AN INTRUSION DETECTION SYSTEM WITH FEATURE SELECTION AND ENSEMBLE MACHINE LEARNING MODELS

BY

OMOROGIE EMMANUEL CHUKWUKA

PSC1808936

DEPARTMENT OF COMPUTER SCIENCE

FACULTY OF PHYSICAL SCIENCES

UNIVERSITY OF BENIN

BENIN CITY

SEPTEMBER 2023

AN INTRUSION DETECTION SYSTEM WITH FEATURE SELECTION AND ENSEMBLE MACHINE LEARNING MODELS

BY

OMOROGIE EMMANUEL CHUKWUKA

PSC1808936

A PROJECT SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE, FACULTY OF PHYSICAL SCIENCES, UNIVERSITY OF BENIN, BENIN CITY IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR AWARD OF BACHELOR OF SCIENCE (B.Sc.) IN COMPUTER SCIENCE

SEPTEMBER 2023

ATTESTATION

I, OMOROGIE EMMANUEL CHUKWUKA, an undergraduate student in the Department of Computer Science, Faculty of Physical Sciences, University of Benin, Edo State, with matriculation number PSC1808936, hereby declare that the work I have submitted is entirely original to me, I attest to have done this project in partial fulfilment of the requirements for the award of Bachelor of Science (B.Sc.) in Computer Science, University of Benin.

E. C. IGODAN, PhD
(Project Supervisor)

Signature/Date

CERTIFICATION

This is to verify that OMOROGIE EMMANUEL CHUKWUKA, an undergraduate student in the Department of Computer Science, Faculty of Physical Sciences, University of Benin, Edo State, with matriculation number PSC1808936 did this project in partial fulfilment of the requirements for the award of Bachelor of Science (B.Sc.) in Computer Science, University of Benin, Under my supervision.

E. C. IGODAN, PhD

(Project Supervisor)

Signature/Date

APPROVAL

This project report prepared by OMOROGIE EMMANUEL CHUKWUKA, an undergraduate student in the Department of Computer Science, Faculty of Physical Sciences, University of Benin, Edo State, with matriculation number PSC1808936 is hereby approved in partial fulfilment of the requirements for the award of Bachelor of Science (B.Sc.) in Computer Science.

PROF. (MRS.). A. O. EGWALI
(Head of Department)

Signature/Date

DEDICATION

I would like to thank God. He has given me strength and encouragement throughout all the challenging moments of completing this project. I am truly grateful for His unconditional and endless love, mercy, and grace.

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to the Almighty for His boundless blessings, guidance, and unwavering support throughout this academic journey. I extend my heartfelt appreciation to the Head Of Department (HOD), Computer Science, Prof. (Mrs.). A. O. Egwali. I owe a debt of gratitude to my project supervisor E. C. Igodan, PhD, for their invaluable guidance, mentorship, and unwavering support throughout this research endeavour.

I also wish to extend my thanks to the esteemed lecturers in the Department of Computer Science, Prof.(Mrs) A.O. Egwali, Prof (Mrs) S. Konyeha, Mrs A. R. Usiobaifo, Mr J. Okhuoya, Prof (Mrs) V. A. Akwukwuma, Prof (Mrs) F. A Egbokhare, Prof A. A Imianvan, Prof G. O Ekuobase, Prof F. I. Amadin, Prof K. C Ukaoha, Engr. Dr. F. A. U. Imouokhome, Dr (Mrs) V. I. Osubor, Dr F. O. Chete, Mr P. E. B. Imiefoh, Mr E. E. Obasohan, Dr (Mrs) G. Aziken, Mr E. Nwelih, Mr S. O. P. Oliomogbe, E. C. Igodan, PhD, Dr F. O. Oliha, Dr E. P. Ebietomere, Dr (Mrs) R. O. Osaseri, Mr K. O. Otokiti, Miss I. O. Usiosofe, Mrs T. Agenmomen, Mr F. Osagie, Mr I. E. Obayagbona, Mrs R. I. Izevbizua, Mr D.N. Idehen, Mrs J. I. Adun. Your lectures, discussions, and academic insights have broadened my knowledge and enhanced my academic journey.

ABSTRACT

This project delves into the realm of network security through the development and evaluation of an Intrusion Detection System (IDS) that harnesses the power of feature selection and ensemble models. In today's digitally interconnected world, the protection of networks against malicious activities and cyber threats is of paramount importance. IDSs serve as the first line of defence in identifying and mitigating these threats, making their enhancement a critical area of research.

This study demonstrates the efficacy of combining feature selection methods with ensemble models to fortify IDS capabilities. By conducting an extensive review of existing IDS methodologies, collecting network data, and employing ensemble techniques, this research showcases that this approach surpasses traditional feature selection methods not only in accuracy but also in computational efficiency.

The use of ensemble models has rendered the IDS more resilient, adaptable to diverse attack patterns, and robust against the inherent noise in network data. These findings contribute significantly to the field of cybersecurity, shedding light on the potential of uniting feature selection and ensemble models to optimise IDS performance.

The practical implications of this research extend to organisations and institutions seeking to bolster their network security posture. Elevating IDS accuracy and efficiency is a pivotal step towards safeguarding networks against the continually evolving landscape of cyber threats.

TABLE OF CONTENTS

ATTESTATION ..... Ⅰ
CERTIFICATION ..... Ⅱ
APPROVAL ..... Ⅲ
DEDICATION ..... Ⅳ
ACKNOWLEDGEMENT ..... Ⅴ
ABSTRACT ..... Ⅵ
TABLE OF CONTENTS ..... Ⅶ
LIST OF TABLES ..... Ⅸ
LIST OF FIGURES ..... Ⅹ
CHAPTER ONE **1**
INTRODUCTION **1**
1.1 Background of the Study 1
1.2 Statement Of The Problem 3
1.3 Aims and Objectives 3
1.4 Methodology 3
1.5 Scope of the Study 4
1.6 Significance of Study 4
CHAPTER TWO **5**
LITERATURE REVIEW **5**
2.1 Overview 5
2.2 Knowledge Discovery Database and Data Mining 5
2.3 Introduction to Intrusion Detection System 7
2.3.1 Host Intrusion Detection System 8
2.3.2 Network Intrusion Detection System 8
2.3.3 Signature-Based Detection 9
2.3.4 Anomaly-Based Detection 9
2.4 Introduction to Machine Learning 10
2.5 Feature Selection 11
2.6 Ensemble Methods 15
2.7 Classification Algorithms 18
2.8 Review of Literature 21
CHAPTER THREE **27**
RESEARCH METHODOLOGY **27**
3.1 Introduction 27
3.2 Data 27
3.2.1 Data Preprocessing 27
3.2.2 Scaling 28
3.3 Feature Selection Methods 28
3.3.1 Chi Square 28

3.3.2 Information Gain....29
3.4 Ensemble Methods....30
3.4.1 Bagging....30
3.4.2 Adaboost....30
3.5.1 Voting....31
3.5 Classification Algorithms....32
3.5.1 Artificial Neural Network....32
3.5.2 Support Vector Machines....32
3.5.3 Logistic Regression....33
3.6 Performance Evaluation Methods....33
3.6.1 Confusion Matrix....33
3.6.2 Accuracy....33
3.6.3 Precision....34
3.6.4 Recall....34
3.6.5 F1-Score....34
CHAPTER FOUR....35
SYSTEM DESIGN AND IMPLEMENTATION....35
4.1 Overview....35
4.2 System Requirements....35
4.2.1 Hardware Requirements....35
4.2.2 Software Requirements....35
4.3 Development Tools....35
4.4 Results....36
4.4.1 Data....36
4.4.3 Discussions....36
CHAPTER FIVE....41
CONCLUSION, RECOMMENDATION AND FUTURE WORKS....41
5.1 Conclusion....41
5.2 Recommendation....42
5.3 Future Works....42
REFERENCES....44
APPENDIX A....46
SOURCE CODE....46

LIST OF TABLES

2.1 Review of Relevant Literature.....22
4.1 Evaluation of Classifiers.....37

LIST OF FIGURES

2.1 Knowledge Discovery process....7
2.2 Feature Selection....11
2.3 Feature Selection Models....12
2.4 Filter Method....13
2.5 Wrapper Method....14
2.6 Embedded Method Flowchart....15
2.7 Bagging Ensemble Method....16
2.8 Boosting Ensemble Method....17
2.9 Stacking Ensemble Method....17
2.10 Decision Tree....18
2.11 Random forest....19
2.12 Support vector machines....20
2.13 Artificial Neural Networks....20
2.14 Logistic Regression....21
4.1 Confusion matrix for Linear Regression Classifier....37
4.2 ROC curve for Linear Regression Classifier....37
4.3 Confusion matrix for Support Vector Machines Classifier....38
4.4 ROC curve for Support Vector Machines Classifier....38
4.5 Confusion matrix for Multilayer Perceptron Classifier....38
4.6 ROC curve for Multilayer Perceptron Classifier....38
4.7 Confusion matrix for Bagging Ensemble Classifier Classifier....39
4.8 ROC curve for Bagging Ensemble CLassifier Classifier....39
4.9 Confusion matrix for Adaboost Ensemble Classifier....39
4.10 ROC curve for Adaboost Ensemble Classifier Classifier....39
4.11 Confusion matrix for Voting Ensemble Classifier....40
4.12 ROC curve for Voting Ensemble Classifier....40

CHAPTER ONE
INTRODUCTION

1.1 Background of the Study

In recent times, the utilisation of the internet has been on a constant rise. It has become an important dependency in several domains, including education, business, healthcare, and more. However, a crucial concern arises: the information obtained via the internet requires safeguarding (Amarudin *et al.,* 2020). Intruders can perform different techniques to crack the system data like password hacking, peer to peer attack, sniffing attack, DOS attack, etc. Conventional methods like firewalls and certain authentication mechanisms have been employed as initial layers of defence for network security but they have their shortcomings. Intrusion Detection System (IDS) can be adopted as a secondary protective measure by identifying the intrusions and preventing systems from getting damaged (Haripriya & Jabbar, 2018).

Intrusion detection system (IDS) monitors network traffic and devices to identify malicious activity, potentially suspicious behaviour, or breaches of security policies. An Intrusion Detection System serves three vital security roles: identifying, observing, and reacting to unauthorised activities (Sreenath & Udhayan, 2015). For decades, researchers have been dedicated to the inquiry of creating IDSs that are proficient, productive, and capable of adaptation. They have been investigating various methodologies to determine their applicability within this realm of research (Sen, 2015).

Firewalls and other conventional methods are extensively used as network safety mechanisms but are not all-encompassing in their ability to stop various forms of attacks and their mutations, as they rely on predefined security policies. They lack the capacity to provide protection if a network breach occurs and are unable to inspect the contents of legitimate traffic. Firewalls are also deficient in their ability to determine the legitimacy and normalcy of traffic, which is where the Intrusion Detection System comes into play. In comparison to firewalls, intrusion detection systems are notably more dynamic. Nevertheless, it's important to note that intrusion detection

systems cannot be regarded as a complete substitute for firewalls; rather, they complement each other effectively (Sreenath & Udhayan, 2015).

Machine learning is the branch of computer science involved with empowering computers to learn from data without requiring explicit programming. Machine learning and artificial intelligence (AI) provides a concise overview of algorithms that enable us to make data predictions. As a result, AI systems acquire knowledge from existing data. ML algorithms are categorised as supervised learning, unsupervised learning, and reinforcement learning (Haripriya & Jabbar, 2018).

In supervised learning a model is trained using a labelled dataset, which consists of input-output pairs. The algorithm learns to map inputs to corresponding desired outputs based on the provided training examples. Supervised learning is categorised into regression and classification tasks. Regression tasks try to predict a continuous output or a numeric value like predicting house prices, temperature, or stock prices. Meanwhile, classification tasks are involved with assigning input data to a specific category or class from a predefined set. Intrusion detection systems are considered a classification problem. Various supervised learning methods like Artificial Neural Networks, Support Vector Machines, and Naive Bayes have been proposed for the creation of intrusion detection models (Belavagi & Muniyal, 2016). However, there may still be room for improvement in these models that may lead to better model accuracy.

Feature selection constitutes a crucial aspect of machine learning, aimed at diminishing data dimensionality. Feature selection methods are usually categorised into the filter method and the wrapper method. In the filter method, features are chosen based on their scores obtained from various statistical tests assessing their relevance through correlation with the dependent or outcome variable. Conversely, the wrapper method identifies a subset of features by gauging their utility in conjunction with the dependent variable (Taher *et al.,* 2019). Feature selection reduces dimensionality and improves accuracy and efficiency of machine learning models.

An ensemble technique is a potent approach within machine learning, combining the forecasts of several separate models to produce a conclusive prediction or choice. It involves methodically

creating and fusing numerous models, like classifiers or regressors, to address a particular computational intelligence challenge. Its typical purpose is to enhance the predictive accuracy of a model or decrease the possibility of haphazardly selecting an ineffective model (Yang *et al.,* 2010).

1.2 Statement Of The Problem

Every computer system is constantly vulnerable to unauthorised access and intrusion, but the risk is even greater when it involves private and sensitive data. The role of Intrusion Detection Systems is crucial in identifying various forms of attacks and safeguarding the network infrastructure (Sreenath & Udhayan, 2015). Conventional security methods like firewalls, user authentication, and data encryption lack the capability to completely ensure network security, primarily due to the rapid evolution of intrusion techniques (Mohammadi *et al.,* 2019). The presence of redundant and irrelevant elements within the data has led to an issue in the categorization of network traffic by machine learning algorithms. This problem has resulted in a delay in the classification process and hindered the ability to achieve precise categorizations (Othman *et al.,* 2018).

1.3 Aims and Objectives

The aim of the project is to create an Intrusion Detection System using an ensemble machine learning model. The objectives are to:

a. extract relevant features from the dataset using an ensemble feature selection technique.
b. build classification models with logistic regression (LR), support vector machines (SVM) and artificial neural networks (ANN), and ensemble models based on these classifiers;
c. implement the proposed models in (b), and
d. assess and compare the models' performance using standard evaluation metrics.

