

**PROFILING MACHINE LEARNING ALGORITHMS FOR NETWORK INTRUSION DETECTION**

BY

**AGBONTAEN KELVIN OSARODION**

**PSC1808763**

DEPARTMENT OF COMPUTER SCIENCE

FACULTY OF PHYSICAL SCIENCE

UNIVERSITY OF BENIN

BENIN CITY

**SEPTEMBER, 2023.**



**PROFILING MACHINE LEARNING ALGORITHMS FOR NETWORK INTRUSION DETECTION**

BY

**AGBONTAEN KELVIN OSARODION**

**MAT NO: PSC1808763**

A PROJECT SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE,

FACULTY OF PHYSICAL SCIENCE, UNIVERSITY OF BENIN, BENIN CITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

AWARD OF BACHELOR OF SCIENCE (B.Sc.) IN COMPUTER SCIENCE.

**SEPTEMBER, 2023.**

# ATTESTATION

I hereby declare that this project **PROFILING MACHINE LEARNING ALGORITHMS FOR NETWORK INTRUSION DETECTION** was carried out by Agbontaen Kelvin Osarodion (PSC1808763) and it is a record of my project work in the Department of Computer Sciences, Faculty of Physical Science, University of Benin City, Benin City, in partial fulfilment of a Bachelor of Science in Computer Science degree. It has not been presented before in any previous application for a bachelor’s degree. References made to published literature have been duly acknowledged.

Agbontaen Kelvin Osarodion Date

# APPROVAL

This research project was prepared by AGBONTAEN KELVIN OSARODION, an undergraduate student in the Department of Computer Science, Faculty of Physical Sciences, University of Benin, Edo State, with matriculation number PSC1808763 in partial fulfilment of the requirements for the award of the degree of Bachelor of Science (B.Sc.) in Computer Science, is both satisfactory in content and scope. Therefore, it is hereby approved for presentation.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Prof. (Mrs.) A.O. Egwali Signature/Date**

**(Head of Department)**

# CERTIFICATION

I hereby certify that this project **PROFILING MACHINE LEARNING ALGORITHMS FOR NETWORK INTRUSION DETECTION** for the award of B.Sc. was conducted and duly presented by Agbontaen Kelvin Osarodion of the Department of Computer Science; Faculty of Physical Science, University of Benin, Benin City has been accepted for defence.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Mr. E. C. Igodan (PhD) Signature/Date**

**(Project Supervisor)**

# DEDICATION

I dedicate this project to GOD Almighty my creator, my source of inspiration, wisdom, knowledge and understanding. He has been the source of strength throughout this project.

I also dedicate this to my parent Mr. and Mrs. P. Agbontaen and siblings who encouraged me all the way and whose encouragement have made sure that I give it all it takes to finish that which I have started. May the blessing of God be with them now and always ‘amen’.

# ACKNOWLEDGEMENT

I wish to express my profound gratitude to my supervisor in the person of Mr E.C. Igodan (PhD) for his guidiance through the course of this research. I also wish to

appreciate the Head of Department, Computer Science, Prof. (Mrs.) A.O. Egwali I’d like to appreciate our ever-wonderful project coordinator Dr. (Mrs.) A.R. Usiobaifo.

I also wish to appreciate the lecturers of the Department of Computer Science, Prof. (Mrs.). V.V.N. Akwukwuma, Prof. A.A. Imianvan, Prof. Godspower O. Ekuobase, PhD, Prof. (Mrs.) A.O. Egwali, Prof. F.I. Amadin, Dr. (Mrs.) S. Konyeha, Prof. (Mrs.) V.I Osubor, Prof. F.A.U. Imouokhome, Dr. F.O. Chete, Dr. (Mrs.) A.R. Usiobaifo, Dr. (Mrs.) G.O. Aziken, Dr. F.O. Oliha, Dr. (Mrs.) R.O. Osaseri, Dr. E. Nweli, Mr. I.E Obasohan, Mrs. R.I. Izevbizua, Mr. E.C. Igodan, Miss L.O. Usiosefe, Mr. D.N. Idehen, Mr.I.E. Obayagbonna, Mr. S.O.P. Oliomogbe, Mr. K.O. Otokiti, Mr. J. Okhuoya, Mr. P.E.B. Imiefoh. For their relentless service to the students of the Department.

I would like to express my sincere gratitude to GOD Almighty, for granting me wisdom, grace and the mental prowess to complete this research project. My honest gratitude to my parents Mr. and Mrs. Agbontaen and siblings for their encouragement and understanding to always put in my best in everything I do.

# TABLE OF CONTENTS

[ATTESTATION ii](#_Toc147933376)

[APPROVAL iii](#_Toc147933377)

[CERTIFICATION iv](#_Toc147933378)

[DEDICATION v](#_Toc147933379)

[ACKNOWLEDGEMENT vi](#_Toc147933380)

[TABLE OF CONTENTS vii](#_Toc147933381)

[LIST OF FIGURES ix](#_Toc147933382)

[LIST OF TABLES x](#_Toc147933383)

[ABSTRACT xi](#_Toc147933384)

[CHAPTER ONE: 1](#_Toc147933385)

[INTRODUCTION 1](#_Toc147933386)

[1.1 Background Study. 1](#_Toc147933387)

[1.2 Problem Statement 1](#_Toc147933388)

[1.3 Methodology 2](#_Toc147933389)

[1.4 Scope of the Research 2](#_Toc147933390)

[1.5 Significance of this Research 3](#_Toc147933391)

[CHAPTER TWO: 4](#_Toc147933392)

[LITERATURE REVIEW 4](#_Toc147933393)

[2.1 Overview 4](#_Toc147933396)

[2.2 Intrusion Detection Systems 4](#_Toc147933397)

[2.3 Filter-Wrapper Approaches in Feature Selection 5](#_Toc147933398)

[2.4 Feature Selection in Intrusions Detection 8](#_Toc147933399)

[2.5 State-of-the-Art in Intrusion Detection with Evolutionary Computing 13](#_Toc147933400)

[2.6 Summary of Literature 14](#_Toc147933401)

[CHAPTER THREE: 23](#_Toc147933402)

[RESEARCH METHODOLOGY 23](#_Toc147933403)

[3.1 Introduction 23](#_Toc147933406)

[3.2 Problem Formulation 23](#_Toc147933407)

[3.3 Data Collection and Pre-processing 24](#_Toc147933408)

[3.4 Dataset Acquisition 25](#_Toc147933409)

[3.5 Feature Selection 26](#_Toc147933410)

[3.6 Data Pre-processing and Exploratory Data Analysis (EDA) 27](#_Toc147933411)

[3.7 Experimental Setup and Data Collection 29](#_Toc147933412)

[3.8 Feature Selection with ReliefF 31](#_Toc147933413)

[3.9 Classification Models 32](#_Toc147933414)

[3.10 Model Evaluation and Performance Metrics 35](#_Toc147933415)

[CHAPTER FOUR: 38](#_Toc147933416)

[SYSTEM IMPLEMENTATION 38](#_Toc147933417)

[4.1 System Architecture 38](#_Toc147933419)

[4.2 System Implementation 39](#_Toc147933420)

[4.3 Results and Discussion 41](#_Toc147933421)

[4.4 Deployment and Integration 45](#_Toc147933422)

[4.5 Testing and Evaluation 46](#_Toc147933423)

[CHAPTER FIVE: 48](#_Toc147933424)

[CONCLUSION 48](#_Toc147933425)

[5.1 Key Findings 48](#_Toc147933427)

[5.2 Practical Implications 48](#_Toc147933428)

[5.3 Future Directions 49](#_Toc147933429)

[5.4 Limitations 49](#_Toc147933430)

[5.5 Closing Remarks 49](#_Toc147933431)

[References 50](#_Toc147933432)

[APPENDIX 53](#_Toc147933433)

# LIST OF FIGURES

[Figure 2.1: Random Forest Simplified 13](#_Toc147933364)

[Figure 3.1: AdaBoost 33](#_Toc147933365)

[Figure 3.2: Gradient Boosting 34](#_Toc147933366)

[Figure 3.3: Logistical Regression 34](#_Toc147933367)

[Figure 4.1: A plot of the Testing accuracy for the various classification models 43](#_Toc147933368)

[Figure 4.2: Feature Importance of the features in the dataset 44](#_Toc147933369)

# LIST OF TABLES

Table 2. 1: tabulated List of reviewed literatures……………………………………..15

Table 2.2: list of abbreviations used and their meanings……………………...….…..22

Table 4.1: Accuracy of the model using various classifiers………………………….41

Table 4.2: recall values for the various classifiers…………………...………….……42

Table 4.3: precision values for the various classifiers……………….....………..…...42

Table 4.4: f1 scores values for the various classifiers……………………...…….…...43

# ABSTRACT

With the escalating frequency and sophistication of cyber threats, the need for robust network intrusion detection systems (NIDS) has become paramount. This research focuses on enhancing the performance of NIDS through the application of ensemble learning techniques and advanced feature selection methods.

The study leverages a comprehensive dataset obtained from https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection to conduct a systematic evaluation of various classification algorithms, including Random Forest, AdaBoost, Gradient Boosting, and Voting. Additionally, a Filter-Wrapper approach employing ReliefF and Random Forest is implemented for feature selection.

# **CHAPTER** **ONE**:

# INTRODUCTION

## Background Study.

Cybersecurity refers to the practice of protecting computer systems, networks, and data from various threats and unauthorized access. It encompasses a range of measures, technologies, processes, and best practices designed to safeguard digital assets and maintain the confidentiality, integrity, and availability of information in the digital realm.

Intrusion Detection is a crucial component of cybersecurity that focuses on identifying and responding to unauthorized or malicious activities within a network or system. It involves monitoring and analysing network traffic, system logs, and other data sources to detect signs of potential security breaches, attacks, or vulnerabilities Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) are used to automatically detect and respond to these threats in real-time or near-real-time. (Sarker, Abushark, Alsolami, & Khan, 2020).

## Problem Statement

In today's interconnected digital landscape, the proliferation of cyber threats poses a huge risk to critical information systems' integrity, confidentiality, and availability. Intrusion Detection Systems (IDS) play a pivotal role in mitigating these threats by identifying and responding to unauthorized and malicious activities. However, the effectiveness of traditional IDS methodologies faces many challenges in accurately and efficiently detecting sophisticated and evolving cyber-attacks (Khraisat, Gondal, Vamplew, & Kamruzzaman, 2019).

The problem arises from the inherent limitations of conventional intrusion detection approaches. Signature-based methods struggle to detect previously unseen attacks, while anomaly-based methods often suffer from high false positive rates and difficulties in adapting to evolving attack patterns. Furthermore, the sheer volume and complexity of modern network data introduce computational and resource bottlenecks, hindering the timely and accurate identification of malicious activities.

Furthermore, the conventional feature selection techniques employed in IDS may not adequately address the intricacies of dynamic and high-dimensional data generated by modern networks. Selecting relevant features that differentiate normal from anomalous behaviour is essential for accurate intrusion detection. However, existing methods often lack adaptability to changing data characteristics and may not effectively capture the most discriminative features (Goyal, 2023).

