ROBERT GORDON UNIVERSITY ABERDEEN

SCHOOL OF COMPUTING

MSc Project Report

Student Name: Felix Oyekunle	Matriculation Number: 2213400
Supervisor: Dr. Harsha Kalutarage	Second Marker: Dr. Chris McDermott
Course: MSc Cyber Security	
Project Title: Anomaly-Based Network Intrusion Detection using Ensemble Learning	
Start Date: 02/02/2024	Submission Date: 25/04/2024

CONSENT

I agree

That the University shall be entitled to use any results, materials or other outcomes arising frommy project work for the purposes of non-commercial teaching and research, including collaboration.

DECLARATION

I confirm:

- **That the work contained in this document has been composed solely by myselfand that I have not made use of any unauthorised assistance.**
- **That the work has not been accepted in any previous application for a degree.**
- **All sources of information have been specifically acknowledged and all verbatimextracts are distinguished by quotation marks.**

Student Signature: Felix Oyekunle	Date Signed: 25/04/2024

ANOMALY-BASED NETWORK INTRUSION DETECTION USING ENSEMBLE LEARNING

FELIX OYEKUNLE

A REPORT SUBMITTED AS PART OF THE REQUIREMENTS FOR THE DEGREE OF
MSC IN COMPUTING: CYBER SECURITY
AT THE SCHOOL OF COMPUTING
ROBERT GORDON UNIVERSITY
ABERDEEN, SCOTLAND

April 2024

Abstract

Growing advancement in cyber-attacks in the form of network intrusion is a major threat to organizational networks and systems. Its efficient and timely detection has been a huge challenge to traditional intrusion detection methods owing to evolving complexity of modern Big Data Networks and increasing sophistication of cyber-attacks tactics. Deep learning and neural network models are computationally expensive in terms of required resources, training, and testing time, especially for a resource constrained environment like the SMEs. Completely tree-based stacking ensemble of DT, RF and XGBoost have performed appreciably well in this domain being tree, bagging and boosting models, however, this work proposed NB-augmented tree-based stacking ensemble models (NB, RF and XGBoost) that have proved by these research experiments (using CICIDS2017 benchmark dataset for NID), to offer improved predictive performance accuracy with comparatively lower computational cost for NIDS. As the performance of an ensemble model is a culmination from the quality of training datasets, base model diversity, ensemble and fusion method used, this study started off with exploring the impact of class-imbalance of training dataset on the performance of ensemble base models with the finding that NB, RF and XGBoost are sensitive to class imbalance issue especially in extreme instances like in the dataset. Bagged undersampling technique was employed to handle this issue. The impact of normalization and choice of normalization technique on the performance of ensemble base models were studied using Z-score and min-max. The results showed that while most ML models (like NB, RF and XGBOOST proved in this work) might be indifferent, in predictive accuracy to normalization data pre-processing, however, their training and testing times most especially are impacted by it. Z-score normalized training dataset-modelled base ML models were eventually used for our ensemble modelling. Mutual Information (MI) and SelectKBest with ANOVA F-value scoring function feature selection methods were deployed in this research to investigate the impact of choice of feature selection technique on the performance of ensemble base models. Mutual Information feature selection method proved superior to SelectKBest with ANOVA method which was more pronounced with NB and RF than XGBoost. The evaluation of these feature selection methods was done across full feature (79), 14, 12, 10, and 8 number of selected features. Results showed an achieved performance accuracy of 99.97%, 99.90%, and 94.62% with RF, XGBoost and NB respectively on 12 best selected features, and 99.49%, 99.45% and 94.60% respectively on eight (8) best selected features using MI. All these modeling experiments were conducted with hyperparameter tuning with cross-validation using RandomSearchCV method. The modelling of the base model was with 80/20 train-test split while the modelling of the ensemble meta-model was done with 70/30 to vary training and testing data subsets which proved to have the capability of improving generalization of performance results. Finally, the best performing NB, RF and XGBoost models were put into ensemble learning using stacking ensemble with meta-learning fusion methods. A thorough evaluation was done on nine (9) ensemble models using performance metrics such as confusion metrics, accuracy, precision, recall, F1-score, False Alarm Rate (FAR), ROC-AUC curve, ensemble model training and testing times. After comparative analysis with literature, two NB-augmented tree-based stacking ensemble models were proposed which achieved performance accuracy of 99.99% and 99.46% at 12 and 8 selected number of features respectively, and with comparatively low training and testing times.

Acknowledgement

I would like to take this opportunity to extend my sincere appreciation to my supervisor, Dr. Harsha Kalutarage, for his invaluable guidance throughout this project. His unwavering support and encouragement make this possible.

I am also grateful to the lecturers who taught me during my master's program. Their knowledge transfer skills are superb and have helped me develop my research skills.

Finally, I extend my thanks to my family and friends for their encouragement, patience, and understanding, especially during the demanding period of my thesis completion. Their support and understanding really motivated me.

Table of Contents

List of Tables

List of Figures

CHAPTER 1: INTRODUCTION

This opens up a comprehensive background on this study coupled with motivation. In addition, a structural breakdown of the content of the rest of this report is also highlighted.

1.1 Background

AV-TEST **(2024)** reported that a lot of businesses and organizations had experienced cyber-attacks in magnitude of over 1.2 billion (Figure **1.1**) in recent years and this number is still growing. Organizations and individuals have suffered a great deal of harm and huge economic losses owing to these malicious attacks. Also recently, Juniper Research **(2023)** affirmed that the number of data breaches had been tripling in the last couple of years and the annual cost of these breaches to businesses would be over 91 trillion USD by the year ending 2028 worldwide. Emerging Big Data network environments such as distributed cloud computing network, Internet-of-Things (IoT)-based and smart networks, plus continuous connectivity, and variations in the types and use of networks vis-a vis continuously evolving intelligent intrusion techniques have exacerbated the vulnerabilities of today's information systems (Jemili, Meddeb and Korbaa **2023**). In addition, BYOD (Bring Your Own Device) and remote work access to corporate network have also opened many opportunities to intruders. This has led to many new network and system security challenges leaving intrusion detection an open issue till today (Chen et al. **2022**). Hence, it is imperative to advance in the sophistication of cybersecurity mechanisms, especially in the area of more robust and efficient intrusion detection solutions to match the increasing complexity of cyber-attacks and thus, ensure system security.

Network security systems are a key component of cyber security framework which focuses on technologies designed to protect networks, programs, computers, and data from attack, modification, and unauthorized access, plus host-based computer system security (Carley **2020**). All of these are made of Intrusion Detection Systems (IDS) developed to identify any anomalous activity by threat actors tampering with network resources and computer systems, stealing sensitive and valuable network resources (Denial of Service Attack), or malicious actions to destabilize normal network operations (Saeed **2022**). Network Intrusion Detection Systems (NIDS) have the ability to prevent, detect and respond to cyber-attacks (Belouch, Hadaj and Idhammad **2018**). A conceptual representation of a NIDS is illustrated in Figure **1.2** (Admkie and Tekle **2020**).

The taxonomy system of IDS as adapted from Liu and Lang (**2019**) classifies IDS based on two methods- Detection Method and Data Source-Based Method as shown in Figure **1.3**. IDS classification based on the detection method are categorised into anomaly-based and misuse or signature-based detections, while classification based on the attack data sources are categorised into host-based intrusion detection system (HIDS) and network-based intrusion detection system (NIDS) which are two-level intrusion detection deployments (Abbas et al. **2023**; Liu and Lang **2019**).

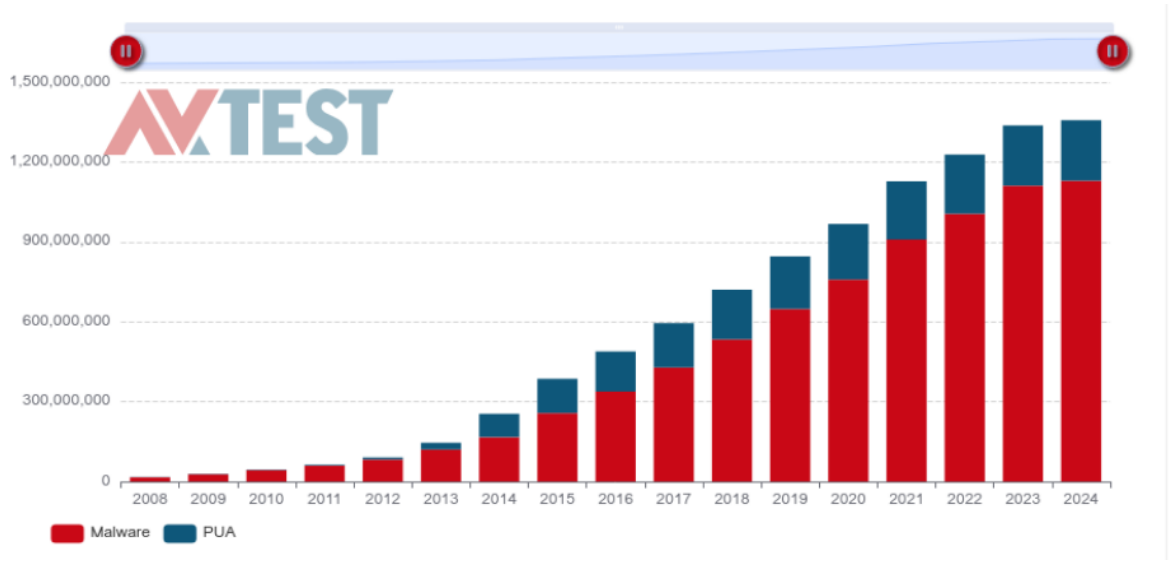


Figure **1.1**: AV-TEST Cyber Attack Trend Report (AV-TEST 2024)

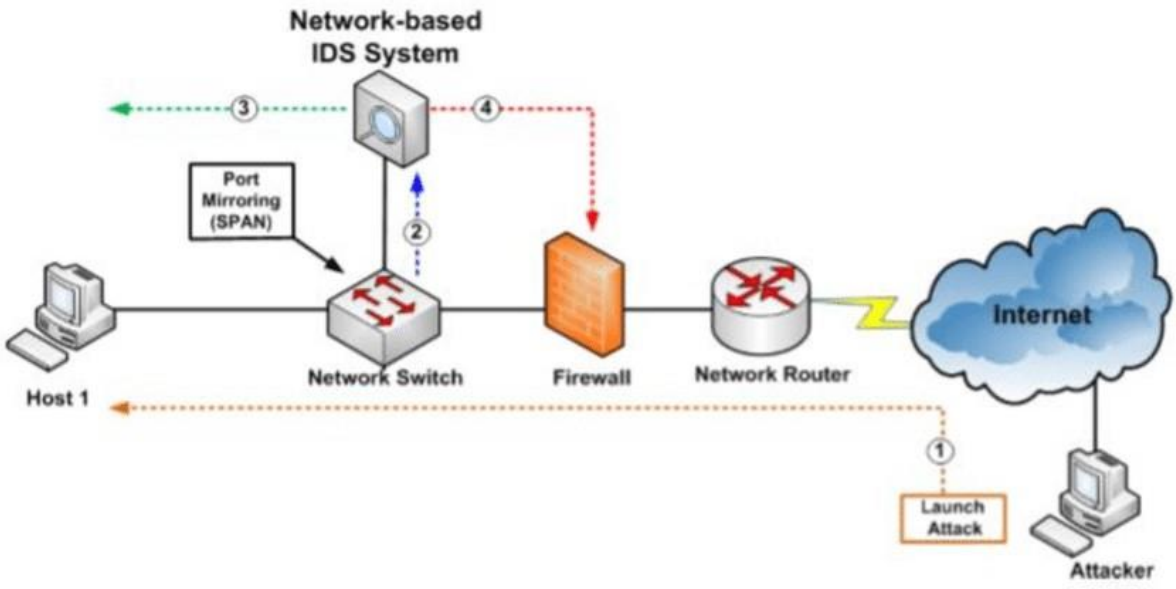


Figure **1.2**: Network-Based Intrusion Detection System (NIDS) (Admkie and Tekle **2020**)

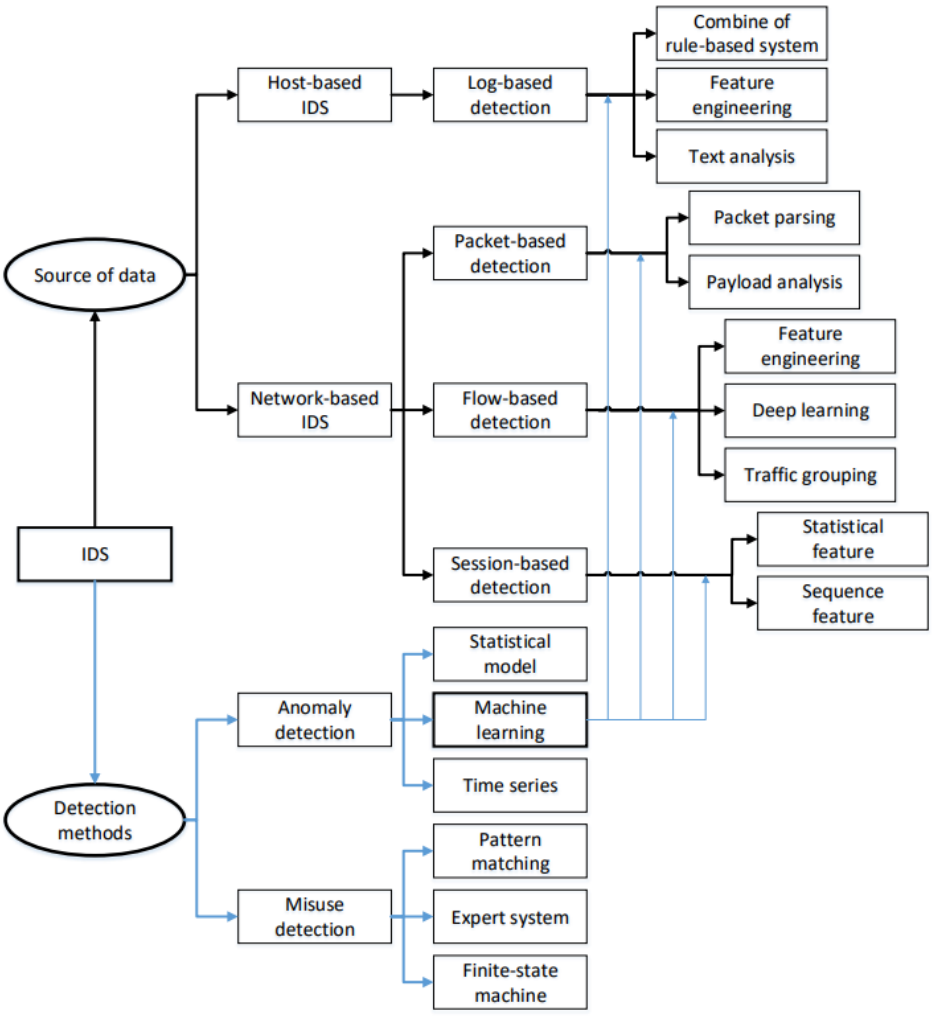


Figure **1.3**: Taxonomy System of IDS (Liu and Lang **2019**)

A misuse or signature-based intrusion detection system relies on predefined vulnerabilities and malware signatures which are limited and fail in zero-day attack detection whereas anomaly-based intrusion detection systems possess predictive capability to detect attacks that were not predefined through detection rules. Although their performances are undermined by instances of false positives, yet they give real-time awakening to administrators of the possibility of attack, thus, perform comparatively well more than signature-based IDS (Xu et al. **2021**; Zhang et al. **2022**). Table **1.1** as adapted from Liu and Lang **(2019)** delineates the differences between misuse or signature-based IDS and anomaly-based IDS.