1.4 Methodology

The publicly available NSL-KDD dataset (Tavallaee *et al.,* 2009) was used in the creation of the proposed model. The methodology of the study consists of five steps:

In the first step, the dataset is preprocessed and then divided into training and testing data.

In the second step, the relevant features are extracted using Chi Square and Information Gain feature selection techniques.

In the third step, support vector machines, artificial neural networks and logistic regression algorithms are trained using the data obtained from the above step.

In the fourth step, the above classifiers are combined using bagging, adaboost and voting ensemble methods.

In the last phase, the effectiveness of the proposed model is evaluated utilising standard assessment measures such as accuracy, precision, recall, and F1-score. The outcomes are subsequently compared with the performance of each classifier to assess the most accurate.

1.5 Scope of the Study

The scope of this study includes Intrusion Detection Systems (IDS), machine learning. supervised learning, classification algorithms and ensemble methods.

1.6 Significance of Study

This research delves into the importance of employing ensemble methods within Intrusion Detection Systems (IDS). By fusing distinct intelligent models, ensemble techniques strive to augment the capability of identifying and thwarting malicious behaviours in networks. The study focuses on the shortcomings inherent in conventional network security methods and standalone classification models and feature selection techniques. It also delves into the real-world application of ensemble strategies within IDS contexts. Overall, this investigation makes a valuable contribution to the advancement of Intrusion Detection Systems.

CHAPTER TWO
LITERATURE REVIEW

2.1 Overview

This chapter serves as an introductory overview of key concepts related to data mining, intrusion detection, machine learning algorithms, and ensemble methods. It aims to equip readers with a foundational understanding of intrusion detection, its significance in network security, and the various approaches employed in this field.

The proposed algorithm comprises four phases. Firstly, the IDS dataset goes through preprocessing and data mining in the initial stage, which refines and readies the data for subsequent analysis. In the second stage, feature selection techniques are applied to the dataset to eliminate unnecessary and redundant data, thereby enhancing classification performance. In the third stage, classification models are trained using the processed data to gain the required insights. Finally, the models undergo testing and evaluation to verify their accuracy in classification.

Following sections in this chapter delve deeper into these ideas by providing a comprehensive examination of pertinent literature. These sections specifically emphasise a range of techniques that have proven effective in the context of intrusion detection systems.

2.2 Knowledge Discovery Database and Data Mining

Knowledge Discovery in Databases (KDD) is a complex process that revolves around the identification of meaningful and comprehensible patterns within data. KDD consists of a number of iterative stages, including data preparation, pattern discovery, knowledge assessment, and refinement (Fayyad *et al.,* 1996). The seven steps involved in the KDD process are:

1. **Understanding the application domain and identifying customer goals:** At the outset, a profound understanding of the subject matter is essential for effective implementation. This stage involves determining how the transformed data and patterns, derived through

data mining, can yield valuable insights. The foundational importance of this step cannot be emphasised enough, as missteps here could lead to erroneous interpretations with adverse consequences for end-users.

2. **Data selection:** Once goals are established, the next step involves meticulously selecting and organising collected data into meaningful subsets. This selection process hinges on factors such as data availability, accessibility, significance, and quality. These criteria bear immense significance in data mining, as they lay the groundwork and substantially influence the types of data models that will be constructed.
3. **Data cleaning and preprocessing:** This phase involves identifying missing data and eliminating noisy, redundant, and low-quality data from the dataset to bolster its reliability and effectiveness. Specific algorithms are applied to detect and remove unwanted data based on attributes pertinent to the specific application.
4. **Data transformation and feature selection:** Here, the focus shifts to selecting the most relevant and informative features (variables or attributes) from the dataset to reduce data complexity while retaining crucial characteristics. Opting for the most significant features enhances model performance, reduces training time, and mitigates overfitting. Several algorithms are available for this purpose.
5. **Data mining:** In this phase, specialised algorithms are employed to extract significant patterns from processed data, which subsequently aid in building prediction models. This analytical tool harnesses artificial intelligence, advanced numerical and statistical techniques, and specialised algorithms to uncover trends within the dataset. It serves as the foundational process for the entire Knowledge Discovery in Databases (KDD) methodology.
6. **Pattern Interpretation:** After extracting trends and patterns through multiple data mining techniques and iterations, it becomes essential to represent these findings visually, using formats such as bar graphs, pie charts, and histograms. These visual representations are crucial for analysing the impact of the data collected and processed in earlier stages, facilitating the evaluation of a specific data model's effectiveness within the given domain.
7. **Knowledge Discovery:** In the final stage of the Knowledge Discovery in Databases (KDD) process, the insights obtained from the preceding step are leveraged and presented

in visual formats such as tables, reports, and other mediums. This step plays a pivotal role in guiding the decision-making process within the relevant application or domain.

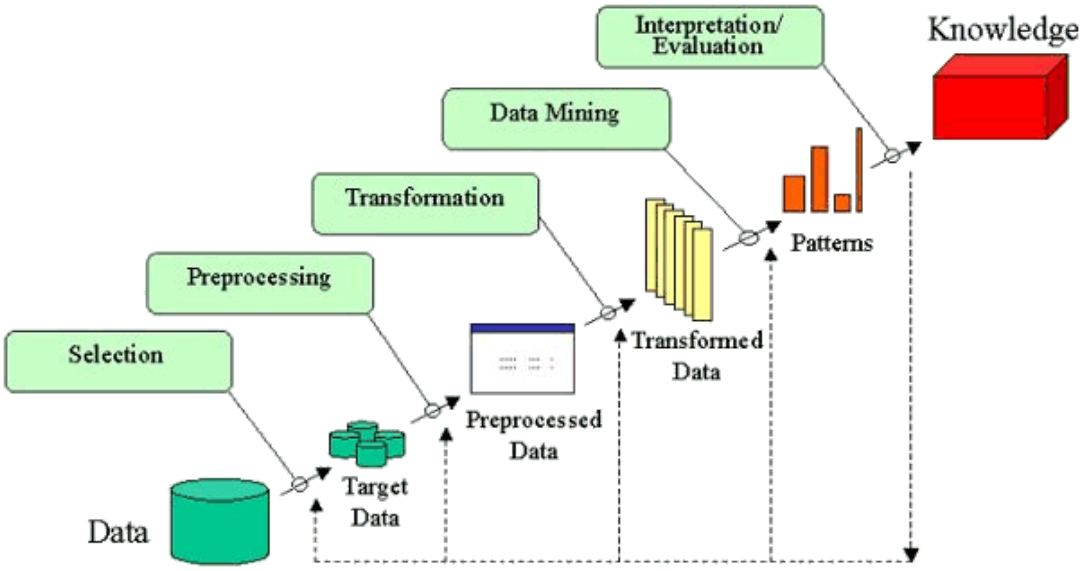

Figure 2.1. Knowledge Discovery process (https://www2.cs.uregina.ca/)

2.3 Introduction to Intrusion Detection System

An Intrusion Detection System (IDS) is a software or hardware solution designed to identify and flag malicious activities occurring on a specific computer or network. When an IDS detects vulnerabilities or suspicious behaviour, it promptly notifies the system administrator. Their primary role is to alert and raise warnings about any potentially harmful incidents within the system. They serve as a last line of defence, capable of capturing intrusions before they can compromise the systems connected to the network. As a result, IDS can be positioned at either the network's perimeter or at the individual host level. Intrusion Detection Systems (IDS) are integral components of network security, with a deep-rooted presence at the host level, making them vital for safeguarding network integrity.

Intrusion detection systems are categorised into two main types; Host Intrusion Detection System (HIDS) and Network Intrusion Detection System (NIDS).

2.3.1 Host Intrusion Detection System

Host Intrusion Detection Systems (HIDSs) are installed on specific endpoints, such as laptops, routers, or servers. HIDSs focus solely on monitoring the activity occurring on the host device, including both incoming and outgoing network traffic. HIDSs typically operate by periodically creating snapshots of critical operating system files and comparing these snapshots over time. If a HIDS detects any changes, such as unauthorised alterations to system configurations or log file modifications, it promptly alerts the security team.

2.3.2 Network Intrusion Detection System

Network Intrusion Detection Systems (NIDSs) play a critical role in monitoring both incoming and outgoing traffic across a network. They are strategically positioned at key locations within the network infrastructure. Typically, NIDSs are placed immediately behind the network's firewalls, which serve as the first line of defence at the network perimeter. This placement allows them to identify and flag any malicious traffic that manages to bypass the initial firewall protection. Additionally, NIDSs can be deployed within the network to detect insider threats or unauthorised access by hackers who have compromised user accounts. For example, NIDSs can be positioned behind internal firewalls in a segmented network to monitor traffic between different subnets.

To ensure that legitimate network traffic flows smoothly without disruption, NIDSs are often placed "out-of-band," meaning that network traffic doesn't directly pass through them. Instead of analysing the actual network packets, NIDSs examine copies of these packets. This approach allows them to inspect network traffic for signs of malicious activity without causing delays for legitimate traffic. There are two detection methods associated with NIDS; signature-based detection and anomaly-based detection.

2.3.3 Signature-Based Detection

Signature-based detection is a method used in intrusion detection systems (IDS) that involves the examination of network packets for specific attack signatures. These attack signatures consist of unique characteristics or behaviours that are associated with a particular type of threat.

In a signature-based IDS, there is a database that stores these attack signatures. When network packets are inspected, the IDS compares them against this database of signatures. If a network packet matches or corresponds to one of the stored attack signatures, the IDS raises an alert or flag to signal a potential intrusion or threat.

However, for signature-based IDS to remain effective, it is crucial to regularly update the signature database with new threat intelligence. This is because new cyberattacks are constantly emerging, and existing attack methods can evolve over time. As a result, if a brand new attack or a modified version of an existing attack is encountered before it has been analysed and added to the signature database, it can potentially go undetected by the signature-based IDS. Therefore, keeping the signature database up-to-date is essential for maintaining the security of the network making this detection method costlier and infeasible.

2.3.4 Anomaly-Based Detection

Anomaly-based Network Intrusion Detection Systems (NIDS) focus on understanding user behaviour. In this approach, it creates a model of normal activity for each user, flagging any deviations from this model as anomalies. The key advantage of using an anomaly-based NIDS is its ability to predict new and previously unknown attacks.

Anomaly-based IDS can be categorised into three types: statistical IDS, knowledge-based IDS, and machine learning-based IDS. Machine learning techniques, in particular, are popular due to their ability to handle noisy data and adaptability. Consequently, many researchers are actively exploring these methods. They often exhibit superior performance, characterised by low false alarm rates and a high detection rate compared to statistical and knowledge-based approaches.

However, machine learning-based techniques have limitations. One of these limitations is the need for manual feature extraction, which can be time-consuming and demanding. Additionally, they may not efficiently handle extensive and diverse datasets. Furthermore, they may struggle to detect multi-classification attacks, where attacks can fall into multiple categories simultaneously. These factors should be considered when selecting the most appropriate approach for implementing an anomaly-based IDS.

In many cases, security teams combine multiple intrusion detection systems and methods for a more comprehensive protection. NIDSs provide a broad view of network traffic, while HIDSs offer an additional layer of security around high-value assets. HIDSs are also valuable for identifying malicious activity originating from a compromised network node, such as the spread of ransomware from an infected device. This combined approach enhances the overall security posture of the network and helps organisations detect and respond to a wide range of cybersecurity threats.

2.4 Introduction to Machine Learning

Machine Learning (ML) is a field within computer science that empowers computers to acquire knowledge and improve their performance without requiring explicit, step-by-step programming. The concept of machine learning was first articulated by Arthur Samuel in 1959. ML revolves around the use of algorithms that enable predictions to be made based on data. ML systems have the capability to learn and adapt by analysing and deriving insights from the data they encounter.

Machine learning algorithms are categorised into three main types, namely, supervised learning, unsupervised learning, and reinforcement learning.

1. **Supervised Learning:** This type of ML involves algorithms that are trained on labelled data, where the correct outcomes are known. The algorithm learns to make predictions or classifications based on this labelled training data.
2. **Unsupervised Learning:** In unsupervised learning, algorithms work with unlabeled data, aiming to discover patterns, structures, or groupings within the data on their own, without explicit guidance.

3. **Reinforcement Learning:** Reinforcement learning is a type of ML where algorithms learn to make decisions through a trial-and-error process. They receive feedback in the form of rewards or penalties based on their actions, allowing them to optimise their behaviour over time.

2.5 Feature Selection

Feature selection is a fundamental concept in machine learning that significantly affects a model's performance. The choice of features used to train machine learning models has a substantial impact on their performance, as features that are only partially relevant or irrelevant can detrimentally affect results. The process of selecting features, whether done manually or automatically from a dataset, plays a crucial role in determining the predictive or output capabilities of the model, aligning it with the desired outcomes.

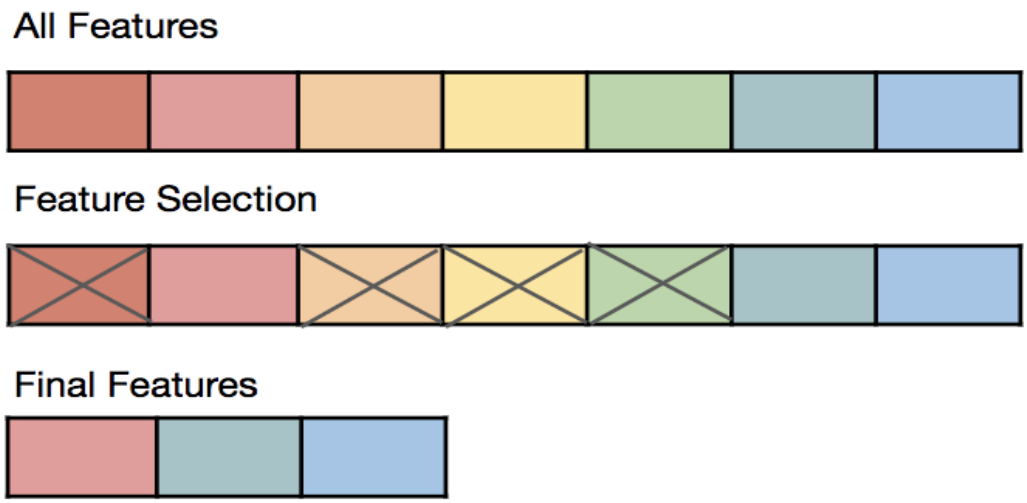

Figure 2.2: Feature Selection (https://www.kdnuggets.com/)

Feature selection techniques can be categorised into two main methods; supervised and unsupervised feature selection.