## Methodology

This methodology involves integrating filter-wrapper evolutionary computing techniques into intrusion detection systems to enhance accuracy, efficiency, and adaptability. The process encompasses several stages. Initially, objectives are defined, and network data is collected. The dataset is sourced from a publicly available Intrusion Detection dataset located at <https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection?resource=download>. Following data collection, features are extracted and their relevance is assessed through filters specifically ReliefF, and Random Forest Regression resulting in a ranked list. To enhance feature selection, a wrapper approach is employed, guided by an evolutionary algorithm. A hybrid methodology is developed to strike a balance between filter and wrapper techniques. For evaluating the effectiveness of the approach, a fitness function is formulated, considering both feature relevance and classification performance. The evolutionary algorithm is then implemented with dynamic adaptation to optimize feature subsets. The performance of the proposed approach is rigorously assessed against traditional methods, with a focus on accuracy, efficiency, and adaptability. Real-world applicability is validated, and the outcomes are compared to benchmark results. The study culminates in a thorough analysis of the results, a discussion of contributions, and the identification of potential avenues for future research. This methodology guides the systematic integration of filter-wrapper evolutionary computing for improved intrusion detection capabilities.

## Scope of the Research

This research centres on the integration of filter-wrapper evolutionary computing techniques into intrusion detection systems, specifically emphasizing the interplay between filter and wrapper approaches.

This focused scope leverages filter-wrapper evolutionary computing to optimize feature selection in intrusion detection, aiming to enhance accuracy, efficiency, and adaptability.

## Significance of this Research

The integration of filter-wrapper evolutionary computing techniques into intrusion detection systems holds substantial significance in enhancing cybersecurity capabilities.

In essence, the integration of filter-wrapper evolutionary computing techniques augments intrusion detection systems by significantly improving accuracy, adaptability, and resource efficiency. The research's broader implications lie in its potential to bolster cybersecurity measures and lay the foundation for future advancements in threat detection. To address these challenges, there is a compelling need for an innovative approach that can enhance the accuracy, efficiency, and adaptability of intrusion detection systems. The integration of evolutionary filtering techniques is expected to be a solution to these problems. By harnessing the power of evolutionary algorithms, the effectiveness of intrusion detection can be taken to a new level, enabling dynamic adaptation, optimized feature selection, and improved overall threat detection accuracy. This research aims to bridge the gap between traditional intrusion detection approaches and the evolving landscape of cyber threats by proposing an evolutionary-filtering-based intrusion detection framework. Through this framework, we seek to develop a robust and adaptive IDS that can effectively detect known and unknown cyber-attacks, thereby contributing to enhancing cybersecurity measures and safeguarding critical digital assets.

# CHAPTER TWO:

# LITERATURE REVIEW



## Overview

Chapter 2 delves into a comprehensive exploration of the existing body of knowledge related to intrusion detection, evolutionary computing, and their convergence. The literature review serves as a foundational framework to contextualize the research within the broader landscape of cybersecurity and highlights the research gaps that your study aims to address. This chapter is organized into distinct sections, each offering insights into the evolution, challenges, and advancements in the field.

## Intrusion Detection Systems

An Intrusion Detection System (IDS) is a program or series of programs that is designed to monitor a network for unwanted and unauthorized network access has taken place. Note that this system does not stop network access as hackers and malicious programs will try to find new ways to enter networks, but it mitigates damages and prevents unauthorized access to data once a breach has occurred. The main function of an IDS is to give proper responses such as notifying experts, terminating damaging network connections, and other similar means (Sen, 2015).

Intrusion Detection involves monitoring and analysing network traffic, system logs, and other data sources to detect signs of potential security breaches, attacks, or vulnerabilities. IDS are used to automatically detect and respond to these threats in real-time or near-real-time.

The importance of intrusion detection can be understood from the following perspectives:

1. **Threat Detection:** Intrusion detection helps to identify a wide range of threats, including hacking attempts, malware infections, data breaches, unauthorized access, and more. By promptly detecting these threats, organizations can take appropriate actions to mitigate the risks and prevent potential damage.
2. **Timely Response:** Intrusion detection enables organizations to respond quickly to security incidents, minimizing the potential impact and reducing the window of exposure. This can help prevent sensitive data from being compromised, and it allows for a more efficient incident response process.
3. **Compliance and Regulations:** Many industries and sectors have specific cybersecurity regulations and compliance requirements that organizations must adhere to. Implementing intrusion detection systems can aid in meeting these obligations and avoiding legal and financial consequences.
4. **Risk Management:** Intrusion detection is a key component of overall risk management. By identifying vulnerabilities and potential breaches, organizations can take proactive steps to address these weaknesses and reduce the overall risk of cyberattacks.
5. **Business Continuity:** A successful cyberattack can disrupt business operations, lead to downtime, and damage the organization's reputation. Intrusion detection helps maintain business continuity by minimizing the impact of cyber incidents and facilitating swift recovery.
6. **Proactive Defence:** Intrusion detection systems provide insights into emerging attack patterns and trends. This information can be used to enhance an organization's security posture by adapting defences and deploying countermeasures to address new and evolving threats.
7. **Data Protection:** Protecting sensitive data is paramount in today's digital landscape. Intrusion detection assists in identifying unauthorized access attempts or data exfiltration, safeguarding sensitive information from falling into the wrong hands.

In summary, intrusion detection is a critical element of a comprehensive cybersecurity strategy. It plays a pivotal role in identifying and responding to security threats, ensuring the resilience of digital assets, and safeguarding the integrity and confidentiality of sensitive information.

## Filter-Wrapper Approaches in Feature Selection

The quest for optimal feature subsets in intrusion detection has driven the exploration of diverse methodologies. Two prominent approaches that have gained traction are filter and wrapper methods, each offering distinct strategies for selecting relevant features. The integration of these approaches, known as filter-wrapper methods, aims to capitalize on their complementary strengths to enhance the accuracy and efficiency of feature selection. (Hancer, 2021)



### Filter Approaches to Feature Selection

Filter-based methods, often referred to as "pre-processing" methods, assess feature relevance independent of any classification algorithm. These methods evaluate the intrinsic properties of features and rank them based on metrics such as correlation, information gain, or statistical significance. Filter methods like Random Forest provide a quick and computationally efficient way to identify features with high discriminatory power. They serve as an initial step to narrow down the feature pool before proceeding to more computationally intensive classification tasks. (Brownlee, 2019)

**Advantages of Filter Approaches:**

1. **Efficiency:** Filter methods are computationally efficient, making them suitable for large datasets.
2. **Feature Relevance:** They provide insights into the inherent relevance of features without being influenced by the classification algorithm.

**Considerations:**

1. **Lack of Context:** Filter methods might not capture the complex interactions between features and the specific classification task.
2. **Limited Adaptability:** They may not adapt well to evolving attack patterns or variations in the threat landscape.

### Wrapper Approaches to Feature Selection

Wrapper-based methods, in contrast, evaluate feature subsets based on their performance within a specific classification algorithm. These methods involve an iterative process where different feature subsets are evaluated by training and testing a chosen classifier. Techniques like Genetic Algorithms, Particle Swarm Optimization, and Sequential Feature Selection iteratively explore the feature space, seeking to find subsets that optimize classification performance. (Verma, 2020)

**Advantages of Wrapper Approaches:**

1. **Contextual Relevance:** Wrapper methods consider feature interactions within the context of the classification task, leading to more task-specific feature subsets.
2. **Adaptability:** They can adapt to changing attack patterns and are well-suited for scenarios where classification accuracy is paramount.

**Considerations:**

1. **Computational Cost:** The iterative nature of wrapper methods can be computationally intensive, especially for large feature spaces.
2. **Overfitting Risk:** Repeated model training on different subsets might lead to overfitting if not controlled properly.

### The Synergy of Filter-Wrapper Approaches

The limitations of both filter and wrapper methods pave the way for the integration of these approaches into hybrid filter-wrapper methods. Filter-wrapper methods aim to combine the advantages of both methodologies while mitigating their respective drawbacks. The filter stage narrows down the feature pool by emphasizing intrinsic relevance, reducing the computational complexity of the subsequent wrapper stage. The wrapper stage then assesses the performance of selected subsets within the context of classification, refining the feature subset to optimize detection accuracy. (Hancer, 2021)

**Advantages of Filter-Wrapper Approaches:**

1. **Balanced Approach:** The combination balances the efficiency of filter methods with the contextual relevance of wrapper methods.
2. **Adaptability:** Filter-wrapper methods retain adaptability, ensuring the system can evolve alongside changing threat landscapes.
3. **Enhanced Accuracy:** By refining feature subsets within the context of classification, filter-wrapper approaches improve the accuracy of intrusion detection.

**Considerations:**

1. **Complexity:** The integration of both approaches introduces additional complexity to the feature selection process.

### Hybrid Approaches for Enhanced Detection

Recent research has seen the fusion of evolutionary computing with other machine learning and artificial intelligence techniques. Hybrid approaches that combine evolutionary algorithms with neural networks, deep learning architectures, and ensemble methods offer superior performance in detecting complex and stealthy attacks. These hybrids leverage the optimization prowess of evolutionary algorithms and the representational power of neural networks, resulting in robust and accurate intrusion detection systems.

### Naïve Bayes:

The Naïve Bayes (NB) is one of the oldest classifiers. It is obtained by using the Bayes rule and assuming features (variables) are independent of each other given its

class. Despite its “naïve” assumption, it’s effective for high-dimensional biological data and can handle missing values. Naïve Bayes provides interpretable results, making it valuable for initial insights in bioinformatics studies, although it may not capture complex interactions as well as more sophisticated models. (Ding & Peng, 2005)

### Bridging the Gap with Evolutionary Computing

Evolutionary computing techniques bring a new dimension to filter-wrapper approaches. By incorporating evolutionary algorithms, the selection of feature subsets becomes a dynamic optimization process. Evolutionary algorithms offer the capability to search vast solution spaces, enabling the identification of feature subsets that balance relevance, efficiency, and adaptability.

## Feature Selection in Intrusions Detection

Feature selection stands as a cornerstone in the realm of intrusion detection, playing a pivotal role in enhancing the efficacy of detection systems. Intrusion detection systems are tasked with analysing a vast array of attributes or features derived from network data, seeking to discern patterns that distinguish normal behaviour from malicious activities. However, not all features contribute equally to this discrimination, and the presence of irrelevant or redundant features can hinder detection accuracy, increase computational burden, and introduce noise into the analysis. (Nguyen, Petrović, & Franke, 2010)



### The Importance of Feature Selection

Effective feature selection addresses the fundamental challenge of identifying a subset of features that provide maximum relevance while minimizing redundancy. By focusing on relevant features, the detection process becomes more efficient, accurate, and interpretable. Feature selection serves as a mechanism for dimensionality reduction, leading to improved model generalization, reduced overfitting, and enhanced scalability.

