feature on both basic and smartphones. SMS spam detection is a crucial task. Traditional machine learning approaches often struggle in this domain due to their reliance on manually crafted features, such as keyword detection, which can result in overly simplistic patterns and misclassification of more complex messages. With this shortcoming, these models can amplify human-induced biases if the training data contains inconsistent labeling or subjective interpretations, leading to unfair treatment of specific keywords or contexts. Conversely, advanced LLMs present effective approaches to addressing such issues, as they can more accurately capture linguistic patterns, contextual nuances, and textual ambiguities than traditional models, representing a substantial advancement in improving label accuracy.

This paper proposes utilizing LLMs to address human-induced labeling bias in spam detection and applying different prompt design methods to guide the process. In text classification, we surveyed two leading-edge LLMs, ChatGPT and Gemini, and evaluated them on the English SMS spam dataset source from UC Irvine's Machine Learning Repository. We explored the highest-performing prompt designs using approaches like in-context learning. The findings indicate that in-context techniques for prompting improve model effectiveness by reducing human-induced (contextual) labeling bias in SMS spam detection with a Balanced Accuracy of 82% - 97% and an Equal Opportunity Difference (EOD) of precisely zero, indicating LLMs' trustworthiness (fairness) in reducing this bias compared to traditional machine learning approaches. Our results also suggested that expanding the sample size can decrease LLMs' ability to reduce human-induced labeling bias in spam detection. In general, this study provides information on the strengths and limitations of LLMs and suggestions for methods to minimize human-induced labeling bias in spam detection and can help guide the selection of appropriate LLMs for this task.

Introduction

Short Message Service (SMS) is not just a mode of communication; it is a global phenomenon. Despite the rise of messaging platforms based on the internet, SMS remains one of the most ubiquitous means of communication. In 2023, 6.89 billion people had smartphones, expected to grow to 7.86 billion by 2028, with 95% of text read and responded to within 3 minutes of receiving it. In the United States (U.S.) alone, 97% of adults own mobile phones (85% being smartphones), and the market is projected to reach a value of 12.6 billion dollars by 2025, with a compound annual growth rate of 20.3%[1]. The simplicity and availability of smartphones and essential mobile devices have made it a major target for spam messages, often leading to fraudulent activities, phishing, and identity theft [80]. A survey conducted in 2022 in the United States indicated that more than 225 billion people received spam texts, an increase 157% compared to 87.8 billion in 2021. An estimated $20 billion lost from scams[2]. It is also important to note that spam causes a burden on network capacity and data storage [90]. As SMS usage grows, so does the need for effective spam detection systems. Traditional spam detection models are heavily relying on machine learning techniques, which depend on labeled datasets for training. However, these data sets are often manually labeled, introducing human-induced bias leading to data imbalances that can compromise the performance and fairness of the model [79, 82].

Bias generally refers to systematic errors in data that can lead to discriminatory results or outcomes. Spam detection is critical in maintaining the security and integrity of SMS messages and economic costs within organisations [94]. Yet, their effectiveness is often downplayed by human-induced label bias in the datasets used for training [79]. This bias could manifest from subjective labeling, inconsistent interpretations, fatigue, lack of visibility into the system, and

[1] https://www.smscomparison.com/sms-statistics/
[2] https://www.robokiller.com/robokiller-2022-phone-scam-report

limited domain expertise. Biases, no matter their source, can lead to inconsistencies in the training data [69] that distort the model learned by a spam classifier.

LLMs have shown their ability to combat spam [56, 62, 95] and provide cyber defenses [59]. LLMs, built on extensive collections of text and code datasets, have proven to excel in natural language processing tasks such as translation, text generation, and text summarization[**?**]. The capacity of LLMs to learn intricate linguistic structures, combined with their understanding of how the meaning of words changes depending on the context, provides an opportunity to help mitigate human-induced data labeling bias in SMS spam datasets. LLMs have proven effective for text classification tasks, where assigning a predefined label to a given text is the objective[42]. Given the potential of large language models in text classification, the spam detection problem can be approached as a text classification task, where the LLM processes a short text message as input and outputs a label as spam or ham. With the massive amount of spam text received daily by phone users, telecommunication companies can use LLMs to detect spam text thereby reducing human-induced data labeling bias and economic costs of employing domain experts.

Several approaches exist for using machine learning, deep learning, and LLMs to tackle the challenge of spam detection[42, 49, 56, 62, 95]. However, most of the work [95] does not mention how LLMs can reduce human-induced data labeling bias in spam datasets. This motivates us to develop an LLM-based SMS spam detection system to help reduce human-induced data labeling bias and costs. Specifically, our research helps answer the following questions:

- **Research Question 1 (RQ1):** What are the most and least successful prompt designs for LLMs to help reduce human-induced labeling bias for SMS spam datasets?

- **Research Question 2 (RQ2):** How well do LLMs reduce human-induced labeling bias for SMS spam datasets compared to state-of-the-art machine learning models?

Using two prompt designs to investigate these research questions, we evaluated two LLMs (ChatGPT and Gemini). We designed several prompts (zero-shot, one-shot, two-shot and three-

shot) to guide the LLMs. We also investigated their empirical performance and capability of distinguishing between spam and ham SMS messages. Our results indicate that most models perform best with prompting. We also found that LLMs can differentiate between spam and ham messages with 100% Balanced Accuracy and low equal opportunity difference. As the number of samples increased, LLMs performed with 82% to 97% balanced accuracy, recall of 86% to 97%, precision of 53% to 82%, and fairness metrics, with an equal opportunity difference of 0 % to 3%. GPT-4 performed the best among the two LLMs and correctly classified 722 out of 747 spam texts with a balanced accuracy of 97% and equal opportunity difference of 0.00%. In addition to testing GPT-4 and Gemini, we tested traditional approaches like logistic regression [35] and XGBoost [30].

Our **key contributions** in this paper include the following [48]:

- Our work evaluated the capabilities of state-of-the-art LLMs in reducing human-induced data labeling bias in SMS spam datasets, achieving a balanced accuracy of 97%, recall of 97%, and an equal opportunity difference of precisely zero, as compared to conventional machine learning approaches with a balanced accuracy of 97%, recall of 81% and an equal difference of 36%.

- In-context learning with prompts is the most significant factor in reducing human-induced data labeling bias in SMS spam detection with LLMs. It enables LLMs to dynamically adapt to specific language tasks and contexts, effectively addressing biases inherent in human-labeled data.

- LLMs can detect 722 out of 747 spam texts (recall of 97%), showing their capability of reducing human-induced data labeling bias.

The remainder of this research paper is structured as follows: Section 7 covers the related work. Section 7 describes the experimental setup, including the dataset description, LLM fine-tuning, bias mitigation, implementation, model performance and fairness metrics. This section

also describes the comparative analysis done using two baseline methods. Section 7 discusses the results of the research questions (RQs). Section 7 discusses the results, and Section 7 concludes the paper.

Related Work

Human-induced bias in data labeling is a persistent challenge in machine learning, particularly for tasks involving subjective interpretations, such as spam detection. When annotators are responsible for labeling large datasets, their perspectives often introduce inconsistencies that significantly hinder the performance of models trained on such data. This variability can be a significant obstacle in tasks like SMS spam detection. Studies, such as those by Fort et al. [43], have emphasized how human labelers" personal biases influence their labeling decisions. This issue is particularly evident in SMS spam datasets, where the line between legitimate and spam messages is often subjective.

Researchers have increasingly turned to large language models (LLMs) like ChatGPT, and Gemini to mitigate human-induced bias [62]. These models have effectively automated the labeling process or assisted human annotators, thereby reducing the subjective nature of manual labeling. Tida and Hsu introduced a BERT model [114], one of the first models that significantly improved downstream task performance with minimal fine-tuning. It has been widely used in NLP tasks, including spam detection, where it helps standardize label generation.

More recently, NLP models have gained attention for their ability to mitigate human bias during data labeling. The model's few-shot learning capabilities, as demonstrated by Brown [25], allow it to generate labels for large datasets with minimal human supervision. The capability of natural language processing models to produce contextually accurate labels has been used in multiple scenarios to reduce human biases, especially in subjective tasks like content moderation and spam detection. Similarly, Hur et al. [62], also using LLMs for SMS

spam detection, achieved an average F1 score of 0.97 and 0.79 for Korean and English datasets, respectively. By integrating their approach into the labeling pipeline, researchers minimized the biases introduced by individual annotators, leading to more consistent dataset annotations.

Beyond fully automated labeling, some NLP tasks have been employed in semi-automated approaches to mitigate labeling bias further. These systems allow human annotators to collaborate with the model, improving label accuracy while reducing individual annotator subjectivity. Schick and Schütze [101] explored prompt-based learning, where LLMs guide human annotators by providing more neutral or standardized label suggestions. This hybrid approach leverages the contextual understanding of large language models while retaining human oversight, crucial for subjective tasks like spam classification.

In addition to automated and semi-automated approaches, active learning frameworks have also been employed to address labeling bias. These frameworks iteratively refine labeled data by combining machine-learning models with human input [103]. In such systems, LLMs like GPT can suggest corrections or flag potential bias in labeled datasets, allowing human annotators to focus on edge cases or ambiguous examples. This collaborative approach helps to improve the overall labeling process and mitigate the biases introduced by individual human annotators.

These studies demonstrate the effectiveness of Natural Language Processing and BERT models for text classification. Our study builds on these papers to extend the works of [25, 62, 101, 103, 114] by using LLMs like to reduce human-induced labeling bias in SMS Spam detection, achieving high Balanced Accuracy. Our proposed study also has a Equal Opportunity Difference (EOD) of zero, indicating how fair, trustworthy, and transparent our study is.