1. **Supervised Feature Selection:** Supervised feature selection is an approach to feature selection that leverages the output label or class information. This method utilises the target variables (output labels) to identify the variables that can enhance the efficiency and performance of the model.

2. **Unsupervised Models:** Unsupervised feature selection is a technique for selecting features that does not rely on the output label or class information. These methods are particularly useful for working with unlabeled data, where there is no predefined output label to guide the feature selection process.

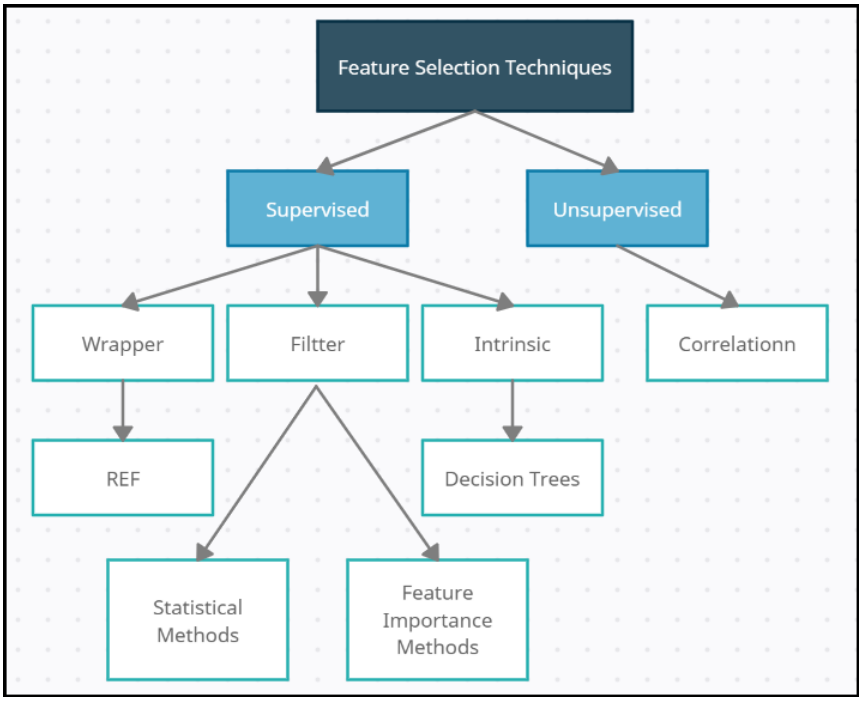

Figure 2.3: Feature Selection Models (https://medium.com/analytics-vidhya/)

Supervised feature selection is in turn classified into different types, including:

1. **Filter Method:** In this approach, features are removed based on their relationship with the output or their correlation with it. Correlation is employed to assess whether features

are positively or negatively linked to the output labels, and features are eliminated accordingly. Common methods used for this purpose include Information Gain, Chi-Square Test, Fisher's Score, and others.

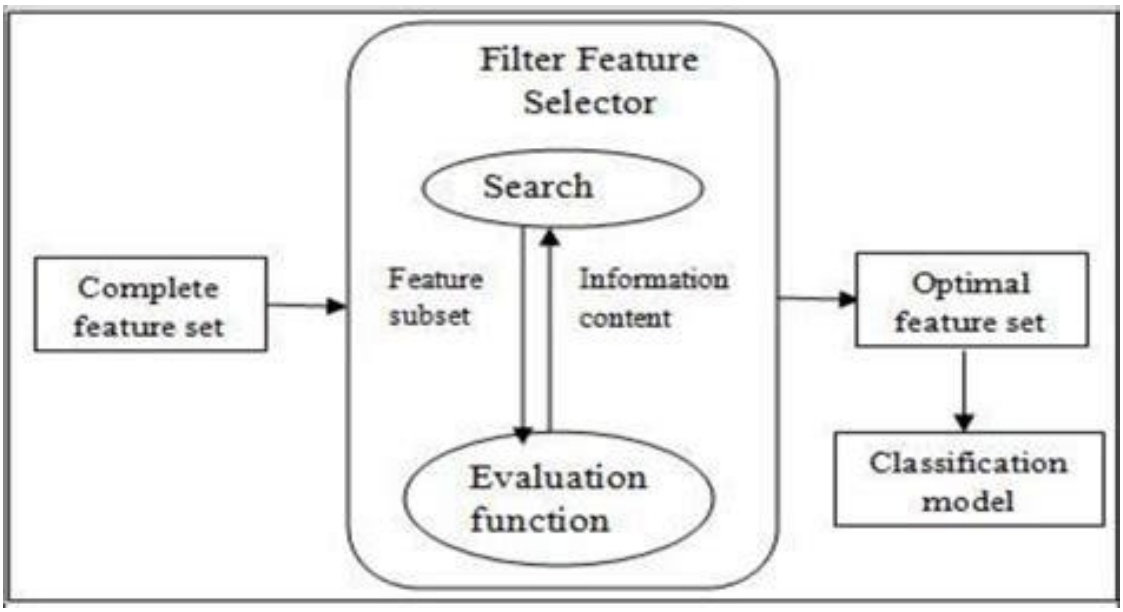

Figure 2.4: Filter Method (Revathy & Lawrance, 2017)

2. **Wrapper Method:** In this approach, we partition our data into subgroups and use them to train a model. Features are added or removed based on the output of the model, and the model is retrained iteratively. This method employs a greedy strategy to generate subsets of features and evaluates the accuracy of all possible combinations. Common techniques following this strategy include forward selection, backward elimination, and similar methods.

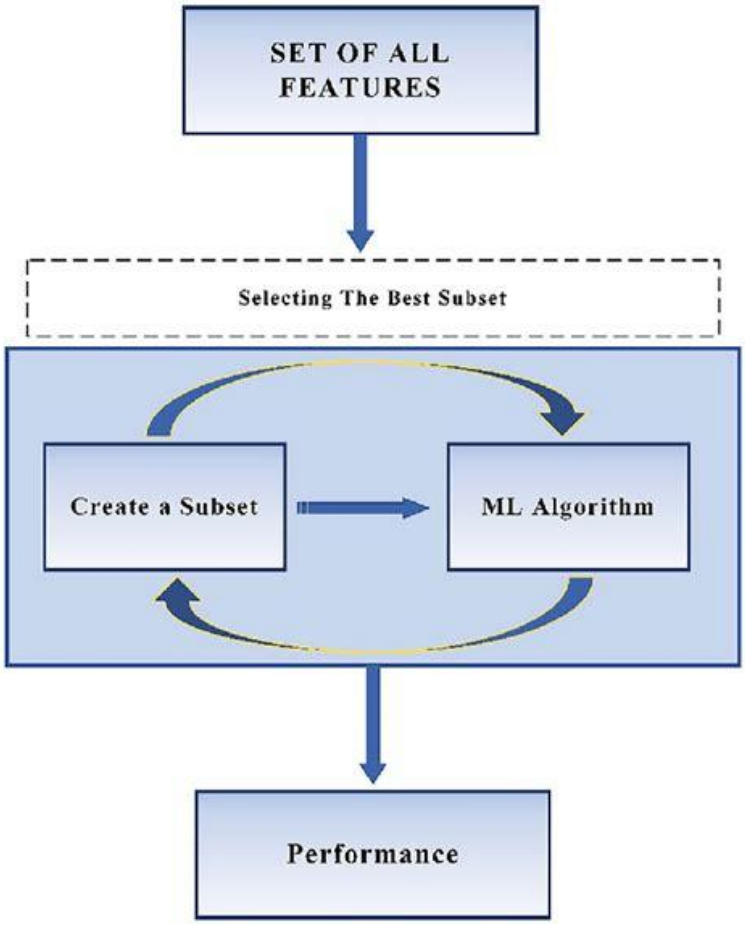

Figure 2.5: Wrapper Method

3. **Embedded Method:** These approaches offer a blend of the benefits seen in both wrapper and filter methods. They incorporate feature interactions while maintaining computational efficiency at a reasonable level. Embedded approaches are characterized by their iterative nature, where each iteration of the model training process is carefully managed to extract the features that make the most significant contributions to the training process during that specific iteration.

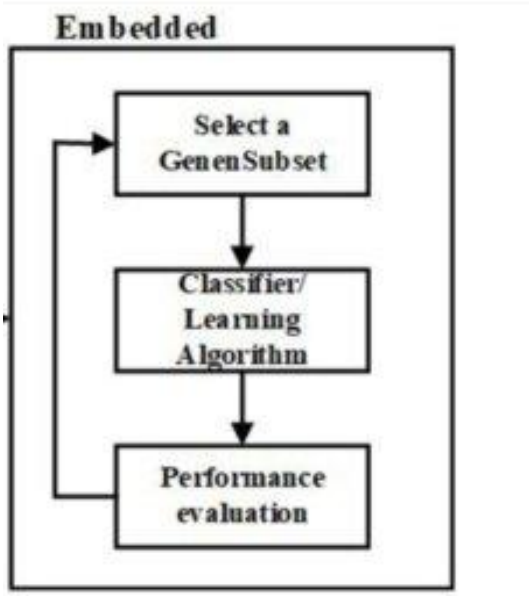

Figure 2.6: Embedded Method Flowchart

2.6 Ensemble Methods

Ensemble learning is a machine learning strategy that harnesses the strength of multiple models, often referred to as "weak learners," to collaboratively address the same problem and produce improved results. The core concept behind ensemble learning is that through smartly combining these less powerful models, we can achieve greater accuracy and robustness.

The motivation behind ensemble learning arises from the belief that individual models might have limitations or biases, but by amalgamating their predictions or decisions, we can compensate for these shortcomings and arrive at a more dependable and precise overall result. Each weak learner may possess distinct strengths, weaknesses, or viewpoints, and ensemble learning aims to capitalise on the diversity and complementary attributes of these models to enhance the final prediction or decision.

The three primary approaches for constructing an ensemble model by aggregating individual machine learning models are as follows:

1. **Bagging:** Bagging, an abbreviation for bootstrap aggregating, is a technique that entails training multiple models in parallel, each on a distinct subset of the training data. These

subsets are created using a method known as bootstrapping, where data samples are randomly chosen with replacement from the original training dataset. The primary objective of bagging is to reduce variance and enhance the generalisation ability of the models by either averaging or combining their predictions. A prominent example of a bagging method is the Random Forest algorithm.

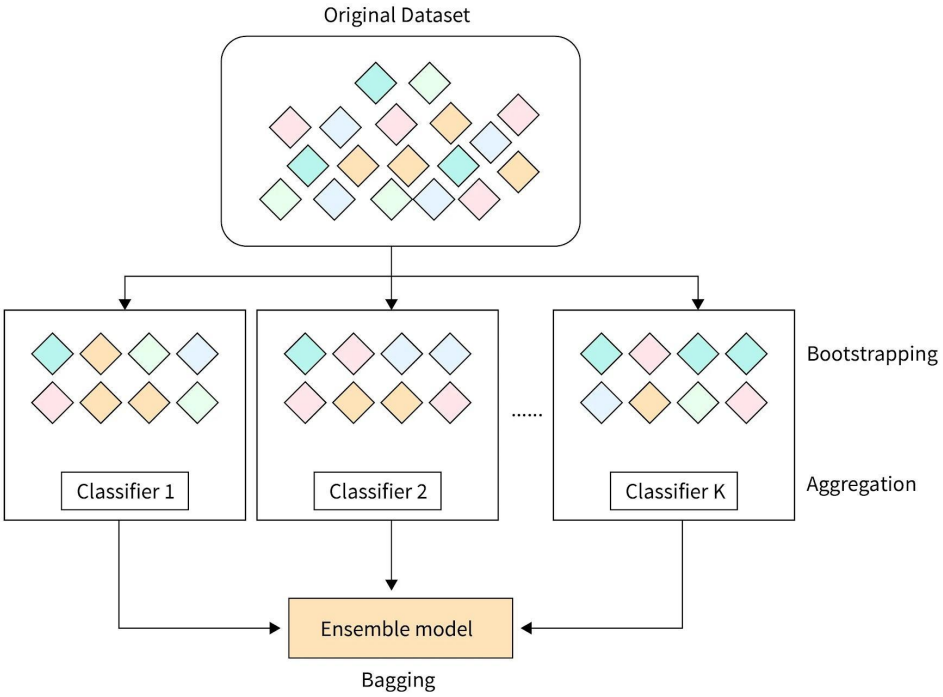

Figure 2.7. Bagging ensemble method (https://www.scaler.com/topics/machine-learning)

2. **Boosting**: Boosting is a sequential approach to ensemble learning that emphasizes training models iteratively. In each iteration, more weight is assigned to instances that were misclassified in previous iterations. Weak models are trained one after the other, and their outputs are aggregated to construct a robust predictive model. Boosting algorithms, like AdaBoost and Gradient Boosting, excel in reducing bias and enhancing accuracy by paying special attention to instances that are challenging to classify.