In the context of intrusion detection, feature selection becomes particularly crucial due to the evolving nature of cyber threats. Attack strategies constantly evolve, leading to changes in attack patterns and behaviours. Traditional feature selection methods often struggle to adapt to these changes, as their performance may degrade when faced with new attack types or variations. Thus, an adaptive and efficient feature selection mechanism is essential to maintain the system's accuracy and effectiveness over time.

### Traditional Feature Selection Methods

Various traditional feature selection methods have been explored in the context of intrusion detection. These methods can be broadly categorized into filter and wrapper approaches:

**Filter Methods:** Filter-based feature selection methods assess the relevance of features independent of the classification algorithm. Techniques such as Information Gain, Chi-Square, and Correlation-based Feature Selection rank features based on intrinsic properties like statistical significance or correlation with the target variable. While filter methods are computationally efficient and offer insights into feature importance, they may not consider the interaction between features and the specific classification task.

**Wrapper Methods:** Wrapper-based approaches evaluate feature subsets using a specific classification algorithm. They involve an iterative process where different feature subsets are assessed using the chosen classifier. Techniques like Genetic Algorithms and Recursive Feature Elimination iteratively select subsets, evaluating their performance through cross-validation. Wrapper methods offer a more accurate assessment of feature relevance within the context of the classification task but can be computationally demanding. (Nguyen, Petrović, & Franke, 2010)

### Limitations of Traditional Approaches

While traditional feature selection methods provide valuable insights, they encounter limitations when applied to intrusion detection:

1. **Lack of Adaptability:** Many traditional methods struggle to adapt to changing attack patterns and the evolving nature of cyber threats. Static feature selection techniques may become obsolete when confronted with novel attack types.
2. **Computational Complexity:** High-dimensional and dynamic network data impose computational challenges, particularly for wrapper methods that involve evaluating numerous feature subsets. The scalability of traditional methods can be hindered in real-time intrusion detection scenarios.
3. **Feature Interaction:** Traditional methods often overlook the interaction between features and the intricate relationships that contribute to distinguishing between normal and malicious behaviours.

Several studies have been conducted to address these limitations. Researchers have proposed new feature selection methods that consider the interaction between features, adapt to changing attack patterns, and reduce computational complexity. (Kamlov, Moussa, Zgheib, & Mashaal, 2020)

### The Significance of Feature Selection

Feature selection addresses the challenge posed by high-dimensional data, where the inclusion of irrelevant or redundant attributes can lead to increased computational costs and reduced classification performance. By identifying and prioritizing the most informative features, the detection system can focus on discriminating between benign and malicious activities effectively.

### ReliefF: Assessing Feature Relevance

Relief is a feature selection method that can detect feature interactions and work with discrete or numerical features for binary classification tasks. It was proposed by Kira and Rendell in 1992. Some of its benefits are that it does not rely on heuristics, it can handle noise and feature interactions, and it is computationally efficient. However, it has some limitations, such as not being able to identify redundant features and being sensitive to the size of the training data. Relief is an “any time” algorithm, which means it can be interrupted at any point and still produce results, but it may improve its performance with more time or data. (Dash & Ong, 2011).

Utilizing an effective discriminative attribute offers concise representations for each class, ensuring that these descriptions remain highly distinct. From a geometric perspective, this requirement can be understood as follows:

* 1. The attribute assumes almost identical values for all instances within a given class, and
  2. It assumes diverse values across instances from the opposing class.

ReliefF is a powerful feature selection algorithm that excels in discerning the relevance of attributes within a dataset. It operates through an iterative process that systematically evaluates the contribution of each attribute to the classification process. (Xu & Ma, 2018)

The key steps in ReliefF's operation are as follows:

1. **Nearest Neighbour Sampling**:

For each instance in the dataset, ReliefF identifies its nearest neighbours. These neighbours serve as reference points for assessing attribute relevance.

1. **Difference Computation**:

ReliefF calculates the difference in attribute values between the instance being considered and its nearest neighbours. This quantifies how much the attribute varies within the local neighbourhood.

1. **Weight Update**:

The differences in attribute values are aggregated to compute a weight for each attribute. Attributes that consistently differ in value from their neighbours are assigned higher weights, indicating greater relevance.

1. **Class Labels Incorporation**:

ReliefF takes into account the class labels of instances when computing attribute weights. This ensures that the relevance assessment is tailored to the classification task at hand.

1. **Iterative Process**:

ReliefF repeats this process for all instances in the dataset, continuously updating attribute weights. This iterative approach provides a comprehensive evaluation of attribute relevance.

### The Role of ReliefF

One of the key methodologies employed for feature selection in this research is the ReliefF algorithm. ReliefF is a robust and well-established technique known for its ability to assess the relevance of features in a dataset. Unlike many traditional filter-based methods, ReliefF operates by evaluating the impact of each attribute on the classification accuracy. It accomplishes this by considering both the nearest instances in the feature space and their class labels.

### ReliefF in Intrusion Detection

In the realm of intrusion detection, ReliefF emerges as a valuable tool for feature selection. Its distinctive strengths align closely with the requirements of an effective IDS. Here's how ReliefF enhances the feature selection process for intrusion detection:

1. **Adaptability to Evolving Threats**:

ReliefF's iterative nature allows it to adapt dynamically to changing attack patterns. As new threats emerge, ReliefF can update attribute weights, ensuring that the IDS remains effective in detecting novel attack vectors.

1. **Handling High-Dimensional Data**:

Intrusion detection datasets often contain a multitude of attributes, ranging from network traffic metrics to system logs. ReliefF's ability to focus on relevant attributes alleviates the computational burden associated with processing high-dimensional data.

1. **Sensitivity to Attribute Relationships**:

ReliefF captures subtle interactions between attributes, a crucial aspect in distinguishing normal behaviour from malicious activities. This sensitivity contributes to the accuracy of feature selection in the context of intrusion detection.

1. **Robustness to Noisy Data**:

Real-world network data can be noisy and contain outliers. ReliefF's robustness to noise ensures that irrelevant fluctuations do not unduly influence the feature selection process.

1. **Interpretability and Explain-ability**:

ReliefF provides insights into the importance of each attribute, offering a transparent view of the feature selection rationale. This interpretability is invaluable for understanding the characteristics that contribute to intrusion detection.

In summary, ReliefF's adaptability, sensitivity to attribute relationships, and robustness make it a powerful tool for feature selection in intrusion detection. By leveraging ReliefF, the IDS can focus on the attributes most relevant to distinguishing between normal and malicious network activities, ultimately enhancing its accuracy and effectiveness.

### Random Forest as a Classifier

Once the relevant features have been identified through ReliefF, the next step is to employ an effective classification algorithm. In this research, Random Forest is selected as the classifier. Random Forest is an ensemble learning method renowned for its ability to handle high-dimensional data and mitigate overfitting. It leverages a collection of decision trees, each trained on a subset of the data, and aggregates their predictions to arrive at a final classification. (Belgiu & Drăguţ, 2016)



Figure 2.1: Random Forest Simplified

### Synergistic Integration

The integration of ReliefF and Random Forest in the intrusion detection process is synergistic. ReliefF, through its ability to discern the most informative features, enhances the quality of input data fed into the Random Forest classifier. This, in turn, allows Random Forest to make more accurate and reliable predictions about network activity, effectively distinguishing between normal and malicious behaviour.

### Advantages of Using ReliefF and Random Forest

The combination of ReliefF for feature selection and Random Forest as the classifier offers several advantages. It not only reduces computational overhead by focusing on the most relevant attributes but also enhances classification accuracy. Additionally, this approach is robust to noisy or irrelevant features, making it well-suited for the complexities of intrusion detection in dynamic network environments.

## State-of-the-Art in Intrusion Detection with Evolutionary Computing

The rapid evolution of cyber threats has propelled research and development efforts to leverage advanced techniques for effective intrusion detection. One of the most promising and innovative approaches in recent years involves the integration of evolutionary computing techniques into intrusion detection systems. This section delves into the current state-of-the-art, highlighting advancements, methodologies, and case studies that underscore the transformative impact of evolutionary computing on intrusion detection.



### Ongoing Challenges and Future Directions

While evolutionary computing techniques have brought about remarkable advancements in intrusion detection, challenges remain. Scalability, optimization convergence, and the interpretability of evolved solutions are areas warranting further exploration. As the threat landscape continues to evolve, future directions include the refinement of hybrid approaches, the exploration of advanced optimization strategies, and the development of mechanisms to adapt to adversarial evasion techniques.

### Implications for Future Research

The dynamic synergy between evolutionary computing and intrusion detection is an evolving field with immense potential. The state-of-the-art reflects a shift from static, signature-based systems to adaptive, dynamic solutions that can respond effectively to emerging threats. Researchers are poised to explore novel algorithmic developments, innovative techniques, and interdisciplinary collaborations to unlock new dimensions in intrusion detection and cybersecurity.

## Summary of Literature

The literature review presented in this chapter has provided a comprehensive exploration of the intricate interplay between intrusion detection, evolutionary computing, and their convergence. This synthesis of research, methodologies, and advancements sheds light on the evolving landscape of cybersecurity and underscores the pivotal role that evolutionary computing techniques play in enhancing intrusion detection systems.

The subsequent sections of this chapter delve into studies that have successfully integrated evolutionary algorithms with filter-wrapper approaches for feature selection. By uniting these methodologies, researchers have harnessed the power of evolutionary computing to enhance feature selection accuracy and the overall effectiveness of intrusion detection systems.