Experimental Setup

We propose a novel approach that uses LLMs to reduce human-induced labeling bias in SMS spam detection. The step-by-step process of our innovative approach is described below:

SMS Spam Data

The SMS Spam collection dataset is provided by UCI Machine Learning Repository [11]. This dataset contains 5572 rows with two columns, with the first column specifying the message category, whether it's "spam" or "ham." Spam here means an unsolicited message, while ham means a standard message. The second column contains the messages. Of the 5572 rows, 4825 are ham messages (86.6%) and 747 spam messages (13.4%).

LLM Fine-Tuning

In this step, we fine-tuned two LLMs: ChatGPT-4 [39] and Gemini-1.0-Pro [110]. We selected these LLMs because they are well known for their ability in text classification [101, 106]. For both LLMs, we used a context length of 128k. Fine-tuning allows the models to adapt to the peculiarities of the SMS data and better understand the nuances of spam vs. non-spam (ham) messages.

Bias Mitigation

The next step involves applying prompt (learning) designs for bias mitigation strategies within the LLM. These prompt designs instruct the model to perform spam detection with minimal bias. We incorporated two prompting methods, namely basic (zero-shot) prompting and in-context (n-shot) prompting, as proposed in existing research [73, 106].

- **Basic (zero-shot):** This prompting design requires no demonstration and provides only a task description. It is the most convenient, robust, and challenging to understand, as no specific example is given. Sometimes, the Basic prompt design setting is identical to the human task.

- **In-context (n-shot):** This involves providing the model with examples of inputs and corresponding responses to facilitate few-shot learning[25, 106]. These examples help guide

the model in generating responses that match the format of the provided examples. The choice of these examples-based learning can significantly affect the model's performance.

Implementation

In this phase, the fine-tuned models are implemented using Python. We utilized the APIs of GPT-4 and Gemini-1.0-Pro hosted by OpenAI[3] and Google AI studio[4]. We selected the text generation parameters for ChatGPT-4 and Gemini-1.0-Pro: Top-p = 1.0, temperature = 0.0, and max. tokens generated = 256. We choose the model's hyper-parameters to ensure deterministic responses, thereby enhancing the model's reliability. Our models typically respond with categorical responses, such as spam/ham, along with their respective probabilities.

Model Evaluation

We used three performance metrics: Recall, Precision, and Balanced Accuracy. Recall (R) is the ratio of correctly predicted positive instances (spam) to the total number of actual positive examples:

$$R = \frac{TP}{TP + FN}$$

Precision (P) is the fraction of positive examples correctly predicted to be in a positive (spam) class out of all predicted positive (spam) examples:

$$P = \frac{TP}{TP + FP}$$

Balanced Accuracy (ACC) [106] is the average accuracy calculated for both the positive (spam) and negative (ham) examples:

$$Balanced\,Accuracy = \frac{\frac{correct_{negative}}{examples_{negative}} + \frac{correct_{positive}}{examples_{positive}}}{2}$$

[3]https://platform.openai.com/docs/overview
[4]https://aistudio.google.com/app/prompts/new_chat

For the fairness metric, we used the statistical parity difference (SPD) [46] and the equal opportunity difference (EOD) [116]. Statistical parity difference measures the difference in the probability of a message being predicted as spam by using spam keywords like "free" and "text."

$$\text{SPD} = \left| P(\text{Spam} \mid \text{"free"}) - P(\text{Spam} \mid \text{"text"}) \right|$$

where,

$$P(\text{Spam} \mid \text{"free"}) = \left| \frac{TP_{free} + FP_{free}}{TP_{free} + FP_{free} + FN_{free} + TN_{free}} \right|$$

and

$$P(\text{Spam} \mid \text{"text"}) = \left| \frac{TP_{text} + FP_{text}}{TP_{text} + FP_{text} + FN_{text} + TN_{text}} \right|$$

Equal opportunity difference (EOD) measures the true positive rate difference for spam messages using spam keywords like "free" and "text."

$$\text{EOD} = |P(\text{True Positive} \mid \text{"free"}) - P(\text{True Positive} \mid \text{"}text\text{"})|$$

where,

$$P(\text{True Positive} \mid \text{"free"}) = \frac{TP_{\text{free}}}{TP_{\text{free}} + FN_{\text{free}}}$$

and

$$P(\text{True Positive} \mid \text{"text"}) = \frac{TP_{\text{text}}}{TP_{\text{text}} + FN_{\text{text}}}$$

SPD and EOD values close to zero means that the algorithm is equally likely to make a mistake for each spam keyword, indicating how fair the algorithms are in making decisions across the keywords in spam messages.

Comparative Analysis

In the last step of the proposed approach, we compare the fine-tuned LLMs with traditional machine-learning models such as Logistic regression and Extreme Gradient Boosting (XGBoost). These state-of-the-art models are commonly used in spam detection but may lack advanced language understanding of LLMs.

Results of RQs

RQ1: What are the most and least successful prompt designs for LLMs-based to help reduce human-induced labeling bias for SMS spam datasets?

In this study, we used a 'task context (natural language) description,' which provides a detailed context to distinguish whether a message is spam or ham, alongside their respective probabilities. This context description was tested on both prompt designs as described in section III B. These natural language descriptions for various prompt designs are as follows:

- **Basic (zero-shot) prompting:** In this prompt design, we used the following context: ***Classify the following SMS message as ham or spam. Also return the probability of it being spam. The output should only contain two words: ham or spam, and the probability with two decimal points***.

- For **in-context prompting**, we explored three different configurations:

  - **One-shot:** An example of a ham or spam message is given in this context: ***Here is one example:***
    Example 1: Message: 'Spam example from the dataset'
    output: spam 0.98
    Now, classify the following message: Message: '{message}' Return the classification

(ham or spam) and the probability, rounded to two decimal points.

– **Two-shot:** Two examples of ham or spam messages are given in this context: ***Here are two examples:***

Example 1:

Message: 'Spam example from the dataset'

output: spam 0.98

Example 2:

Message: 'Ham example from the dataset'

output: ham 0.01

Now, classify the following message: Message: '{message}' Return the classification (ham or spam) and the probability, rounded to two decimal points.

– **Three-shot:** Three examples of ham or spam messages are given in this context: ***Here are three examples:***

Example 1:

Message: 'Spam example from the dataset'

output: spam 0.98

Example 2:

Message: 'Ham example from the dataset'

output: ham 0.01

Example 3:

Message: 'Spam example from the dataset'

output: spam 0.99

Now, classify the following message: Message: '{message}' Return the classification (ham or spam) and the probability, rounded to two decimal points.

Table 16 shows the performance of all prompting methods using the mean class (balance) accuracy metric. The findings indicate that in-context prompting methods deliver the best performance, with two-shot and three-shot achieving comparable results.

Table 16: Evaluation of various prompting techniques across models.

Model	**Basic**	**In-Context**		
		One-Shot	**Two-Shot**	**Three-Shot**
	Balanced Accuracy (ACC)			
ChatGPT-4	0.93	**0.96**	**0.97**	**0.97**
Gemini-1.0-Pro	0.82	**0.85**	**0.88**	**0.87**

Summary for RQ1: In-context prompts performed better for both ChatGPT and Gemini in reducing human-induced labeling bias for SMS spam datasets.

RQ2: How well do LLMs reduce human-induced labeling bias for SMS spam datasets compared to state-of-the-art machine learning models?

We used each model's best prompting method (ChatGPT-4 with two-shot and Gemini with two-shot) to compare the LLMs' capabilities for reducing human-induced labeling bias. ChatGPT-4 achieved a Balanced Accuracy of 97%, recall of 97%, and precision of 82%, showcasing its ability to classify spam and ham SMS messages accurately while maintaining fairness with an equal opportunity Difference (EOD) of 0, as shown in Table 19. Traditional models, with higher precision but comparable accuracy, are better suited for scenarios prioritizing fewer false positives. High precision but low recall in these models allow more spam to bypass filters. In contrast, ChatGPT -4's high recall and lower precision reflect an aggressive filtering approach, minimizing spam at the cost of some ham misclassification. Figure 9 highlights ChatGPT -4's confident and decisive classifications, with ham near 0 and spam near 1. Figure 9 further

illustrates ChatGPT -4's decisive classifications, with ham predictions clustering near 0 and spam near 1, confirming its confident and low-variability decision-making.

Table 18 presents the performance of each model based on samples classified as spam or ham. Notably, ChatGPT-4 continues to stand out with a Balanced Accuracy, precision, and recall of 100%, and an equal opportunity difference of 0%.

Table 17: Precision, Recall, Balanced Accuracy (ACC), SPD (free, text), and EOD (free, text) for Models in best Prompting (Two-Shot) and State-of-the-art Methods

Model	**P**	**R**	**ACC**	**SPD**	**EOD**
ChatGPT-4	0.82	0.97	0.97	0.03	0.00
Gemini-1.0-Pro	0.53	0.86	0.88	0.09	0.02
Logistic regression	0.96	0.64	0.95	0.04	0.36
XGBoost	0.96	0.81	0.97	0.04	0.36

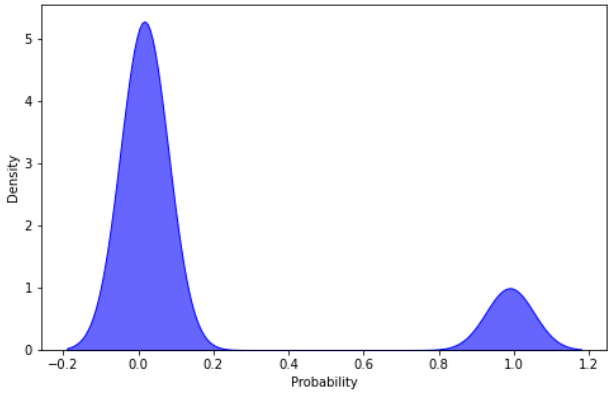

Figure 9: Probability Density Plot showing the distribution of Ham (0) and Spam (1) Messages for ChatGPT-4 with the best In-contexting prompting (two-shot)

Table 18: Precision, Recall, Balanced Accuracy (ACC), SPD (free, text), and EOD (free, text) for Models with the best In-context prompting (Two-shot) with Different Sample Sizes

Model	Sample Size	P	R	ACC	SPD	EOD
ChatGPT-4	10	1.00	1.00	1.00	0.00	0.00
ChatGPT-4	20	1.00	1.00	1.00	0.00	0.00
ChatGPT-4	30	1.00	1.00	1.00	0.00	0.00
ChatGPT-4	60	1.00	1.00	1.00	0.00	0.00
ChatGPT-4	100	0.80	1.00	0.98	0.03	0.00
Gemini-1.0-Pro	10	0.67	1.00	0.94	0.13	0.13
Gemini-1.0-Pro	20	0.57	1.00	0.91	0.19	0.19
Gemini-1.0-Pro	30	0.50	1.00	0.90	0.20	0.20
Gemini-1.0-Pro	60	0.47	0.89	0.86	0.07	0.07
Gemini-1.0-Pro	100	0.42	0.92	0.87	0.08	0.08

Summary for RQ2: ChatGPT outperformed Gemini, achieving a balanced accuracy of 97% and an EOD of 0.00, highlighted in yellow, compared to leading machine learning models.