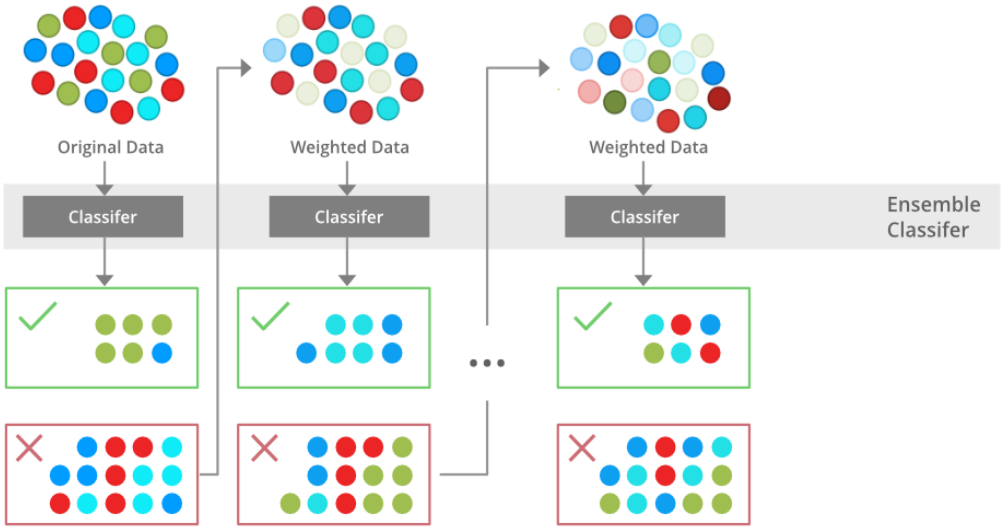

Figure 2.8. Boosting ensemble method (https://www.geeksforgeeks.org/)

3. **Stacking**: Stacking, sometimes referred to as stacked generalisation, is a technique that encompasses the training of multiple models to produce predictions, followed by the utilisation of a meta-model to consolidate these predictions. Unlike straightforward averaging of predictions, in stacking, the meta-model is trained to understand how to optimally weigh the predictions generated by the individual models. This approach permits the creation of more intricate and refined combinations of models, often resulting in enhanced overall performance.

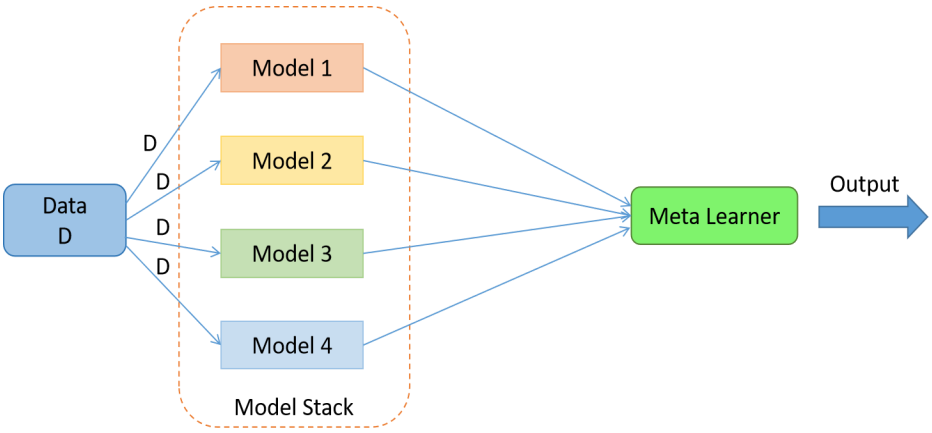

Figure 2.9. Stacking ensemble method (https://www.analyticsvidhya.com/)

2.7 Classification Algorithms

Classification is a supervised learning task that involves categorising input data into predefined classes or categories. The primary objective of classification is to build a predictive model that can assign new, unseen data points to one of these predefined classes based on their features or attributes. Classification problems can be binary, where the goal is to classify data into one of two classes (e.g., spam or not spam), or multi-class, where there are more than two possible classes (e.g., classifying animals into categories like "dog," "cat," "horse," and so on).

Common algorithms used for classification tasks include:

a. **Decision Trees**: Decision trees create a hierarchical structure where each node represents a feature, and branches represent decisions based on those features. They are interpretable and can handle both classification and regression tasks. Ensembles like Random Forests and Gradient Boosted Trees improve their predictive power and reduce overfitting.

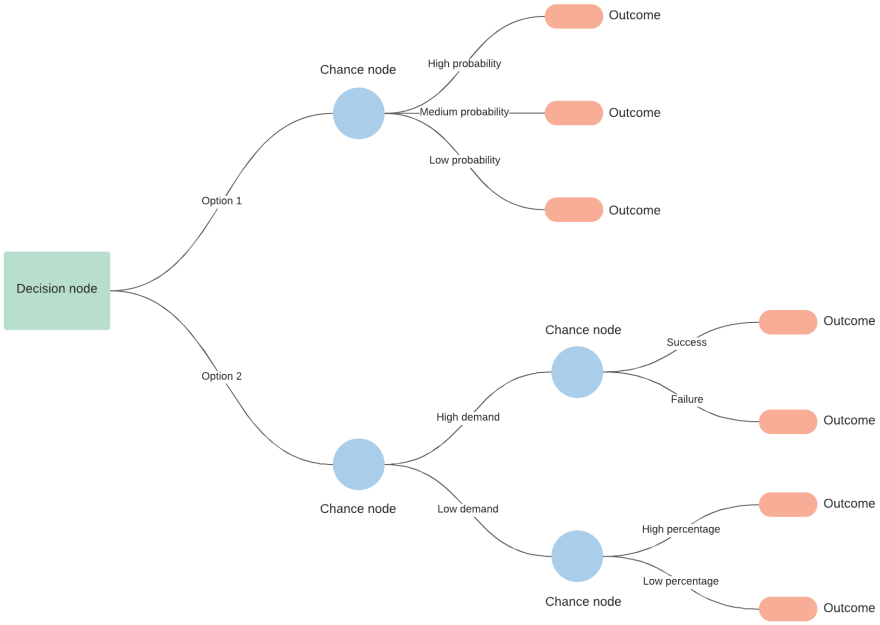

Figure 2.10. Decision Tree (https://lucidspark.com/)

b. **Random Forests**: Random Forest is an ensemble learning method that combines multiple decision trees to make predictions. It reduces overfitting by using bootstrapped

data and random feature selection. This ensemble approach often provides higher accuracy and robustness compared to individual decision trees.

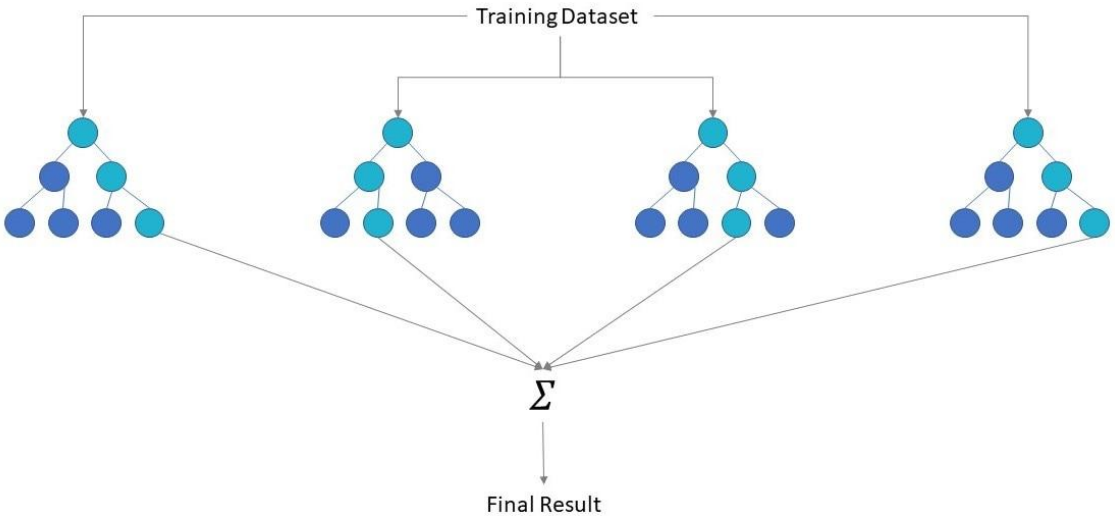

Figure 2.11. Random Forest (https://lucidspark.com/)

c. **Support Vector Machines (SVMs)**: Support Vector Machines are versatile classifiers capable of handling both linear and nonlinear classification tasks. SVM aims to find a hyperplane that maximises the margin between data points of different classes. This margin ensures robust generalisation to unseen data. SVM's kernel trick allows it to handle complex data distributions.

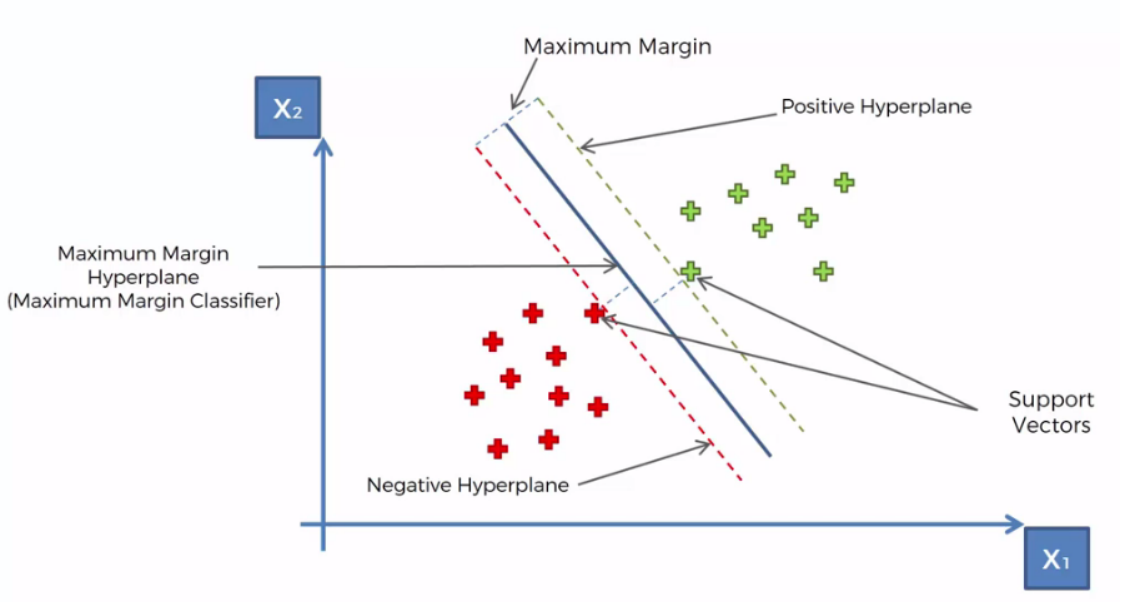

Figure 2.12. Support vector machines (http://arun-aiml.blogspot.com/)

d. **Neural Networks**: Neural networks, especially deep neural networks, are highly flexible classifiers capable of handling complex tasks. They consist of layers of interconnected neurons that learn hierarchical representations of data.

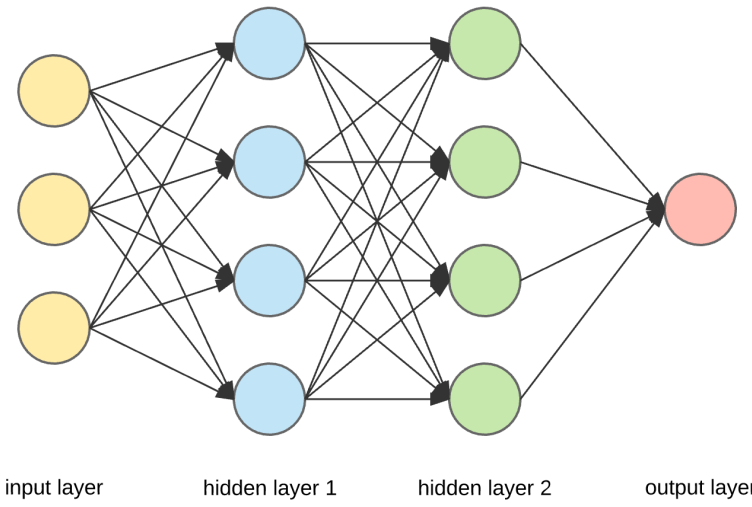

Figure 2.13. Artificial Neural Networks (https://medium.com/@ksusorokina/)

e. **Logistic Regression**: Logistic regression is a fundamental classification algorithm used when we want to predict binary outcomes, such as whether an email is spam or not. It models the probability of an event occurring by fitting the data to the sigmoid function. The output is interpreted as the likelihood of belonging to a particular class. It's a simple yet effective choice when the decision boundary between classes is linear.

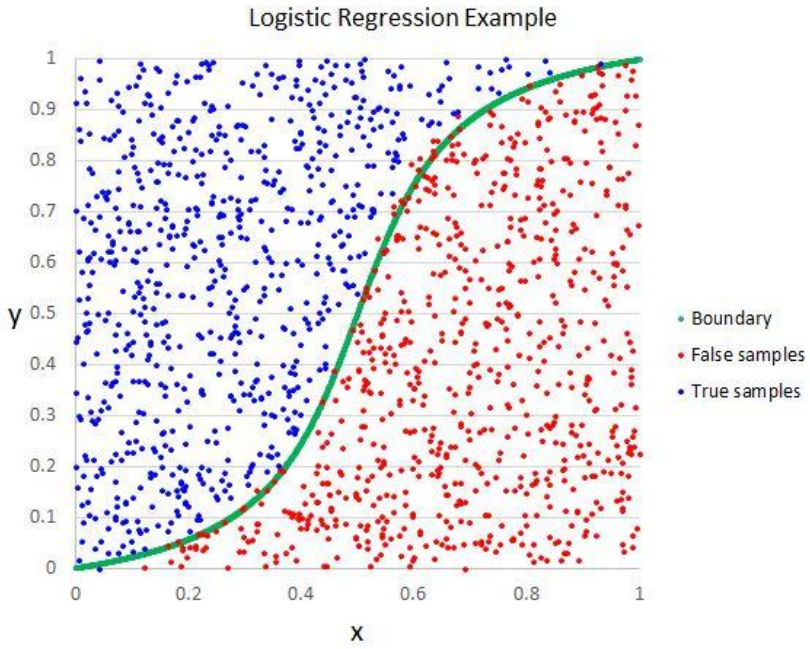

Figure 2.14. Logistic Regression (https://helloacm.com/)

2.8 Review of Literature

In this section, we present an in depth review of contemporary literature on the application of machine learning techniques in Intrusion Detection Systems.