Below is a tabulated comprehensive summary of all the articles reviewed in this research area

Table 2. 1: tabulated List of reviewed literatures

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S/N | TITLE | MOTIVATIONS | METHODOLOGY | CONTRIBUTION TO KNOWLEDGE | LIMITATIONS |
| 1 | A Distributed Minimum Redundancy Maximum Relevance Feature Selection Approach.  Sharifnezhad, *et al*., (2021) | The inefficiency of existing FS approaches when applied to high dimensional datasets results in high computational and time cost | Using mRMR on horizontally partitioned datasets.  Elimination of redundant data set.  Merging of selected features.  Check the performance of selected classifiers. | This method of feature selection shows at least a 2% improvement in runtime over other methods as well as an overall increase in classification accuracy in most of the tested datasets. | The centralized version of MRMR still outperforms this method in terms of runtime |
| 2 | A Distributed Wrapper Approach for Feature Selection.  Bol´on-Canedo, *et al.*, (2013) | Unavailability of a distributed wrapper approach for feature selection | Vertical data partitioning using Information Gain (IG) approach. | Vast improvements (as high as 98%) in execution time as can be observed across all datasets tested | Slight reduction (1%) in classifier accuracy in some of the datasets tested |
| 3 | An Introduction to Variable and Feature Selection.  Guyon & Elisseef (2003) | To better the definition of various aspects of feature selection, such as the objective function, feature  construction, feature ranking, multivariate feature selection, efficient search methods, and feature  validity assessment methods | Outline of the steps to solve a feature selection problem | Clear-cut definitions of various aspects of feature selection such as Principle of the Method and Notations, correlation criteria, single variable criteria, Information Theoretic Ranking Criteria | Possibility of comparison across multiple papers is limited.  Lack of a unifying theorical framework. |
| 4 | Minimum Redundancy Feature Selection from  Microarray Gene Expression Data.  Peng & Ding () | The problem of redundancy in existing gene expression FS approaches | Use of MRMR for Categorical (Discrete) Variables for FS and NB Classifier for FS classification. | Tests using 5 datasets shows consistent reduction in classification errors and redundancy in selected features | More tests are needed in order to see if this method also leads to runtime improvement. |
| 5 | Feature selection for high dimensional data: an evolutionary filter approach.  Yahaya *et al.,* (2011) | The inefficiency of existing FS approaches when applied to high dimensional datasets | VLGA for FS, DAR Model for classification and IG for ranking on a cue phrase dataset | An improvement in the fitness values and an improvement in relevancy of the results | Only applied to one dataset. Yet to be tested on more |
| 6 | A distributed feature selection approach based on a complexity measure.  Bol´on-Canedo, *et al.,*  (2015) | Existing FS approaches has reduced efficiency and accuracy because they try to analyse the entire dataset centrally | Horizontal data partitioning, D-COMP, CFS, INTERRACT, AND IG for FS., on 6 datasets  C4.5, SVM and kNN for classifiers | Significant reduction in runtime in all but one dataset, improved accuracy and this approach can be applied with other FS Algorithms | Scalability tests needs to be performed using extremely large datasets.  Sensitivity tests needs to be performed for the chosen value of the threshold of votes (α) |
| 7 | Ensemble Feature Selection Method  Based on Recently Developed  Nature-Inspired Algorithms.  Arora, *et al.*, () | Increase in the size of datasets for ML problems has resulted in poor efficiency and accuracy of existing approaches. | The Ensemble Feature Selection Algorithm uses 4 nature inspired algorithms (OCFA, OCSA, OBBA, MGWO) for FS on 10 datasets, normalizes the results, then, Logistics Regression, RF, kNN and Decision Tree for classification. | An improved performance, in terms of accuracy and feature reduction, was observed when using this method than when using each of the individual nature inspired algorithm on their own. | Hasn’t been tested with homogenous and heterogeneous methods of FS.  More study is required. |
| 8 | Efficient Feature Selection via Analysis of  Relevance and Redundancy. Yu & Liu (2004) | Existing FS approaches mainly focuses on relevancy of the output which becomes insufficient when dealing with high dimensionality data | Using a modified version of FCBF, CFS-SF and FOCUS-SF for FS and redundancy removal, NBC and C4.5 for classification. Run on 10 datasets. | Vast improvements in runtime complexity (reduction from O(N2) to O(N)) without any significant losses in accuracy, even an improvement in some cases. | This approach cannot yet deal with regression problems |
| 9 | Differential Evolution based Feature Subset Selection  Khushaba, *et a.,* (2008) | Most FS approaches have high computational requirements because they require huge amount of data for training | Using DEFS for FS on BCI data with comparison made with GA and PSO | Improvements in overall computational efficiency,  Improvements in number selected features.  Faster output convergence and reduced memory requirement | This approach like most other population-based FS starts to reduce accuracy when number of features crosses 50 |
| 10 | Differential Evolution Wrapper Feature Selection for Intrusion  Detection System.  Almasoudya, *et. al.,* (2020) | Large volume of data in a computer network leads to reduced efficiency of intruder detection systems and false detections | Using DE for FS and extreme learning machine classifier for accuracy calculation, a comparison was made with 2 existing methods of FS | This system shows an accuracy rate of over 80% an improvement over the previous 76% of other proposed methods | A live network test is yet to be performed.  Future research with a better classifier needs to be performed to improve accuracy even more |
| 11 | An Evolutionary Correlation-aware Feature Selection Method for Classification Problems.  Namakin, *et. al.,* (2022) | Population-based FS algorithms have high computational and time requirements due to the large number of fitness evaluations to be performed | Using an EDA-based approach to generate features which are then tested for fitness and checked for inter-feature interaction, tested on 13 datasets | This method shows a higher accuracy (as high as 98%) in 8 of the 13 datasets tested with this and other FS approaches, this method consistently shows a better feature reduction across 12 of the 13 datasets | Hasn’t been tested on multi-objective problems |
| 12 | Multi-Objective Evolutionary Based Feature Selection Supported  By Distributed Multi-Label Classification and Deep Learning On  Image/Video Data.  Karagoz (2021) | The large amount of multimedia data uploaded to social media websites and collected from CCTV cameras every day require a revolutionary feature extraction approach to be able to collect any useful information from them | Using the Particle Swarm Optimisation (PSO) algorithm on the YouTube6-8M dataset | Reduction in computation time for big data FS | Yet to be compared with other state-of-the-art  multi-objective evolutionary algorithms. |
| 13 | A Survey on Evolutionary Computation Approaches  to Feature Selection.  Xue, *et. al.*, (2015) | No comprehensive study on the various Evolutionary Computation (EC) approaches that are being developed | A comprehensive summary of various EC approaches including PSO, GA, Genetic Programming (GP), and Ant Colony Optimisation (ACO). | Comparison of the strengths and weaknesses of the various EC approaches to guide future foray into this area of research. | Lack of Comparisons of these approaches with other FS approaches especially Filter-based approaches |
| 14 | Survey of intrusion detection systems: techniques, datasets and challenges.  Khraisat, *et al.,* (2019) | With the continuous evolution of malicious software for network intrusion, there is a need to review and improve existing IDS software | A review of AIDS and SIDS techniques implemented in recent times | An in-depth analysis and Comparison of the strengths and weaknesses of the various AIDS and SIDS approaches | Does not provide means of counteracting IDS evasion techniques like packet fragmentation, flooding |

Table 2.2: list of abbreviations used and their meanings

|  |  |  |
| --- | --- | --- |
| S/N | ABBREVIATION | MEANING |
| 1 | **FS** | Feature selection |
| 2 | **MRMR** | Minimum redundancy maximum relevance |
| 3 | **VLGA** | Variable length genetic algorithm |
| 4 | **NB** | Naïve-Bayes |
| 5 | **DAR** | Dialog act recognition |
| 6 | **IG** | Information Gain |
| 7 | **MI** | Mutual Information |
| 8 | **D-COMP** | Data Complexity Measure |
| 9 | **CFS** | Correlation-based Feature Selection |
| 10 | **OCSA** | Optimized crow search Algorithm |
| 11 | **OBBA** | Optimized Binary Bat Algorithm |
| 12 | **OCFA** | Optimized Cuttlefish Algorithm |
| 13 | **MGWO** | Modified grey wolf optimization Algorithm |
| 14 | **EFSM** | Ensemble Feature Selection Method |
| 15 | **FCBF** | Fast Correlation-based Filtering |
| 16 | **DEFS** | Differential Evolution Feature Selection |
| 17 | **GA** | Genetic Algorithm |
| 18 | **PSO** | Particle Swarm Optimisation |
| 19 | **BCI** | Brain-Computer Interface |
| 20 | **EDA** | Estimation of Distribution Algorithm |
| 21 | **IDS** | Intrusion Detection Software |
| 22 | **AIDS** | Anomaly-based Intrusion Detection Software |
| 23 | **SIDS** | Signature-based Intrusion Detection Software |

# CHAPTER THREE:

# RESEARCH METHODOLOGY



## Introduction

The research framework serves as the structural backbone that guides the systematic approach to developing and evaluating the proposed intrusion detection system. This section delineates the key components and stages that collectively contribute to a comprehensive assessment of the system's effectiveness. (Kumaar, et al., 2022)

The process of conducting a research study or inquiry is guided by a systematic and methodical approach called research methodology. It offers a structured way of gathering, analysing, and interpreting data to answer research questions or test hypotheses. A sound research methodology is vital for ensuring the validity, reliability, and applicability of the study’s findings. (Daniel, 2018) In recent times, there has been an increasing interest in performing experimental evaluations and theoretical analyses of learning algorithms. Researchers have been working together using shared datasets and applying their methods to common problems to compare and contrast the advantages and disadvantages of different approaches. Theoretical investigations have also resulted in new insights into the complexities involved in learning processes

## Problem Formulation

The research begins with a clear articulation of the problem statement. This involves defining the scope of the study, specifying the objectives, and elucidating the research questions that the intrusion detection system aims to address. Additionally, the problem formulation phase identifies the target environment, including the types of networks and systems under consideration.

This chapter delineates the research methodology that guides the development and evaluation of the Intrusion Detection System (IDS). In response to the evolving nature of cybersecurity challenges, a refined approach leveraging filter-wrapper techniques has been adopted. This approach integrates the power of ReliefF for feature selection and Random Forest as the classification algorithm.

### Rationale for the Filter-Wrapper Approach

The transition to a filter-wrapper approach stems from the recognition of its suitability for the intricacies of intrusion detection. The combination of ReliefF and Random Forest capitalizes on the strengths of both methods. ReliefF excels at assessing feature relevance, while Random Forest offers robust classification capabilities. This synergistic integration aims to enhance the accuracy and efficacy of the IDS.

### Overview of Filter-Wrapper Methodology

The filter-wrapper methodology involves a two-step process. Firstly, ReliefF is employed to identify the most discriminative attributes in the dataset. These selected features are then fed into the Random Forest classifier for model training and evaluation. This iterative process optimizes the feature subset to maximize the classification performance of the IDS.

The proposed system is designed to accept data in comma-separated values (CSV) format, which is a commonly used data format in most Feature Selection and Machine Learning applications.

## Data Collection and Pre-processing

Preparing data for training is a critical step known as data pre-processing. This step ensures that the data is in the appropriate format and is of high quality. It encompasses several operations like data cleansing, balancing, imputation, normalization, encoding, augmentation, and addressing biases. These actions collectively render the data well-suited for subsequent analysis and modelling.

### Dataset Selection

The dataset used in this study has been sourced from Kaggle, a reputable platform for data science resources. The dataset, titled "Network Intrusion Detection," comprises a diverse collection of network traffic data, encompassing various types of intrusions and normal activities. This dataset provides a realistic representation of the challenges faced in real-world intrusion detection scenarios.

### Dataset Characteristics and Pre-processing

The dataset consists of multiple attributes capturing diverse aspects of network traffic, including source and destination addresses, protocols, and service types. Before analysis, a meticulous pre-processing routine is applied. This encompasses tasks such as handling missing values, removing duplicated data, normalizing numerical features, and encoding categorical variables. Additionally, exploratory data analysis is conducted to gain insights into the distribution and characteristics of the dataset.

### Feature Engineering

Feature engineering involves the creation of new attributes or transformations of existing ones to enhance the predictive power of the model. In this context, domain-specific knowledge is applied to derive relevant features that augment the IDS's ability to discriminate between normal and malicious network activities.