Table **1. 1**: Difference between Misuse/Signature-Based and Anomaly-Based IDSs (Liu and Lang **2019**)

	Misuse/Signature-Based IDS	**Anomaly-Based IDS**
Detection performance	Low false alarm rate; High missed alarm rate	Low missed alarm rate; High false alarm rate
Detection efficiency	High, decrease with scale of signature database	Dependent on model complexity
Dependence on domain knowledge	Almost all detections depend on domain knowledge	Low, only the feature design depends on domain knowledge
Interpretation	Design based on domain knowledge, strong interpretative ability	Outputs only detection results, weak interpretative ability
Unknown attack detection	Only detects known attacks	Detects known and unknown attacks

Host-based IDS (HIDS) are antivirus software, and host-based firewall for discovering, determining, and identifying illegal duplication, use, alteration, and destruction of data. They can identify intrusion positionally precisely to initiate responses based on the anomalous behaviour of certain host objects such as sensitive files, programs, or system ports. They are unable to detect network attacks because they rely on the host system and its operating system (OS). Conversely, Network-based IDS (NIDS) can detect types of protocol and network attacks, though, they monitor only the network traffic passing through network segments. Network traffic packets are units of network communication whose flow is within time stamps or a time window, while a network traffic session is a packet sequence combination of network information- client IP, client port, server IP, server port, protocol (Liu and Lang **2019**). Table **1.2** highlights the differences between host-based IDS and network-based IDS.

As already shown in Figure **1.3**, Machine Learning (ML) is a technique used in the development of more reliable and efficient anomaly-based IDS which requires learning from experience dataset in order to make predictive detection of intrusion attack. When it is supervised, ML algorithms are trained on labelled dataset of known attacks and normal activity, and then use the knowledge to detect new and potentially anomalous behaviour. Whereas, when it is unsupervised, ML algorithms learn and classify patterns on their own without being pre-trained on specific attack (Jemili, Meddeb and Korbaa **2023**). This has made detection of intrusion quicker and more accurate leading to more timely and effective response (Hussein et al. **2019**). However, ML-based IDS performance has been plummeting of late owing to emergence of Big Data networks and increasing computational complexity (Umar et al. **2024**). Feature selection techniques for high dimensional data reduction have been used to manage this issue and improve the performance of ML-based IDS (Bridges et al. **2020**; Albulayhi et al. **2022**). Furthermore, the use of normalization in handling IDS dataset features with large value ranges has proven to be very impactful on the implementation of ML-based IDS models, reducing learning time and improving IDS model performance (Kasongo and Sun **2020;** Wang et al. **2009**). The

performance of an ML-based IDS model depends not only on feature selection and normalization techniques, but also the dataset, and particularly the machine learning algorithms used (Umar et al. **2024**). Much research has been conducted in the application of machine learning techniques (simple learning) for improving the performance of IDS (Saxena, Sinha and Shukla **2017**; Sarker et al. **2020**). However, these systems are limited due to their use of single classifiers and consequently being unable to detect and prevent severe attacks. Combining multiple ML algorithms (simple learners like NB, DT, KNN, LR, SVM, etc) in ensemble learning has proven to improve the robustness, generalization, and performance of anomaly-based NIDS (Hansen and Salamon **1990**). This underscores the importance of exploring ensemble-based IDS from various respects (Pham et al. **2018**).

Table **1. 2**: Difference between Host-Based IDS and Network-Based IDS (Liu and Lang **2019**)

	Host-Based IDS	**Network-Based IDS**
Source of data	Logs of operating system or application programs	Network traffic
Deployment	Every host; Dependent on operating systems; Difficult to deploy	Key network nodes; Easy to deploy
Detection efficiency	Low, must process numerous logs	High, can detect attacks in real time
Intrusion traceability	Trace the process of intrusion according to system call paths	Trace position and time of intrusion according to IP addresses and timestamps
Limitation	Cannot analyze network behaviours	Monitor only the traffic passing through a specific network segment

Ensemble learning is a machine learning technique where multiple base ML models are combined into a single, more powerful model leveraging the aggregated strengths and predictions of the individual base classifiers (Jemili, Meddeb and Korbaa **2023**). The resulting predictions are usually more reliably accurate and robust than that of a single model (Chand et al. **2016**). Classifier ensembles better handle bias-variance trade-off complexity more effectively than individual models (Oza and Tumer **2008**). one of the drawbacks of simple machine learning is overfitting where an ML model fits to training data and fails to generalize to unseen data. Ensemble learning reduces overfitting by combining the different biases of multiple models, reducing the overall effect of every single model's overfitting (Arass **2019**). Ensemble learning combines the strengths of various simple learners. Therefore, it addresses the several difficulties in determining the best single model selection per problem domain, and thus improving performance through model diversity (Hussain and Rahim **2015**).

1.2 Motivation

Growing advancement in cyber-attacks in the form of network intrusion is a major threat to organizational networks and systems. Its efficient and timely detection has been a huge challenge to traditional intrusion detection methods owing to evolving complexity of modern Big Data Networks and increasing sophistication of cyber-attacks tactics. The situation is even worse for Small and Medium Enterprises (SMEs) of developing countries which suffer from scarce resources and limited investment in the deployment of powerful security controls and solutions (Abbas et al. **2023**). Several detection solutions such as signature-based or misuse detection and anomaly-based machine learning techniques have been developed but with plummeting performance in terms of robustness, adaptability, and predictive accuracy in the face of ever-changing attack vectors, expanding attack landscapes and heterogeneity of network environments-traditional, cloud, smart and IoT networks (Jemili, Meddeb and Korbaa **2023**). By leveraging the power of ensemble learning which operates by combining the strengths, and compensating the weaknesses of individual traditional simple learners (ML Models) to produce more reliably accurate predictions, improved robustness, and adaptability of anomaly-based network intrusion detection system (NIDS) to varying modern network environments, we shall be able to continuously innovate more efficient and cost-effective anomaly-based NIDS in terms of performance, required resources and training time. The main motivation to conduct this research work is the prospects of making significant knowledge contributions towards the development of more efficient and cost-effective anomaly-based NIDS using ensemble learning technique.

1.3 Content of the Rest of the Report

The remainder of this report is structured as follows:
Chapter 2: The Literature Review chapter offers critical overview of existing research works relating to anomaly-based NIDS using traditional machine learning and ensemble learning techniques plus research gaps.

Chapter 3: The Project Specification chapter outlines the project objectives, functional and non-functional requirements, the methodology employed, considerations of legal, ethical, social, and environmental factors, identification of security risks, and mitigating measures. It also addresses adherence to the code of practice and industry standards as related to this research work.

Chapter 4: The Design and Implementation chapter covers the design alternatives, justification for chosen design, and system specifications. It also gives a step-by-step account of the model implementation workflow which comprises data understanding, data pre-processing, feature selection, model training, optimization with hyperparameter tuning and cross-validation settings.

Chapter 5: The Evaluation chapter offers a thorough analysis of the outcomes derived from the various experimentations on both the base ML Models, and the Ensemble Models, plus the evaluation of the proposed ensemble learning models based on several measures.

Chapter 6: The Conclusion and Future Work chapter provides a comprehensive overview of this research work, highlighting its findings and achievements, limitations, and potential opportunities for future research.

CHAPTER 2: LITERATURE REVIEW

In the recent past, the industry and academics have been researching on intrusion detection using machine learning techniques. This chapter will be providing a critical review of research works relating to network intrusion detection systems (NIDS) using the traditional Machine Learning, and the Ensemble Learning techniques. In addition, a section would be dedicated to discussing research gaps capturing the target objectives of this project work.

2.1 NIDS using Traditional Machine Learning Techniques

In a study by Alghushairy et al. **(2024)**, an SVM-based Network Outlier Detection System (SVM-NODS) was proposed for efficient anomaly-based network intrusion detection. The NSL-KDD and CICIDS2017 intrusion datasets were employed to build and evaluate the proposed model which recorded an outstanding detection accuracy of 99.25% with 22 selected features. Network features were comparatively normalized by min-max and Z-Score methods, while the relevant features were comparatively selected individually by the principal component analysis (PCA) and correlated features selection (CFS) techniques. The Genetic algorithm (GA) was employed to tune the SVM's RBF control parameters to improving the robustness of the proposed NODS model. However, this performance accuracy can still be improved with other ML techniques to a much higher performance accuracy in the face of today's increasing attack sophistication and complexity. In another study by Vibhute et al. **(2024)** that same year, a network intrusion detection was implemented separately on three machine learning models- support vector machine (SVM), logistic regression, and K-nearest neighbour's (KNN) with 87.58, 88.86, and 98.24 validation accuracies respectively using NSL-KDD Benchmark Dataset. KNN model produced satisfactory performance in detecting and classifying network anomalies with a 98.24% success rate. Omar **(2023)** conducted a project on harnessing the power and simplicity of Decision Trees to detect IoT Malware. MaleVis publicly available dataset was used in the experiment with entropy decision procedure as the feature selection method. The solution achieved 97.23% precision and 95.89% recall, specificity of 96.58%, F1-score of 96.40%, and an accuracy of 96.43. This work is limited in that the dataset, MaleVis is focus on Malware Intrusion and cannot generalize for other forms of intrusion attack characterization like DDOS attack, Mirai botnet, etc.

In a study by Prasad, Tripathi and Dahal **(2022),** a Bayesian was applied to classify the behaviour of mobile nodes using incoming network packets. The feature ranking method was applied to extract a set of more significant features that provided a lightweight training set. Experimental results showed the average detection accuracy as 94.37% for blackhole intrusion attack and 99% for wormhole intrusion attack. However, the study suffered some limitations in the computational complexities and low memory capacity of the mobile nodes. Gadal et al. **(2022)** proposed an anomaly-based network intrusion detection model that used K-mean clustering with sequential minimal optimization (SMO) for intrusion classification using NSL-KDD network intrusion dataset. Consistency-subset level and genetic search algorithm were used for feature selection by SMO, and the proposed model was confirmed to have improved detection rate with accuracy of 97.40%, with 22 NSL-KDD features. However, the solution was observed to be limited in pattern and anomaly detection in massive datasets in real-time, requiring practically huge running time and processing power. In the same year, Bong and Kim **(2022)** analysed the change in performance of Gaussian Naïve Bayes model according to the smoothing factor. The experiment was based on the NSL-KDD dataset which showed that the performance of Gaussian NB model changed from at least 38.80% to 94.53% depending on the smoothing factor (0 and 1e-01

respectively), and 87.99% at default value. The experiments also revealed that Gaussian NB model can be used in the future for zero-day intrusion detection system. It also clarifies that the smoothing factor of the Gaussian NB model determines the shape of gaussian distribution that is connected to the probability. However, it is very much likely that the model performed highly due to overfitting on the dataset. Therefore, it is necessary to learn a high-quality dataset that reflects the latest attacks and network techniques so as to improve the model when applying to a practical environment.

In another study by Guezzaz et al**. (2021)** a reliable network intrusion detection approach using decision tree with enhanced data quality using entropy decision feature selection method was presented. The DT classifier was trained and evaluated on the NSL-KDD and CICIDS2017 datasets separately, achieving 99.42% and 98.80% accuracy with NSL-KDD and CICIDS2017 datasets, respectively, but still completely tree-based. In the same year, Jie and Shan **(2021)** proposed an effective intrusion detection framework based on SVM with naïve Bayes feature embedding as a data quality improvement technique. Experiments on multiple datasets in network intrusion detection domain validate achieved good and robust performances with 93.75% accuracy on UNSW-NB15 dataset, 98.92% accuracy on CICIDS2017 dataset, 99.35% accuracy on NSL-KDD dataset and 98.58% accuracy on Kyoto 2006+ dataset. Furthermore, this method possesses huge advantages in terms of high accuracy, high detection rate, low false alarm rate and rapid training speed, which demonstrates the effectiveness of naïve Bayes feature embedding in facilitating intrusion detection. However, little or no concern was given to handling severe class imbalance issue inherent in modern day networks.

Table **2.1** provides a compact view of reviewed research works on the most optimum performance- category using traditional machine learning techniques (simple learning) and focusing on anomaly-based network intrusion detection. It highlights the proposed machine learning models, specific feature selection method, datasets used, and maximum performance accuracy achieved in each study.

Table **2. 1:** Compact view of reviewed research works on the most optimum performance category using traditional machine learning techniques.

Author & Year	Traditional ML Model	Dataset	Feature Selection Method	Maximum Achieved Prediction Accuracy
Alghushairy et al. (2024)	SVM and NB	NSL-KDD and CICIDS2017	PCA and CFS	99.25%
Vibhuti et al. (2024)	SVM, LR, and KNN	NSL-KDD	Random Forest-based	98.24%
Omar (2023)	DT	MaleVis	Entropy decision	96.43%
Gadal et al. (2022)	K-mean clustering	NSL-KDD	Genetic Search Algorithm	97.40%
Prasad, Tripathi and Dahal (2022)	Bayesian rough set classifier	MANETs	Feature ranking	94.37%
Bong and Kim (2022)	Gaussian NB	NSL-KDD	All Features	94.53%
Guezzaz et al. (2021)	DT	NSL-KDD and CICIDS2017	Entropy decision	99.42%
Jie and Shan (2021)	SVM	UNSW-NB15 CICIDS2017 NSL-KDD	NB-Feature Embedding	99.35%

The performance prediction accuracies (Maximum of 99.42% using DT) in the above Table **2.1** are not as high as efficient enough for prevailing complex and modern big data network intrusion attacks. Besides, simple learning models are vulnerable to overfitting, where a model fits to the training data too much to fail to adapt to unseen data (Jemili, Meddeb and Korbaa 2023). Ensemble learning promises to mitigate this.