Table 2.1: Review of Relevant Literature

Author's Name / Year	**MOTIVATIONS**	**METHODOLOGY**	**CONTRIBUTION TO KNOWLEDGE**	**LIMITATIONS**
Gulzanesther *et al.,* 2018	Enhancing the effectiveness of current methods is necessary to improve their basic accuracy and detection rate.	A Robust Fast Machine Learning approach was put forward for the development of an effective intrusion detection system. NSL-KDD dataset with 41 features was used to evaluate the train and evaluate the proposed model The Random Forest machine learning method was applied in a network-based Intrusion Detection System to classify regular and anomalous activities within the network.	The outcomes of the suggested model indicate that by employing data preprocessing and feature selection, the classification algorithm can attain an accuracy level of 99%.	While achieving a heightened detection accuracy, the categorization accuracy remains comparatively lower due to the structural resemblance in network traffic behaviour.
Othman *et al.,* 2018	The high dimensionality of big data makes the classification process complex and time consuming.	The model introduced utilised the Spark Big Data platform for rapid processing and analysis of data. KDD99 dataset with 41 features was used to evaluate the train and evaluate the model. SVM classifier was also used in building the model	The experimental findings demonstrated that the model exhibits exceptional performance and speed.	The proposed method does not support multi-class classification
Hadi & Mohammed, 2022	Low efficiency of machine learning algorithms in Network Intrusion Detection Systems (NIDS) due to huge amounts of data that are unclassified.	The suggested approach involves constructing the Network Intrusion Detection System Deep Learning (NIDS-DL) by employing multiple classifiers within the realm of deep learning and training them using 12 selected features extracted from the original 41 features in the NSL-KDD dataset. The performance of the proposed NIDS was assessed using standard evaluation metrics including, Accuracy (AC), Precision (P), recall (R), and F1-score (F).	Utilising the feature selection technique to train classifiers focuses on incorporating the strongest feature correlations while preventing any misguided training that could compromise achieving optimal outcomes. CNN classifier was proven to achieve the highest results of all tested	

			classifiers.	
Taher *et al.,* 2019	High false alarm rate of proposed methods.	The suggested approach employs both Support Vector Machines (SVM) and Artificial Neural Networks (ANN) as machine learning techniques on the NSL-KDD dataset.	The study shows that the model constructed through Artificial Neural Networks (ANN) and employing a wrapper feature selection method exhibited superior performance compared to all alternative models. This resulted in the accurate classification of network traffic, achieving a detection rate of 94.02%.	
Mohammadi *et al.,* 2019	Current Intrusion Detection System (IDS) struggle to handle an extensive volume of data that contains irrelevant and redundant attributes, leading to issues categorising network traffic accurately.	The suggested approach merges filter and wrapper techniques to pick the best-suited set of attributes from a substantial dataset (KDD Cup 99). This aims to enhance the effectiveness of the IDS by boosting its performance.	The introduced FGLCC method yielded superior outcomes in terms of detection, accuracy, and false positive rates when contrasted with LCFS and FGMI approaches.	
Amarudin *et al.,* 2020	To ascertain the most recent trends in the field of Intrusion Detection Systems (IDS), including the techniques, methods, and datasets employed by researchers.	Several literature was reviewed and their data was extracted.	The study shows that SVM based models are the most proposed models for IDS.	

Haripriya & Jabbar, 2018	To discuss and compare different ML methods for Intrusion Detection System	Several models were reviewed using standard evaluation metrics.	The study shows that every algorithm has its own importance in the improvement of IDS.	
Ayyagari *et al.,* 2021	To provide a structured analysis of intrusion detection methods and systems utilised across diverse network environments.	In the survey protocol, several research questions are investigated, and corresponding answers for each question are also furnished.	A systematic survey of various techniques and systems for intrusion detection in network security was presented.	
Tama & Lim, 2021	High false positive rate (FPR) or failure to detect novel attacks by IDS techniques	An empirical benchmark was executed to assess the comparative effectiveness of ensemble methods within Intrusion Detection Systems (IDSs) using different classification algorithms and performance metrics.	This study unveiled a notable surge of interest in the utilisation of the random forest classifier for Intrusion Detection Systems (IDSs). The findings indicate that ensemble learning has led to substantial enhancements over standalone classifiers. Nevertheless, this isn't a consistent outcome, as its effectiveness relies on diverse elements like the choice of base classifiers and voting methodologies.	
Alagappan & Sahayam, 2023	The growing complexity and sophistication of cyber-attacks	Various machine learning methods were investigated using standard evaluation techniques	Machine learning algorithms were shown to improve Intrusion Detection Systems (IDSs), as they have the capability to grasp patterns and anomalies from extensive datasets and can adjust to evolving attack patterns.	

Sreenath & Udhayan, 2015	There is a requirement for a streamlined model that addresses numerous prevalent performance metrics or a fusion of these metrics.	An intrusion detection system using bagging ensemble selection is proposed. The NSL-KDD dataset with 41 features was used. The intrusion detection system was evaluated based on cross-validation and accuracy.	Among the four classification algorithms, Bagging Ensemble Selection achieved the highest accuracy percentage.	
Belavagi & Muniyal, 2016	Need for an investigation into the use of supervised learning for IDS	The supervised machine learning algorithms namely Logistic Regression, Gaussian Naive Bayes, Support Vector Machine and Random Forest were tested with the result represented as a Reliability curve	Based on the findings, it can be inferred that the Random Forest classifier surpasses other classifiers for the analysed dataset and parameters. It achieves an accuracy of 99%.	Multi-class classifiers were not considered
Pai *et al.,* 2021	Need for a comparative study on which machine learning algorithms perform better in detecting attacks like Denial of Service, Probe, User to Root and Remote to Local	The NSL-KDD dataset was employed to train the machine learning model for detecting network intrusions. The chosen machine learning algorithms for the study include SVM, Decision Tree, Logistic Regression, Naive Bayes, J48, and Random Forest. Prior to inputting the dataset into the model, normalisation was performed. The performance of the machine learning model was assessed based on accuracy, precision, recall, F1-score, and ROC metrics.	The most effective classifier was the random forest classifier, achieving a 99.997% accuracy for DoS attacks. Similarly, the best classifier for U2R attacks was J48, demonstrating a 99.947% accuracy.	
Chaibi *et al.,* 2020	Proven lack of accuracy of other IDS machine learning methods	Two methodologies were used; Artificial Neural Network (ANN) and Recurrent Neural Network (RNN) with information gain (IG) for feature selection, and an RNN with information gain (IG), grain ratio (GR) and	The results indicate that in both approaches, RNN demonstrates superior performance compared to ANN. However, among various machine learning classifiers, ANN performs better.	

		correlation attribute (CA) as feature selection methods NSL-KDD data-set was used to create the model. The evaluation of the outcome was conducted using standard mtrics such as Accuracy, True Positive Rate (TPR), False Positive Rate (FPR), Precision, and Recall.		
Krishnaveni *et al.,* 2021	Traditional intrusion detection systems (IDS) are inadequately equipped to address intricate and widespread security breaches and attacks effectively.	An innovative univariate ensemble filter feature selection method (UEFFS) is introduced, which employs five distinct filter feature selection techniques to derive optimal feature subsets from the provided intrusion datasets. Three datasets were used namely Honeypot, NSL-KDD, Kyoto. The study utilised various performance evaluation metrics, including prediction accuracy, true positive rate, false positive rate, precision, recall, F-measure, detection rate (DR), false alarm rate (FAR), and ROC-AUC analysis.	It demonstrated that the suggested approach led to a significant enhancement in both accuracy and the robustness of diverse classification tasks, thereby contributing to feature selection (FS) and pivotal stages in the intrusion detection system.	

CHAPTER THREE
RESEARCH METHODOLOGY

3.1 Introduction

Research methodology is a systematic and organised approach employed by researchers to conduct investigations or studies. It offers a structured framework for the collection, analysis, and interpretation of data in order to address research questions or test hypotheses. A well-established research methodology is vital to ensure the trustworthiness, precision, and relevance of the study's findings.

In this research, an examination is conducted to assess how machine learning algorithms contribute to intrusion detection systems. The initial data preparation phase entails addressing missing values and standardising attributes, following the KDD (Knowledge Discovery in Databases) and data mining approach as outlined by (Fayyad *et al.,* 1997). Subsequently, the processed data is utilised for constructing a classification model aimed at enhancing intrusion detection.

3.2 Data

The dataset used was the NSL_KDD (Tavallaee *et al.,* 2009), it contains records that hold the traces of internet traffic observed by a basic intrusion detection network. It has 43 features with over 40,000 samples and is an improvement of the KDDCup99 dataset which contains many issues that have been resolved by NSL_KDD (Amarudin et al., 2020).

3.2.1 Data Preprocessing

Data preparation is a vital phase in readying data for training, and it encompasses various procedures to ensure the data is in the correct format and of high quality. This includes tasks like data cleaning, balancing, filling missing values, standardising, encoding, enhancing, and mitigating bias, all of which are performed to make the data suitable for further analysis and modelling.

3.2.2 Scaling

In machine learning, feature scaling stands out as a pivotal stage in data preprocessing before constructing a machine learning model. The act of scaling can be the differentiating factor between a subpar machine learning model and an improved one (Roy, 2020). A robust scaling algorithm was used for this project which involves removing the median and scaling the data according to the quantile range between the 1st quartile (25th quantile) and the 3rd quartile (75th quantile), this gives better results than other scaling and normalisation methods for datasets with outliers (Pedregosa *et al.,* 2011).

3.3 Feature Selection Methods

Feature selection plays a crucial role in machine learning by reducing data dimensionality and increasing performance of machine learning models. Two common methods employed for this purpose are the filter method and the wrapper method. In the filter method, features are chosen based on their scores from various statistical tests that gauge their relevance through their correlation with the dependent or outcome variable. In contrast, the wrapper method identifies a subset of features by assessing the subset's utility in relation to the dependent variable. Filter methods are agnostic to any specific machine learning algorithm, whereas the wrapper method selects the best feature subset depending on the machine learning algorithm utilised for model training (Taher *et al.,* 2019). The filter methods used for the implementation of this project are the CHI square and Information gain.

3.3.1 Chi Square

The chi-square test is a statistical method employed to assess the independence of two events or variables. When provided with data for these two variables, we can determine the observed count (O) and the expected count (E) for each category or outcome. The chi-square statistic quantifies the extent to which the observed count (O) differs from the expected count (E), helping us evaluate the degree of association or independence between the variables (Babienko, 2019). The mathematical model is defined as:

$$x_c^2 = \sum_{i=1}^{N} \frac{(O_i - E_i)^2}{E_i} \qquad (3.1)$$

Where;

1. c is the degree of variation independence
2. O is the observed value
3. E is the expected value

3.3.2 Information Gain

Entropy is a measure of impurity or disorder within a set of data. It's used to quantify the uncertainty or randomness associated with the distribution of classes or labels in a dataset (Seth, 2020). It is defined mathematically as;

$$H(S) = -\sum_{i=1}^{N} (p_i * log_2(p_i)) \quad (3.2)$$

Where:

1. H(S) is the entropy of the dataset.
2. p_i represents the proportion (probability) of the i-th class within the dataset.
3. The sum Σ is taken over all distinct classes in the dataset.

Information gain (IG) measures the reduction in uncertainty (entropy) achieved by partitioning a dataset based on a particular feature (Gopalan, 2020). The information gain is obtained through the following steps;

1. Calculate the entropy of the original dataset before any divisions. This initial entropy quantifies the dataset's uncertainty concerning class labels**,**
2. Using a specific feature, divide the dataset into subsets, with each subset linked to a unique feature value**,**
3. Compute the entropy for each subset produced by the split. This computation assesses the level of uncertainty within each subset,
4. Calculate Information Gain by subtracting the average entropy of all the subsets from the initial entropy,

5. Choose the feature maximising information gain as the splitting criterion to reduce uncertainty and create homogenous subsets.
6. Repeatedly apply the process for child nodes using remaining features.

3.4 Ensemble Methods

Ensemble methods in machine learning combine the predictions of multiple individual models to improve their overall performance and robustness. The three ensemble methods used in this project are Bagging, boosting and voting.

3.4.1 Bagging

Bagging employs bootstrap sampling to acquire subsets of data for training these base learners. To implement bagging for our dataset we take a sample of m training examples by drawing with replacement. This means that some original examples may appear multiple times in the sample, while others may not be included at all. We repeat this process T times, T samples of m training examples are generated. Subsequently, from each of these samples, we train a base learner using the base learning algorithm. (Zhou, 2012).

Mathematically this is expressed by;

$$H(x) = arg\,max_{y \in Y} \sum_{t=1}^{T} I(h_t(x) = y) \quad (3.3)$$

where:

1. y is the output label
2. x is the input
3. H is the final model

3.4.2 Adaboost

Adaboost is a boosting ensemble algorithm focused on minimising the exponential loss function which measures the difference between predicted and actual classes, by adjusting weights and selecting the best weak classifiers (Zhu *et al.,* 2006).

$$F_T(x) = \sum_{t=1}^{T} f_t(x) \qquad (3.4)$$

where:

a. $F_T(x)$ is the final predictor,

b. $F_t(x)$ is base classifier,

c. x is the input.

3.5.1 Voting

In the voting process, each model makes a prediction that adds a vote to a candidate class. These votes are gathered and employed to decide the ultimate outcome, often by selecting the class with the highest vote count or appying a predefined voting scheme. In this project, the chosen voting method is the majority voting approach, in which every classifier makes a choice for a particular class label, and the ultimate class label is chosen based on the one that receives more than half of the votes. If none of the class labels obtains over half of the votes, the combined classifier abstains from making a prediction.

$$H(x) = \left\{ \sum_{i=1}^{T} h_i^j(x) > \frac{1}{2} \sum_{k=1}^{l} \sum_{i=1}^{T} h_i^k(x) \right\} \qquad (3.5)$$

Where:

a. h_i is an individual classifier

b. c_j is the rejection option in the case no classifier acquires more than half of the votes

3.5 Classification Algorithms

Classification in machine learning analyses patterns and features in the training data to estimate the likelihood of new data belonging to specific classes. The classification algorithms utilised in this project are support vector machines, artificial neural network and linear regression.