Discussion

Our experimental findings emphasize the transformative potential of LLMs in mitigating human-induced labeling bias in SMS spam datasets. These LLMs demonstrate superior recall, statistical parity difference (SPD), and equal opportunity difference (EOD) compared to traditional methods, although with lower precision tradeoffs. This suggests a promising avenue for further exploration in prompt fine-tuning to enhance performance. By reducing reliance on human-labeled training data, our proposed approach offers a more efficient and adaptive solution for environments with rapidly evolving SMS spam techniques, outperforming conventional machine learning models.

The availability of English only SMS messages is a limitation to the generalisation of this work and poses an external threat to validity. Using spam messages in other languages was not considered. Secondly, the SMS spam dataset can be considered outdated, as it was made

available in 2012. Third, we only tested two models and two prompting methods, which does limit the conclusion validity of our results.

Conclusion and Future work

Our research evaluated the capabilities of LLMs using Basic and In-context prompting methods to reduce human-induced labeling bias. We found that LLMs are ineffective with the Basic (zero-shot) prompting method; results improved from one-shot to 2-shot. From two-shot to three-shot, the performance is decreased slightly. The study also reveals that in In-context prompting methods, LLMs achieve high Balanced Accuracy, recall, and precision, and low statistical parity, and equal opportunity differences. This demonstrates the potential of leveraging LLMs in an In-context setting for efficient and accurate SMS spam detection without previous state-of-the-art machine learning or deep learning, requiring large labeled datasets, which are costly in terms of labeling by humans and pre-processing. In-context prompting only requires a few examples of messages, and using this method, patterns of recent SMS spam messages can be quickly updated and filtered in digital communication channels. A key strength of our research is its ability to eliminate the need for manual data collection and labeling. The labels generated by LLMs are largely free from the bias typically introduced by human annotators.

For future work, one could expand the study to different languages and explore different LLMs and prompting techniques [106]. Secondly, one can explore the benefit of integrating LLM-based labeling into machine learning or deep learning pipelines. Thirdly, one research direction is to develop auto-optimization prompting techniques instead of relying on manual prompting.

REDUCING HUMAN-INDUCED LABEL BIAS IN SMS SPAM WITH CONTEXT-ENHANCED CLUSTERING (CEC)

Contribution of Authors and Co-Authors

Manuscript in Chapter 8

Author: Gerard Shu Fuhnwi

Contributions: Problem identification and proposing solution, conducting experiment, manuscript writing, creating tables. Primary writer

Co-Author: Dr. Ann Marie Reinhold and Dr. Clemente Izurieta

Contributions: Contribution in manuscript editing/writing, provided feedback, guidance and advice.

Manuscript Information Page

Gerard Shu Fuhnwi, Dr. Ann Marie Reinhold, and Dr. Clemente Izurieta

IEEE International Conference on Cyber Security and Resilience (CSR)

Status of Manuscript:
____ Prepared for submission to a peer-reviewed journal
____ Officially submitted to a peer-reviewed journal
_× Accepted by a peer-reviewed journal
____ Published in a peer-reviewed journal

IEEE
7th April 2025

Abstract

Short Message Service (SMS) is a widely used text messaging feature available on both basic and smartphones, making SMS spam detection a critical task. Supervised machine learning approaches often face challenges in this domain due to their dependence on manually crafted features, such as keyword detection, which can result in simplistic patterns and misclassification of more complex messages. Furthermore, these models can exacerbate human-induced bias if the training data include inconsistent labeling or subjective interpretations, leading to unfair treatment of specific keywords or contexts.

We propose a Context-Enhanced Clustering (CEC) approach to address these challenges by leveraging contextual metadata, adaptive thresholding, and modified similarity measures for clustering. We evaluate our approach using the English SMS spam dataset source from UC Irvine's Machine Learning Repository. CEC identifies representative samples from the SMS dataset to fine-tune LLMs such as ChatGPT-4, improving the robustness and fairness of spam classification. Our approach outperforms traditional clustering techniques such as K-means and DBSCAN in mitigating bias, as demonstrated through experiments measuring a balanced accuracy of 85% and a treatment equality difference (TED) of precisely zero. When used to identify representative samples to fine-tune ChatGPT-4, the CEC achieves a balanced accuracy of 98%, an equal opportunity of difference (EOD), and a treatment equality difference (TED) of zero. These results significantly reduce human-induced bias while maintaining high classification accuracy.

Introduction

Short Message Service (SMS) is not merely a mode of communication but a global phenomenon. Despite the increasing dominance of Internet messaging platforms, SMS is one of the most ubiquitous and reliable communication methods. In 2023, there were 6.89 billion

smartphone users worldwide, projected to increase to 7.86 billion by 2028. SMS remains highly effective, with 95% of text messages read and responded to within three minutes after receiving. In the United States alone, 97% of adults own mobile phones (85% of which are smartphones), and the SMS market is projected to reach a value of 12.6 billion dollars by 2025, experiencing a significant growth compound annual rate of 20.3% [1]. However,the simplicity and widespread availability of mobile devices have made SMS a significant target for spam messages, leading to fraud, phishing, and identity theft [80]. In 2022, a survey in the United States revealed that more than 225 billion spam texts were sent, a staggering 157% increase from 87.8 billion in 2021. These scams resulted in an estimated 20 billion dollars in financial losses[2].

Furthermore, spam significantly burdens network capacity and data storage, further compounding its negative impacts [90]. As SMS usage increases, the demand for effective and robust spam detection systems becomes increasingly critical. Addressing this challenge is essential to ensure user safety, maintaining network efficiency, and preserving SMS's trust and reliability as a communication platform. Existing spam detection methods often rely on manually labeled datasets, which introduce human biases arising from inconsistencies in the way messages are labeled, affecting classification fairness and accuracy [79, 82]. Although clustering techniques such as K-means [58] and DBSCAN [45] are widely used for pre-processing and semisupervised learning, they fail to address these biases due to static configurations and limited adaptability to contextual features.

Human-induced bias refers to systematic patterns of unfairness introduced into data due to human decisions, perspectives, or behaviors that can result in discriminatory outcomes. Spam detection plays a crucial role in ensuring the security and reliability of SMS communications while minimizing economic losses for organizations [94]. However, its effectiveness is often compromised by the human-induced label bias present in the training datasets [79]. This

[1] https://www.smscomparison.com/sms-statistics/
[2] https://www.robokiller.com/robokiller-2022-phone-scam-report

bias can arise from subjective labeling, inconsistent interpretations, annotator fatigue, limited domain expertise, or a lack of transparency in the labeling process. Such biases introduce inconsistencies in the training data [69], ultimately distorting the models developed for spam classification.

Several clustering techniques have been proposed for SMS spam detection, offering a combination of methods to enhance effectiveness [3, 13, 88, 92, 105]. However, in most existing work, incorporating contextual metadata, such as spam-related keywords and fairness, still needs to be addressed, limiting their applicability in real-world scenarios. This motivates the development of a fair, efficient, and context-aware clustering approach, such as Context-Enhanced Clustering for SMS spam detection, which incorporates adaptive techniques and contextual metadata to improve clustering quality and reduce human-induced labeling bias. Our research specifically addresses the following questions:

- **RQ1:** How effective is the context-driven clustering approach in reducing human-induced label bias compared to traditional clustering techniques such as DBSCAN and K-means in SMS spam detection?
- **RQ2:** Can a context-driven clustering technique effectively automate the selection of representative samples to generate prompts to fine-tune large-language models in SMS spam detection?

The main contributions of our approach are as follows:

- We introduce a novel context-driven clustering framework that incorporates contextual metadata, modified cosine similarity, and adaptive thresholding to improve clustering quality in SMS spam detection.

- We automate the selection of representative samples from clusters, enabling efficient and unbiased prompt generation for LLM fine-tuning and eliminating manual effort.
- We demonstrate significant reductions in human-induced label bias and improved fairness in SMS spam detection using metrics such as equal opportunity, statistical parity, and treatment equality differences while maintaining high classification performance.

The rest of this paper is organized as follows: Section II discusses related work. Section III introduces the Context-Enhanced SMS Clustering (CESC) framework, detailing the integration of contextual metadata, modified cosine similarity, adaptive thresholding, and the selection of representative samples for fine-tuning LLMs. Section IV describes model performance evaluation, fairness metrics, and results addressing the research questions (RQs). Section V provides an in-depth discussion of the empirical results. Section VI presents a threat to the validity of our approach. Finally, Section VII concludes the paper and suggests potential directions for future research.