2.2 NIDS using Ensemble Learning Techniques

In a study by Umar et al**. (2024),** Random Forest as a bagging ensemble model of Decision Trees (DTs) was explored and achieved the best performance on both NSL-KDD and UNSW-NB15 datasets with prediction accuracies of 99.87% and 98.5% respectively. An in-depth analysis of the effects of feature selection and normalization (decision tree wrapper-based, and min-max normalization) techniques on various IDS models built using two IDS datasets NSL-KDD and UNSW-NB15, and five different ML algorithms was carried out in this research work. The five algorithms were support vector machine, k-nearest neighbours, random forest, naive bayes, and artificial neural network. The results showed that both normalization and feature selection can impact IDS modelling. In another research work (Jemili, Meddeb and Korbaa **2023**), a stacking ensemble-based hybrid model was experimented on Random Forest, XGBoost, and Decision Trees on diverse datasets, N-BaIoT, NSL-KDD, and CICIDS2017. The performance accuracies were 96.36% for

CICIDS2017, 98.21% for NSL-KDD, 96.39% for N-BaIot with Average total accuracy of 97%. This work offers promising opportunity for bolstering intrusion classification in Big Data network environment in the face of evolving threats. Further that year, GANDHI et al. **(2023)** proposed an ensemble learning-based network intrusion detection system with Random Forest, AdaBoost, and LGBM ensemble models combined with a soft voting scheme. The proposed model achieved an accuracy of 99.90% on the NSL-KDD and UNSW-NB15 dataset benchmark datasets. However, these were all tree-based ensemble models. In a study by Abbas et al. **(2023),** an ensemble model based on RF, MLP and SVM classifiers using random forest-recursive feature elimination (RF-RFE) method for selection of 15 features was proposed to increase the predictive performance of intrusion detection system (IDS). The evaluation of the proposed ensemble machine leaning model shows 99%, 98.53% and 99.90% overall accuracy for NSL-KDD, UNSW-NB15 and CSE-CIC-IDS2018 datasets, respectively. The outcome of the research is significant in contributing to the performance efficiency of intrusion detection systems and developing secure systems and applications. Rashid et al. **(2022)** proposed a tree-based stacking ensemble model with selectkbest feature selection approach (20 features) to detect intrusion. The stacking ensemble model consists of DT, RF, and XGBoost, and the effectiveness of the proposed model was tested with NSL-KDD and UNSW-NB15 datasets. Performance evaluation showed that the proposed model achieved 99.90% and 95.26% accuracy for NSL-KDD and UNSW-NB15 datasets. The proposed stacking model took advantage of the multiple heterogeneous classifiers and thus, was capable of overcoming potential limitations of classifiers involving single or homogeneous ML models, demonstrating better performance in identifying unseen data.

Gupta, Jindal and Bedi **(2021)** in a study to address the challenges on increasing false alarms of NIDS proposed Cost-Sensitive Deep Learning and Ensemble algorithm (CSE-IDS) comprising a Deep Neural Network (DNN), Extreme Gradient Boosting (XGBoost), and Random Forest (RF). The performance accuracy of the ensemble on CICIDS2017 datasets was 92% with reduction in false alarms. Moreover, in the same year, Singh and Ranga **(2021)** presented an effective network-based intrusion detection model using an ensemble-based machine learning approach with four classifiers (Boosted tree, bagged tree, subspace discriminant, and RUSBooted) along with a voting scheme. CICIDS2017 and CloudSim datasets were used for simulation and testing of the proposed model. The model proved to be efficient in the identification of intrusions in the cloud environment with a high rate of detection and generation of minimal false alarm warnings. The implementation results showed an accuracy of 97.24%. In another research work by Krishnaveni et al. **(2021)** a majority voting ensemble model comprising SVM, NB, Logistic Regression, and DT applied on univariate ensemble filter feature selection method (UEFFS) over three intrusion datasets, NSL_KDD dataset, Kyoto 2006+, Real-time Honeypot Dataset (2018) was investigated. The proposed method achieved a considerable amount of performance improvement in accuracy with 98.89% and robustness of various classification tasks, thus contributing to Feature Selection in intrusion detection system.

In Rajadurai and Gandhi **(2020),** a stacked ensemble of Random Forest and Gradient Boosting was proposed for a wireless network-based IDS. The proposed model yielded an accuracy of 91.09% over NSL-KDD dataset. It was confirmed to have increased attack detection rates and reduced the training time compared to single decision algorithm, and also showed better accuracy compared with ML algorithms such as ANN, CART, random forest, and SVM. Further that year, Rajagopal, Kundapur and Hareesha **(2020)** in their research work, presented a stacking ensemble model with

logistic regression (LR), K-nearest neighbor (KNN), and random forest (RF) as the base classifiers, while support vector machine (SVM) was the meta-classifier over two heterogeneous datasets- UNSW NB-15 (a packet-based dataset), and UGR'16 (a flow-based dataset), captured in emulated as well as real network traffic environment, respectively. The evaluation results were 97% accuracy and 94% accuracy on real-time dataset, UGR'16, and emulated one, UNSW NB-15. This indicated that stacking ensemble was capable of generating superior predictions. In another research, Tama, Comuzzi and Rhee **(2019)** proposed an improved IDS based on hybrid feature selection and two-level classifier ensemble based on two meta- learners (rotation forest and bagging). The hybrid feature selection technique comprises three methods- particle swarm optimization, ant colony algorithm, and genetic algorithm for reducing NSL-KDD and UNSW-NB15 datasets. On the NSL-KDD dataset, the proposed classifier showed 85.8% accuracy, 86.8% sensitivity, and 88.0% detection rate.

Illy et al. **(2019)** developed multi-expert ensembles that employ voting functions with the NSL-KDD dataset. It was discovered that ensemble Bagging with J48 as the base classifier was effective for binary classification using voting, while ensemble of RF, MLP, Boosting, and Bagging, performed well for multi-class classification. Overall accuracy of 85.81% and 84.25% were achieved for the benign and attack classification, respectively. A deployment architecture was also proposed where anomaly detection was performed in fog nodes to allow faster detection and response, and attack classification in the cloud would benefit from more resources to run a more complex ensemble for better classification that guides the intrusion prevention tasks. In the same year, Gu et al. **(2019)** proposed an effective intrusion detection framework (DT-EnSVM) based on SVM ensemble with feature augmentation, feature data Density-ratio Transformation (DT). The data quality-improvement technique (DT) was used to reconstruct the original features to provide high-qualified and concise training data. The detection accuracy of the proposed ensemble model was 99.41%. Pham et al. **(2018)** carried out a study to improve performance of IDS using ensemble methods and feature selection. The proposed model was built on bagging of boosting algorithm (AdaBoost-J48 classifier). The experimental results showed that the bagging ensemble model with J48 as the base classifier produced high classification accuracy with low False Alarm Rate (84.25% accuracy and 2.79% FAR) when working with the subset of 35 selected features of NSL-KDD dataset using Gain Ratio (GR) technique. However, this work can be better validated with another varying valid NIDS dataset like CICIDS2017 and improved with diverse ML models not limited to tree-based classifiers.

Table **2.2** provides a compact view of reviewed research works on the most recent and optimum performance-category models using ensemble learning, focusing on anomaly-based network intrusion detection. It highlights the proposed ensemble models, ensemble methods, fusion methods, specific feature selection method, number of used features (if available), datasets used, and maximum achieved performance accuracy in each study. This is also to make obvious the commonalities in the design of existing best preforming ensemble of traditional ML models, as a result, unveiling likely research gaps.

Table **2. 2**: Compact view of reviewed research works on the most recent and optimum performance- category using ensemble learning of traditional ML models.

Author & Year	Ensemble Model	Ensemble Method	Fusion Method	Dataset	Feature Selection Method	Number of Used Features	Maximum Achieved Accuracy
Umar et al. (2024)	Random Forest	Bagging	Voting	NSL-KDD and UNSW-NB15	Decision tree wrapper-based approach	-	99.87%
Jemili, Meddeb and Korbaa (2023)	Random Forest, XGBoost	Stacking	Meta-learning	CICIDS2017, NSL-KDD, N-BaIot	Feature Importance	12, 34, 115	98.21%
GANDHI et al. (2023)	Random Forest, AdaBoost, and LGBM	Bagging and Boosting	Voting	NSL-KDD and UNSW-NB15	-	-	99.0%
Abbas et al. (2023)	RF, MLP and SVM	Bagging	Voting	NSL-KDD, UNSW-NB15, CSE-CIC-IDS2018	Random Forest-Recursive Feature Elimination (RF-RFE)	15	99.90%
Rashid et al. (2022)	DT, RF, and XGBoost	Stacking	Meta-learning	NSL-KDD and UNSW-NB15	SelectKBest	20	99.90%
Gupta, Jindal and Bedi (2021)	DNN, XGBoost, RF	Stacking	Meta-learning	CICIDS2017	-	-	92.0%
Singh and Ranga (2021)	Boosted tree, bagged tree, subspace discriminant, and RUSBooted	Boosting, and Bagging	Voting	CICIDS 2017 and CloudSim	-	-	97.24%
Krishnaveni et al. (2021)	SVM, NB, LR, DT	Bagging	Voting	NSL-KDD, Kyoto, Honeypot (2018)	Univariate Ensemble Filter Feature Selection (UEFFS)	-	98.89%
Rajadurai and Gandhi (2020)	RF and Gradient Boosting	Stacking	Meta-learning	NSL-KDD	-	-	91.09%
Rajagopal, Kundapur and Hareesha (2020)	LR, KNN, RF, and SVM	Stacking	Meta-learning	UNSW NB-15, and UGR'16	Information gain (IG) and Hashing	11	97.0%

Author & Year	Ensemble Model	Ensemble Method	Fusion Method	Dataset	Feature Selection Method	Number of Used Features	Maximum Achieved Accuracy
Tama, Comuzzi and Rhee (2019)	Rotation Forest and Bagging	Stacking	Meta-learning	NSL-KDD and UNSW-NB15	Particle Swarm Optimization, Ant Colony Algorithm, and Genetic Algorithm	37, 19	85.80%
Illy et al. (2019)	RF, MLP, Boosting,	Bagging	Voting	NSL-KDD	-	38	85.81%
Gu et al. (2019)	SVM Ensemble	Boosting	Voting	-	Feature Data Density-ratio Transformation (DT)	-	99.41%
Pham et al. (2018)	AdaBoost (J48)	Bagging	Voting	NSL-KDD	Gain Ratio	35	84.25%

Stemming from the summary table of reviewed research works above (Table **2.2)**, the most prominent commonalities in the works are that most optimally performing ensemble models of traditional ML models are tree-based (DT, RF, Boosting- 99.90%). Aside from Neural Network ML types (MLP), all recorded performance accuracies are less than 99.99%, and there was no exploration of model performance below 11 selected features (Major Research Gaps).

2.3 Research Gaps

These highlight areas of opportunities for research from reviewed existing works.

2.3.1 Limited Comparative Analysis of Impacts of ML Data Pre-processing Techniques on Performance of Base Classifiers:

Normalization and class imbalance handling techniques are essential tools when dealing with modern Big Data network intrusion datasets (Gupta, Jindal and Bedi **2021**; Wang et al. **2009**). In addition, Mohammed and Kora **(2023)** affirm that the success of an ensemble model can also be influenced by the properties of datasets used in training. It is obvious that most research works do not carry out comparative analysis of the effect of choice of dataset pre-processing techniques on the performance of ensemble baseline models before deciding on one.

2.3.2 Limited Comparative Analysis of Impacts of Choice of Feature Selection Methods on Performance of Base Classifiers:

Choice of feature selection method plays a critical role in optimum base model performance as the largeness of dataset's features does not necessarily yield improved performance (Jemili, Meddeb and Korbaa **2023**). While a lot of studies have employed various feature selection, and dimensionality reduction methods, there is quite a few that carried out comparative analysis of the impact of choice of methods on the performance of ensemble baseline classifiers before deciding on one.

2.3.3 Exploration of Minimum Feature Dimensionality:

Construction of effective AI models with the smallest possible data to successfully move modern big data network intrusion attack dataset to small data environment has always been an open opportunity for research (ENISA **2023**). While some studies have experimented with reduced number of features for modelling, a lot of other works seem to stop once they get a comparatively improved performance based on existing works without exploring the performance of base classifiers with fewer number of features, thus, losing out from the possibilities of spotting comparatively better model performance at the least possible cost of resources, training and testing times.

2.3.4 The Prevalence of Optimum Model Performance in Completely Tree-Based Ensembles:

The goal of ML-based IDS is to achieve consistently almost 100% performance accuracy with zero misclassification (zero false positive and negative). However, there is a noticeable trend in literature of the dominance of completely tree-based models (DT, RF, XGBoost) being presented (aside Neural Network and Deep learning) as gearing towards this goal. This prevalence leaves open the possibilities of improved performance by augmenting a fully tree-based model with other traditional ML algorithms (like NB, SVM, etc) especially in the case of ensemble learning.

CHAPTER 3: PROJECT SPECIFICATION

This outlines the project's aim and objectives, functional and non-functional requirements, the methodology employed, considerations of legal, ethical, social, and environmental factors, identification of security risks, and mitigating measures. It also addresses adherence to the code of practice and industry standards as related to this research work.

3.1 Aim and Objectives

The aim of this project is to review the performance of current ensembles of traditional machine learning models in anomaly-based Network Intrusion Detection (NID) and propose an ensemble model with improved accuracy level. The objectives are as follows:

1. Review existing literature on network intrusion detection using ensemble of traditional machine learning techniques.
2. Evaluate the impact of class imbalance on the performance of ensemble baseline ML models.
3. Examine the impact of training dataset normalization techniques on the performance of ensemble baseline ML models. Analyse the performance of the ensemble baseline ML models under normalized and unnormalized training datasets.
4. Evaluate the impact of feature selection methods on the performance of ensemble baseline ML models. Analyse the performance of the ensemble baseline ML models across range of number of selected features fewer than 16.
5. Develop ensemble models from the best performing traditional baseline ML models under balanced and unbalanced training datasets, plus hyperparameter tuning and cross-validation. Evaluate the performance of the ensemble models under balanced and unbalanced training datasets.
6. Analyse the performance of the developed ensemble models trained with balanced dataset across range of number of selected features. Assess the performance of the ensemble models based on training and testing times, confusion matrix, and metrics such as accuracy, precision, recall, F1 score, False Alarm Rate (FAR), and ROC-AUC curve. Propose the most efficient ensemble model with superior performance than the reviewed literature.

3.2 Functional and Non-Functional Requirements

The MOSCOW prioritization technique is employed in this work to delineate the functional and non-functional requirements based on their level of significance and criticality to the successful delivery of this project. This method would help prioritize and categorize these requirements into four main groups whose initial letters form the abbreviation, MoSCoW: Must-have, Should-have, Could-have, and Won't-have (Product Plan **2024**; Volkerdon **2024**). The functional and non-functional requirements are presented in Table **3.1** and Table **3.2** respectively.

3.2.1 Functional Requirements

The functional requirements are presented in Table **3.1**.