3.5.1 Artificial Neural Network

Artificial Neural Networks (ANNs) draw inspiration from the human brain, replicating the way biological neurons communicate. ANNs consist of layers of nodes, including an input layer, one or more hidden layers, and an output layer. Each node, also referred to as an artificial neuron, links to others and carries an associated weight and threshold. When the output of a neuron exceeds the designated threshold, it activates and transmits data to the subsequent network layer. If the output remains below the threshold, no data is forwarded to the next layer (Taher *et al., 2019)*.

3.5.2 Support Vector Machines

A support vector machine creates a hyperplane or multiple hyperplanes within a high-dimensional or potentially limitless space, which can then be used for tasks such as classification, regression, or other computational tasks (Pedregosa *et al.,* 2011).

For a linearly separable dataset, the classifier or the decision function will have the form:

$$w^t x = (-b) \times (1) + (-a) \times x + 1 \times y \qquad (3.6)$$

where:

a. w is the weight,
b. b is the bias, and
c. x is the feature vector.

3.5.3 Logistic Regression

Logistic regression is a predictive modelling technique that calculates the probability of a certain outcome based on one or more independent variables. The logistic function, often depicted as an S-shaped curve (sigmoid curve), generates an output value ranging between 0 and 1 which can effectively represent a probability (Nield, 2022). It is represented mathematically by the formula:

$$y = \frac{1}{1 + e^{-z}} \tag{3.7}$$

where:

a. y is the output value ranging from 0 to 1
b. x is the input variable
c. z is the coefficient

3.6 Performance Evaluation Methods

The following are the performance evaluation methods that would be used in evaluating the performance of the model proposed in this project:

3.6.1 Confusion Matrix

The confusion matrix is used to visualise performance of a classification model. The process involves contrasting the class labels predicted by the model with the actual class labels in a test dataset. It then classifies the results into four categories: true positives, true negatives, false positives, and false negatives (Krishnaveni *et al.,* 2021).

3.6.2 Accuracy

Accuracy is the most commonly used metric for model assessment. It indicates the general performance of the classifier. If its value is 0 it indicates misclassification, if it is 1 it indicates accurate classification (Krishnaveni *et al.,* 2021). It is expressed by the formula:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \quad (3.8)$$

3.6.3 Precision

Precision in a classification model measures its ability to correctly classify the relevant data points within the dataset. It is calculated by dividing the number of true positives by the total number of predicted positive samples (Krishnaveni *et al.,* 2021). The precision is determined as follows:

$$\text{Precision} = \frac{TP}{TP + FP} \quad (3.9)$$

3.6.4 Recall

Recall measures a model's ability to identify all the important data points within a dataset. This is calculated by dividing the number of true positives by the total number of actual positive samples (Krishnaveni *et al.,* 2021). The calculation for recall is as follows:

$$\frac{TP}{TP + FN} \quad (3.10)$$

3.6.5 F1-Score

This metric uses the mean to combine the recall and precision and give a general indication of a classification model's performance (Krishnaveni *et al.,* 2021). It is estimated as follows.

$$F1\ score = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} \quad (3.11)$$

CHAPTER FOUR
SYSTEM DESIGN AND IMPLEMENTATION

4.1 Overview

System implementation involves the practical creation and evaluation of a suggested model. The aim during this stage is to bring the model into existence, document the procedure, and showcase the final results.

4.2 System Requirements

System requirements include the fundamental hardware and software needed to run a system.

4.2.1 Hardware Requirements

The hardware requirements of the system are:

1. Processor: Intel Core i5 or higher
2. RAM: 16 GB of RAM or higher
3. Drive: 16GB of space or higher
4. Processor Speed: 1.8GHZ
5. GPU: A GPU is required.
6. Internet: An internet connection is required.

4.2.2 Software Requirements

1. Operating System: modern windows, linux and mac OS versions are supported.
2. An IDE (Integrated Development Environment) configured for python development is required for implementing and debugging the program.

4.3 Development Tools

Software development tools are used to help developers create, debug, test, and maintain software applications and systems. The following tools were used in the implementation of the proposed model:

1. **Python**: a general-purpose programming language that is easy to learn and use.

2. **Scikit-learn**: a machine learning library for Python that provides a variety of machine learning algorithms.
3. **Kaggle**: a data science and machine learning platform that provides datasets, competitions, and forums.

4.4 Results

This following section contains tables and graphs displaying the final results of the implementation.

4.4.1 Data

The data used for the implementation is the NSL_KDD dataset (Tavallaee *et al.,* 2009), which contains 43 features with over 40,000 samples, of which we use only a subset of 25191 samples. There are no missing data, duplicates, or significant outliers in the dataset. The output label is categorised into 1 for an intrusion detected and 0 for the opposite case. We split the data into 70% training and 30% testing and scale it using min max scaler.

4.4.3 Discussions

In table 4.1, We can observe the performance of the models based on accuracy, precision, recall and F1-Score evaluation metrics. The Logistic regression model is proven to have the highest accuracy of the other models at 0.884 accuracy score while Bagging and Adaboost are shown to have the highest f1 score. SVM achieved the lowest accuracy of all the models at 0.855 with MLP and Voting ensemble scoring higher in accuracy with 0.864 and 0.859 respectively.

Table 4.1. Evaluation of classifiers

	LR	SVM	MLP	Bagging (RF)	Adaboost	Voting
Accuracy	**0.884**	0.855	0.864	0.868	0.868	0.859
Precision	**0.952**	0.904	0.909	0.861	0.861	0.912
Recall	0.952	0.922	0.909	**1.0**	**1.0**	0.917
F1 Score	0.911	0.913	0.924	**0.925**	**0.925**	0.914

The figures provided below depict the confusion matrix and the ROC curves of the classifiers that have been implemented and evaluated. These graphical representations give us valuable insights into the performance of the models. From this, we can observe a significant true negative rate when it comes to the detection of attacks. However, it is worth highlighting that there is also a noteworthy amount of false negatives predictions in the classification of normal instances, particularly noticeable in certain models such as bagging and adaboost. This occurrence might be attributed to data imbalance in the dataset.