Related Work

Human-induced bias in data labeling remains a persistent challenge in machine learning, especially for tasks that require subjective interpretations, such as spam detection. When annotators label large datasets, their perspectives can introduce inconsistencies that adversely affect the performance of models trained on these data. This variability poses a considerable challenge in tasks such as SMS spam detection. Research, including the work by Fort et al. [43], highlights how human labelers' backgrounds and personal biases can shape their labeling decisions. Although several research studies have been conducted about data labeling in the SMS spam detection domain using clustering algorithms such as K-means [58] and DBSCAN

[45], little research has been conducted to address human-induced biases due to the large number of short messages in SMS.

A contextual term is defined by its meaning and the surrounding words and sentences that decorate the word. Some terms such as "free" or "win" are more suspect in short SMS messages. Therefore, special clustering techniques are needed to reduce human-induced label bias in SMS text messages for spam detection, since existing clustering methods have limitations in adapting to contextual terms in short text data such as SMS.

Specifically, with respect to SMS detection, Nagwani and Sharaff [88] investigated a bi-level text classification and clustering approach employing K-means to improve SMS spam filtering and thread identification. Their approach demonstrated that integrating clustering with classification techniques can effectively enhance spam detection but has limited adaptability to varying message densities.

Anjali et al. [105] introduced optical character recognition for image data using unsupervised and deep semi-supervised learning for the detection of SMS scams. This study employs a combination of K-means, Non-Negative Matrix Factorization, and Gaussian Mixture Models along with feature extraction techniques such as TF-IDF and PCA, with K-means feature extraction and vectorizer achieving a superior accuracy.

Hind and Rachid [13] explored unsupervised and supervised learning techniques, a hybrid model combining K-means and Naive Bayes, Random Forests, Logistic regression, and a Support Vector Machine for SMS spam detection, with K-means-SVM having outstanding precision. Similarly, Darshit [92] also combines clustering with the Support Vector Machine for SMS spam detection.

A comparative analysis by Songfeng et al. [107] evaluated the performance of K-means and DBSCAN on synthetic datasets. The study provided valuable insights into the strengths and limitations of each algorithm, informing their application in SMS spam detection. In a similar research work by Ahmad and Shilpa [3], they analyze four clustering algorithms, namely K-

means, DBSCAN, HCA, and MDBCA, and compare their performance using different datasets.

These studies demonstrate the ability of clustering for text classification. Our study builds on these papers to extend the works of [13, 88, 92, 105, 107] by using context-enhanced clustering to reduce human-induced labeling bias in SMS spam detection. We expand on previous approaches by addressing the evolving nature of spam messages and contextual features that are present in short SMS texts.

Our Approach

This section describes the proposed Context-Enhanced Clustering (CEC) framework for mitigating human-induced labeling bias in SMS spam detection. CEC leverages contextual metadata, modified cosine similarity, and adaptive clustering to help create high-quality clusters of SMS messages as shown in Figure 10. Our CEC approach automatically selects representative samples from the generated clusters, ensuring a diverse and unbiased selection of messages. These representative samples are then used to create prompts for fine-tuning a Large Language Model (LLM), enhancing its ability to classify SMS spam more fairly and accurately while mitigating human-induced labeling bias.

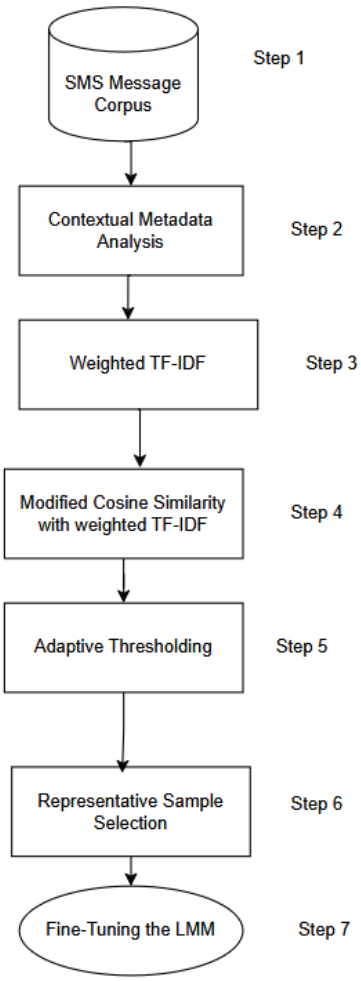

Figure 10: This is a flow chart of the proposed CEC approach. The rectangles represent the CEC processes, the cylinders represent stored data, the oval shape represents the final step, and the arrows indicate the flow direction.

SMS Spam Data (Step 1)

The SMS spam collection dataset from the UCI Machine Learning Repository comprises 5572 entries with two columns [10]. The first column specifies the message category, either spam (unsolicited message) or ham (regular message), while the second column contains the message text. Of the 5572 entries, 4825 are ham messages (86.6 %) and 747 spam messages (13.4

%).

Contextual Metadata Analysis (Step 2)

CEC begins by analyzing the contextual metadata of SMS messages to assign importance to terms that are more relevant to spam messages. Spam-related keywords ("free," "win," "offer") are identified based on domain knowledge and their frequency in spam-labeled messages. Each term t_{ij} in a message m_i is assigned a weight based on its relevance and frequency:

$$w_{ij} = \text{Relevance}(t_{ij}) \times \text{Frequency}(t_{ij}, M_{\text{spam}})$$

Where w_{ij} is the contextual weight for term t_{ij}, calculated based on term relevance and its frequency in spam, and M_{spam} denotes the subset of messages labeled as spam.

Weighted TF-IDF (Step 3)

A modified cosine similarity metric is applied to measure similarity between SMS messages, enhancing traditional similarity computation by assigning a greater weight to spam-indicative keywords. Consider two messages $A = [a_1, a_2, \ldots, a_n]$ and $B = [b_1, b_2, \ldots, b_n]$. Let $W = [w_1, w_2, \ldots, w_n]$ represent the vector of contextual weights. Then, the modified cosine similarity is defined as:

$$\text{Modified Cosine Similarity}_{AB} = \frac{\sum_{k=1}^{n} w_k \cdot a_k \cdot b_k}{\sqrt{\sum_{k=1}^{n} w_k \cdot a_k^2} \cdot \sqrt{\sum_{k=1}^{n} w_k \cdot b_k^2}}$$

Where w_k is the contextual weight of term k, and a_k, b_k are the weighted term frequencies for messages A and B, respectively.

Adaptive Thresholding (Step 5)

To allow clusters to adapt dynamically, thresholds for grouping messages must be determined. For each message m_i, the local density is calculated using the average similarity of its

k-nearest neighbors :

$$D_i = \frac{1}{k} \sum_{j \in \text{Neighbors}_k(i)} S_{ij}$$

where S_{ij} is the similarity between messages m_i and m_j. The thresholding for clustering is then calculated as:

$$\epsilon_i = \alpha \cdot D_i$$

The scaling factor α was tuned in the range $\alpha \in [0.3, 1.5]$, with optimal performance in $\alpha = 0.5$, where α adjusts the sensitivity to the density of local messages. This adaptive thresholding enables tighter grouping in high-density regions to prevent overgeneralization and looser thresholds in low-density areas to capture rare spam patterns effectively.

Adaptive Thresholding (Step 5)

To allow clusters to adapt dynamically, thresholds for grouping messages must be determined. For each message m_i, the local density is calculated using the average similarity of its k-nearest neighbors :

$$D_i = \frac{1}{k} \sum_{j \in \text{Neighbors}_k(i)} S_{ij}$$

where S_{ij} is the similarity between messages m_i and m_j. The thresholding for clustering is then computed as:

$$\epsilon_i = \alpha \cdot D_i$$

where α is a scaling factor that adjusts sensitivity to message density, this adaptive threshold ensures that clustering accounts for local variations in message density. Messages in high-density areas (many similar messages) require higher similarity to be grouped, preventing overgeneralization. In contrast, messages in low-density regions (rare or unique messages) are allowed looser similarity thresholds, ensuring rare spam patterns are captured.

Representative Sample Selection (Step 6)

When the adaptive clusters are formed based on local message density as described in step 5, representative samples are selected from these clusters to provide a balanced and unbiased subset for fine-tuning LLMs. These samples are selected based on the centrality score for a message x in the cluster C_i. Messages with the highest centrality score are selected as representative samples, ensuring they are most indicative of the cluster characteristics. This centrality score is calculated as the average similarity S_{xy} between x and all other messages y in the cluster C_i as follows:

$$R(x) = \frac{1}{|C_i|} \sum_{y \in C_i} S_{xy}$$

Fine-Tuning the LLM (Step 7)

The selected representative samples generate prompts for fine-tuning a pre-trained LLM, such as ChatGPT, for SMS spam detection. LLM fine-tuning trains the model to classify messages as spam or non-spam, using the selected samples as input. This process reduces human-induced label bias by relying on the contextually balanced generated prompts from the clusters, which helps improve both fairness and model performance in classification.

The pseudo-code of our approach is shown in Figure 11.

[3]https://github.com/gshufuhnwi/CEC-Approach

Algorithm 1 Context-Enhanced Clustering (CEC)

Require: SMS corpus M, spam-related keywords K, TF-IDF, LLM model

Preprocess messages and compute TF-IDF vectors
Assign contextual weights to terms in K
Compute weighted TF-IDF vectors
Calculate modified cosine similarity for all messages
Perform adaptive thresholding:
for each message m_i **do**
    Compute local density using k-nearest neighbors
    Calculate threshold $\epsilon_i = \alpha \times \text{mean\_density}$
end for
Perform adaptive clustering:
Initialize clusters as empty
for each pair of messages (m_i, m_j) **do**
    if $S_{ij} \geq \min(\epsilon_i, \epsilon_j)$ **then**
        Assign m_i and m_j to the same cluster
    end if
end for
for each cluster **do**
    Calculate centrality scores for messages
    Select representative samples with the highest scores
end for
Fine-tune LLM using representative samples as prompts
return Fine-tuned LLM model for spam detection

Figure 11: Pseudocode for Context-Enhanced Clustering (CEC) Approach.[3]

Results Evaluation

Here, we describe how we evaluated our approach and provide the results that answer our research questions.