Table **3. 1**: Functional Requirements Using MoSCoW Analysis

S/N	**Functional Requirement**	**MoSCoW**
1	Implement data cleaning techniques to remove inconsistencies, duplicate columns, missing values, infinity entries, errors, and anomalies in the public dataset, and label encoding.	Must Have
2	Address class imbalance problem and compare the performance of the ensemble baseline models before and after class balancing.	Must Have
3.	Deploy at least two high dimensional dataset normalization techniques and evaluate impact on performance of the ensemble baseline models.	Must Have
4	Deploy at least two feature selection methods and evaluate impact on performance of the ensemble baseline models.	Must Have
5	Build ensemble baseline models with hyperparameter tuning and cross-validation and evaluate the performance of the ensemble baseline models over a range of number of selected features.	Should Have
6	Build ensemble models from the best performing baseline models and evaluate performance under balanced and unbalanced training datasets.	Must Have
7	Analyse the performance of the developed ensemble models trained with balanced dataset across range of number of selected features.	Should Have
8	Assess the performance of the ensemble models using appropriate evaluation metrics.	Must Have
9	Comparatively analyse the performance of the ensemble models and propose the best ensemble model.	Must Have
10	Present visual representations of both the data and the performance of the model to facilitate a clear interpretation.	Should Have
11	Integrate the proposed model into an Intrusion Detection System (IDS).	Won't Have
12	Generate real-time Network Intrusion traffic data to evaluate the performance of the Model.	Could Have

3.2.2 Non-Functional Requirements

The non-functional requirements are presented in Table **3.2**.

Table **3. 2**: Non-functional Requirements Using MoSCoW Analysis

S/N	Non-Functional Requirement	MoSCoW
1	Scalability: The proposed model must be able to handle increasing big data volumes efficiently and effectively without significantly degrading in performance.	Must Have
2	Performance: The proposed model should have the capability to attain better results in aspects like training duration, accuracy,recall, precision, and f1-score.	Should Have
3	Fault Tolerance: The proposed model should have the ability to continue functioning and provide reliable outputs even in the presence of unexpected events or exceptionally robust datasets.	Should Have
4	Security: The proposed model should be able to ensure the security and privacy of its data, and outputs.	Should Have

3.3 Methodology

This section covers details of the systematic stages of a workflow encapsulated in the designed machine learning pipeline in this study. Figure **3.1** illustrates this with a flow chart.

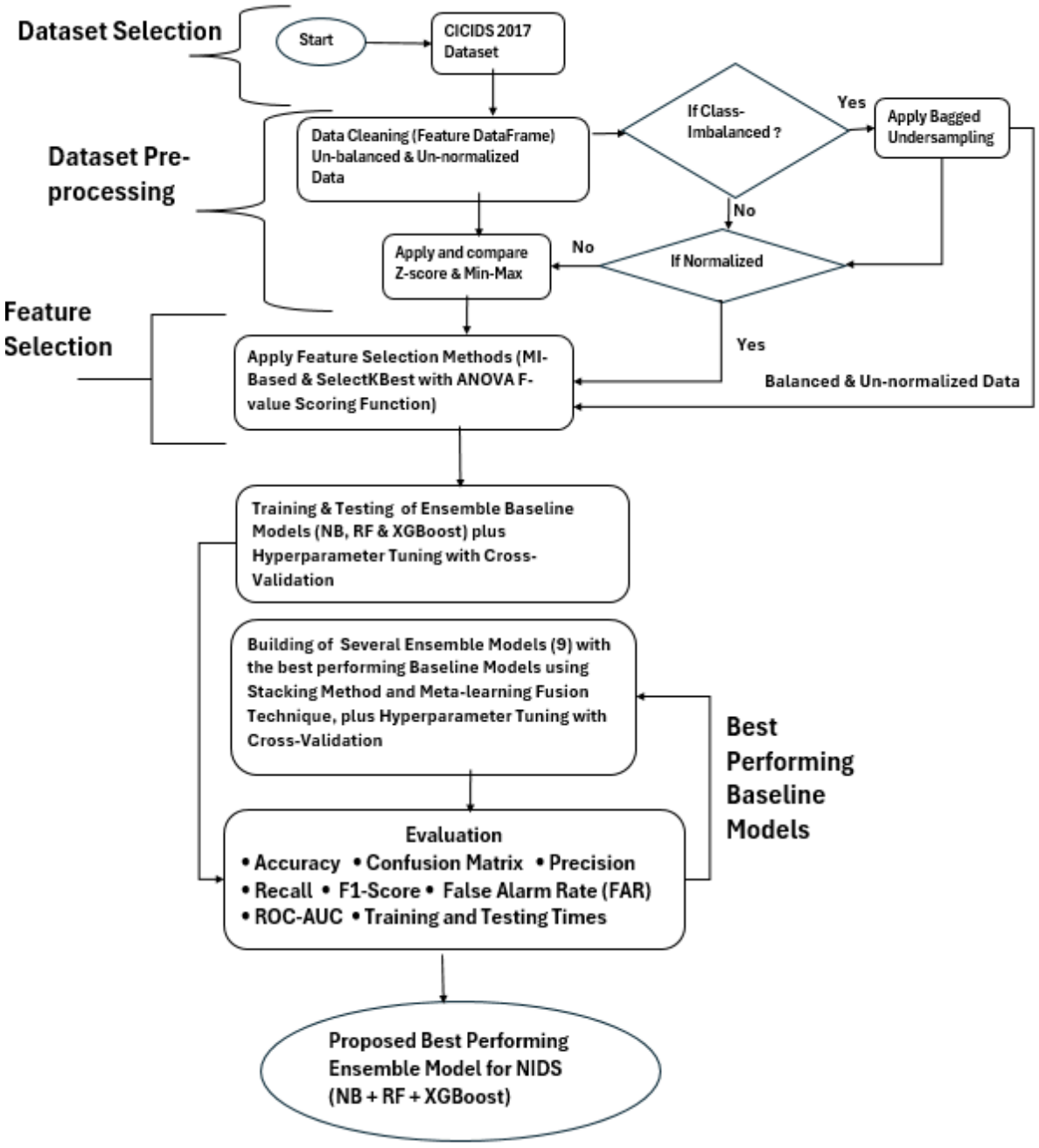


Figure **3. 1: Workflow Diagram for the Proposed NIDS Ensemble Model**

3.3.1 Dataset Selection

The performance of ML models heavily relies on the quality of available training dataset (Mohammed and Kora **2023**). From the literature review, Tables **2.1** and **2.2**, it is obvious that there are several publicly available benchmark datasets for network intrusion detection evaluation such as NSL-KDD, UNSW-NB15, N-BaIoT, CSE-CIC-IDS2018, and CICIDS2017 etc. (Sharafaldin, Lashkari and Ghorbani **2018**). According to Gharib et al. **(2016)**, CICIDS2017 intrusion detection evaluation dataset has proven to have satisfied all the 11 characteristics that are critical for a comprehensive and valid IDS dataset- attack diversity, anonymity, available protocols, complete capture, complete interaction, complete network configuration, complete traffic, feature set, heterogeneity, labelling, and metadata. CICIDS2017 contains benign and the most up-to-date common attacks, which resembles the true real-world intrusion data. It also includes the results of the network traffic analysis captured using CICFlowMeter. The dataset captures 5 days (Monday–Friday) of data, showing the abstract behaviour of 25 users and their interactions with diverse protocols such as HTTP, HTTPS, FTP, SSH, and email. This study has used the Wednesday dataset which has 85 columns and 692703 rows. It contains BENIGN traffic and different type of Dos/DDoS network intrusion traffic classes such as DoS slowloris, DoS Slowhttptest, DoS Hulk, DoS GoldenEye and Heartbleed. Figure **3.2** shows the public download of the dataset from the Canadian Institute for Cybersecurity database.

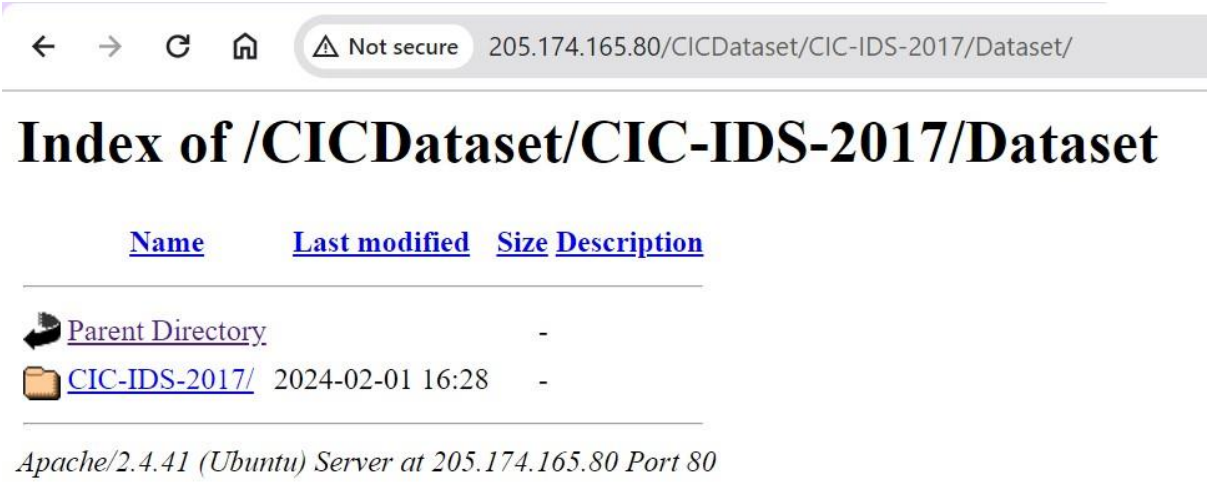


Figure **3. 2:** CICIDS2017 Dataset Download (https://www.unb.ca/cic/datasets/ids-2017.html)

3.3.2 Dataset Pre-processing

This stage involves dataset preparation and cleaning to ensure error-free computations. It begins with the removal of inconsistencies in the column names to the removal of duplicate columns, handling of missing values and infinity entries. The CICIDS2017 dataset is characterized with huge volume as it comprises over 600,000 data samples with some entries having incomplete or excessively large values, and it is highly class-imbalanced (Panigrahi and Borah **2018**). To address this, data understanding was carried out which led to the removal of some insignificant columns as far as intrusion detection modelling is concerned such as the "flow id", "source ip", "destination ip", and "timestamp". Initially, the dataset consists of multi-class traffic labels, and as this study is focused on binary classification, it became imperative to re-label the traffic classes into two categories- "ATTACK" and "BENIGN". All distinct types of intrusive DoS/DDoS attacks are re-labelled as "ATTACK". Label encoding was applied to encode the label numerically as "0" for "BENIGN" and "1" for "ATTACK" to aid machine learning computation. The "Label" column represents the class prediction, and this together with the newly generated "Encoded_label" column was extracted as independent variables. As earlier mentioned, the data distribution of this dataset is fraught with imbalance class proportions. This is capable of biasing model training and degrading performance (ENISA **2023**). When faced with imbalanced data, the model tends to favour the majority class, leading to sub-optimal prediction of the minority class. Ensuring both classes have equal representation during model training is key. Bagged undersampling technique was deployed in this study to address high dimensional data class imbalance issue, as this technique has proven to be superior to several other methods (including SMOTE- Synthetic Minority Over-sampling Technique) in improving predictive accuracy of the minority class without largely decreasing that of the majority class especially in this case of extreme class imbalance (Jeong et al. **2022**; Blagus and Lusa **2013**). In bagged undersampling, multiple balanced subsets are created using random undersampling (removing instances from the majority class to balance the class distribution), each containing an equal number of instances from both classes. This process is repeated multiple times, with replacement (bootstrap sampling) to create diverse subsets. Finally, Normalization as the last part of this data cleaning and pre-processing phase ensures that wide-range numerical values in the dataset are transformed into having a common scale without difference in the value range, and that all given dataset features possess equal weight. This is helpful in speeding up model training (Kumar, Gupta and Arora **2021**). The variations in value ranges can result in a situation where the model assigns greater importance to the highest value than the lowest value (Alghushairy et al. **2024**). Owing to the significance of this to model training, we have explored Z-score (Standard Scaler) and Min-Max normalization techniques in this study for the sake of performance impact comparison and decision-making (Santos, Miani and Silva **2024**; Obaid, Dheyab and Sabry **2019**).

- **Z-score/Standard Scaler:** This technique measures how far a given value deviates from the mean regarding the standard deviation. It is represented in Equation **3.1**.

$$\mathbf{X} = \frac{\boldsymbol{x} - \mathbf{mean}\ (\boldsymbol{x})}{\boldsymbol{Standard\ deviation}\ (\boldsymbol{x})} \qquad \textbf{(3.1)}$$

- **Min-max Scaler**: This normalization technique transforms the data to fit within the range of 0 to 1 on an interval scale. If the dataset contains negative values, the interval scale can range between -1 and 1. It is represented in Equation **3.2**.

$$\mathbf{X} = \frac{x - \min(x)}{\max(x) - \min(x)} \qquad (3.2)$$

3.3.3 Feature Selection

The CICIDS2017 dataset is a high-volume dataset characterized with high dimensionality. Training ML model on the whole dataset without any form of appreciable dimensionality reduction can be inaccurate and computationally costly in terms of required resources and training time, and such models tend to overfit. Overfitting occurs when models learn the training data too well that they fail to generalize or adapt to unseen data (Jemili, Meddeb and Korbaa **2023**). This happens because the model has memorized the noise in the training data to the extent that it becomes unable to accurately make predictions on new data. The cost of training time is because the model has to essentially learn a large number of parameters whereas, the largeness of these features does not guarantee improved performance accuracy (Rashid et al. **2022**). Feature selection method as a means of dimensionality reduction is used to address this challenge.

In this study, the most significant features in the dataset were selected for ensemble baseline model training purposes based on feature importance (super features). This implies that less relevant features were discarded for memory and process optimization. This study comparatively explored two feature selection methods (Mutual Information-based method, and the SelectKBest with ANOVA F-value Scoring Function method) to extract feature subsets from the dataset with the goal of identifying the method whose selected model training features better contribute to an improved performance of the ensemble baseline models (Alduailij et al. **2022**). Figure **3.3** shows a list of the entire features including the label column of the CICIDS2017 dataset. Figure **5.3** (Appendix **B)** shows a correlation Heat Map of these features.