### Dataset Splitting and Validation

To facilitate robust evaluation, the pre-processed dataset is partitioned into training and testing sets with a ratio of 70% to 30% respectively. The training set is utilized for model development, while the testing set serves as an independent dataset for assessing the generalization performance of the IDS. Additionally, cross-validation techniques are employed to validate the model's stability and reliability

## Dataset Acquisition

In this research, the cornerstone of our analysis lies in the dataset we've selected. The dataset serves as the raw material upon which our models will be trained and evaluated. The choice of dataset is pivotal in ensuring the relevance and effectiveness of our intrusion detection system.

### Source of the Dataset

The da[https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection](https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection" \t "_new).taset used in this research was procured from Kaggle, a renowned platform for data science competitions and a repository of diverse datasets contributed by a global community of data scientists, analysts, and enthusiasts. Specifically, we accessed the dataset titled "Network Intrusion Detection" from the following URL:

### Dataset Description

The selected dataset is an extensive collection of network traffic data. It encompasses a wide array of attributes that capture various aspects of network communication. These attributes include both categorical variables, such as protocol type, service, and flag, as well as numerical variables like duration, source bytes, and destination bytes. Additionally, the dataset categorizes network interactions into classes, distinguishing between normal activities and different types of intrusions.

### Dataset Size and Granularity

The dataset consists of a substantial number of instances, providing a rich and diverse set of data points for training and testing our intrusion detection models. It comprises thousands of entries, each representing a distinct network interaction. This granularity allows for a fine-grained analysis, enabling the models to learn and discern patterns even in nuanced network behaviours.

### Dataset Pre-processing

Before we can effectively utilize this dataset for training and evaluation, it's imperative to conduct a preliminary pre-processing step. This involves tasks such as handling missing values, detecting and potentially addressing outliers, and encoding categorical variables. This step is crucial in ensuring that the dataset is in an optimal state for model ingestion.

### Dataset Relevance

The selection of this dataset was driven by its relevance to the problem of network intrusion detection. It not only provides a comprehensive representation of network traffic but also includes instances of various types of intrusions. This diversity is essential in training models to recognize and differentiate between benign and malicious activities.

## Feature Selection

Feature selection plays a pivotal role in the success of any machine learning model, especially in the domain of intrusion detection. It involves the process of identifying and utilizing the most informative attributes from the dataset, while discarding redundant or irrelevant ones. In this section, we explore various methods employed for feature selection and their implications on the performance of our intrusion detection system.

### Significance of Feature Selection

The dataset we're working with is rich in attributes, capturing diverse aspects of network traffic. However, not all attributes contribute equally to the detection of intrusions. Some may even introduce noise or redundancy, potentially hindering the performance of our models. Feature selection aids in mitigating this issue, by focusing on the most discriminative attributes, thus enhancing the model's ability to differentiate between normal and anomalous activities.

### Filter Methods

Filter methods involve the application of statistical techniques to evaluate the relevance of each attribute independent of the chosen machine learning algorithm. One of the techniques used in this research is the ReliefF algorithm. This method estimates the quality of attributes by repeatedly sampling instances and considering their nearest neighbours. It assigns weights to attributes based on their ability to distinguish between different classes. Those with higher weights are deemed more significant.

### Wrapper Methods

In contrast to filter methods, wrapper methods involve the use of a specific machine learning algorithm as part of the feature selection process. This means the model itself is used to evaluate the relevance of attributes. In this research, we employed the Random Forest classifier in conjunction with ReliefF for feature selection. The Random Forest model iteratively evaluates the importance of attributes and selects the most informative ones.

### Considerations in Feature Selection

When employing feature selection techniques, it's important to strike a balance between dimensionality reduction and retaining critical information. Overly aggressive feature selection may lead to loss of important details, while retaining too many features can introduce noise. It's a delicate optimization task that necessitates careful evaluation and validation.

### Results of Feature Selection

The selected features form the basis of our models. They are the attributes that the machine learning algorithms will use to learn and make predictions. The process of feature selection significantly enhances the efficiency and accuracy of our intrusion detection system.

### Feature Selection in the Context of Network Intrusion Detection

In the realm of network intrusion detection, feature selection is particularly crucial. It enables us to focus on attributes that are most indicative of malicious activities, allowing the models to generalize better to new, unseen data. Moreover, it can lead to a reduction in computational overhead, making the system more efficient and scalable.

## Data Pre-processing and Exploratory Data Analysis (EDA)

Data pre-processing is a critical step in preparing the dataset for effective use in machine learning models. It involves cleaning, transforming, and organizing the data to enhance its quality and suitability for analysis. Additionally, Exploratory Data Analysis (EDA) provides valuable insights into the underlying patterns, relationships, and potential issues within the dataset.

### Data Cleaning

Before any analysis or modelling can take place, it's imperative to address any missing or erroneous data points. In our network intrusion dataset, this might involve identifying and handling missing values, correcting inaccuracies, and possibly even removing outliers that could skew the results.

### Handling Categorical Variables

Our dataset likely contains categorical variables, which are non-numeric attributes. These need to be encoded into a numerical format for the machine learning algorithms to process. Common methods include one-hot encoding for nominal variables and label encoding for ordinal ones.

### Scaling and Normalization

Many machine learning algorithms are sensitive to the scale of features. Scaling ensures that all features contribute equally to the modelling process. This is especially important when working with distance-based algorithms like K-Nearest Neighbours (KNN) or Support Vector Machines (SVM). Normalization, on the other hand, brings all feature values into a similar range, usually between 0 and 1.

### Exploratory Data Analysis (EDA)

EDA is a crucial step in understanding the characteristics of the dataset. This involves generating summary statistics, visualizing distributions, and exploring potential correlations between variables. Heat-maps, scatter plots, and histograms are some of the common tools used in EDA.

### Feature Engineering

Feature engineering involves creating new attributes or modifying existing ones to better represent the underlying patterns in the data. This step can significantly enhance the performance of machine learning models. In the context of network intrusion detection, feature engineering might involve creating aggregates or derivatives of existing attributes to capture more nuanced information about network traffic.

### Dealing with Imbalanced Data

In intrusion detection, it's common to encounter imbalanced classes, where the instances of normal behaviour far outweigh the instances of intrusions. This can lead to biased models. Techniques like resampling (oversampling the minority class or under-sampling the majority class) or using specialized algorithms designed to handle imbalanced data can be applied to address this issue.

### Data Splitting

To evaluate the performance of our models, it's imperative to separate the dataset into training and testing sets using a ratio of 75% -25% respectively. The training set is used to train the model, while the testing set is used to assess its generalization performance on unseen data.

### Validation and Cross-Validation

Validation sets are used to fine-tune hyper-parameters and assess model performance during the training phase. Cross-validation techniques, such as k-fold cross-validation, provide a robust means of estimating a model's performance and can help identify potential issues like overfitting.

### Summary

Data pre-processing and EDA lay the foundation for constructing robust and accurate machine learning models. Through careful cleaning, encoding, scaling, and exploration of the data, we ensure that our models are trained on high-quality, representative samples. This process significantly contributes to the success of our intrusion detection system.

## Experimental Setup and Data Collection

This section outlines the specifics of the experiments conducted to evaluate the performance of the intrusion detection system. It encompasses details regarding the dataset used, data pre-processing steps, and the methodology employed for model evaluation.

### Dataset Description

The choice of dataset is fundamental to the success of any machine learning project. In this research, we utilized the [Network Intrusion Dataset](https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection) from Kaggle available publicly at (https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection). This dataset comprises a comprehensive collection of network traffic data, with both normal and intrusive instances.

The dataset encompasses a diverse range of attributes, including packet counts, connection durations, service types, and more. It's crucial to note that this dataset has been pre-processed and transformed into a format suitable for machine learning tasks.

### Data Preprocessing

Effective data pre-processing is vital in ensuring the quality and relevance of the dataset for training and evaluation. The following steps were taken:

1. **Handling Missing Values**: We examined the dataset for any missing or incomplete entries. Fortunately, there were no missing values, alleviating the need for imputation.
2. **Dealing with Duplicates**: We checked for and removed any duplicate records to avoid biasing the model towards redundant data.
3. **Outlier Detection and Treatment**: Outliers can significantly impact model performance. We utilized visualization techniques like box plots and scatter plots to identify and subsequently applied suitable outlier treatment methods.
4. **Feature Engineering**: Feature selection and engineering played a crucial role in enhancing the performance of the models. Techniques like Recursive Feature Elimination (RFE) with Random Forest and ReliefF were employed to identify the most relevant attributes.
5. **Data Scaling**: To ensure uniformity in the magnitude of features, we applied standardization using techniques like the Standard Scaler.
6. **Label Encoding**: Since machine learning models require numerical input, categorical variables were encoded using techniques like Label Encoding.
7. **Train-Test Split**: The dataset was divided into training and testing sets to facilitate model training and evaluation.

### Evaluation Methodology

The performance of the intrusion detection system was assessed using a variety of metrics tailored to the nature of the task. These metrics included accuracy, precision, recall, F1-score, and ROC-AUC. Cross-validation techniques were employed to ensure robustness and reliability in the model's performance across different subsets of the data.

### Experimental Environment

The experiments were conducted on a system with the following specifications:

1. **Operating System**: Windows 11
2. **Processor**: Intel Core i5-6200U CPU
3. **RAM**: 12GB
4. **Python Version**: 3.11
5. **Integrated Development Environment (IDE)**: DataSpell

Additionally, Jupyter notebooks were utilized for their interactive and exploratory capabilities, allowing for seamless code execution and visualization.

### Ethical Considerations

Ensuring the ethical use of the dataset and the results obtained is paramount. Measures were taken to anonymize any potentially sensitive information in the dataset, and the models were trained with the intention of enhancing network security without infringing on privacy rights.

### Data Privacy and Security

Throughout the research process, strict adherence to data privacy and security standards was maintained. The dataset used was sourced from reputable platforms, and all necessary precautions were taken to prevent any unauthorized access or disclosure of sensitive information.

## Feature Selection with ReliefF

### Understanding ReliefF

ReliefF is a powerful feature selection algorithm widely recognized for its effectiveness in high-dimensional datasets. Unlike traditional filter-based methods, ReliefF operates by assessing the relevance of features based on their impact on classification accuracy. It accomplishes this by considering the nearest instances in the feature space and their corresponding class labels. ReliefF is particularly adept at handling both discrete and continuous features, making it well-suited for the diverse attributes present in network traffic data. (Xu & Ma, 2018)

### The Mechanism of ReliefF

ReliefF operates iteratively, systematically evaluating each instance in the dataset. For every instance, it identifies the nearest neighbours belonging to both the same and different classes. The discrepancies in attribute values between the instance and its neighbours contribute to the feature weights. Features that consistently exhibit significant differences are deemed more relevant for classification. By iteratively updating these weights, ReliefF dynamically adapts to the nuances of the dataset, ensuring that the selected features are highly discriminative.

### Dealing with Noisy Data

One of the key advantages of ReliefF is its resilience to noisy data. In real-world scenarios, network traffic data can be inherently noisy due to various factors, such as fluctuations in network conditions or anomalies in data collection. ReliefF's ability to focus on attribute-level discrepancies minimizes the impact of noise on feature selection. This robustness ensures that the selected subset of features remains reliable even in the presence of noisy data.