Model Performance

The performance of our approach was evaluated using three performance metrics: precision (P), recall (R), and balanced accuracy (ACC). Precision (P) measures the proportion of correctly predicted positive (spam) examples relative to all examples classified as positive (spam):

$$P = \frac{TP}{TP + FP}$$

Recall is the proportion of positive instances (spam) correctly predicted out of the total number of actual positive examples:

$$R = \frac{TP}{TP + FN}$$

Balanced Accuracy (ACC) is the mean accuracy calculated across both the positive (spam) and negative (ham) classes [106]:

$$ACC = \frac{\frac{correct_{negative}}{examples_{negative}} + \frac{correct_{positive}}{examples_{positive}}}{2}$$

Fairness Metric

Fairness refers to the ability of a spam detection model to classify spam and ham messages equitably without introducing biases that disproportionately affect specific message patterns, keywords, or context. We evaluated fairness using three metrics: statistical parity difference (SPD), equal opportunity difference (EOD), and treatment equality difference (TED) [116]. Statistical parity difference quantifies the difference in the likelihood of a message being classified as spam based on the presence of keywords such as "free" and "win."

$$\text{SPD} = \left| P(\text{Spam} \mid \text{"free"}) - P(\text{Spam} \mid \text{"win"}) \right|$$

where,

$$P(\text{Spam} \mid \text{"free"}) = \left| \frac{TP_{free} + FP_{free}}{TP_{free} + FP_{free} + FN_{free} + TN_{free}} \right|$$

and

$$P(\text{Spam} \mid \text{"win"}) = \left| \frac{TP_{win} + FP_{win}}{TP_{win} + FP_{win} + FN_{win} + TN_{win}} \right|$$

Equal opportunity difference (EOD) calculates the difference in true positive rates for spam messages containing keywords such as "free" and "text."

$$\text{EOD} = |P(\text{True Positive} \mid \text{"free"}) - P(\text{True Positive} \mid \text{"}text\text{"})|$$

where,

$$P(\text{True Positive} \mid \text{"free"}) = \frac{TP_{\text{free}}}{TP_{\text{free}} + FN_{\text{free}}}$$

and

$$P(\text{True Positive} \mid \text{"win"}) = \frac{TP_{\text{win}}}{TP_{\text{win}} + FN_{\text{win}}}$$

Treatment equality difference (TED) is satisfied if messages containing spam keywords like "free" and "text" have an equal ratio of false negatives (FN) and false positives (FP).

$$\text{TED} = |\left(\frac{\text{FN}}{\text{FP}}\right)_{\text{free}} - \left(\frac{\text{FN}}{\text{FP}}\right)_{\text{win}}|$$

SPD, EOD, and TED values near zero indicate that the algorithm is equally likely to make errors for each spam keyword, reflecting the fairness of its decision-making across keywords in spam messages.

Addressing the Research Questions (RQs)

RQ1: *How effective is the context-driven clustering approach in reducing human-induced label bias compared to traditional clustering techniques such as DBSCAN and K-means in SMS spam detection?*

To answer this research question, we applied our approach described in Section 3, which uses contextual metadata, modified cosine similarity, and adaptive clustering techniques to generate high quality clusters of SMS messages to help reduce human-induced label bias.

Our approach demonstrated a balanced accuracy of 85%, recall of 68%, and precision of 100%, highlighting its effectiveness in accurately classifying spam and ham SMS messages. Furthermore, it maintained fairness with a treatment equality difference of 0, as shown in Table 19. For comparison, K-means and DBSCAN were configured with key parameters set as follows: The K-means were run with the number of clusters = 20, while DBSCAN used eps = 0.9 and minimum samples = 2. DBSCAN, with similar precision and low accuracy, is better suited for scenarios where minimizing false positives is a priority but might fail to deliver consistent results across all messages. In contrast, K-means with low precision and low accuracy are unsuitable for applications where reliable and accurate message classification is essential. DBSCAN, with high precision but low recall, allows more spam messages to evade detection and pass through the filters. In contrast, our approach with high precision and low recall demonstrates a conservative filtering strategy that accurately identifies legitimate messages while allowing some spam to bypass detection.

Table 19: Precision, Recall, Balanced Accuracy (ACC), SPD (free, win), EOD (free, win), and TED (free, win) for our approach (CEC), DBSCAN and K-Means

Model	**P**	**R**	**ACC**	**SPD**	**EOD**	**TED**
CEC	1.00	0.68	0.85	0.13	0.02	0.00
DBSCAN	0.99	0.59	0.79	0.14	0.07	0.02
K-Means	0.82	0.44	0.71	0.14	0.06	0.08

Summary for RQ1: As highlighted in yellow, CEC surpassed DBSCAN and K-Means, achieving a balanced accuracy of 85% and a TED score of 0.00.

RQ2: *Can a context-driven clustering technique effectively automate the selection of representative samples to generate prompts to fine-tune large-language models in SMS spam detection?*

Today, most applications rely on manual prompts to bridge the gap between human and LLM language to achieve the best performance. To address this problem in SMS spam detection, we used our CEC approach to help automate the selection of samples, which can be used as prompts for fine-tuning large-language models (LLMs). This eliminates the need for manual labeling and prompt generation, which is time-consuming and prone to human-induced bias. In this study, we used our CEC approach to select samples, which are used as input to fine-tune ChatGPT-4. CEC, together with ChatGPT-4, achieved a balanced accuracy of 98%, recall of 97%, and precision of 88%, showing the ability of ChatGPT-4 to learn from unbiased, high-quality examples from our CEC approach while maintaining fairness with an EOD and a TED of 0, as shown in Table 20.

Table 20: Precision, Recall, Balanced Accuracy (ACC), SPD (free, win), EOD (free, win), and TED (free, win) for ChatGPT-4 using CEC for prompt selection

Model	**P**	**R**	**ACC**	**SPD**	**EOD**	**TED**
ChatGPT-4	0.88	0.97	0.98	0.06	0.00	0.00

Discussion

Our experimental results highlight the significant potential of CEC to effectively mitigate human-induced label bias by incorporating context-aware clustering and adaptive thresholding in SMS spam datasets. Compared to traditional methods like DBSCAN and K-means, which have lower recall and balanced accuracy, CEC demonstrates superior balanced accuracy, recall, treatment equality difference (TED), and equal opportunity difference (EOD), and also ensures balanced representation of spam-related context, reducing over-reliance on specific keywords. Using CEC to select samples for fine-tuning LLMs such as ChatGPT-4 further enhances fairness and classification accuracy, addressing the limitations of existing clustering methods. Our approach minimizes the dependence on human-induced labeled training data, providing a

more adaptive and efficient solution for rapidly evolving SMS spam patterns while surpassing traditional machine learning models.

Threats to validity

While our Context-Enhanced Clustering (CEC) approach effectively mitigates human-induced label bias in SMS spam detection, certain factors may impact its generalizability and validity:

- The reliance on English-only SMS messages restricts the applicability of this approach to non-English languages. Spam characteristics, contextual meanings, and linguistic structures vary across different languages, potentially affecting model performance when applied to multilingual datasets.
- CEC relies on adaptive clustering with contextual weighting, which improves fairness but may be sensitive to hyperparameter selection (clustering thresholds, similarity metrics). Variability in parameter tuning could impact model outcomes, requiring further optimization strategies for different datasets and spam trends.
- SPD, EOD, and TE assess spam detection's fairness, minimizing classification disparities across different message types. However, these metrics alone may not capture all forms of subtle bias in spam detection.

Conclusion And Future Work

Our research presents a novel approach to mitigate human-induced label bias in SMS spam detection using a context-enhanced clustering (CEC) framework. The study demonstrates that CEC and using CEC to select samples for fine-tuning large language models such

as ChatGPT-4 achieved highly balanced accuracy, recall, and precision while maintaining low equal opportunity and treatment equality differences compared to the work of G.S. fuhnwi et al. [**?**]. This highlights the potential of CEC for efficient and accurate SMS spam detection, eliminating the need for traditional state-of-the-art machine learning or deep learning approaches that rely on large, labeled datasets, which are costly in terms of human labeling.

Future work will focus on extending the CEC approach to multilingual datasets, sentiment analysis, explore additional fairness evaluation frameworks and implementing real-time spam filtering capabilities.

THREATS TO VALIDITY

This dissertation undertakes a comprehensive investigation into mitigating bias in anomaly detection systems using advanced machine learning and statistical techniques. While the experimental designs across the different chapters were carefully constructed, certain limitations may affect the validity of the findings. Threats to validity are categorized into internal, external, construct, and statistical conclusion validity.

Internal validity refers to the degree to which causal conclusions drawn from the study are warranted. Across the experiments, particularly in supervised learning tasks, internal validity may be threatened by factors such as improper preprocessing (e.g., data leakage), confounding variables within datasets, or model overfitting. For example, the application of feature selection or transformation techniques prior to dataset partitioning can introduce information leakage, inflate performance metrics, and misrepresent the effectiveness of bias mitigation strategies. To address this, rigorous protocols were enforced to ensure that all preprocessing was conducted on training data only, with parameters then applied to test data consistently. A specific threat to internal validity in Chapter 4, which focuses on mitigating selection bias using feature selection within intrusion detection systems, arises from the risk of data leakage during the normalization phase. If normalization parameters (e.g., mean and standard deviation) are computed using the entire dataset before the train/test split, information from the test set can inadvertently influence the training process. This violates the principle of keeping the test data unseen during model development and may lead to overestimated performance outcomes. To safeguard against this, all normalization procedures should be applied after data splitting. This strategy ensured the preservation of experimental integrity and strengthened the internal validity by accurately reflecting the model's generalization capability.