1	flow_id	22	flow_packets/s	43	fwd_packets/s	64	fwd_avg_packets/bulk
2	source_ip	23	flow_iat_mean	44	bwd_packets/s	65	fwd_avg_bulk_rate
3	source_port	24	flow_iat_std	45	min_packet_length	66	bwd_avg_bytes/bulk
4	destination_ip	25	flow_iat_max	46	max_packet_length	67	bwd_avg_packets/bulk
5	destination_port	26	flow _iat_min	47	packet_length_mean	68	bwd_avg_bulk_rate
6	protocol	27	fwd_iat_total	48	packet_length_std	69	subflow_fwd_packets
7	timestamp	28	fwd_iat_mean	49	packet_length_variance	70	subflow_fwd_bytes
8	flow_duration	29	fwd_iat_std	50	fin_flag_count	71	subflow_bwd_packets
9	total_fwd_packets	30	fwd_iat_max	51	syn_flag_count	72	subflow_bwd_bytes
10	total_backward_packets	31	fwd_iat_min	52	rst_flag_count	73	init_win_bytes_forward
11	total_length_of_fwd_packets	32	bwd_iat_total	53	psh_flag_count	74	init_win_bytes_backward
12	total_length_of_bwd_packets	33	bwd_iat_mean	54	ack_flag_count	75	act_data_pkt_fwd
13	fwd_packet_length_max	34	bwd_iat_std	55	urg_flag_count	76	min_seg_size_forward
14	fwd_packet_length_min	35	bwd_iat_max	56	cwe_flag_count	77	active_mean
15	fwd_packet_length_mean	36	bwd_iat_min	57	ece_flag_count	78	active_std
16	fwd_packet_length_std	37	fwd_psh_flags	58	down/up_ratio	79	active_max
17	bwd_packet_length_max	38	bwd_psh_flags	59	average_packet_size	80	active_min
18	bwd_packet_length_min	39	fwd_urg_flags	60	avg_fwd_segment_size	81	idle_mean
19	bwd_packet_length_mean	40	bwd_urg_flags	61	avg_bwd_segment_size	82	idle_std
20	bwd_packet_length_std	41	fwd_header_length	62	fwd_header_length.1	83	idle_max
21	flow_bytes/s	42	bwd_header_length	63	fwd_avg_bytes/bulk	84	idle_min
						85	label

Figure **3. 3**: CICIDS 2017 Entire Dataset Features

- **Mutual Information (MI)-Based Feature Selection Method**

Mutual Information is a statistical measure of the dependency between two random variables. In feature selection, it is deployed to quantify the amount of information that one feature provides about the target variable. It is a filter-based feature selection method that selects features based on their individual Mutual Information scores- MI Score. It is a relatively fast and model-agnostic method that can be used to select features from large datasets in a reasonable amount of time. Features with high mutual information scores are considered highly informative and relevant for predicting the target variable, potentially improving the performance and interpretability of machine learning models (Albulayhi et al. **2022**). Equation **(3.3)** shows the calculation of MI Score.

$$I(X; Y) = H(X) - H(X|Y) \quad \textbf{(3.3)}$$

where I (X; Y) is the calculated MI score for X and Y; H(X) is the entropy for X, and H(X|Y) is a conditional entropy for attributes X and Y.

- **SelectKBest with ANOVA F-value Scoring Function Feature Selection Method**

SelectKBest method is used to select the top K number of features from a feature dataframe based on univariate statistical tests such as ANOVA F-value. It selects features based on the ANOVA F-value between each feature and the target variable. The ANOVA F-value scoring function is typically used for ML for regression and classification when dealing with numerical features dataframe and a continuous or categorical target variable with multiple classes. This method is also model-agnostic, as it can generally be used with any model that accepts feature arrays as input, thereby improving the model's performance by selecting the most informative features and reducing the dimensionality of the feature space, leading to faster training times, better generalization, and improved model interpretability. It assumes that features with a higher variance contain more useful information. The ANOVA F-value measures the difference in means between different classes in the target variable. Features with high F-values are considered important for discriminating between different classes. At first, the variance of each feature is calculated, and then a subset of features is selected based on a user-specified threshold value (K). SelectKBest with ANOVA method selects K features according to the highest F-value scores (Rashid et al. **2022**).

Mathematically, If X represents the input features and Y represents the target variable. Then, The F-value for feature X_i is calculated as in Equation **(3.4)**.

$$F(X_i, Y) = \frac{\text{variance within groups}}{\text{Variance between groups}} \quad \textbf{(3.4)}$$

Where:

The variance within groups measures the variation of the feature within each class of the target variable, Y, and the variance between groups measures the variation in the means of the feature across different classes of the target variable, Y.

The SelectKBest with ANOVA method selects K number of features with the highest F-values between each feature X_i and the target variable ,Y. If S represents the set of selected features, K. The SelectKBest with ANOVA F-value Scoring Function-based Feature Selection Method can be represented mathematically as in Equation **(3.5)**.

$$S = \text{SelectKBest}(X,Y,K) , \quad \textbf{(3.5)}$$

where:

X is the input feature dataframe matrix or array, Y is the target variable, K is the number of features to select, and S is the set of selected features (K).

3.3.4 Training and Testing

In this phase, series of selected feature subsets (dataframes) are divided into a training set comprising 80% of the dataframe and a testing set comprising the remaining 20%. The testing set is used for prediction (validation). Subsequently, the training set is employed to train three chosen ensemble baseline models which are Naive Bayes (NB), Random Forest (RF), and Extreme Gradient Boosting (XGBoost) with automated Hyperparameter Tuning and Cross-Validation settings (RandomSearchCV). The ensemble baseline model training and testing was done severally across a number of scenarios for the purpose of research and analysis as follows:

- Z-score normalized-balanced and Min-Max normalized-balanced feature dataset (Impact of Choice of Normalization Technique on Baseline Model Performance).
- Balanced-normalized and balanced-unnormalized (Impact of Normalization on Baseline Model Performance)
- Balanced-normalized and unbalanced-normalized feature dataset (Impact of Class-imbalance on Baseline Model Performance).
- MI-based and SelectKBest with ANOVA F-value Scoring Function feature selection methods for 14, 12, 10, and 8 number of selected features (Baseline Model Performance Comparison on Feature Selection Methods).

(a) Naive Bayes (NB)

Naïve Bayes is a conditional probability-based algorithm based on the Bayes' Theorem that calculates and predicts the probabilities of each feature belonging to each class with a simple assumption that all features are independent of each other, and each feature can only affect classification result independently. This can often be an unrealistic assumption, but it allows the algorithm to be computationally efficient and work well with high-dimensional data. Naive Bayes classifier requires a small amount of training data to estimate the necessary parameters. It is easy to build and scalable to larger datasets. It is also a versatile and effective ML algorithm that is often used in text

classification, spam filtering, intrusion detection, and sentiment analysis in cybersecurity using statistical theory. There are different forms of the feature probability distribution, like Gaussian, Bernoulli etc. Gaussian Naïve Bayes (GNB) was used in this study. It assumes that the values of each successive class are mutually independent and follow the Gaussian Distribution. It is an effective choice for identifying anomalous network activities and potential security threats. By considering the statistical distribution of features related to network traffic, such as packet sizes, response times, and connection duration, the model can learn patterns of normal behaviour. During the testing phase, it can efficiently classify incoming data as either normal or malicious based on the learned likelihood distributions. NB is robust to noise, and able to learn incrementally (Alghushairy et al. **2024**; Sahu et al. **2020**; Liu and Lang **2019**). Aside the cross-validation setting, default NB smoothing factor (var_smoothing) was used to enhance stability and generalization of the model.

Bayes theorem can be expressed by Equation **3.6**:

$$P(c|x) = \frac{P(c)P(x|c)}{P(x)} \qquad \textbf{(3.6)}$$

Where P(c) is a kind of prior probability, P(x|c) is the conditional probability of a connection record x relative to class label c**,** and P(x) is the evidence factor used for normalization. For a given connection record x, the evidence factor P(x) has no relationships with class labels. So, P(c|x) is only related to P(c) and P(x|c) (Bong and Kim 2**022**).

(b) Random Forest (RF)

RF is a supervised ensemble of decision trees (DT) that develops multiple decision trees into a forest during training. Each tree is trained on a random subset and bootstrap sample of the training data, and a measure is used to determine the optimal feature for data separation. The final prediction is made by aggregating the predictions of the individual trees by taking a majority vote for classification tasks or averaging for regression tasks (Alduailij et al. **2022**). RF uses bagging ensemble method to combine the decision tree's simplicity with the flexibility to increase predictive accuracy. The use of randomness is to reduce variance and overfitting issues in DT (Jeong et al. **2022**; Sahu et al. **2020**) Comparing with SVMs, RF is faster and works well with a mixture of numerical and categorical features. Random forest performs better in training speed, prediction performance, generalization, robustness in handling inconsistent datasets, the ability to classify embedded features than other traditional machine learning methods. It possesses internal metrics for assessing feature importance (Rashid et al. **2022**). RF model possesses hyperparameter with cross-validation settings to optimize its performance (Jemili, Meddeb and Korbaa **2023**):

- n_estimators
- 'min_samples_split,
- max_depth
- CV

(c) Extreme Gradient Boosting (XGBoost)

XGBoost is a widely used scalable tree boosting algorithm in the ensemble family (gradient-boosted DTs). It is an advanced implementation of gradient boosting that optimizes a differentiable loss function by iteratively fitting new models to the negative gradient of the loss function which involves building of an ensemble of weak learners (decision trees) sequentially with each subsequent model focusing on the examples that were misclassified by previous models, especially with structured features. (Vijay et al. **2021**). It uses a regularized model structure and advanced optimization techniques to improve performance and scalability, including regularization to prevent overfitting, parallelization for efficient training on large datasets, and tree pruning to reduce model complexity and enhance performance while ensuring variance-bias trade-offs. XGBoost supports various functions such as regression, classification, and ranking. One of the major advantages of XGBoost is that it is highly efficient in reducing computing time and providing optimal use of memory resources (Rajagopal, Kundapur and Hareesha **2020**). It possesses hyperparameter with cross-validation settings to optimize its performance (Jemili, Meddeb and Korbaa **2023**):

- n_estimators
- max_depth
- learning_rate
- CV

(d) Hyperparameter Tuning and Model Performance Cross-Validation

RandomizedSearchCV as an ML package is a combination of randomized search hyperparameter tuning and model performance cross-validation techniques. Randomized search involves selection of random combinations of hyperparameters from specified ranges. It is fast and scalable even when the hyperparameter search space is large. Cross-Validation is used to estimate best model performance, selecting corresponding best hyperparameters reliably and complementing hyperparameter tuning technique. It is used to assess the performance generalization of a model by partitioning the dataset into multiple subsets (folds). The model is trained on training set and evaluated on the testing or validation set. This process is repeated for each fold, and the performance metrics are averaged (Rashid et al. **2022**; Rajagopal, Kundapur and Hareesha **2020**).

(e) Ensemble Modelling

Furthermore, in tandem with the focus of this study as indicated in Figure **3.1**, next is the building of Ensemble Models by combining the best performing trained and tested baseline ML models (NB, RF, and XGBoost) with automated hyperparameter tuning and cross-validation settings (RandomizedSearchCV). The following nine (9) ensemble models were developed and evaluated under balanced and unbalanced feature datasets scenarios using stacked generalization (stacking) ensemble and meta-learning fusion methods for the purpose of analysis and research (Mohammed and Kora **2023**):

- NB + RF,
- NB + XGBoost,
- RF + NB,
- XGBoost + NB,
- RF + XGBoost,
- XGBoost + RF,
- (NB + RF) + XGBoost,
- (NB + XGBoost) + RF, and
- (RF + XGBoost) + NB

To further enhance generalization of the ensemble models, aside base model diversity, a different subsets of the stacked data was use for the train-test split (70/30) for the meta-model as against 80/20 used for the base models.

3.3.5 Evaluation

The evaluation of the project will be carried out in seven phases:

1. Impact of Choice of Normalization Technique on Performance of Ensemble Baseline Models: This will be evaluated by comparatively analysing the performance of ensemble baseline models trained with Z-score and Min-Max normalized datasets.
2. Impact of Normalization on Performance of Ensemble Baseline Model: This will be evaluated by comparatively analysing the performance of ensemble baseline models trained with normalized and unnormalized datasets.
3. Impact of Class Imbalance on Ensemble Baseline Model Performance: This will be evaluated by comparatively analysing the performance of ensemble baseline models trained on balanced and unbalanced datasets.
4. Impact of Feature Selection Method on Performance of Ensemble Baseline Model: This will be evaluated by comparatively analysing the performance of ensemble baseline model trained with different feature selection methods for 14, 12, 10 and 8 number of selected features.
5. Performance analysis for all developed Ensemble Models to select the ones whose accuracies are greater than 99.90%.
6. Performance analysis of selected Ensemble Models on balanced and unbalanced datasets.
7. Evaluation of confusion matrix, accuracy, precision, recall, F1-score, False Alarm Rate (FAR), ROC-AUC curves, model training and testing times based on the selected Ensemble models.

In this study, model performance is evaluated using a number of metrics as described below (Mohammed and Kora **2023**; Krishnaveni et al. **2021**).

- **Confusion Matrix**

A confusion matrix is a performance measurement for machine learning classification problems. It shows the number of True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) as predictions made by the model. The FP and FN are Type I and Type II Misclassification Errors respectively.

In a binary classification problem (0,1), the confusion matrix is structured as shown in Figure **3.4**:

			Predicted	
			Benign	Attack
			0	1
Actual	Benign	0	TN	FP
	Attack	1	FN	TP

Figure **3. 4**: Confusion Matrix

- True Positive (TP): The number of Attack instances that were correctly predicted as Attack by the model.
- True Negative (TN): The number of benign instances that were correctly predicted as benign by the model.
- False Positive (FP): The number of benign instances that were incorrectly predicted as Attack by the model.
- False Negative (FN): The number of Attack instances that were incorrectly predicted as benign by the model

- **Accuracy**

Performance Accuracy (%) represents the percentage of correctly predicted instances out of the total instances in the dataset (Equation **3.7).**

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad \textbf{(3.7)}$$

- **Precision**

Precision measures the accuracy of the positive predictions made by the model. It is defined as the ratio of true positive (TP) predictions to the total number of positive predictions made by the model (Equation **3.8**)

$$\text{Precision} = \frac{TP}{TP+FP} \quad \textbf{(3.8)}$$

- **Recall/Detection Rate (DR)/Sensitivity/True Positive Rate (TPR)**

Recall measures the ability of a classification model to correctly identify all positive instances in a dataset. It is defined as the ratio of true positive (TP) predictions to the total number of actual positive instances (Equation **3.9**).

$$\text{Recall} = \frac{TP}{TP+FN} \quad \textbf{(3.9)}$$

- **F1 – Score**

F1 score combines both precision and recall providing a balanced evaluation of a machine learning model's performance in binary classification tasks. The F1 score ranges from 0 to 1, where a higher value indicates better performance. It is the harmonic mean of sensitivity and precision. it is often used as a performance metric to assess the efficacy and generalization of intrusion detection system model especially with irregular class distribution of datasets (Equation **3.10**).

$$\text{F1-score} = 2 * \frac{Precision * Recall}{Precision + Recall} \quad \textbf{(3.10)}$$

- **False Alarm Rate (FAR)**

Also known as Fall-out or False positive rate (FPR), it is the measure of incorrectly classified negative (benign) instances as positive (attack or anomaly) from all the actual benign instances. The lower its value the better.

$$\text{FAR/FPR} = \frac{FP}{FP = TN} \quad \textbf{(3.9)}$$

- **Receiver Operating Characteristic (ROC) curve**

ROC curves are used to plot the relationship between detection rate (DR/TPR) and false positive rate (FPR) of a model at different threshold settings. The performance accuracy of a model using ROC curves can be measured in terms of the Area under the Curve (AUC) which ranges from 0 to 1. An AUC value closer to 1 indicates that the model has good performance, and it can effectively distinguish between the attack and benign classes across different threshold settings. An AUC of 0.5 means the model is performing no better than random guessing, while an AUC below 0.5 indicates that the model is performing worse than random guessing (Krishnaveni et al. **2021**).