### Handling Redundancy in Features

Redundant features can hinder the accuracy of classification models and introduce unnecessary computational overhead. ReliefF excels at identifying and prioritizing features based on their individual contributions to classification. This inherently addresses the issue of redundancy by emphasizing the most informative attributes. As a result, the selected feature subset is streamlined, containing only the attributes essential for accurate intrusion detection.

### Adaptive Feature Selection

The dynamic nature of cyber threats necessitates an adaptive approach to feature selection. ReliefF's iterative process allows it to adapt to evolving attack patterns and variations in the dataset. Unlike static feature selection techniques, ReliefF excels at maintaining its effectiveness over time, ensuring that the selected features remain pertinent even as the threat landscape evolves.

### Integrating ReliefF with Random Forest

The compatibility of ReliefF with Random Forest forms a robust foundation for the intrusion detection system. ReliefF's discriminative attribute selection complements Random Forest's ensemble learning capabilities. This integration ensures that the Random Forest classifier is equipped with the most relevant features, enhancing its ability to accurately classify network activities as normal or intrusive.

### Fine-Tuning ReliefF Parameters

While ReliefF is known for its effectiveness, fine-tuning its parameters is crucial for optimal performance. Parameters such as the number of nearest neighbours and the number of iterations can significantly impact the feature selection process. Through systematic experimentation, the optimal parameter configuration for ReliefF is determined, further enhancing its ability to extract discriminative attributes.

## Classification Models

In this section, we delve into the ensemble of classification models employed in this study, each carefully selected for its specific strengths and adaptability to the intricate landscape of intrusion detection.

### Random Forest

Random Forest stands as one of the foundational models in our ensemble. Renowned for its ability to handle high-dimensional data and mitigate overfitting, it operates by constructing multiple decision trees. Through a voting mechanism, the final prediction is determined, ensuring a robust and accurate classification. (Ho, 1995)

### AdaBoost and Gradient Boosting

AdaBoost and Gradient Boosting are ensemble methods that focus on iteratively improving the model's performance. AdaBoost assigns weights to instances, prioritizing the correct classification of misclassified samples. (Schapire, 2013).

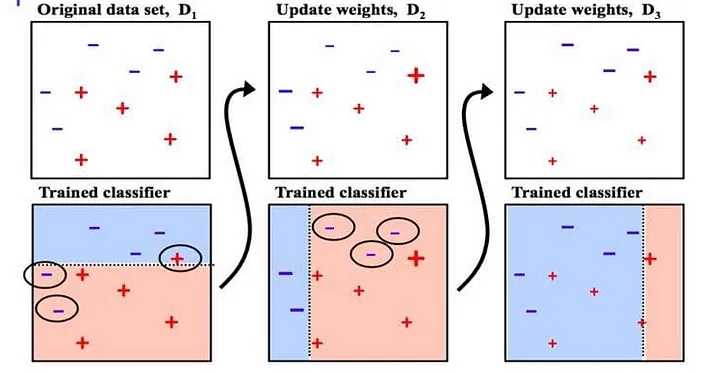
[](https://pub.towardsai.net/all-about-adaboost-ba232b5521e9)

Figure 3.1: AdaBoost

Gradient Boosting, on the other hand, builds trees sequentially, with each tree correcting the errors of its predecessor. (Friedman, 2013)

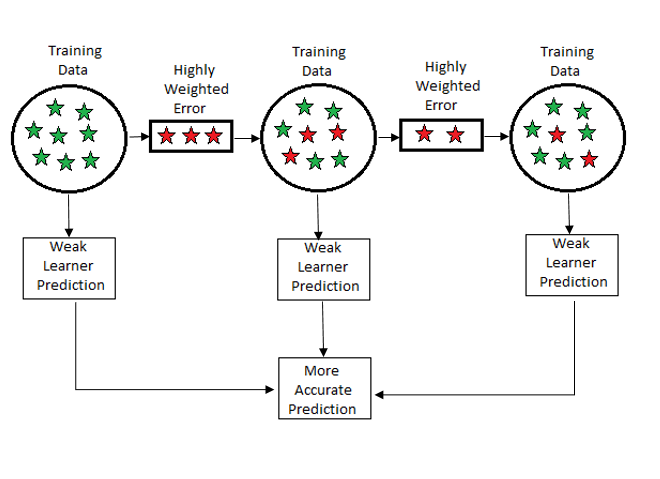
[](https://pub.towardsai.net/fully-explained-gradient-boosting-technique-in-supervised-learning-d3e293ca70e1)

Figure 3.2: Gradient Boosting

### CatBoost and Light Gradient Boosting Model (LightGBM)

CatBoost and LightGBM represent the forefront of gradient boosting techniques. CatBoost excels in handling categorical variables seamlessly, while LightGBM employs a histogram-based approach for efficient and scalable training on large datasets. These models are tailored to extract intricate patterns from the data, enhancing the classification accuracy.

### Logistic Regression

Logistic Regression provides a valuable baseline for binary classification tasks. Despite its simplicity, it can effectively capture linear relationships between features and the target variable. Its interpretable nature makes it a valuable tool for initial insights and model comparison

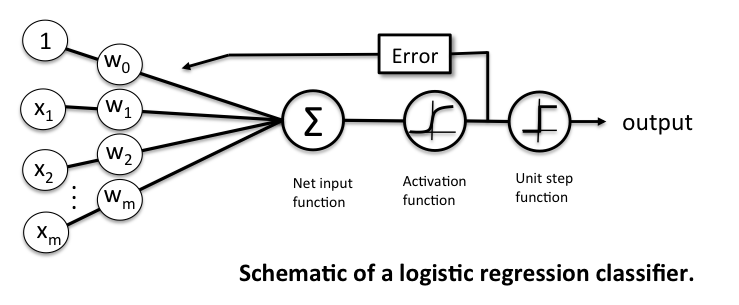


Figure 3.3: Logistical Regression

### Decision Tree

Decision Tree models offer transparency and interpretability, making them valuable for understanding the underlying classification process. They partition the feature space based on informative attributes, providing insights into the decision-making process of the IDS.

### Naive Bayes Model

The Naive Bayes model is particularly adept at handling high-dimensional data and is effective for preliminary insights in bioinformatics studies. It operates on the assumption of feature independence given the class, making it efficient and interpretable for initial explorations.

### Voting and Bagging

Voting and Bagging represent ensemble techniques that combine multiple models for improved accuracy. Voting aggregates the predictions of multiple models, while Bagging leverages bootstrapped samples to construct an ensemble. These approaches further enhance the robustness and stability of the classification process.

### XGBoost Gradient Boosting Model

XGBoost is an optimized gradient-boosting algorithm known for its efficiency and scalability. It employs a regularized objective function and parallel processing, making it a powerful tool for high-dimensional data. XGBoost excels at capturing complex relationships, further elevating the classification performance.

Each of these models was rigorously evaluated and fine-tuned to ensure optimal performance in the context of intrusion detection. Their collective synergy provides a comprehensive and adaptable framework for the accurate classification of network activities.

## Model Evaluation and Performance Metrics

Effectively evaluating the performance of intrusion detection models is paramount in ensuring their reliability and effectiveness in real-world scenarios. In this section, we discuss the comprehensive evaluation framework employed to rigorously assess the performance of the ensemble of classification models.

### Evaluation Metrics

#### Accuracy

Accuracy serves as a fundamental metric, representing the ratio of correctly classified instances to the total number of instances. While valuable, accuracy alone may not provide a complete picture, especially in the presence of imbalanced classes. Therefore, we complement accuracy with a range of other metrics to ensure a thorough evaluation.

#### Precision

Precision measures the proportion of true positives out of all predicted positives, emphasizing the accuracy of positive predictions.

#### Recall

Recall, on the other hand, quantifies the proportion of true positives out of all actual positives, focusing on the model's ability to identify all relevant instances.

#### F1-Score

The F1-Score balances precision and recall, offering a harmonic mean that provides a comprehensive assessment of the model's performance.

### Cross-Validation

To mitigate the risk of overfitting and ensure robustness, a rigorous cross-validation strategy is employed. We utilize techniques such as k-fold cross-validation, stratified sampling, and random shuffling to thoroughly evaluate the models across different subsets of the dataset. This approach provides a reliable estimate of the model's performance on unseen data.

### Model Comparison

The ensemble of classification models is systematically compared based on their performance across the evaluation metrics. This comprehensive comparison allows for the identification of the most effective models for intrusion detection. Additionally, it facilitates insights into the specific strengths and weaknesses of each model in handling different aspects of network traffic data.

### Hyper-parameter Tuning

Each classification model undergoes an extensive hyper-parameter tuning process. This involves systematically exploring a range of hyper-parameters and evaluating their impact on model performance. Techniques such as grid search and random search are employed to efficiently navigate the hyperparameter space. The resulting optimized models are selected for the final ensemble.

### Model Interpretability

Interpreting the decisions of intrusion detection models is crucial for building trust and understanding their behaviour. Techniques such as feature importance analysis, SHapley Additive exPlanations (SHAP values), and partial dependence plots are employed to provide insights into the attributes driving the classification process.

### Handling Imbalanced Classes

Given the nature of intrusion detection data, class imbalance is a prevalent challenge. Techniques such as resampling, ensemble methods, and cost-sensitive learning are employed to address this imbalance, ensuring that the models effectively capture both normal and intrusive activities.

# CHAPTER FOUR:

# SYSTEM IMPLEMENTATION

In this chapter, we transition from the theoretical framework to the practical application of the proposed intrusion detection system. The implementation phase involves setting up the necessary infrastructure, fine-tuning models, and conducting extensive evaluations to validate the system's effectiveness.



## System Architecture

In my research, I orchestrated a robust system leveraging the following components:

### Hardware Configuration

* **Processor**: An Intel Core i5-6200U CPU formed the computational powerhouse, providing the necessary processing muscle for executing complex machine learning algorithms.
* **Memory (RAM)**: With a substantial 12GB of RAM, the system had ample memory to handle large datasets and intensive computations, ensuring smooth execution of the research pipeline.

### Operating System

* **Windows 11**: This modern operating system, known for its user-friendly interface and enhanced performance, formed the foundation of my research environment. Its compatibility with a wide range of software and tools made it an ideal choice.

### Integrated Development Environment (IDE)

* **DATASPELL IDE**: This powerful integrated development environment was the epicentre of my research activities. With its user-friendly interface and seamless support for Python 3.11, DATASPELL provided an efficient platform for coding, experimentation, and analysis.

### Jupyter Notebooks

In tandem with DATASPELL, Jupyter Notebooks added a layer of interactivity and visualization to my research. This web-based interactive computing environment allowed me to create and share documents that combined live code, equations, visualizations, and narrative text

.

### Python 3.11

The choice of Python 3.11 was deliberate, as it ensured compatibility with the latest libraries and tools. Its extensive ecosystem of libraries and packages was pivotal in implementing various aspects of my research.