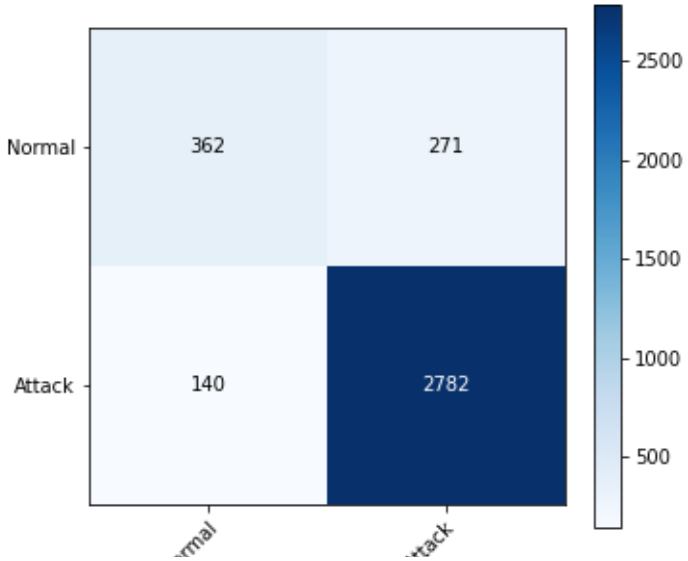

Figure 4.1. Confusion matrix for Linear Regression classifier

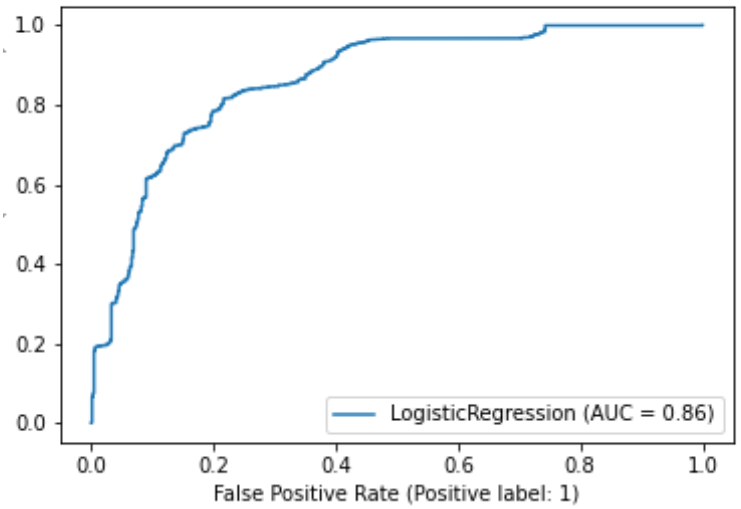

Figure 4.2. ROC curve for Logistic Regression classifier

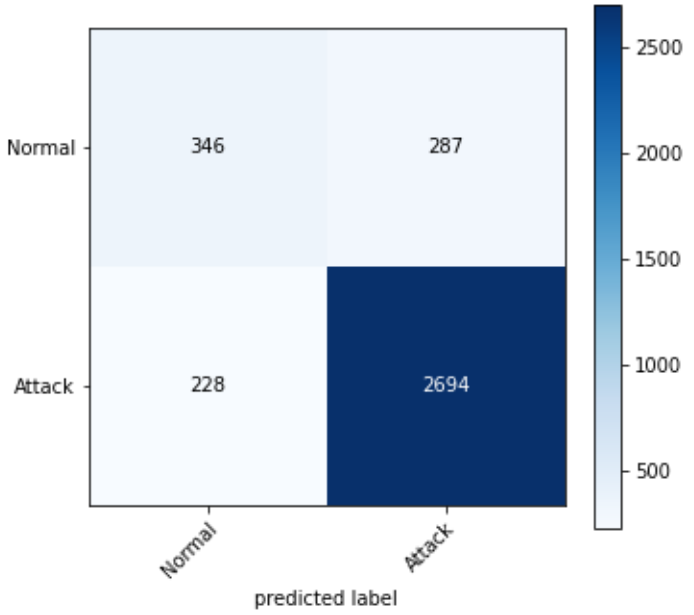

Figure 4.3. Confusion matrix for Support Vector Machines classifier

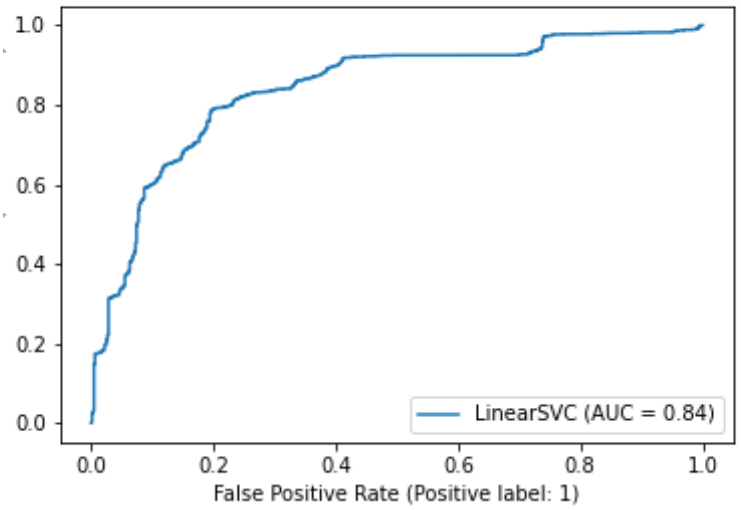

Figure 4.4. ROC curve for Support Vector Machines classifier

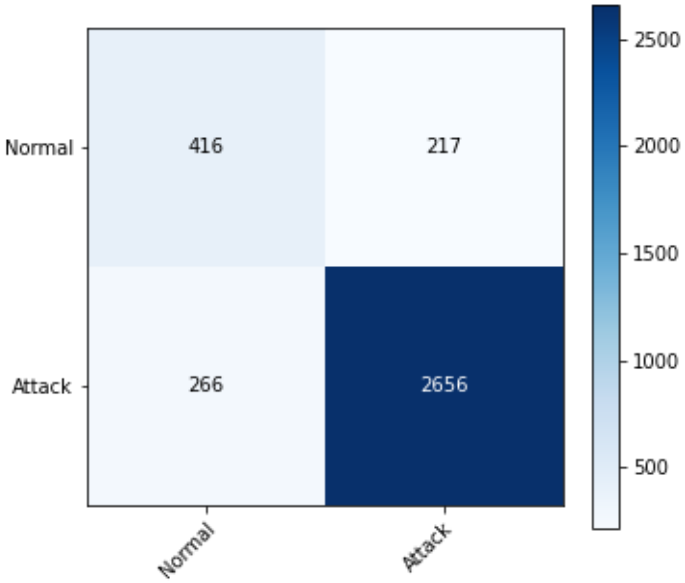

Figure 4.5. Confusion matrix for Multilayer Perceptron classifier

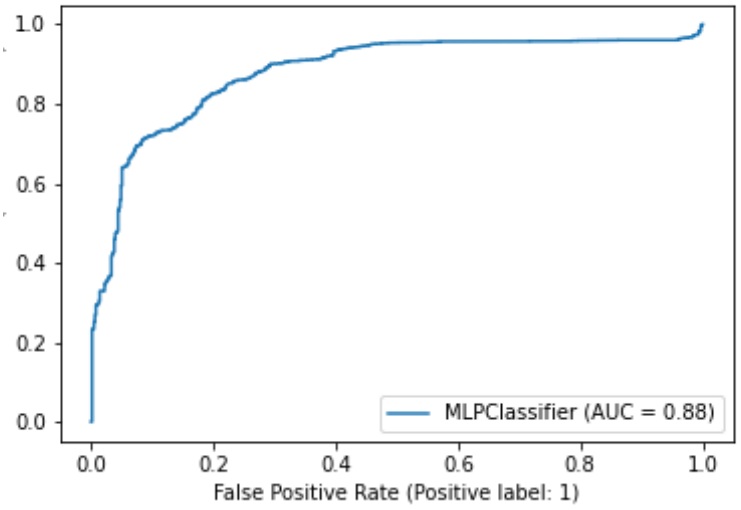

Figure 4.6. ROC curve for MLP classifier

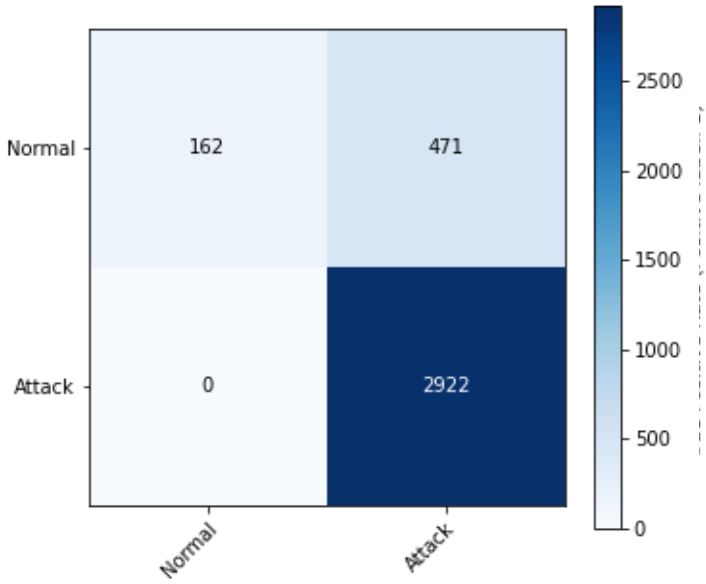

Figure 4.7. Confusion matrix for Bagging Ensemble classifier

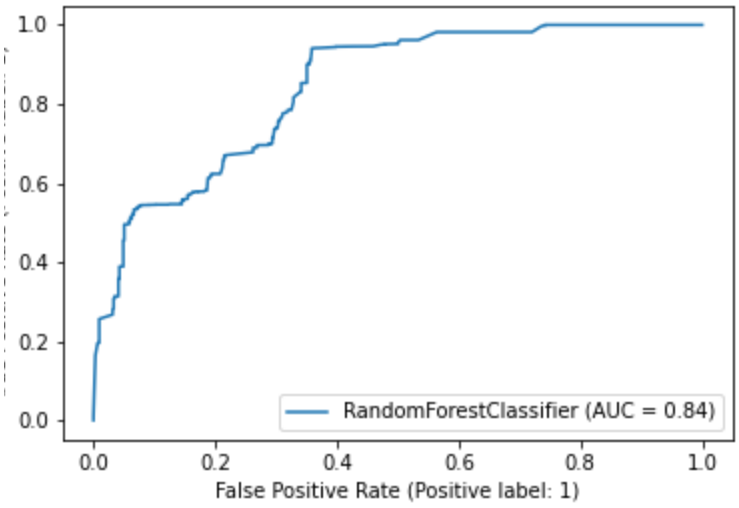

Figure 4.8. ROC curve for Bagging Ensemble classifier

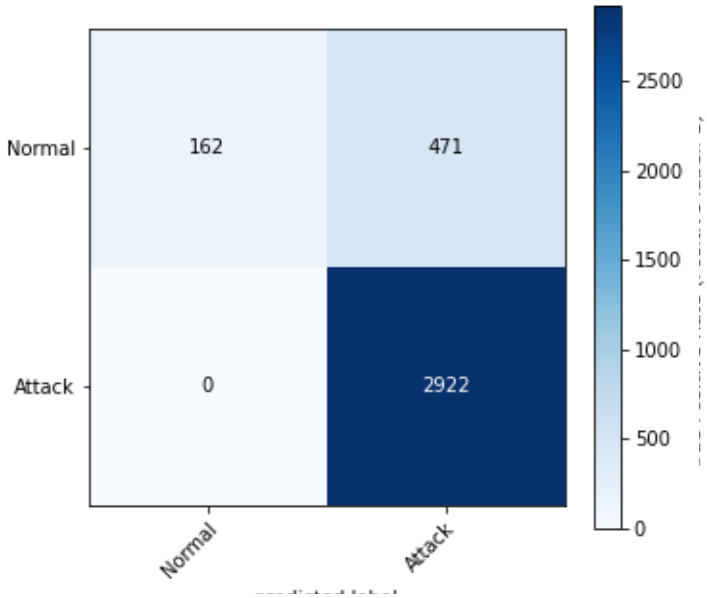

Figure 4.9. Confusion matrix for Adaboost Ensemble classifier

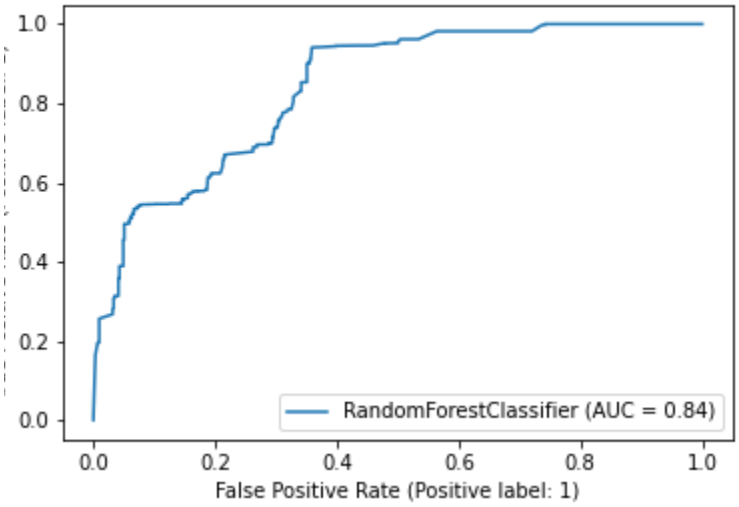

Figure 4.10. ROC curve for Adaboost Ensemble classifier

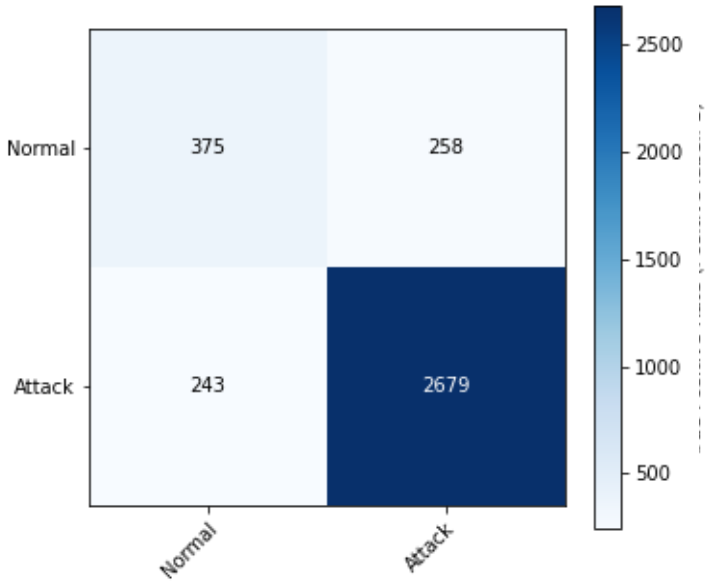

Figure 4.11. Confusion matrix for Voting Ensemble classifier

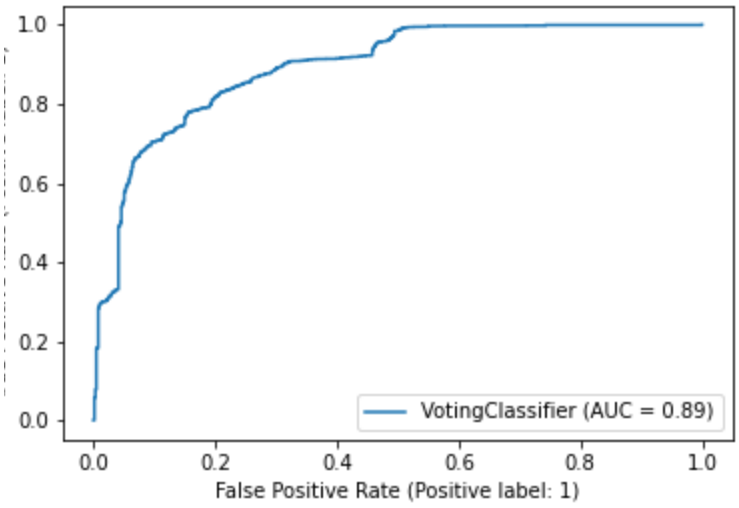

Figure 4.12. ROC curve for Voting Ensemble classifier

CHAPTER FIVE

CONCLUSION, RECOMMENDATION AND FUTURE WORKS

5.1 Conclusion

In this project, we have delved into the development and evaluation of an Intrusion Detection System (IDS) that combines feature selection techniques and ensemble models to enhance network security. In today's interconnected digital landscape, safeguarding sensitive information and networks against malicious activities and cyberattacks is paramount. IDSs are fundamental tools in identifying and mitigating these threats, making their improvement an essential research endeavour.

Our results have demonstrated the effectiveness of leveraging feature selection methods in tandem with ensemble models for IDS enhancement. We conducted an extensive review of existing IDS methodologies, collected network data, and implemented ensemble techniques to select and classify the most relevant features for intrusion detection. Our results showcase that this approach outperforms traditional feature selection methods, not only in accuracy but also in computational efficiency.

By utilising ensemble models, our IDS has shown itself to be more robust, capable of handling diverse attack patterns, and resistant to the noise present in network data. The findings of this study contribute significantly to the field of cybersecurity by illustrating the potential of combining feature selection and ensemble models to optimise IDS performance.

5.2 Recommendation

Based on the findings of this project, we offer several recommendations. Firstly, organisations should consider integrating the proposed feature selection and ensemble model-based IDS into their existing network security infrastructure. This can substantially improve the accuracy and efficiency of intrusion detection, resulting in better protection against cyber threats. Secondly, maintaining continuous data monitoring is imperative to keep the feature selection model up-to-date with evolving network behaviours and threats. Regular updates and retraining are essential to ensure its continued effectiveness. Thirdly, interdisciplinary collaboration between cybersecurity experts, data scientists, and network administrators is crucial for the successful implementation of this IDS. Such collaboration can lead to refined models and insights into emerging threats. Lastly, organisations should establish alerting mechanisms and well-defined incident response plans to address security incidents promptly.

5.3 Future Works

While this dissertation represents a significant step forward in enhancing intrusion detection systems through feature selection and ensemble models, there remain several avenues for future research and improvement. First, the exploration of behavioural analysis techniques alongside feature selection could further enhance the IDS's ability to detect anomalies and novel attacks. Second, the integration of deep learning models in conjunction with ensemble methods holds promise for creating more robust and adaptive intrusion detection systems. Third, the incorporation of real-time threat intelligence feeds could provide the IDS with the ability to

identify emerging threats and zero-day vulnerabilities. Finally, addressing scalability challenges in large-scale network environments is essential to ensure that the IDS remains effective in protecting expansive networks.

REFERENCES

Alagappan, V. V., & Sahayam, A. L. (2023). A Comprehensive Study on the influence of Machine Learning Algorithms on Intrusion Detection System. https://www.researchgate.net/publication/370466404

Amarudin, Ferdiana, R., & Widyawan. (2020). A Systematic Literature Review of Intrusion Detection System for Network Security: Research Trends, Datasets and Methods. *2020 4th International Conference on Informatics and Computational Sciences (ICICoS)*, 1-6. doi.org/10.1109/ICICoS51170.2020.9299068`

Ayyagari, M. R., Kesswani, N., Kumar, M., & Kumar, K. (2021). Intrusion detection techniques in network environment: a systematic review. *Wireless Networks*, *27*(2), 1269–1285. https://doi.org/10.1007/s11276-020-02529-3

Babienko, V. (2019). *Chi-Square Test for Feature Selection in Machine learning*. Towards Data Science. Retrieved September 9, 2023, from https://towardsdatascience.com/chi-square-test-for-feature-selection-in-machine-learning-206b1f0b8223

Belavagi, M. C., & Muniyal, B. (2016). Performance Evaluation of Supervised Machine Learning Algorithms for Intrusion Detection. *Procedia Computer Science*, *89*, 117 – 123. doi.org/ 10.1016/j.procs.2016.06.016

Chaibi, N., Atmani, B., & Mokaddem, M. (2020). Deep Learning Approaches to Intrusion Detection:A new Performance of ANN and RNN on NSL-KDD. *Proceedings of the 1st International Conference on Intelligent Systems and Pattern Recognition.* 10.1145/3432867.3432889

Gopalan, B. (2020, December 10). *What is Entropy and Information Gain? How are they used to construct decision trees?* Numpy Ninja. Retrieved September 9, 2023, from https://www.numpyninja.com/post/what-is-entropy-and-information-gain-how-are-they-used-to-construct-decision-trees

Gulzanesther, A., W. Kathrine, J., Shibin, D., & Susanna, A. (2018). Efficient intrusion detection using machine learning techniques. *Journal of Adv Research in Dynamical & Control Systems*, *10*(03-Special Issue), 1045-1050. https://www.researchgate.net/publication/331274698

Hadi, M. R., & Mohammed, A. S. (2022). A Novel Approach to Network Intrusion Detection System using Deep Learning for SDN: Futuristic Approach. *Academy and Industry Research Collaboration Center (AIRCC)*, 69-82. https://doi.org/10.5121/csit.2022.121106

Haripriya, L., & Jabbar, M. A. (2018). Role of Machine Learning in Intrusion Detection System: Review. *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, 925-929. doi.org/10.1109/ICECA.2018.8474576