- **Ensemble Model Computational Time (Running Time)**

In this study, the computational time is the entire time taken to train and test (Training and Testing Times) a model excluding the time taken in the feature selection or dataset pre-processing process. However, the training time is inclusive of the time consumption for the modelling k-fold cross-validation with optimum hyperparameter search which led to the substantial computational time values across all the experimental modelling.

3.4 Project Plan and Timelines

Deep learning and neural network models are computationally expensive in terms of required resources, training and testing time for NIDS, especially for a resource constrained environment like the SMEs (Abbas et al. **2023**). Therefore, the scope of this study is to research into the machine learning pipeline of the ensemble of simpler traditional ML models and optimize every phase towards producing an ensemble model with improved predictive performance accuracy over existing ones, while still benefitting from the lower computational cost and training times inherent in the chosen ensemble baseline models plus the robustness and adaptability that ensemble learning offers. This produced a proposed NB-augmented tree-based stacking ensemble model with 99.99% performance

accuracy on 12 number of selected training features and lower training and testing times compared with its counterpart which is completely tree-based.

This project scope fulfilment is captured in a project plan in **Appendix A** detailing the tasks taken and their timelines.

3.5 Review of Legal, Ethical, Social, Professional and Environmental issue

This section of the project discusses the legal, ethical, social, professional and environmental issues that may arise during the development of this project.

3.5.1 Legal Issues

The dataset utilized in this study was sourced from the Canadian Institute for Cybersecurity's (CIC) publicly accessible online database, which is housed on the University of New Brunswick's official website. The CICIDS2017 dataset comprises network traffic information that were specifically produced for scholarly and experimental uses. It's important to emphasize that since this dataset doesn't contain any personal information, users don't need to obtain any additional authorization in order to access it. However, it's crucial to understand the legal ramifications of accessing this dataset, including license, copyright, and data usage rights. To prevent any infringement of data utilisation rights or intellectual property rights, this study has made sure that all citations are correctly attributed, as specified in the licence statement.

3.5.2 Ethical Issues

Transparency is one of the most important ethical considerations when it comes to Network Intrusion Detection using Ensemble Learning. It's critical to be clear about the project's objectives and to make sure the outcomes are applied lawfully, avoiding any possibility of misuse. In order to maintain this, the research will be totally open about the dataset and the decision-making algorithms that were used for the project. This helps stop the model from being used maliciously and enables independent verification of the model's fairness and accuracy. It is imperative that the machine learning model handle all data equally, make fair conclusions, and not favour any particular type of network traffic. Therefore, it's imperative to take precautions to shield this data against unauthorized access and nefarious usage.

3.5.3 Social Issues

Using ensemble learning to network intrusion in real time could result in extensive personal surveillance. Nevertheless, real-time production deployments that could have substantial impact on the corporate environment and society are not a must-have functional requirement of this study. As a result, no noteworthy environmental or social issues have been found in relation to the study's scope.

3.5.4 Professional Issues

Objective analysis, transparent approach, and accurate reporting are essential to upholding professionalism. Every aspect of this project, including the phases of implementation and methodology, has been well documented. This approach improves comprehension among colleagues and facilitates the validation of findings. It is crucial to stress that this study was conducted with no commercial goals, but a major focus on academic aims.

3.5.5 Environmental Issues

One of the most important factors in determining the environmental impact is the usage of computational resources for model testing and training. Using a cloud-based infrastructure such as Apache Spark, to build the machine learning models, along with resource and code optimization, proves to be a powerful approach to reducing the project's environmental effect. This is in good accordance with the concepts of responsible environmental management, encouraging a strategy for sustainable practices that both lessen the impact on the environment and increase the project's overall beneficial impact.

3.6 Identification of Risks, Including Security Risks, and Mitigating Measures Taken

When models are developed with biased training data, it may lead to predictions that cannot generalize over new attack traffic circumstances. Because the model could overlook some attack types, this can present serious difficulties for NID. This study employed a benchmark dataset with actual network traffic to address this. To guarantee peak performance, a number of crucial actions are required before the model is trained. These consist of class imbalance correction, feature engineering, and dataset preprocessing, and training of the meta-model with a data subset different from the population ratio used for the train-test split used for the baseline models.

False negative is an additional concern. It poses a significant risk for detecting network intrusion attacks. This is due to the fact that a false negative happens when the model misclassifies attack traffic for benign traffic. This could make an attack go unnoticed, which could have detrimental effects. There are several strategies to lower the possibility of false negative results. Ensemble baseline model diversity is one method- by employing different models, each of which will have unique strengths and limitations. This will contribute to raising the detecting system's overall accuracy.

3.7 Identification of Codes of Practice and Industry Standards Related to the Work

3.7.1 The AI Ethics Guidelines for Trustworthy AI

The Partnership on AI has produced the AI Ethics Guidelines for Trustworthy AI (European Commission **2021**) which contains a set of principles designed to direct the development and application of artificial intelligence. The purpose of these strategically stated principles is to advance the ideals of safety, privacy, responsibility, justice, and

transparency in the context of AI development and deployment. Guidelines for AI developers, users, and policymakers were created by a group of professionals from academia, business, and civil society.

3.7.2 The IEEE Ethically Aligned Design (AED)

The Institute of Electrical and Electronics Engineers (IEEE) is responsible for organizing the IEEE Ethically Aligned Design (AED) framework (IEEE **2019**), which addresses the moral creation of AI systems. With the help of the framework, ethical AI development and application may be ensured by following a set of rules, suggestions, and best practices. Seven principles make up the EAD framework: role-based design, privacy, security, fairness, accountability, transparency, and human-centric design. Network intrusion attack detection projects can benefit from the application of these IEEE EAD framework concepts in a variety of methods.

CHAPTER 4: DESIGN AND IMPLEMENTATION

This chapter offers chronological and comprehensive details of this project design and implementation process.

4.1 Design Alternatives and Justification for Chosen Design

4.1.1 Design Alternatives

Researching a robust, efficient and cost-effective solution for anomaly-based NIDS demands thorough evaluation and consideration of different design alternatives. Every ML design type has its own benefits and limitations.

Some of the most common ML design alternatives for NIDS are as follows:

- **Simple Learning:** These are typically straightforward algorithms in traditional machine learning used to solve specific tasks without incorporating complex techniques like deep learning architectures. They are often easy to implement and interpret, making them suitable for various applications- Binary classification, Regression, Clustering, Rule-based learning, and Anomaly detection. Examples of simple machine learning models are Logistic Regression, Decision Trees, K-Nearest Neighbors, Naive Bayes, K-Means Clustering, Support Vector Machines etc. They serve as building blocks in machine learning and are often used as baseline models for more complex models like ensembles. Their limitations are difficulty in best model selection for a given problem, and they are prone to bias and overfitting to training data and poor adaptability to evolving threat landscape (Jemili, Meddeb and Korbaa **2023**).

- **Ensemble Learning:** This technique combines multiple ML models to produce improved accuracy and stability of a single ensemble model. Each individual classifier has strengths and weaknesses and combining them is to implement a strong classifier. This results in a more robust solution that is less likely to be impacted by the specific strengths and weaknesses of anyone. Ensemble learning can improve the performance of simple learning methods by leveraging the diversity of individual models, producing more accurate prediction than each individual and overcoming simple learning limitations like overfitting. The success of an ensemble learning lies in the type of utilized baseline models, the data samples techniques used in training, the diversity of employed different baseline classifiers, and the fusion methods of the baseline ensemble models (Abbas et al. **2023**).

- **Deep Learning:** This refers to a class of machine learning techniques that are based on artificial neural networks with multiple layers. These neural networks are able to automatically learn complex patterns and representations from data by hierarchically extracting features at different levels of abstraction (Saleh et al. **2022**). However, the training process of deep learning models requires a massive effort, and tuning for the best hyperparameters requires expertise and extensive trial, which is tedious and time-consuming, and can increases the chance of

overfitting (Mohammed and Kora **2023**). Deep learning models are highly scalable and can handle large amounts of data. They are usually trained on powerful hardware, such as GPUs or TPUs to efficiently process massive datasets and complex models with millions or even billions of parameters. It has revolutionized many areas of artificial intelligence and has achieved state-of-the-art performance in various domains including computer vision, speech recognition, natural language understanding, and reinforcement learning. Moreover, as network edge devices do not usually have huge computation power, deep learning methods may not be efficient in real-time for NIDS especially for resource constrained environment like the SMEs. SMEs can readily implement ensemble of traditional simple machine learning-based systems and fulfil their need of combating cyber-attacks. Ensemble models can also process labelled Big Data, and still achieved the same performance as deep learning without requiring high-end GPUs (Abbas et al. **2023**).

4.1.2 Justification for Chosen Design

Ensemble learning of traditional simple learners is a choice for NIDS in complex and modern big data network environments owing to its robustness, adaptability and excellent predictive performance accuracy, which are comparable to that of deep learning alternatives, and yet still cost-effective in terms of required resources and running time (training and testing time), a benefit to the SMEs (Abbas et al. **2023**).

Ensemble learning methods are of three types, namely, bagging, boosting and stacking, while ensemble learning fusion methods are of two types, namely, voting and meta-learning methods (Mohammed and Kora **2023**). Random Forest (RF) is an example of a bagging ensemble model of DTs utilizing voting fusion technique, while Extreme Gradient Boosting (XGBoost) is an example of a boosting ensemble model of DTs using voting fusion technique.

Bagging ensemble method combines bootstrapping and aggregating to create multiple diverse predictive models. is a completely data-specific algorithm where small changes in the training data set will lead to significant changes in the model predictions. It involves creating small subsets of data from the original dataset, aiming to increase model diversity and reduce variance, thus avoiding overfitting. Bootstrapping involves training ensemble models on bootstrap replicates of the training dataset, while aggregation combines predictions through majority voting. Bagging performs well on high-dimensional data. Bagging utilizes parallel ensemble techniques and different voting methods for fusion.

Boosting ensemble method is a sequential method that aims to improve model performance by iteratively correcting the errors of previous models. It involves fitting multiple weak learners in a sequence, with each subsequent model giving more emphasis to data points that were poorly handled by earlier models. Boosting helps reduce variance and bias in machine learning ensembles and offers relatively easy model interpretation. Overall, boosting employs sequential ensemble techniques with data dependency and utilizes various fusion methods for combining model outputs.

Stacking ensemble method, also called Stacked Generalization, is an ensemble technique that combines predictions from multiple base models to train a new model (meta-model). It involves base models at level 0 and a meta-model

at level 1. The base models are trained on the data, and their predictions are then used as input for the meta-model to learn how to best combine them. This method typically outperforms individual models and offers a deeper understanding of the training data than bagging and boosting. Stacking utilizes parallel ensemble techniques and relies on meta-learning methods for fusion (Xiong, Ye and Wu **2021**; Ma et al. **2018**; Divina et al. **2018**)

Voting fusion methods are commonly employed in classification and regression tasks to enhance predictive accuracy, particularly in conjunction with bagging and boosting ensemble methods. There are three main voting fusion methods: max voting, averaging voting, and weighted average voting. Max voting (majority voting) selects the class label with the highest number of votes among predictions. Averaging voting computes the average of predictions from multiple models to make the final prediction. Weighted average voting assigns different weights to baseline learners based on their importance, resulting in a weighted sum of predictions. These weights determine the contribution of each model to the final prediction (Krishnaveni et al. **2021**).

Meta-learning fusion method involves the concept of learning from learners, aiming to enhance the performance of learning algorithms by modifying certain aspects based on experimental results. It encompasses multiple learning stages where the outputs of individual learners serve as inputs to a meta-learner (meta-classifier), which produces the final output. Meta-learning addresses challenges in training learning algorithms, such as high operational costs and lengthy training times, by improving algorithms and identifying those that perform better. Benefits include accelerated learning processes, improved adaptability to changing conditions, and optimization of hyperparameters for optimal results. Meta-learning also offers opportunities to address challenges in deep learning, including data size and computational complexities (Hospedales et al. **2021**; Kuruvayil and Palaniswamy **2021**)

Owing to higher robustness and generalization capability of stacking ensemble with meta-learning fusion method as highlighted in the foregoing, this study has worked with its architecture as depicted in Figure **4.1**.

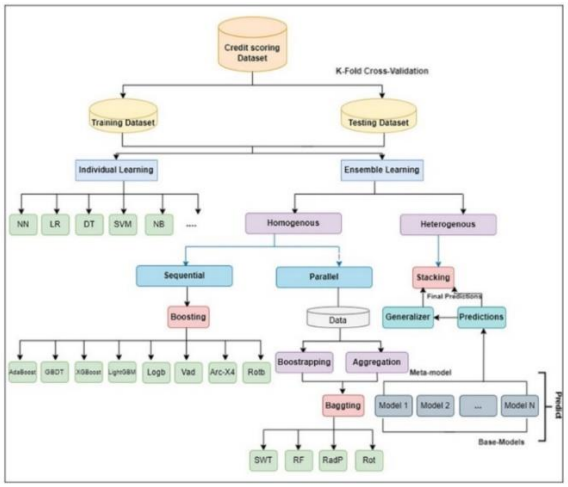


Figure **4. 1:** Architecture of a Stacking Ensemble Learning (Jemili, Meddeb and Korbaa **2023**).

From the literature review in Chapter 2, RF and XGBoost as completely tree-based ensembles of bagging and boosting categories respectively, have always had record of good model performance with their advantages

discussed in Chapter 3. These models were augmented with NB in this study to research on ensemble learning for NIDS following the framework designs shown in Figures **4.2**, and **4.3** based on fewer number of selected features.

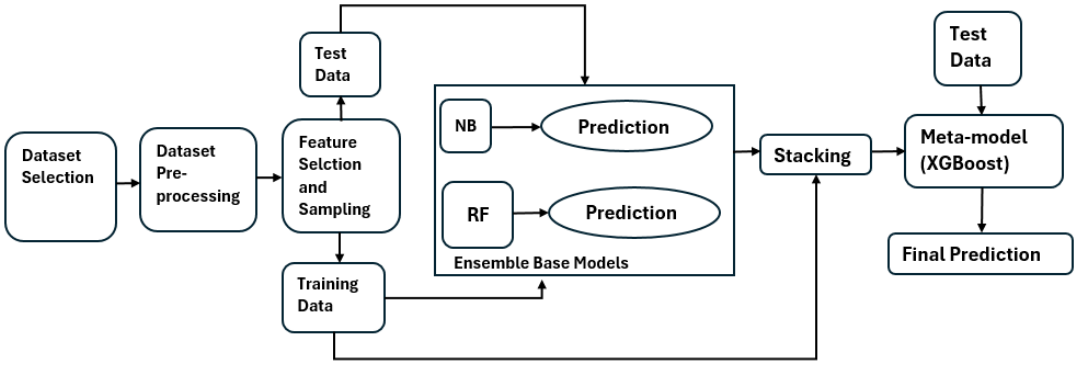


Figure **4. 2: Proposed NB-Augmented Tree-Based Stacking Ensemble for NIDS Case 1**

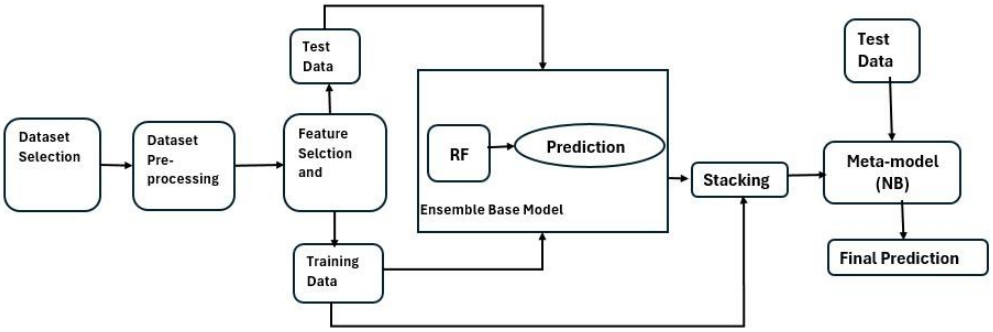


Figure **4. 3: Proposed NB-Augmented Tree-Based Stacking Ensemble for NIDS Case 2**

The implementation of these designs and seven (7) other experimental designs would be the discussion in subsequent sections.