External validity concerns the extent to which findings generalize beyond the studied datasets. While multiple public datasets were employed, ranging from malware traffic to

intrusion detection and SMS spam, the representativeness of these datasets for real-world environments is a limitation. For instance, the use of benchmark datasets such as NSL-KDD or SMS spam collections may not fully capture the diversity of traffic patterns, adversarial behavior, or labeling inconsistencies in operational systems.

A potential threat to **construct validity** arises from the inherent challenge of accurately defining and operationalizing the concept of bias in anomaly detection systems. Although quantitative fairness metrics such as Statistical Parity Difference (SPD), Equal Opportunity Difference (EOD), and Treatment Equality Difference (TED) are utilized to measure bias, these indicators may not fully capture the nuanced, domain-specific interpretations of fairness, particularly in cybersecurity and natural language processing contexts. The use of surrogate features or proxy variables to represent sensitive or latent attributes may also introduce assumptions that deviate from the intended theoretical constructs. As a result, the models may not be measuring bias in the way it is conceptually defined, thereby affecting the validity of the conclusions drawn about fairness and bias mitigation.

Finally, **statistical conclusion validity** could be threatened by factors such as small sample sizes for rare anomalies, model instability across different initializations, or multiple hypothesis testing without appropriate correction. These risks were mitigated through repeated cross-validation, significance testing (e.g., Wilcoxon Signed-Rank Test), and the use of robust evaluation metrics like MCC, ROC AUC, and Weighted F1 Score.

CONCLUSION AND FUTURE WORK

Conclusion

This dissertation advances the field of trustworthy and fair machine learning by developing a unified, bias-aware framework for anomaly detection systems. Motivated by the increasing integration of artificial intelligence in critical decision-making domains such as cybersecurity and spam filtering, this work systematically addresses multiple forms of bias, including representation, selection, and label bias, which can significantly compromise the fairness, robustness, accuracy, and reliability of anomaly detection models. The research unfolds across four major approaches, each targeting a distinct class of bias and corresponding to the core chapters of the dissertation. In Chapter 4, the first approach introduces a hybrid deep learning framework that combines autoencoders with logistic regression to mitigate representation bias in malware detection. By coupling the feature extraction strength of deep neural networks with the interpretability of classical models, this method enhances both fairness and explainability. The fairness of the model was assessed using statistical parity measures, and the findings were published in IEEE Access (2024). In Chapter 5, the second approach addresses selection bias in network intrusion detection through a pipeline that integrates Extreme Gradient Boosting (XGBoost) with Recursive Feature Elimination (RFE). The combined model improves predictive performance while minimizing false alarms, and a statistical hypothesis testing framework was used to benchmark its validity rigorously. This work was published in IEEE Access (2024). Chapter 6 presents the third approach, which investigates the use of unsupervised learning models, specifically Isolation Forests and One-Class SVMs, to overcome label bias in anomaly detection. These models are particularly effective in scenarios where ground truth labels are either unavailable or unreliable, as is often the case in intrusion detection tasks. The effectiveness and bias resilience of these models were empirically validated and published in IJACA (2023). Expanding the study into natural language processing, Chapter 7 and Chapter 7 introduce

two approaches to mitigating human-induced bias in SMS spam classification. First, large language models (LLMs) such as ChatGPT and Gemini were employed through prompt-based, in-context learning to reduce dependence on potentially biased training data. Second, a novel Context-Enhanced Clustering (CEC) framework was developed to group data more fairly by incorporating contextual metadata and a modified cosine similarity metric. These techniques collectively promote more equitable fine-tuning and inference in spam classification systems. Results from these chapters have been disseminated through two publications: one published at IEEE Access (2025) and another accepted at the 2025 IEEE International Conference on Cyber Security and Resilience (CSR), Greece. Together, these contributions establish a principled and multifaceted strategy for mitigating bias across different stages of anomaly detection, such as data preprocessing, model training, and evaluation. Notably, the research not only introduces novel algorithmic interventions but also grounds them in rigorous experimental design, fairness metrics, and empirical validation. By publishing these contributions across high-impact venues, this work extends the literature on algorithmic fairness and positions itself within the ongoing discourse on ethical and transparent AI. The broader significance of this dissertation lies in its interdisciplinary application of machine learning, statistical analysis, and natural language processing to advance the development of fair and trustworthy detection systems. These innovations hold practical implications for real-world deployment in cybersecurity, fraud detection, and automated moderation systems, where fairness, robustness, accuracy and reliability are paramount.

Future Work

This dissertation presents a comprehensive study on bias mitigation in anomaly detection systems using improved machine learning and statistical frameworks. Addresses human-induced measurement and representation biases in malware detection, network intrusion detection, and SMS spam classification. In the future, we intend to make additional contributions

to this evolving research space by extending the scope, methods, and practical applicability of the proposed approaches. First, we plan to incorporate model interpretability techniques such as SHAP (SHapley Additive Explanations) and LIME to better understand the decision-making processes of the hybrid autoencoder-logistic regression model for malware detection [46]. This will not only help identify sources of bias, but will also improve transparency and accountability. We also aim to extend this approach to other forms of obfuscated malware and evaluate its adaptability in real-time detection environments.

Second, building on our XGBoost-based intrusion detection approach [47], our goal is to explore deep learning techniques such as GANs and deep autoencoders. These models are more capable of capturing complex patterns in high-dimensional data and can reduce the computational burden introduced by Recursive Feature Elimination (RFE). In addition, we will explore the integration of fairness-aware objectives into the training loop of these models and assess their impact on bias metrics such as Statistical Parity Difference (SPD) and Equal Opportunity Difference (EOD).

Third, in addressing human-induced labeling bias [44], future work will involve a comparative evaluation of unsupervised models like Isolation Forest and One-Class SVM against supervised models, such as cost-sensitive SVMs and ensemble classifiers. This will help determine under which conditions each category of model is better suited for fair and robust intrusion detection. We also plan to extend these studies to datasets that include dynamic and evolving threat patterns, thus improving the generalization between attack types.

Fourth, to further investigate the role of large language models (LLMs) in reducing bias [48] in NLP tasks, we plan to (i) extend the current work to multilingual SMS datasets, (ii) evaluate different LLMs and in-context prompting strategies, and (iii) develop automated prompt optimization methods to eliminate manual intervention. We also aim to explore the integration of LLM-generated annotations into machine learning training pipelines to improve both scalability and fairness.

Lastly, we intend to expand the Context-Enriched Clustering (CEC) approach to other applications, such as sentiment analysis and real-time SMS spam filtering. We aim to test CEC's scalability under streaming conditions and evaluate additional fairness frameworks to support responsible deployment. In addition, we plan to optimize the adaptive thresholding mechanism within the CEC using data-driven techniques to improve its robustness across various distributions.

Through these future efforts, we aim to broaden the impact of this work and further strengthen the intersection between fairness, machine learning, and cybersecurity. Our ultimate goal is to enable the development of trustworthy anomaly detection systems that can operate reliably in sensitive, real-world environments where accuracy and fairness are both critical.

REFERENCES CITED

[1] Naoki Abe, Bianca Zadrozny, and John Langford. Outlier detection by active learning. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 504–509, 2006.

[2] Narasimha R Adiga, George Almási, George S Almasi, Yariv Aridor, Rajkishore Barik, D Beece, Ralph Bellofatto, Gyan Bhanot, Randy Bickford, M Blumrich, et al. An overview of the bluegene/l supercomputer. In *SC'02: Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*, pages 60–60. IEEE, 2002.

[3] HP Ahmad and Shilpa Dang. Performance evaluation of clustering algorithm using different dataset. *International Journal of Advance Research in Computer Science and Management Studies*, 8, 2015.

[4] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60:19–31, 2016.

[5] Saeed Ahmed, YoungDoo Lee, Seung-Ho Hyun, and Insoo Koo. Mitigating the impacts of covert cyber attacks in smart grids via reconstruction of measurement data utilizing deep denoising autoencoders. *Energies*, 12(16):3091, 2019.

[6] Mousa Alalhareth and Sung-Chul Hong. An improved mutual information feature selection technique for intrusion detection systems in the internet of medical things. *Sensors*, 23(10):4971, 2023.

[7] Shadi Aljawarneh, Monther Aldwairi, and Muneer Bani Yassein. Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. *Journal of Computational Science*, 25:152–160, 2018.

[8] Mouhammad Alkasassbeh and Mohammad Almseidin. Machine learning methods for network intrusion detection. *arXiv preprint arXiv:1809.02610*, 2018.

[9] Julia Allen and Howard Lipson. Technical leadership. *CERT RESEARCH ANNUAL REPORT*, page 72.

[10] Tiago Almeida and Jos Hidalgo. SMS Spam Collection. UCI Machine Learning Repository, 2011. DOI: https://doi.org/10.24432/C5CC84.

[11] Tiago A Almeida, José María G Hidalgo, and Akebo Yamakami. Contributions to the study of sms spam filtering: new collection and results. In *Proceedings of the 11th ACM symposium on Document engineering*, pages 259–262, 2011.

[12] Nari Sivanandam Arunraj, Robert Hable, Michael Fernandes, Karl Leidl, and Michael Heigl. Comparison of supervised, semi-supervised and unsupervised learning methods in network intrusion detection system (nids) application. *Anwendungen und Konzepte der Wirtschaftsinformatik*, (6):10–19, 2017.

[13] Hind Baaqeel and Rachid Zagrouba. Hybrid sms spam filtering system using machine learning techniques. In *2020 21st International Arab Conference on Information Technology (ACIT)*, pages 1–8. IEEE, 2020.

[14] Ricardo Baeza-Yates. Bias on the web. *Communications of the ACM*, 61(6):54–61, 2018.

[15] Mohammad Bagher Bahador, Mahdi Abadi, and Asghar Tajoddin. Hlmd: a signature-based approach to hardware-level behavioral malware detection and classification. *The Journal of Supercomputing*, 75:5551–5582, 2019.

[16] Solon Barocas and Andrew D Selbst. Big data's disparate impact. *Calif. L. Rev.*, 104:671, 2016.