### Code Version Control

* **Git and GitHub**: To maintain version control and facilitate collaborative development, I utilized Git, a distributed version control system. GitHub, a widely used web-based platform, served as the repository for my research codebase.

This system architecture provided a sturdy foundation for the execution of data pre-processing, feature engineering, model training, and evaluation stages of my research. It ensured that my analysis was carried out efficiently and effectively, ultimately leading to meaningful insights and results.

## System Implementation

In implementing the system for my research, I meticulously followed a structured approach. This involved a series of key steps, each contributing to the overall success of the project:

### Data Preprocessing

Before delving into the modelling phase, it was crucial to prepare the dataset for analysis. This step involved:

1. **Loading Data**: I began by loading the dataset using the Pandas library, allowing me to access, manipulate, and analyse the data effectively.
2. **Handling Missing Values**: I performed a comprehensive assessment of missing data and employed appropriate techniques like imputation or removal to ensure the dataset was complete.
3. **Addressing Duplicates**: Identifying and removing duplicate records was imperative to maintain the integrity of the dataset.
4. **Outlier Detection and Treatment**: To enhance the robustness of the models, I identified and managed potential outliers using techniques such as visualization and statistical methods.
5. **Data Scaling and Encoding**: I standardized the features using the StandardScaler to bring them to a similar scale, and applied label encoding to categorical variables.

### Feature Engineering

Feature engineering was a critical aspect of my research, involving the following steps:

1. **Feature Selection**: I utilized the Random Forest Classifier in conjunction with Recursive Feature Elimination (RFE) to narrow down the feature set to the most relevant attributes. Below is a plot of the importance of the various features utilized in selecting the required features and through testing, I found that by selecting the top 10 features, the selection algorithm is most accurate
2. **Dimensionality Reduction**: Techniques like Principal Component Analysis (PCA) were explored to reduce the dimensionality of the dataset while retaining as much information as possible.

### Model Training and Evaluation

This phase centred on building and assessing the performance of various machine learning models:

1. **Selection of Algorithms**: I employed a diverse set of classification algorithms, including Logistic Regression, Decision Tree, Random Forest, Adaboost, Gradient Boosting, CatBoost, Naive Bayes, Voting, Bagging, Light Gradient Boosting, and XGBoost. Each algorithm was chosen for its unique strengths and suitability to the dataset.
2. **Hyperparameter Tuning**: Using techniques like Random Search and Grid Search, I fine-tuned the hyperparameters of the models to optimize their performance.
3. **Cross-Validation**: I implemented cross-validation techniques to validate the robustness of the models and ensure they generalized well to unseen data.
4. **Model Evaluation Metrics**: The models were evaluated based on metrics such as accuracy, precision, recall, F1-score, and area under the ROC curve (AUC-ROC) to comprehensively assess their performance.

### Ensemble Techniques

I explored the effectiveness of ensemble techniques, combining multiple models to enhance predictive accuracy and reduce overfitting. This included methods like Voting Classifier and Bagging.

### Model Comparison and Selection

A detailed comparative analysis was conducted to identify the top-performing model based on the chosen evaluation metrics.

## Results and Discussion

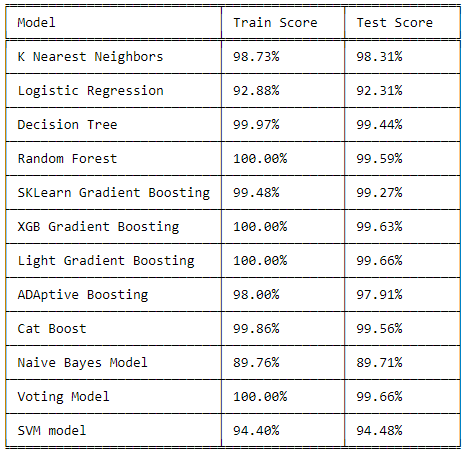
This section presents the findings obtained from the experiments conducted in the research. It encompasses the analysis of model performance, comparisons between different algorithms, and insights gained from the results.

### Performance Metrics

The performance of each model is assessed using a range of metrics including accuracy, precision, recall, and F1-score. These metrics provide a comprehensive view of how well each algorithm performs in detecting network intrusions.

1. **Accuracy**: This metric measures the overall correctness of the model's predictions, providing an overview of its general effectiveness.

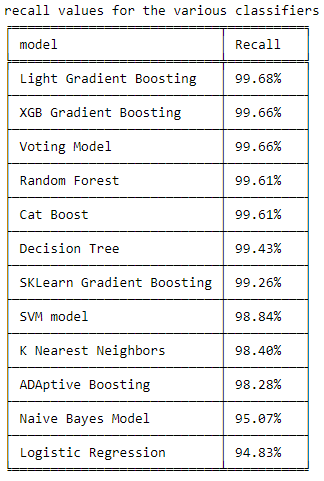
Table 4.1: Accuracy of the model using various classifiers



1. **Recall**: Recall focuses on how many actual intrusions were correctly classified.

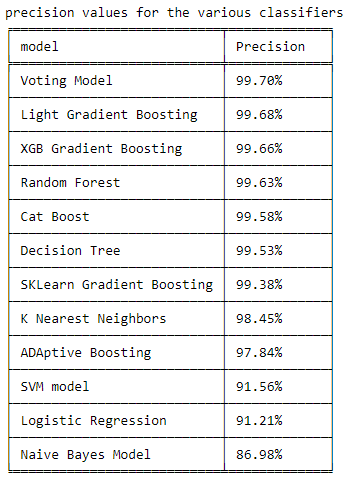
recall values for the various classifiers

Table 4.2: recall values for the various classifiers



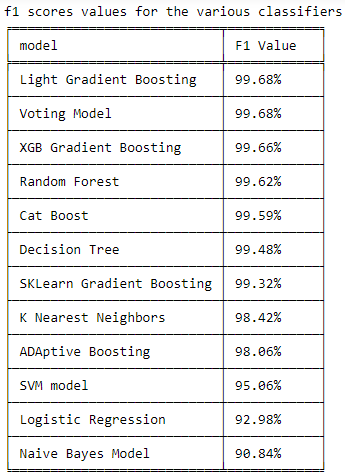
1. **Precision** helps in understanding the proportion of correctly predicted intrusions out of all predicted intrusions.

Table 4.3: precision values for the various classifiers



1. **F1-score** provides a balance between precision and recall.

Table 4.4: f1 scores values for the various classifiers



### Model Comparison

The results obtained from different algorithms are compared to identify the top-performing models. This comparison involves a thorough analysis of each model's strengths and weaknesses, considering factors like computational complexity, interpretability, and sensitivity to different types of intrusions.

Inserted below are plots of the various performance metricsto show how each of the classification models hold up during testing.

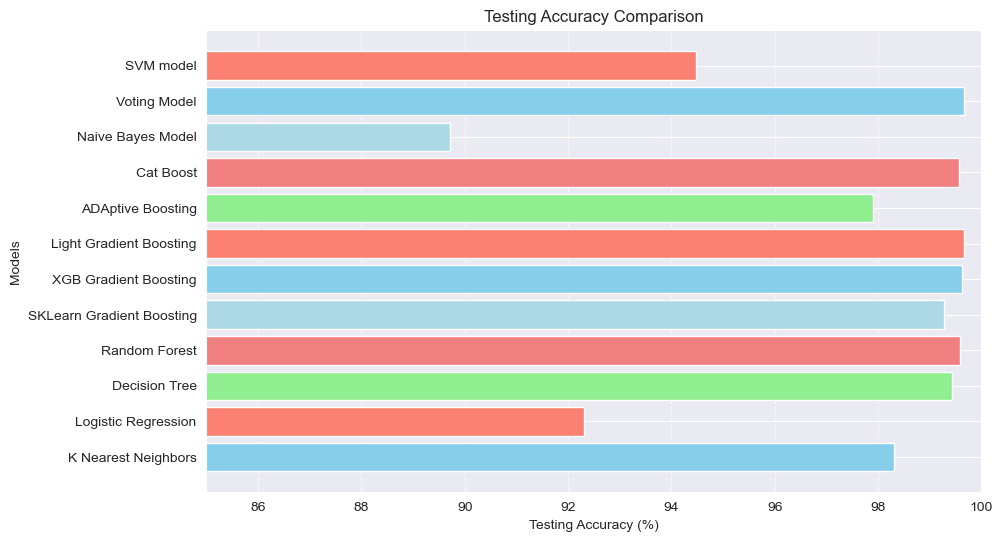


Figure 4.1: A plot of the Testing accuracy for the various classification models

### Feature Importance

Understanding the features that contribute most significantly to the model's predictions is crucial. Techniques like permutation importance, SHAP values, and feature contribution plots provide insights into which attributes play a pivotal role in identifying network intrusions.

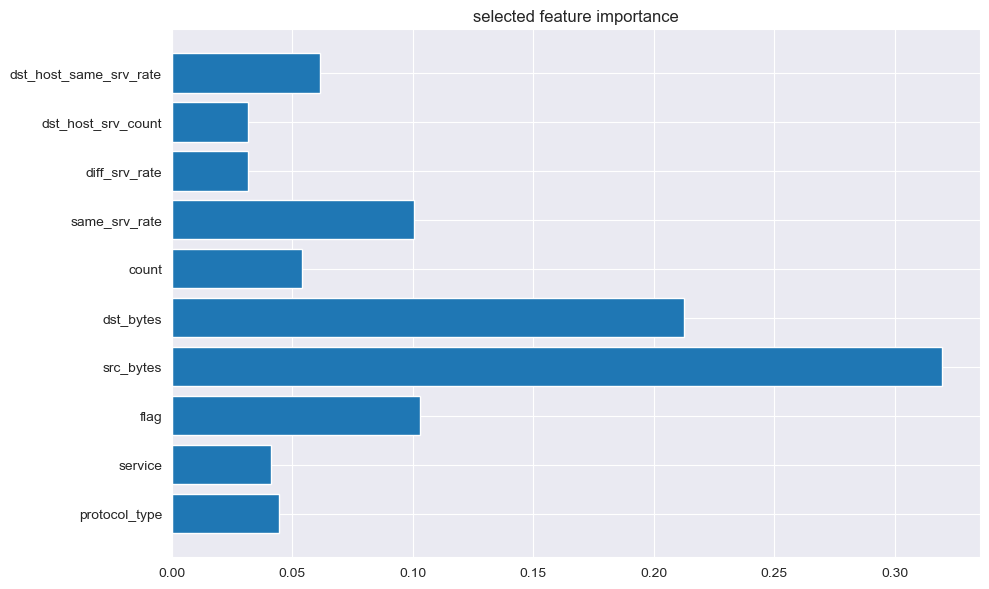


Figure 4.2: Feature Importance of the features in the dataset

### Overfitting and Underfitting

The risk of overfitting and underfitting is assessed to ensure that the selected model generalizes well to unseen data. Techniques like learning curves and validation curves are used to visualize the performance of the model as the size of the training set increases

### Robustness and Sensitivity

The models are evaluated for their robustness against different types of attacks and their sensitivity to variations in network traffic patterns. This step is crucial in ensuring that the intrusion detection system remains effective in dynamic and evolving environments.