Kira, K., & Larry, R. (1992). Practical Approach to Feature Selection. *Proceedings of the Ninth International Workshop on Machine Learning*, 249-256. https://doi.org/10.1016/B978-1-55860-247-2.50037-1

Krishnaveni, S., Sivamohan, S., Sridhar, S. S., & Prabakaran, S. (2021). Efficient feature selection and classification through ensemble method for network intrusion detection on cloud computing. *Cluster Computing*, *24*(3), 1761-1779. https://doi.org/10.1007/s10586-020-03222-y

Mohammadi, S., Mirvaziri, H., Ghazizadeh-Ahsaee, M., & Karimipour, H. (2019). Cyber intrusion detection by combined feature selection algorithm. *Journal of Information Security and Applications*, *44*, 80-88. https://doi.org/10.1016/j.jisa.2018.11.007

Othman, S. M., Ba-Alwi, F. M., Alsohybe, N., & Al-Hashida, A. (2018). Intrusion detection model using machine learning algorithm on Big Data environment. *Journal of Big Data*, *5*(1), 1-12. https://doi.org/10.1186/s40537-018-0145-4

Pai, V., Devidas, & Adesh, N. D. (2021). Comparative analysis of Machine Learning algorithms for Intrusion Detection. *IOP Conference Series: Materials Science and Engineering*, *1013*. https://doi.org/10.1088/1757-899X/1013/1/012038

Revathy, R., & Lawrance, R. (2017). Classifying crop pest data using C4.5 algorithm. *2017 IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS)*, 1-6. 10.1109/ITCOSP.2017.8303122

Roy, B. (2020, April 5). *All about Feature Scaling. Scale data for better performance of... | by Baijayanta Roy*. Towards Data Science. Retrieved September 8, 2023, from https://towardsdatascience.com/all-about-feature-scaling-bcc0ad75cb35

Seth, N. (2020, November 9). *Entropy in Machine Learning: Definition, Examples and Uses*. Analytics Vidhya. Retrieved September 9, 2023, from https://www.analyticsvidhya.com/blog/2020/11/entropy-a-key-concept-for-all-data-science-beginners/

Sreenath, M., & Udhayan, J. (2015). Intrusion Detection System using Bagging Ensemble Selection. *2015 IEEE International Conference on Engineering and Technology (ICETECH)*. doi.org/10.1109/icetech.2015.7275015

Taher, K. A., Mohammed Yasin Jisan, B., & Rahman, M. M. (2019). Network Intrusion Detection using Supervised Machine Learning Technique with Feature Selection. *2019 International Conference on Robotics,Electrical and Signal Processing Techniques (ICREST)*. doi.org/10.1109/icrest.2019.864416

Tama, B. A., & Lim, S. (2021). Ensemble learning for intrusion detection systems: A systematic mapping study and cross-benchmark evaluation. *Computer Science Review*, *39*. https://doi.org/10.1016/j.cosrev.2020.100357

Zhou, Z.-H. (2012). *Ensemble Methods: Foundations and Algorithms*. CRC Press.

APPENDIX A

SOURCE CODE

```
# %% [markdown]
#
# ## Importing Libraries

# %% [code] {"jupyter":{"outputs_hidden":false}}
!pip install skrebate

# %% [code] {"jupyter":{"outputs_hidden":false}}
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import statsmodels.api as sm
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, VotingClassifier
from sklearn import tree
from sklearn import svm
from sklearn.svm import SVC
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_roc_curve
from sklearn.model_selection import GridSearchCV
from mlxtend.plotting import plot_confusion_matrix
from skrebate import ReliefF
from sklearn.feature_selection import chi2, mutual_info_classif, SelectKBest

# Turn off the warnings.
warnings.filterwarnings(action='ignore')
%matplotlib inline

# %% [markdown]
# ------
# ## Reading Data

# %% [code] {"jupyter":{"outputs_hidden":false}}
data = pd.read_csv("../input/nslkdd/KDDTest-21.txt" , sep = "," , encoding = 'utf-8')

# %% [markdown]
# ---------
# ## Exploring Data
```

```
# %% [code] {"jupyter":{"outputs_hidden":false}}
data

# %% [markdown]
# ---------
# ## Columns Modification

# %% [code] {"jupyter":{"outputs_hidden":false}}
Columns =
(['duration','protocol_type','service','flag','src_bytes','dst_bytes','land','wrong_fragment','urgent','hot',

'num_failed_logins','logged_in','num_compromised','root_shell','su_attempted','num_root','num_file_creat
ions',

'num_shells','num_access_files','num_outbound_cmds','is_host_login','is_guest_login','count','srv_count',

'serror_rate','srv_serror_rate','rerror_rate','srv_rerror_rate','same_srv_rate','diff_srv_rate','srv_diff_host_rat
e',

'dst_host_count','dst_host_srv_count','dst_host_same_srv_rate','dst_host_diff_srv_rate','dst_host_same_sr
c_port_rate',

'dst_host_srv_diff_host_rate','dst_host_serror_rate','dst_host_srv_serror_rate','dst_host_rerror_rate',
        'dst_host_srv_rerror_rate','attack','level'])

# %% [code] {"jupyter":{"outputs_hidden":false}}
data.columns = Columns

# %% [code] {"jupyter":{"outputs_hidden":false}}
data.head(10)

# %% [markdown]
# -------
# ## Data Description

# %% [code] {"jupyter":{"outputs_hidden":false}}
data.info()

# %% [code] {"jupyter":{"outputs_hidden":false}}
data.describe()

# %% [code] {"jupyter":{"outputs_hidden":false}}
data.nunique()

# %% [code] {"jupyter":{"outputs_hidden":false}}
data.max()

# %% [markdown]
# -----
# ### Exploring Responses
```

```
# %% [code] {"jupyter":{"outputs_hidden":false}}
Results = set(data['attack'].values)
print(Results,end=" ")

# %% [markdown]
# ## Classifying The Attack Results
# Converting the output labels to numeric values

# %% [code] {"jupyter":{"outputs_hidden":false}}
attack = data.attack.map(lambda a: 0 if a == 'normal' else 1)

data['attack_state'] = attack

# %% [code] {"jupyter":{"outputs_hidden":false}}
data.head(10)

# %% [code] {"jupyter":{"outputs_hidden":false}}
data.head(10)

# %% [markdown]
# --------
# ## Data preprocessing
# ### Chechking for Missing Data

# %% [code] {"jupyter":{"outputs_hidden":false}}
data.isnull().sum()

# %% [code] {"jupyter":{"outputs_hidden":false}}
data.isnull().sum()

# %% [markdown]
# -----
# - **There is no missing data**

# %% [markdown]
# --------
# ### Checking for Duplicates

# %% [code] {"jupyter":{"outputs_hidden":false}}
data.duplicated().sum()

# %% [markdown]
# ---------
# - **There is no duplicated data**

# %% [markdown]
# -----
# ### Handling Outliers

# %% [code] {"jupyter":{"outputs_hidden":false}}
data.shape
```

```
# %% [code] {"jupyter":{"outputs_hidden":false}}
data.plot(kind='box',subplots=1,layout=(8,5),figsize=(20,40))
plt.show()

# %% [markdown]
# -----
# - **No significant outliers in the data**

# %% [markdown]
# -------------
# ## Data Encoding

# %% [code] {"jupyter":{"outputs_hidden":false}}
data = pd.get_dummies(data,columns=['protocol_type','service','flag'],prefix="",prefix_sep="")

# %% [code] {"jupyter":{"outputs_hidden":false}}
LE = LabelEncoder()
attack_LE= LabelEncoder()
data['attack'] = attack_LE.fit_transform(data["attack"])

# %% [markdown]
# 25191----
# ## Data Splitting

# %% [code] {"jupyter":{"outputs_hidden":false}}
X = data.drop(['attack','level', 'attack_state'], axis = 1)
y = data['attack_state']

# %% [code] {"jupyter":{"outputs_hidden":false}}
X_train,X_test,y_train,y_test = train_test_split(X, y, test_size= 0.3 , random_state=42)

# %% [markdown]
# ---------
# ## Data Scaling

# %% [code] {"jupyter":{"outputs_hidden":false}}
mm_scaler = MinMaxScaler()
X_train = mm_scaler.fit_transform(X_train)
X_test = mm_scaler.fit_transform(X_test)

# %% [code] {"jupyter":{"outputs_hidden":false}}
X_train.shape, y_train.shape

# %% [code] {"jupyter":{"outputs_hidden":false}}
X_test.shape, y_test.shape

# %% [code] {"jupyter":{"outputs_hidden":false}}
X_train.min(), X_test.min()

# %% [markdown]
```

```
# ------
# ## Feature selection

# %% [markdown]
# ### Chi Square

# %% [code] {"jupyter":{"outputs_hidden":false}}
# apply SelectKBest class to extract top 50 best features
chi_sq = SelectKBest(score_func=chi2, k=50)
X_train = chi_sq.fit_transform(X_train,y_train)
X_test = chi_sq.fit_transform(X_test,y_test)

# %% [markdown]
# ### Information gain

# %% [code] {"jupyter":{"outputs_hidden":false}}
infoGain = SelectKBest(mutual_info_classif, k=15)
X_train = infoGain.fit_transform(X_train, y_train)
X_test = infoGain.fit_transform(X_test, y_test)

X_train.shape,X_test.shape

# %% [markdown]
# --------
# ### Data Modeling

# %% [markdown]
# #### Evaluating Function

# %% [code] {"jupyter":{"outputs_hidden":false}}
def Evaluate(Model_Name, Model_Abb, X_test, Y_test):

        Pred_Value= Model_Abb.predict(X_test)
        Accuracy = metrics.accuracy_score(Y_test,Pred_Value)
        Sensitivity = metrics.recall_score(Y_test,Pred_Value)
        Precision = metrics.precision_score(Y_test,Pred_Value)
        F1_score = metrics.f1_score(Y_test,Pred_Value)
        Recall = metrics.recall_score(Y_test,Pred_Value)

        print('-----------------------------------------------------\n')
        print('The {} Model Accuracy   = {}\n'.format(Model_Name, np.round(Accuracy,3)))
        print('The {} Model Sensitvity = {}\n'.format(Model_Name, np.round(Sensitivity,3)))
        print('The {} Model Precision  = {}\n'.format(Model_Name, np.round(Precision,3)))
        print('The {} Model F1 Score   = {}\n'.format(Model_Name, np.round(F1_score,3)))
        print('The {} Model Recall       = {}\n'.format(Model_Name, np.round(Recall,3)))
        print('-----------------------------------------------------\n')

        Confusion_Matrix = metrics.confusion_matrix(Y_test, Pred_Value)
        plot_confusion_matrix(Confusion_Matrix,class_names=['Normal', 'Attack'],figsize=(5.55,5),
colorbar= "blue")
        plt.savefig(f'{Model_Name}-cm')
```

```
        plot_roc_curve(Model_Abb, X_test, Y_test)
        plt.savefig(f'{Model_Name}-roc')

# %% [markdown]
# #### Logistic Regression

# %% [code] {"jupyter":{"outputs_hidden":false}}
LR= LogisticRegression()
LR.fit(X_train , y_train)

# %% [code] {"jupyter":{"outputs_hidden":false}}
LR.score(X_train, y_train), LR.score(X_test, y_test)

# %% [code] {"jupyter":{"outputs_hidden":false}}
Evaluate('Logistic Regression', LR, X_test, y_test)

# %% [markdown]
# ----------------
# ## SVM Classifier

# %% [code] {"jupyter":{"outputs_hidden":false}}
Linear_SVC = svm.LinearSVC(C=1)
Linear_SVC.fit(X_train, y_train)

# %% [code] {"jupyter":{"outputs_hidden":false}}
Linear_SVC.score(X_train, y_train), Linear_SVC.score(X_test, y_test)

# %% [code] {"jupyter":{"outputs_hidden":false}}
Evaluate('SVM Linear SVC Kernel', Linear_SVC, X_test, y_test)

# %% [markdown]
# -----
# # MLP classifier

# %% [code] {"jupyter":{"outputs_hidden":false}}
MLP = MLPClassifier(random_state=1)
MLP.fit(X_train, y_train)

# %% [code] {"jupyter":{"outputs_hidden":false}}
MLP.score(X_train, y_train), MLP.score(X_test, y_test)

# %% [code] {"jupyter":{"outputs_hidden":false}}
Evaluate('MLP-ANN', MLP, X_test, y_test)

# %% [markdown]
# -----
# # Ensembles

# %% [markdown]
# # Bagging with RandomForest
```

```
# %% [code] {"jupyter":{"outputs_hidden":false}}
RF = RandomForestClassifier(max_depth=2, random_state=0)
RF.fit(X_train, y_train)

# %% [code] {"jupyter":{"outputs_hidden":false}}
RF.score(X_train, y_train), RF.score(X_test, y_test)

# %% [code] {"jupyter":{"outputs_hidden":false}}
Evaluate('Random Forest', RF, X_test, y_test)

# %% [markdown]
# ----
# # Adaboost

# %% [code] {"jupyter":{"outputs_hidden":false}}
ADA_ensemble = AdaBoostClassifier(n_estimators=100, random_state=0)
ADA_ensemble.fit(X_train, y_train)

# %% [code] {"jupyter":{"outputs_hidden":false}}
ADA_ensemble.score(X_train, y_train), ADA_ensemble.score(X_test, y_test)

# %% [code] {"jupyter":{"outputs_hidden":false}}
Evaluate('Adaboost', RF, X_test, y_test)

# %% [markdown]
# ----
# # Voting

# %% [code] {"jupyter":{"outputs_hidden":false}}
Voting_ensemble = VotingClassifier(
        estimators=[('MLP', MLP), ('LR', LR)],voting="soft"
)
Voting_ensemble.fit(X_train, y_train)

# %% [code] {"jupyter":{"outputs_hidden":false}}
Voting_ensemble.score(X_train, y_train), Voting_ensemble.score(X_test, y_test)

# %% [code] {"jupyter":{"outputs_hidden":false}}
Evaluate('Voting', Voting_ensemble, X_test, y_test)
```