[17] Victor R Basili. Software modeling and measurement: the Goal/Question/Metric paradigm. Technical report, 1992.

[18] E Beauxis-Aussalet and L Hardman. Ieee conference on visual analytics science and technology (vast)-poster proceedings. In *IEEE Conference on Vi-sual Analytics Science and Technology (VAST)-Poster Proceedings*, pages 1–2, 2014.

[19] Pooja Bhoria and Kanwal Garg. Determining feature set of dos attacks. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(5):875–878, 2013.

[20] Christopher M Bishop. Pattern recognition and machine learning. *Springer google schola*, 2:5–43, 2006.

[21] David Bisson. How to foil the 6 stages of a network intrusion,tripwire state of security news. *https://www.tripwire.com/state-of-security/security-data-protection/security-hardening/6-stages-of-network-intrusion-and-how-to-defend-against-them*, 2019.

[22] Martin Boldt and Bengt Carlsson. Analysing privacy-invasive software using computer forensic methods. *ICSEA, Papeetee*, 2006.

[23] C Allan Boneau. A comparison of the power of the u and t tests. *Psychological Review*, 69(3):246, 1962.

[24] Ahmet Selman Bozkir, Ersan Tahillioglu, Murat Aydos, and Ilker Kara. Catch them alive: A malware detection approach through memory forensics, manifold learning and computer vision. *Computers & Security*, 103:102166, 2021.

[25] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[26] Victor R Basili1 Gianluigi Caldiera and H Dieter Rombach. The goal question metric approach. *Encyclopedia of software engineering*, pages 528–532, 1994.

[27] Tristan Carrier. Detecting obfuscated malware using memory feature engineering. 2021.

[28] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[29] Thomas M Chen. Guarding against network intrusions. In *Computer and information security handbook*, pages 149–163. Elsevier, 2013.

[30] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[31] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, Rory Mitchell, Ignacio Cano, Tianyi Zhou, et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4, 2015.

[32] Davide Chicco and Giuseppe Jurman. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics*, 21:1–13, 2020.

[33] Hamid Darabian, Sajad Homayounoot, Ali Dehghantanha, Sattar Hashemi, Hadis Karimipour, Reza M Parizi, and Kim-Kwang Raymond Choo. Detecting cryptomining malware: a deep learning approach for static and dynamic analysis. *Journal of Grid Computing*, 18:293–303, 2020.

[34] Hamid Darabian, Sajad Homayounoot, Ali Dehghantanha, Sattar Hashemi, Hadis Karimipour, Reza M Parizi, and Kim-Kwang Raymond Choo. Detecting cryptomining malware: a deep learning approach for static and dynamic analysis. *Journal of Grid Computing*, 18:293–303, 2020.

[35] Bilge Kagan Dedeturk and Bahriye Akay. Spam filtering using a logistic regression model trained by an artificial bee colony algorithm. *Applied Soft Computing*, 91:106229, 2020.

[36] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

[37] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[38] Wilfred J Dixon. Analysis of extreme values. *The Annals of Mathematical Statistics*, 21(4):488–506, 1950.

[39] Adrian Egli. Chatgpt, gpt-4, and other large language models: the next revolution for clinical microbiology? *Clinical Infectious Diseases*, 77(9):1322–1328, 2023.

[40] Wisam Elmasry, Akhan Akbulut, and Abdul Halim Zaim. Evolving deep learning architectures for network intrusion detection using a double pso metaheuristic. *Computer Networks*, 168:107042, 2020.

[41] Nabila Farnaaz and MA Jabbar. Random forest modeling for network intrusion detection system. *Procedia Computer Science*, 89:213–217, 2016.

[42] John Fields, Kevin Chovanec, and Praveen Madiraju. A survey of text classification with transformers: How wide? how large? how long? how accurate? how expensive? how safe? *IEEE Access*, 12:6518–6531, 2024.

[43] Karën Fort, Gilles Adda, and Kevin Bretonnel Cohen. Amazon mechanical turk: Gold mine or coal mine? *Computational Linguistics*, 37(2):413–420, 2011.

[44] Gerard Shu Fuhnwi, Victoria Adedoyin, and Janet O. Agbaje. An empirical internet protocol network intrusion detection using isolation forest and one-class support vector machines. *International Journal of Advanced Computer Science and Applications*, 14(8), 2023.

[45] Gerard Shu Fuhnwi, Janet O Agbaje, Kayode Oshinubi, and Olumuyiwa James Peter. An empirical study on anomaly detection using density-based and representative-based clustering algorithms. *Journal of the Nigerian Society of Physical Sciences*, pages 1364–1364, 2023.

[46] Gerard Shu Fuhnwi, Matthew Revelle, and Clemente Izurieta. A hybrid anomaly detection approach for obfuscated malware. In *2024 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 159–165, 2024.

[47] Gerard Shu Fuhnwi, Matthew Revelle, and Clemente Izurieta. Improving network intrusion detection performance : An empirical evaluation using extreme gradient boosting (xgboost) with recursive feature elimination. In *2024 IEEE 3rd International Conference on AI in Cybersecurity (ICAIC)*, pages 1–8, 2024.

[48] Gerard Shu Fuhnwi, Matthew Revelle, Bradley Whitaker, and Clemente Izurieta. Using large language models to mitigate human-induced bias in sms spam: An empirical approach. In *2025 IEEE 4th International Conference on AI in Cybersecurity (ICAIC)*, pages 1–7, 2025.

[49] Sridevi Gadde, A Lakshmanarao, and S Satyanarayana. Sms spam detection using machine learning and deep learning techniques. In *2021 7th international conference on advanced computing and communication systems (ICACCS)*, volume 1, pages 358–362. IEEE, 2021.

[50] Isabell Gaylord. Network intrusion: How to detect and prevent it. *Retrieved from United States Cybersecurity Magazine: https://www. uscybersecurity. net/network-intrusion*, 2021.

[51] Aurélien Géron. Hands-on machine learning with scikit-learn, keras, and tensorflow: concepts. *Aurélien Géron-Google Kitaplar, yy https://books. google. com. tr/books*, 2019.

[52] Rubén González-Sendino, Emilio Serrano, Javier Bajo, and Paulo Novais. A review of bias and fairness in artificial intelligence. 2023.

[53] Frank Ephraim Grubbs. *Sample criteria for testing outlying observations*. University of Michigan, 1949.

[54] Dikshant Gupta, Suhani Singhal, Shamita Malik, and Archana Singh. Network intrusion detection system using various data mining techniques. In *2016 International Conference on Research Advances in Integrated Navigation Systems (RAINS)*, pages 1–6. IEEE, 2016.

[55] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[56] Muhammad Usman Hadi, Rizwan Qureshi, Abbas Shah, Muhammad Irfan, Anas Zafar, Muhammad Bilal Shaikh, Naveed Akhtar, Jia Wu, Seyedali Mirjalili, et al. A survey on large language models: Applications, challenges, limitations, and practical usage. *Authorea Preprints*, 3, 2023.

[57] Tarfa Hamed, Rozita Dara, and Stefan C Kremer. Network intrusion detection system based on recursive feature addition and bigram technique. *computers & security*, 73:137–155, 2018.

[58] John A Hartigan, Manchek A Wong, et al. A k-means clustering algorithm. *Applied statistics*, 28(1):100–108, 1979.

[59] Mohammed Hassanin and Nour Moustafa. A comprehensive overview of large language models (llms) for cyber defences: Opportunities and directions. *arXiv preprint arXiv:2405.14487*, 2024.

[60] Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern recognition letters*, 24(9-10):1641–1650, 2003.

[61] Steven A Hofmeyr, Stephanie Forrest, and Anil Somayaji. Intrusion detection using sequences of system calls. *Journal of computer security*, 6(3):151–180, 1998.

[62] Minseok Hur, Sooyon Seo, Jaeho Hwang, Hyelim Lim, and Moohong Min. Utilizing large language models for detection of sms spam in few-shot settings. *Available at SSRN 4815382*.

[63] Nwokedi Idika and Aditya P Mathur. A survey of malware detection techniques. *Purdue University*, 48(2):32–46, 2007.

[64] Danial Javaheri and Mehdi Hosseinzadeh. A framework for recognition and confronting of obfuscated malwares based on memory dumping and filter drivers. *Wireless Personal Communications*, 98:119–137, 2018.

[65] Jueun Jeon, Jong Hyuk Park, and Young-Sik Jeong. Dynamic analysis for iot malware detection with convolution neural network model. *Ieee Access*, 8:96899–96911, 2020.

[66] Wang Ke. Anomalous payload-based network intrusion detection. *Proc. of Recent Advance in Intrusion Detection (RAID), September 2004*, 2004.

[67] Amit Kumar, Harish Chandra Maurya, and Rahul Misra. A research paper on hybrid intrusion detection system. *International Journal of Engineering and Advanced Technology (IJEAT) Vol*, 2, 2013.

[68] Nilesh Kunhare, Ritu Tiwari, and Joydip Dhar. Particle swarm optimization and feature selection for intrusion detection system. *Sādhanā*, 45:1–14, 2020.

[69] Maxime Labonne and Sean Moran. Spam-t5: Benchmarking large language models for few-shot email spam detection. *arXiv preprint arXiv:2304.01238*, 2023.

[70] Wei-Jen Li, Ke Wang, Salvatore J Stolfo, and Benjamin Herzog. Fileprints: Identifying file types by n-gram analysis. In *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop*, pages 64–71. IEEE, 2005.

[71] Yihua Liao and V Rao Vemuri. Use of k-nearest neighbor classifier for intrusion detection. *Computers & security*, 21(5):439–448, 2002.

[72] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(1):1–39, 2012.

[73] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM computing surveys*, 55(9):1–35, 2023.

[74] Maya Hilda Lestari Louk and Bayu Adhi Tama. Dual-ids: A bagging-based gradient boosting decision tree model for network anomaly intrusion detection system. *Expert Systems with Applications*, 213:119030, 2023.