4.2 System Specifications

The following information includes the system specification, tools and Technologies used:

1. Operating system Windows 11

2. Processor Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz 2.50

3. RAM 16.0 GB

4. Anaconda Navigator 2.4.0

5. Jupyter Notebook 6.5.2

6. Python

7. RStudio (for some Plots)

4.3 Data Understanding

Import the required libraries, such as matplotlib, pandas, numpy, and seaborn, into the Jupyter notebook first. Next, use the read_csv() function to load the dataset and put it in the DataFrame variable df. According to Figure **4.4** the 692,703 rows and 85 characteristics make up the dataset. A summary of the data distribution inside each column is given by the describe() command. Further information about the dataset's columns can be obtained via the df.info() function. Making sure that column header names are proper is crucial since inaccurate headers may cause computational issues in later stages. The dataset's column names were changed by eliminating brackets and spaces, as seen in figure **4.4**

```
[1]: # Importing first set of python ML libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import time
```

```
[3]: # Load CICIDS2017 Choice Dataset
dataset_path = "Wednesday-workingHours.pcap_ISCX.csv"
df = pd.read_csv(dataset_path)
print(df.shape)
df.describe()
```

```
(692703, 85)
```

[3]:

	Source Port	Destination Port	Protocol	Flow Duration	Total Fwd Packets	Total Backward Packets	Total Length of Fwd Packets	Total Length of Bwd Packets	Fwd Packet Length Max	Fwd Packet Length Min	...	act
count	692703.000000	692703.000000	692703.000000	6.927030e+05	692703.000000	692703.000000	6.927030e+05	6.927030e+05	692703.000000	692703.000000	...	
mean	42583.995907	5686.869462	9.219523	2.800168e+07	9.556261	10.214079	5.550930e+02	1.699644e+04	233.593936	15.022183	...	
std	19535.697710	15727.423560	5.009796	4.276680e+07	747.197814	984.204633	6.163663e+03	2.241175e+06	603.751856	51.068835	...	
min	0.000000	0.000000	0.000000	-1.000000e+00	1.000000	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	...	
25%	36235.000000	53.000000	6.000000	2.010000e+02	2.000000	1.000000	1.200000e+01	0.000000e+00	6.000000	0.000000	...	

Figure **4. 4**: Initial load plus Dataset Description

It is critical to comprehend both the dataset's numerical and category columns. For model training, categorical columns must be handled properly as in Figure **4.5**This is due to the fact that categorical data cannot be properly understood by machine learning models. They cannot be used unless they are transformed into numerical data. Notably, there are 80 number columns and 5 category columns in the CICIDS2017 dataset.The columns "flow_id," "source_ip," "destination_ip," "timestamp," and "label" are among these categorical columns. There are many label kinds in the label column, which correspond to different categories of DDoS attacks. The distribution of distinct DDoS attack types within the dataset is shown graphically in Figure **4.6**

```
[11]: # Print unique label types from the "Label" column of the DataFrame, df4
category_types=df4["label"].unique()
print("unique label types: ")

for category in category_types:
    print(category)
```

```
unique label types:
BENIGN
DoS slowloris
DoS Slowhttptest
DoS Hulk
DoS GoldenEye
Heartbleed
```

Figure **4. 5**: Categorical classes in the label column

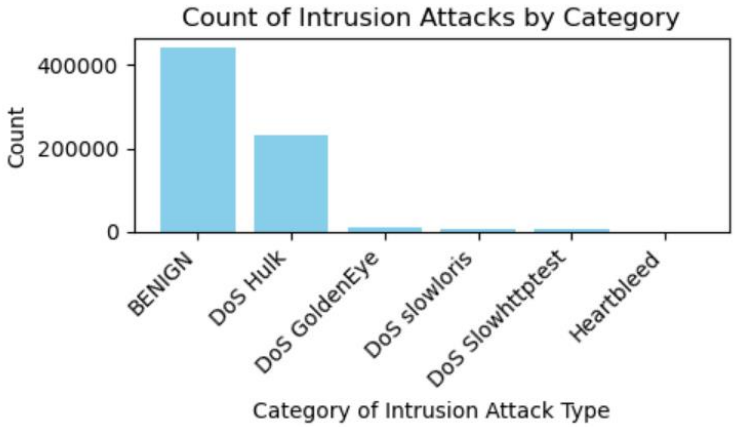


Figure **4. 6:** Category of the Intrusion Attack Type

4.4 Data Pre-processing

First, it begins with label encoding from multi-class into binary as earlier revealed in Figure **4.6**, the label column contains multiclass attack instances such as DoS slowloris, DoS Slowhttptest, DoS Hulk, DoS GoldenEye, and Heartbleed. These were replaced with the label "ATTACK" as in Figure **4.7.** Therefore, the classification task now involves two distinct classes: "ATTACK" and "BENIGN". Figure **4.8** reveals the new binary class distribution which show an extreme case of class imbalance as the “ATTACK” class is barely 50% population of the “BENIGN” class

```
# Create a copy of df4 and assign it to df5
df5 = df4.copy()

# Replace other category labels except BENIGN with ATTACK
df5.loc[df5['label'] != 'BENIGN', 'label'] = 'ATTACK'

# Display unique label types after replacement
print("Unique label types after replacement:")
print(df5['label'].unique())
```

Figure **4. 7:** Conversion of multi-class to binary by replacement.

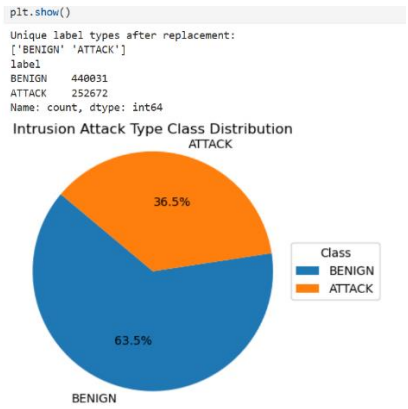


Figure **4. 8**: Intrusion attack type class distribution.

Next is Label encoding as in Figure 9, was implemented by representing the "ATTACK" class by 1, while the "BENIGN" class was encoded as 0, thus, creating a new column “encoded_label” in df5. Hence, the new column contains numerical digits 0 and 1. This transform aids data processing and pattern learning in machine learning.

```
# Define label mapping dictionary for converting categorica
label_mapping = {"BENIGN": 0, "ATTACK": 1}

# Map labels using the label mapping dictionary
df5["encoded_label"] = df5["label"].map(label_mapping)

# Display unique label types after label mapping
print("Unique encoded_label types after label mapping:")
print(df5["encoded_label"].unique())

df5.info()
```

```
# Create a copy of df4 and assign it to df5
df5 = df4.copy()

# Replace other category labels except BENIGN with ATTACK
df5.loc[df5['label'] != 'BENIGN', 'label'] = 'ATTACK'

# Display unique label types after replacement
print("Unique label types after replacement:")
print(df5['label'].unique())
```

```
# Display unique encoded_label types (0,1)
print("Unique encoded_label types:")
print(df5["encoded_label"].unique())
```

```
Unique encoded_label types:
[0 1]
```

Figure **4. 9:** Label Encoding

Figure 4.10 defines the “x” and “y” variables which represent the input feature dataframe variable and the target variable of class predictions respectively in ML. Some columns were dropped from the “x” dataframe since those columns by domain knowledge are less significant to intrusion attack. They could vary and change.

```
[17]: # The encoded_label represents the prediction class (target variable, y) while the x stands for the collection of input features for ML Training
# Dropping of flow_id, source_ip, destination_ip, timestamp, label and encoded_label categorical features owing to their lack of significance to Anomaly-

X = df5.drop(["flow_id                    ", "source_ip", "destination_ip", "timestamp", "label", "encoded_label"], axis=1)
y = df5["encoded_label"]
X.info()
y.info()
```

Figure **4. 10**: Input Feature and Target Variable Defined

These were ’flow_id,’ ’label,’ ’timestamp,’ ’source_ip,’ ’destination_ip,’ and ’encoded_label. The old label column needs to go as the “encoded_label” is now the new target variable y. It was observed from the dataset that it containe some missing values and infinity entries, anout 1008 missing values, 289 infinity entries for flow_bytes/s columns and 1297 infinity entries in flow_packets/s columns as showm in Figure **4.11.** This was cleaned up by replacing with NaN. Mean imputation is applied using the ’SimpleImputer’ class from ’scikit-learn’. This function replaces missing values with the mean of non-missing values within the same column for data integrity in modelling. Figure 4.11 explain this further.

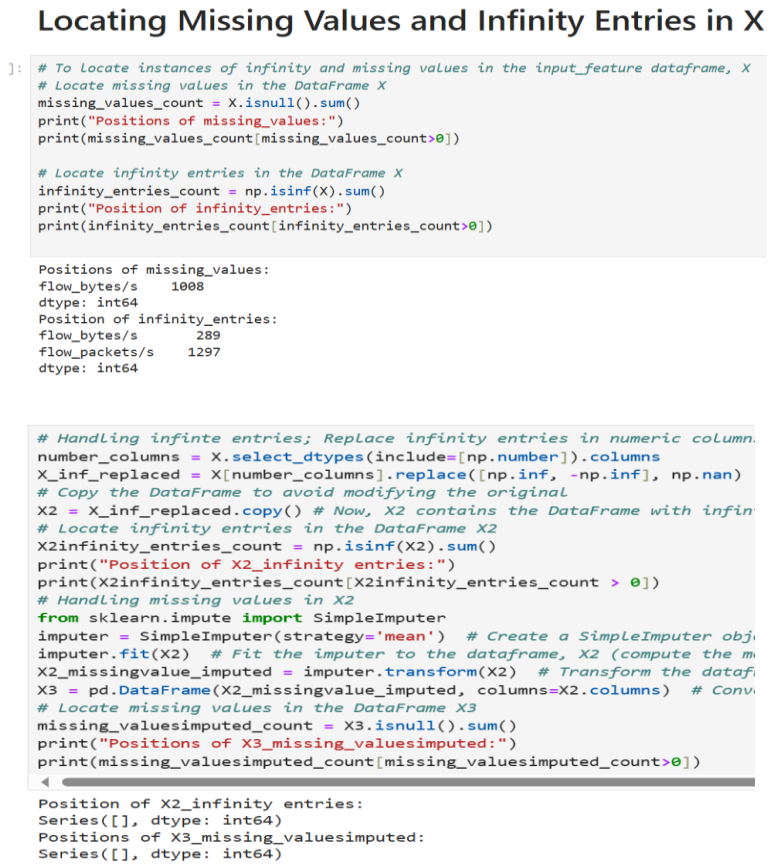


Figure **4. 11:** Locating and cleaning up missing values and infinity entries

As seen in Figure **4.8 ,** 440,031 instances of BENIGN traffic and 252,672 instances of attack traffic were contained in the dataset. This is a gross a class imbalance issue. Bagged undersampling technique was employed to resolve this issue as explained in Chapter 3. Hence, both ATTACK and BENIGN classes have the same number of entries as shown in Figure **4.12.**

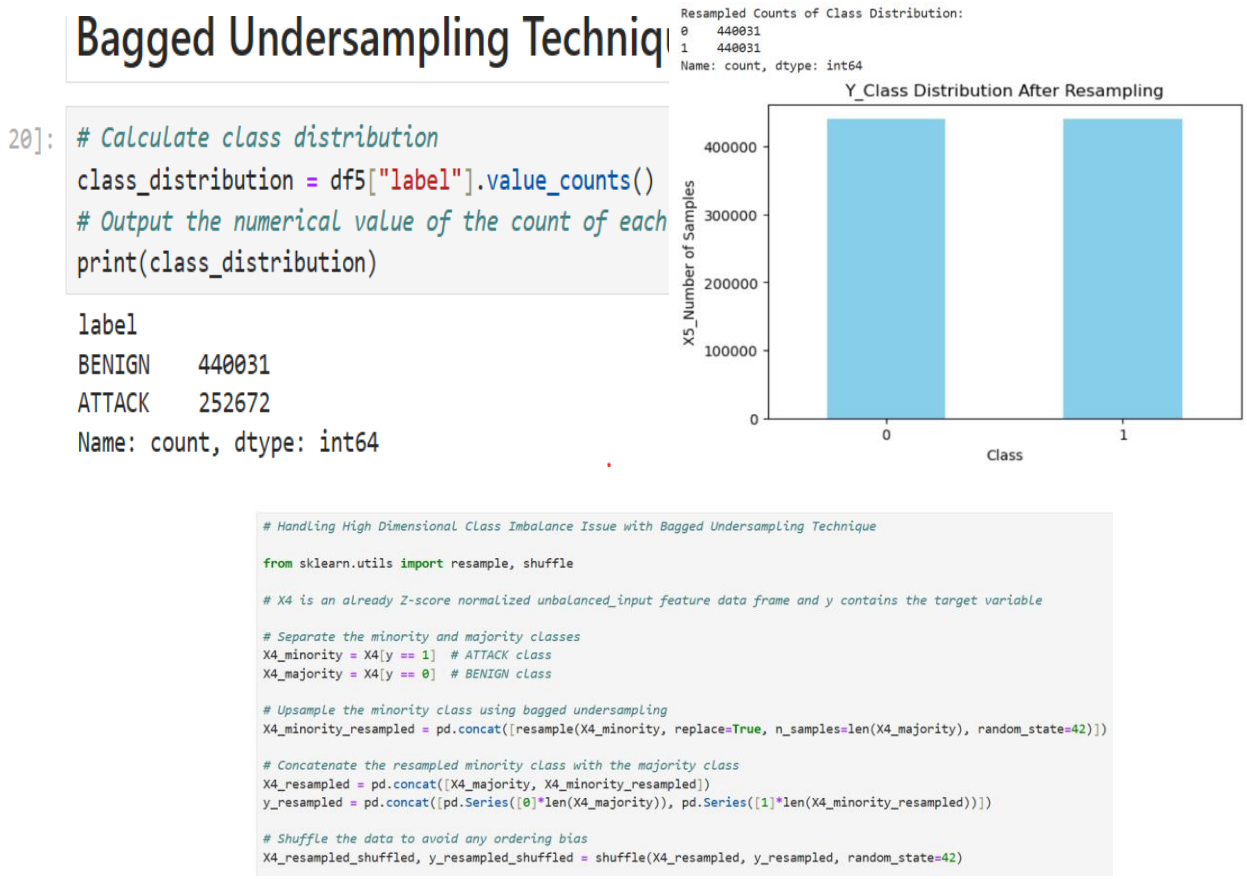


Figure **4. 12**: Balancing Classes using Bagged Undersampling Technique

Next is normalization of the dataset as explained in Chapter 3. Two techniques were explored in this study, Z-score and the Min-Max to evaluate impact on performance on base models. Figures **4.13** and **4.14** show the scripting of these followed by the description of the data output by running the .head() commands on X3_scaled and X3_scaled2 dataframes. The standardscaler and the min-maxscaler package from the sklearn processing library did this.

```
[22]: # using z-score normalization to rescale and normalize the unbalanced input_fea

from sklearn.preprocessing import StandardScaler

X3_scaler = StandardScaler()  # Creating a StandardScaler object, X3_scaler

# Apply z-score normalization to the numerical columns of X3
X3_scaled = X3.copy()  # Make a copy of the original DataFrame, X3 so as to pre
numerical_cols = X3_scaled.select_dtypes(include=['float64', 'int64']).columns
X3_scaled[numerical_cols] = X3_scaler.fit_transform(X3_scaled[numerical_cols])

# Display the normalized DataFrame
X3_scaled.head()
```

```
# Display the normalized DataFrame
X3_scaled.head()
```