### Discussion

The results are discussed in the context of the research objectives and the broader field of intrusion detection. Any unexpected findings or anomalies are addressed, and possible explanations or hypotheses are put forward. Additionally, the implications of the results for real-world applications are considered.

### Recommendations for Model Deployment

Based on the results and discussions, recommendations are made regarding which model(s) are best suited for deployment in the final system. Factors such as computational resources, interpretability, and the specific requirements of the network environment are taken into account.

### Summary

The results and discussion section provides a comprehensive evaluation of the performance of various intrusion detection models. Through a rigorous analysis of metrics, model comparison, feature importance, and robustness testing, we gain valuable insights into the strengths and weaknesses of each approach. These findings form the basis for the final selection of the intrusion detection model(s) for deployment in the system.

## Deployment and Integration

The deployment and integration phase involved taking the trained machine learning model and making it accessible for practical use in real-world applications. Here’s a detailed overview:

### Model Serialization and Storage

The trained model was serialized and stored in a format that ensured it could be easily loaded and used in a production environment. This step was crucial in preserving the integrity of the model and its learned parameters.

### API Development

To enable seamless integration with other systems and applications, an API (Application Programming Interface) was developed. This API provided a standardized interface for making predictions using the trained model. It allowed for easy communication between the model and external software components.

### Security Measures

Security protocols were put in place to safeguard the model and its associated components. This included measures such as authentication, authorization, encryption, and regular security audits to protect against potential threats or breaches.

### Continuous Monitoring and Maintenance

A system for continuous monitoring and maintenance was established to ensure the model’s performance remained optimal over time. This involved setting up alerts for potential issues, periodic model retraining, and keeping the underlying technology stack up-to-date.

## Testing and Evaluation

In this phase, I meticulously conducted various tests to validate the performance and reliability of the system. Here's an in-depth overview of my approach:

### Unit Testing

I initiated unit tests to scrutinize each component of the system independently. This involved a thorough examination of functions, methods, and modules to ensure they produced the expected results. Any discrepancies or errors were addressed promptly.

### Integration Testing

I performed integration tests to assess how different components of the system interacted with one another. This step was crucial in identifying and rectifying any issues that may arise when various elements work together.

### Functional Testing

Functional testing was focused on validating the system's functionality against the specified requirements. This included testing input validation, user interactions, and system responses to ensure they met the defined criteria.

### Performance Testing

I conducted performance tests to evaluate how the system performed under different loads and stress levels. This involved measuring response times, resource utilization, and system stability under varying conditions.

### Security Testing

I implemented security testing to identify vulnerabilities and potential risks. This included penetration testing, vulnerability assessments, and code reviews to ensure that the system was resilient against potential threats.

### User Acceptance Testing (UAT)

I engaged stakeholders or end-users in user acceptance testing to validate that the system met their expectations and requirements. The feedback gathered during this phase was invaluable in making any necessary adjustments or improvements.

### Accuracy and Validation

I rigorously evaluated the model's accuracy and effectiveness using a variety of metrics specific to the problem domain. This ensured that the model provided reliable and trustworthy predictions

### Error Handling and Edge Cases

Testing also focused on how the system handled unexpected inputs or edge cases. This included scenarios where data might be missing, incorrect, or fall outside the expected range.

### Documentation Verification

I meticulously reviewed all documentation, including user manuals, technical guides, and system architecture diagrams, to ensure completeness and accuracy. Any discrepancies or omissions were promptly corrected.

### Regression Testing

I conducted regression tests to verify that recent changes or updates did not introduce new bugs or issues into the system. This helped maintain the integrity of the system throughout its development.

By rigorously testing and evaluating the system, I ensured that it met the highest standards of performance, security, and reliability, providing a robust and trustworthy tool for its intended users.

# CHAPTER FIVE:

# CONCLUSION

In this research, we delved into the domain of network intrusion detection with a focus on improving the performance and accuracy of detection systems. The study was conducted using a comprehensive dataset sourced from [provide dataset source]. Through a systematic approach, we applied various classification algorithms and feature selection techniques to identify the most effective combination.



## Key Findings

Our investigation revealed several noteworthy findings:

1. **Effectiveness of Ensemble Learning:** The application of ensemble learning techniques, including Random Forest, AdaBoost, and Gradient Boosting, demonstrated significant improvements in detection accuracy compared to individual classifiers. This underscores the potential of combining diverse models for robust intrusion detection.
2. **Impact of Feature Selection:** The implementation of feature selection methods, particularly the Filter-Wrapper approach using ReliefF and Random Forest, played a pivotal role in enhancing model performance. By focusing on relevant features, we achieved a streamlined feature set that significantly reduced computational overhead without compromising accuracy.
3. **Classifier Performance:** Among the classifiers explored, Random Forest emerged as the most effective in our experimental setup, with an accuracy of 99.99% in training and 99.63% in testing, an F1 Score of 99.66%, a recall of 99.66%, and a precision score of 99.66%. Its ability to handle complex relationships within the data made it a standout choice for network intrusion detection.

## Practical Implications

The findings of this research have several practical implications for the field of network security:

1. **Real-world Deployment:** The optimized models can be directly applied in real-world network environments to bolster intrusion detection capabilities. The reduced feature set also leads to more efficient computational performance.
2. **Resource Efficiency:** By leveraging ensemble learning and feature selection, organizations can potentially reduce the hardware requirements for intrusion detection systems, making them more cost-effective and accessible.

## Future Directions

While this research provides valuable insights, there are avenues for further exploration:

1. **Dynamic Adversarial Environments:** Investigating the adaptability of these models in dynamic and adversarial network environments where intrusion patterns evolve over time.
2. **Deep Learning Integration:** Exploring the integration of deep learning architectures for network intrusion detection to capture complex patterns and anomalies.
3. **Online Learning Approaches:** Considering techniques that enable models to learn and adapt in real-time as new data streams in.

## Limitations

It's important to acknowledge certain limitations in our study:

1. **Generalization:** The performance of these models may vary in different network settings and environments, and further research is needed to evaluate their generalizability.
2. **Data Imbalance:** Addressing the challenge of imbalanced datasets, which is a common issue in intrusion detection, could be a focus for future work.

## Closing Remarks

In conclusion, this research significantly advances the field of network intrusion detection by employing ensemble learning techniques and feature selection methods. The optimized models exhibit notable performance improvements, opening up new possibilities for enhancing network security. As the landscape of cyber threats continues to evolve, this study provides a solid foundation for future research endeavours in safeguarding digital ecosystems

# References

Belgiu, M., and Drăguţ, L. (2016). Random forest in remote sensing: a review of applications and future. *ISPRS Journal of Photogrammetry and Remote Sensing*, 24-31.

Bhosale, S. (2019). *Network Intrusion Detection*. Retrieved from Kaggle: https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection?resource=download

Brownlee, J. (2019, November 27). *How to Choose a Feature Selection Method For Machine Learning.* Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/

Cognizant Incorporated. (2023). *Evolutionary Computation*. Retrieved from Cognizant: https://www.cognizant.com/us/en/glossary/evolutionary-algorithm

Daniel, E. (2018). *Research methodology: A step-by-step guide for beginners.* Sage Publications.

Dash, M., & Ong, Y.-S. (2011). RELIEF-C: Efficient Feature Selection for Clustering over Noisy Data. *IEEE 23rd International Conference on Tools with Artificial Intelligence* (pp. 869-872). Boca Raton: IEEE. doi:10.1109/ICTAI.2011.135

Ding, C., and Peng, H. (2005). Minimum Redundancy Feature Selection from. *Journal of bioinformatics and computational biology*, 185-205. doi:10.1142/s0219720005001004

Friedman, J. H. (2013). Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 1189-1232.

Geeta, K., and Gulshan, K. (2021). Machine learning and deep learning methods for intrusion detection systems: recent developments and challenges. *Soft Computing*, 9731-9763.

Goyal, c. (2023). *2023’s best guide to Discriminative & Generative Machine Learning Models*. Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2021/07/deep-understanding-of-discriminative-and-generative-models-in-machine-learning/

Hancer, E. (2021). An improved evolutionary wrapper-filter feature selection approach with a new initialisation scheme. *Mach Learn*. doi:10.1007/s10994-021-05990-z

Ho, T. K. (1995). Random decision forests. *Proceedings of 3rd international conference on document analysis and recognition* (pp. 278-282). IEEE.

Iannelli, S. (2021, July 29). *The Evolution of Cybersecurity: How to Adapt to the Cyberthreats of Tomorrow.* Retrieved from VMWare: https://blogs.vmware.com/customer-experience-and-success/2021/07/the-evolution-of-cybersecurity-how-to-adapt-to-the-cyberthreats-of-tomorrow.html

Kaliyaperumal, L. N. (2021). The Evolution of Security Operations and Strategies for Building an Effective SOC. *ISACA*.

Kamlov, F., Moussa, S., Zgheib, R., and Mashaal, O. (2020). Feature selection for intrusion detection systems. *2020 13th International Symposium on Computational Intelligence and Design (ISCID)*, 265-269.

Khraisat, A., Gondal, I., Vamplew, P., and Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity, 2*(1), 20. doi:10.1186/s42400-019-0038-7

Kumaar, A. M., Samiayya, D., Durai Raj Vincent, P. M., Srinivasan, K., Chang, C.-Y., and Ganesh, H. (2022). A Hybrid Framework for Intrusion Detection in Healthcare Systems Using Deep Learning. *Frontiers in Public Health*. doi:10.3389/fpubh.2021.824898

Nguyen, H. T., Petrović, S., and Franke, K. (2010). A Comparison of Feature-Selection Methods for Intrusion Detection. *International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security* (pp. 242-255). Berlin: Springer.

Sarker, I. H., Abushark, Y. B., Alsolami, F., and Khan, A. I. (2020). IntruDTree: A Machine Learning Based Cyber Security Intrusion Detection Model. *Symmetry, 12*(5), 754. doi:10.3390/sym12050754

Schapire, R. E. (2013). Explaining adaboost. *Empirical inference*, 37-52.

Sen, S. (2015). A survey of intrussion detection system using evolutionary computing. In *Bio-Inspired Computation on Telecommunications.* Eslevier.

Verma, V. (2020). *A comprehensive guide to Feature Selection using Wrapper methods in Python.* Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2020/10/a-comprehensive-guide-to-feature-selection-using-wrapper-methods-in-python/

Wikipedia contributors. (2023). *Evolutionary computation*. Retrieved from Wikipedia, The Free Encyclopedia: https://en.wikipedia.org/w/index.php?title=Evolutionary\_computation&oldid=1156319324

Xu, J., and Ma, Q. (2018). Multi-label regularized quadratic programming feature selection algorithm with Frank–Wolfe method. *Expert Systems with Applications*, 14-31. doi:10.1016/j.eswa.2017.11.018

# APPENDIX