[75] Batta Mahesh et al. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, 9(1):381–386, 2020.

[76] Larry M Manevitz and Malik Yousef. One-class svms for document classification. *Journal of machine Learning research*, 2(Dec):139–154, 2001.

[77] Goeffrey J McLachlan. Mahalanobis distance. *Resonance*, 4(6):20–26, 1999.

[78] Souhail Meftah, Tajjeeddine Rachidi, and Nasser Assem. Network based intrusion detection using the unsw-nb15 dataset. *International Journal of Computing and Digital Systems*, 8(5):478–487, 2019.

[79] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM computing surveys (CSUR)*, 54(6):1–35, 2021.

[80] Moohong Min, Jemin J Lee, and Kyungho Lee. Detecting illegal online gambling (iog) services in the mobile environment. *Security and Communication Networks*, 2022(1):3286623, 2022.

[81] Tom M Mitchell and Tom M Mitchell. *Machine learning*, volume 1. McGraw-hill New York, 1997.

[82] Ugochukwu Mmaduekwe. Bias and fairness issues in artificial intelligence-driven cybersecurity. *Current Journal of Applied Science and Technology*, 43(6):109–119, 2024.

[83] Bilal Mohammed and Ekhlas K Gbashi. Intrusion detection system for nsl-kdd dataset based on deep learning and recursive feature elimination. *Engineering and Technology Journal*, 39(7):1069–1079, 2021.

[84] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[85] Robert Moskowitz. Network intrusion: Methods of attack. In *RVS Conference*, 2020.

[86] Nour Moustafa and Jill Slay. The significant features of the unsw-nb15 and the kdd99 data sets for network intrusion detection systems. In *2015 4th international workshop on building analysis datasets and gathering experience returns for security (BADGERS)*, pages 25–31. IEEE, 2015.

[87] Snehal A Mulay, PR Devale, and Goraksh V Garje. Intrusion detection system using support vector machine and decision tree. *International journal of computer applications*, 3(3):40–43, 2010.

[88] Naresh Kumar Nagwani and Aakanksha Sharaff. Sms spam filtering and thread identification using bi-level text classification and clustering techniques. *Journal of Information Science*, 43(1):75–87, 2017.

[89] Nir Nissim, Omri Lahav, Aviad Cohen, Yuval Elovici, and Lior Rokach. Volatile memory analysis using the minhash method for efficient and secured detection of malware in private cloud. *Computers & Security*, 87:101590, 2019.

[90] Tu Ouyang, Soumya Ray, Mark Allman, and Michael Rabinovich. A large-scale empirical analysis of email spam detection through network characteristics in a stand-alone enterprise. *Computer Networks*, 59:101–121, 2014.

[91] Swati Paliwal and Ravindra Gupta. Denial-of-service, probing & remote to user (r2l) attack detection using genetic algorithm. *International Journal of Computer Applications*, 60(19):57–62, 2012.

[92] Darshit Pandya. Spam detection using clustering-based svm. In *Proceedings of the 2019 2nd International Conference on Machine Learning and Machine Intelligence*, pages 12–15, 2019.

[93] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *proceedings of the 26th Symposium on Operating Systems Principles*, pages 1–18, 2017.

[94] Justin M Rao and David H Reiley. The economics of spam. *Journal of Economic Perspectives*, 26(3):87–110, 2012.

[95] Sergio Rojas-Galeano. Zero-shot spam email classification using pre-trained large language models. In *Workshop on Engineering Applications*, pages 3–18. Springer, 2024.

[96] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[97] Yvan Saeys, Inaki Inza, and Pedro Larranaga. A review of feature selection techniques in bioinformatics. *bioinformatics*, 23(19):2507–2517, 2007.

[98] Yakub Kayode Saheed, Micheal Olaolu Arowolo, and Abdulrauf U Tosho. An efficient hybridization of k-means and genetic algorithm based on support vector machine for cyber intrusion detection system. *International Journal on Electrical Engineering and Informatics*, 14(2):426–442, 2022.

[99] Oluwafemi A Sarumi, Adebayo O Adetunmbi, and Fadekemi A Adetoye. Discovering computer networks intrusion using data analytics and machine intelligence. *Scientific African*, 9:e00500, 2020.

[100] Izuru Sato, Yoshinori Okazaki, and Shigeki Goto. An improved intrusion detection method based on process profiling. *IPSJ Journal*, 43(11):3316–3326, 2002.

[101] Timo Schick and Hinrich Schütze. Exploiting cloze questions for few shot text classification and natural language inference. *arXiv preprint arXiv:2001.07676*, 2020.

[102] R Sekar, Mugdha Bendre, Dinakar Dhurjati, and Pradeep Bollineni. A fast automaton-based method for detecting anomalous program behaviors. In *Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001*, pages 144–155. IEEE, 2000.

[103] Burr Settles. Active learning literature survey. 2009.

[104] Neha V Sharma and Narendra Singh Yadav. An optimal intrusion detection system using recursive feature elimination and ensemble of classifiers. *Microprocessors and Microsystems*, 85:104293, 2021.

[105] Anjali Shinde, Essa Q Shahra, Shadi Basurra, Faisal Saeed, Abdulrahman A AlSewari, and Waheb A Jabbar. Sms scam detection application based on optical character recognition for image data using unsupervised and deep semi-supervised learning. *Sensors*, 24(18):6084, 2024.

[106] Benjamin Steenhoek, Md Mahbubur Rahman, Monoshi Kumar Roy, Mirza Sanjida Alam, Earl T Barr, and Wei Le. A comprehensive study of the capabilities of large language models for vulnerability detection. *arXiv preprint arXiv:2403.17218*, 2024.

[107] Songfeng Sun, Kaibo Lei, Zunshun Xu, Wubin Jing, and Guang Sun. Analysis of k-means and k-dbscan commonly used in data mining. In *2023 International Conference on Intelligent Media, Big Data and Knowledge Mining (IMBDKM)*, pages 37–41, 2023.

[108] Harini Suresh and John V Guttag. A framework for understanding unintended consequences of machine learning. *arXiv preprint arXiv:1901.10002*, 2(8):73, 2019.

[109] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*, pages 1–6. Ieee, 2009.

[110] Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[111] Cao Tien Thanh. A study of machine learning techniques for cybersecurity. In *2021 15th International Conference on Advanced Computing and Applications (ACOMP)*, pages 54–61. IEEE, 2021.

[112] Sebastian Thrun and Lorien Pratt. Learning to learn: Introduction and overview. In *Learning to learn*, pages 3–17. Springer, 1998.

[113] Li Tian and Wang Jianwen. Research on network intrusion detection system based on improved k-means clustering algorithm. In *2009 International Forum on Computer Science-Technology and Applications*, volume 1, pages 76–79. IEEE, 2009.

[114] Vijay Srinivas Tida and Sonya Hsu. Universal spam detection using transfer learning of bert model. *arXiv preprint arXiv:2202.03480*, 2022.

[115] Jacob W Ulvila and John E Gaffney Jr. Evaluation of intrusion detection systems. *Journal of Research of the National Institute of Standards and Technology*, 108(6):453, 2003.

[116] Sahil Verma and Julia Rubin. Fairness definitions explained. In *Proceedings of the international workshop on software fairness*, pages 1–7, 2018.

[117] Jin Wang, Chang Liu, Xin Shu, Hui Jiang, Xiao Yu, Jie Wang, and Wenna Wang. Network intrusion detection based on xgboost model improved by quantum-behaved particle swarm optimization. In *2019 IEEE Sustainable Power and Energy Conference (iSPEC)*, pages 1879–1884. IEEE, 2019.

[118] Wei Wang and Roberto Battiti. Identifying intrusions in computer networks based on principal component analysis. 2005.

[119] Y-M Wang, Doug Beck, Binh Vo, Roussi Roussev, and Chad Verbowski. Detecting stealth software with strider ghostbuster. In *2005 International Conference on Dependable Systems and Networks (DSN'05)*, pages 368–377. IEEE, 2005.

[120] Y-M Wang, Doug Beck, Binh Vo, Roussi Roussev, and Chad Verbowski. Detecting stealth software with strider ghostbuster. In *2005 International Conference on Dependable Systems and Networks (DSN'05)*, pages 368–377. IEEE, 2005.

[121] Michael West. Preventing system intrusions. In *Network and system security*, pages 29–56. Elsevier, 2014.

[122] Wikipedia. Performance analysis of nsl-kdd dataset using ann. In *2015 international conference on signal processing and communication engineering systems*, pages 92–96. IEEE, 2015.

[123] Wikipedia. Wikipedia contributors: Isolation forest. *The Free Encyclopedia*, 2020.

[124] Jenif D Souza WS and B Parvathavarthini. Machine learning based intrusion detection framework using recursive feature elimination method. In *2020 International Conference on System, Computation, Automation and Networking (ICSCAN)*, pages 1–4. IEEE, 2020.

[125] Zhixing Xu, Sayak Ray, Pramod Subramanyan, and Sharad Malik. Malware detection using machine learning based analysis of virtual memory access patterns. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pages 169–174. IEEE, 2017.

[126] M.J. Zaki and W. Meira. *Data Mining and Machine Learning: Fundamental Concepts and Algorithms*. Cambridge University Press, 2020.

[127] Bin Zhang, Wentao Xiao, Xi Xiao, Arun Kumar Sangaiah, Weizhe Zhang, and Jiajia Zhang. Ransomware classification using patch-based cnn and self-attention network on embedded n-grams of opcodes. *Future Generation Computer Systems*, 110:708–720, 2020.

[128] Hao Zhang, Yongdan Li, Zhihan Lv, Arun Kumar Sangaiah, and Tao Huang. A real-time and ubiquitous network attack detection based on deep belief network and support vector machine. *IEEE/CAA Journal of Automatica Sinica*, 7(3):790–799, 2020.