2]:

	source_port	destination_port	protocol	flow_duration	total_fwd_packets	total_backward_packets	total_length_of_fwd_packets	total_length_of_bwd_packets	fwd_pack
0	0.351920	-0.356503	-0.642646	-0.653857	-0.011451	-0.009362	-0.089086	-0.007581	
1	0.351613	-0.336856	-0.642646	-0.654742	0.001932	-0.005298	-0.062154	-0.007438	
2	0.181207	-0.355994	-0.642646	-0.654728	0.000594	-0.004282	0.421001	-0.006178	
3	0.351664	-0.336856	-0.642646	-0.654398	0.009962	0.001815	0.469998	-0.004612	
4	0.181309	-0.355994	-0.642646	-0.654728	-0.000744	-0.004282	0.421001	-0.006177	

5 rows × 79 columns

Figure **4. 13**: Z-score Normalization

```
# using Min-Max normalization to rescale and normalize the unbalanced input_feature
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# X3 is your unbalanced input feature DataFrame

# Create a MinMaxScaler object
X3_scaler2 = MinMaxScaler()

# Make a copy of the original DataFrame to preserve its integrity
X3_scaled2 = X3.copy()

# Select numerical columns of both Float64 & int64 data types
numerical_cols = X3_scaled2.select_dtypes(include=['float64', 'int64']).columns

# Apply min-max scaling to the numerical columns
X3_scaled2[numerical_cols] = X3_scaler2.fit_transform(X3_scaled2[numerical_cols])
```

	source_port	destination_port	protocol	flow_duration	total_fwd_packets	total_backward_packets	total_length_of_fwd_packets	total_length_of_bwd_packets	fwd_packe
0	0.754696	0.001222	0.352941	0.000319	0.000000	0.000004	0.000005	9.569378e-09	
1	0.754604	0.005940	0.352941	0.000004	0.000049	0.000018	0.000141	5.199362e-07	
2	0.703807	0.001344	0.352941	0.000009	0.000044	0.000022	0.002573	5.023923e-06	
3	0.754620	0.005940	0.352941	0.000127	0.000078	0.000044	0.002820	1.062201e-05	
4	0.703838	0.001344	0.352941	0.000009	0.000039	0.000022	0.002573	5.027113e-06	

5 rows × 79 columns

X3 --->> X6

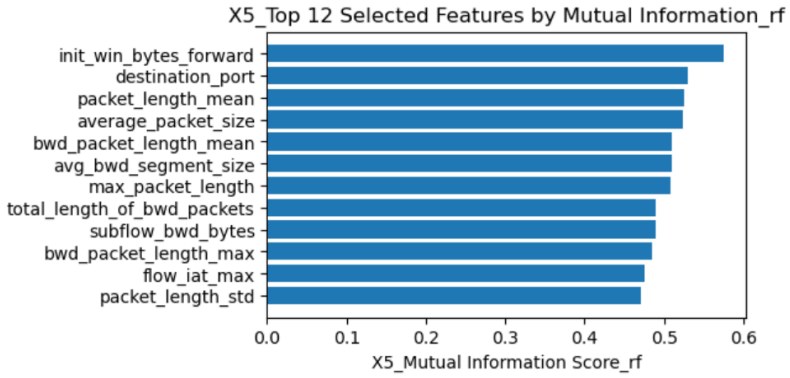
Figure **4. 14**: Min-Max Normalization

4.5 Feature Selection

Two method of features selection were employed in this study for analysis of impact on the performance of base models, the Mutual Information and the SelectKBest with ANOVA F-value scoring function as already explained in Chapter 3. From sklearn.feature_selection mutual_info_classif and SelectKBest with f_classif were imported. In mutual information-based method, argsort() gets the indices for the best 12 features using the mi_scores while in the SelectKBest coding the selector uses the f_classif function to select k best features bases on the f-values. Figures **4.15** and **4.16** show the portion if the coding for MI and SelectKBest respectively. These are just samples for 12 number of selected features.

```
# Implementing Mutual Information-based feature selection method for RF classifier along with

from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.feature_selection import mutual_info_classif
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Split the data into train and test sets
X5_train, X5_test, Y_train, Y_test = train_test_split(X5, Y, test_size=0.2, random_state=42)

# Perform feature selection using Mutual Information-based technique
mi_scores = mutual_info_classif(X5_train, Y_train, random_state=42)

# Get the indices of the top 12 features based on mutual information scores
selected_feature_indices = (-mi_scores).argsort()[:12]

# Select the top 12 features from the training and test sets
X5rf_train_selected = X5_train.iloc[:, selected_feature_indices]
X5rf_test_selected = X5_test.iloc[:, selected_feature_indices]

# Start the training timer
start_trt = time.time()
```

```
X5_Test set accuracy_rf: 0.999727292870413
```




```
Best hyperparameters: {'n_estimators': 100, 'min_samples_split': 2, 'max_depth': None}
Training Time: 3667.8282046318054 seconds
Testing Time: 2.107238531112671 seconds
```

Figure **4. 15**: Mutual Information-based feature selection method implemented

```
# Identify constant features
constant_features_idx = np.where(X5_train.var() == 0)[0]

# Remove constant features
X5_train = X5_train.drop(columns=X5_train.columns[constant_features_idx])
X5_test = X5_test.drop(columns=X5_test.columns[constant_features_idx])

# Perform feature selection using SelectKBest with ANOVA F-value scoring
selector = SelectKBest(score_func=f_classif, k=12)
X5_train_selected = selector.fit_transform(X5_train, Y_train)
X5_test_selected = selector.transform(X5_test)

# Get the indices of the selected features
selected_feature_indices = selector.get_support(indices=True)
selected_features = X5.columns[selected_feature_indices]
```

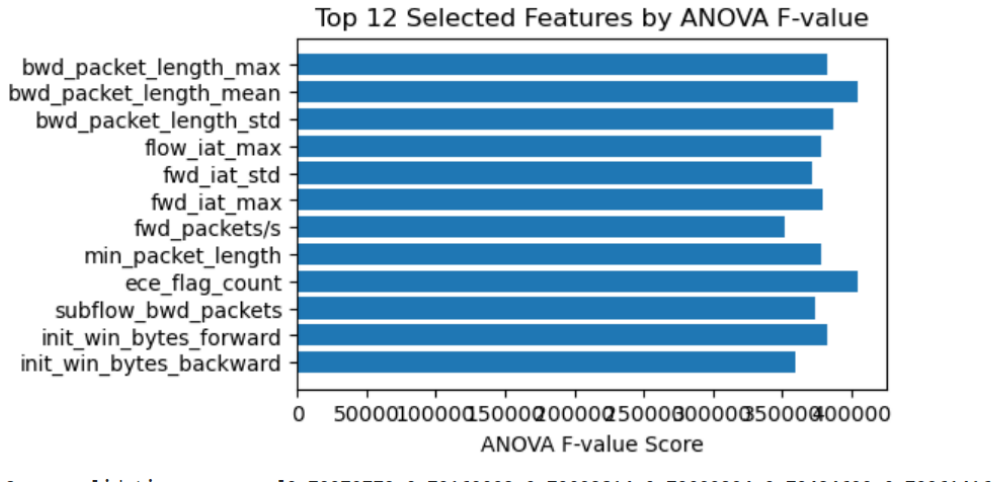


Figure **4. 16**: SelectKBest with ANOVA F-value feature selection method implemented

4.5 Ensemble Base Model Training and Testing

Base model training was done by using a train_test split of 80/20. Random state was set to 42 for all model trainings to ensure reproducibility. Hyperparameter tuning with cross-validation was achieved RandomSearchCV method for RF and XGBoost, while default smoothing was used for NB and 10-fold cross-validation was used to validate NB performance accuracy result. 5-fold cross-validation setting was set for RF and XGBoost constantly in all modelling involving them to save time. Figures **4.17** to **4.18** below are just samples of base model training and testing for the three base models.

```
# Initialize XGBoost model
X5xgb_model = XGBClassifier()

# Define the parameter grid for hyperparameter tuning for XGBoost Model
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [3, 5],
    'learning_rate': [0.1, 0.01]
}
```

```
# Initialize RandomizedSearchCV with 5-fold cross-validation
X5xgb_random_search = RandomizedSearchCV(estimator=X5xgb_model, param_distributions=param_grid, n_iter=5, cv=5, random_state=42)
```

```
# Fit RandomizedSearchCV to the training data
X5xgb_random_search.fit(X5_xgb_train_selected, Y_train)

# Print the best hyperparameters
print("X5_xgb_Best hyperparameters:", X5xgb_random_search.best_params_)

# Train XGBoost model with the best hyperparameters
X5best_xgb_model = XGBClassifier(**X5xgb_random_search.best_params_)
X5best_xgb_model.fit(X5_xgb_train_selected, Y_train)

# Make predictions on the test set
X5_Y_pred_xgb = X5best_xgb_model.predict(X5_xgb_test_selected)

# Calculate accuracy
X5xgb_accuracy = accuracy_score(Y_test, X5_Y_pred_xgb)
print("X5_Test set accuracy_xgb:", X5xgb_accuracy)

# Plot the feature importance scores
plt.figure(figsize=(5, 3))
```

Figure **4. 17**: XGBoost Model Training and Testing

```
# Initialize Random Forest model
X5rf_model = RandomForestClassifier()

# Define the parameter grid for hyperparameter tuning for Random Forest Model
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [None, 5],
    'min_samples_split': [2, 5]
}
```

```
# Initialize RandomizedSearchCV with 5-fold cross-validation
X5rf_random_search = RandomizedSearchCV(estimator=X5rf_model, param_distributions=param_grid, n_iter=5, cv=5, random_state=42)
```

```
# Fit RandomizedSearchCV to the training data
X5rf_random_search.fit(X5rf_train_selected, Y_train)

# Print the best hyperparameters
print("Best hyperparameters:", X5rf_random_search.best_params_)

# Train Random Forest model with the best hyperparameters
X5best_rf_model = RandomForestClassifier(**X5rf_random_search.best_params_
X5best_rf_model.fit(X5rf_train_selected, Y_train)

# Make predictions on the test set
X5Y_pred_rf = X5best_rf_model.predict(X5rf_test_selected)

# Calculate accuracy
X5rf_accuracy = accuracy_score(Y_test, X5Y_pred_rf)
print("X5_Test set accuracy_rf:", X5rf_accuracy)
```

Figure **4. 18**: RF Model Training and Testing

```python
# Initialize Naive Bayes model
X5nb_model = GaussianNB()

# Perform k-fold cross-validation
cv_scores = cross_val_score(X5nb_model, X5_train_selected, Y_train, cv=10)

# Print cross-validation scores
print("Cross-validation scores:", cv_scores)
print("Mean cross-validation score:", cv_scores.mean())

# Train Naive Bayes model on the entire training set
X5nb_model.fit(X5_train_selected, Y_train)

# Make predictions on the test set
X5nb_Y_pred = X5nb_model.predict(X5_test_selected)

# Calculate accuracy
X5nb_accuracy = accuracy_score(Y_test, X5nb_Y_pred)
print("X5nb_Test set accuracy:", X5nb_accuracy)
```

```
Cross-validation scores: [0.94621121 0.94623961 0.94533059 0.94517435 0.94531638 0.94694979
 0.94517435 0.94567147 0.94531638 0.94487529]
Mean cross-validation score: 0.9456259425826167
X5nb_Test set accuracy: 0.9460096697403033
```

Figure **4. 19**: NB Modelling with 10-fold Cross-Validation

4.5 Ensemble Model Training and Testing

Figure **4.20** shows an end-to-end modelling of (NB + RF) + XGBoost proposed model to describe the coding pipeline. This starts off with loading essential libraries . The predictions of NB and RF are stacked with the original selected training feature dataframe to train the meta-data using the ".npcolumn_stack" function. The stacked prediction dataframe is the divided using 70/30 as against the 80/20 used for each of the base models and the final prediction is made by the meta-model. Table **4.1** shows the RandomSearchCV found hyperparameters.

Table **4. 1:** Best Hyperparameters for the Proposed ensemble model at 12 by RandomSearchCV

Ensemble Model	Base Classifier Hyperparameter	Best Meta-Classifier Hyperparameter
NB +	NB does not use Hyperparameters, CV(10)	n_estimators': 200, 'max_depth': 3, 'learning_rate': 0.1; CV(5)
RF +	n_estimators': 100, 'min_samples_split': 2, 'max_depth': None; CV(5)	
XGBoost		

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split, RandomizedSearchCV, cross_val_score
from xgboost import XGBClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from sklearn.feature_selection import mutual_info_classif
import numpy as np
import time



# X5 is an already Z-score normalized and Bagged-Undersampling Balanced_input feature data fr

# Split the data into train and test sets
X5_train, X5_test, Y_train, Y_test = train_test_split(X5, Y, test_size=0.2, random_state=42)

# Perform feature selection using Mutual Information-based technique
mi_scores = mutual_info_classif(X5_train, Y_train, random_state=42)

# Get the indices of the top 12 features based on mutual information scores
selected_feature_indices = (-mi_scores).argsort()[:12]

# Select the top 12 features from the training and test sets
```

```python
# Start the training timer
start_trt = time.time()
start_trt3 = time.time()

# Initialize Naive Bayes model
X5nb_model = GaussianNB()

# Train Naive Bayes model on the entire training set
X5nb_model.fit(X5_train_selected, Y_train)

end_trt3 = time.time()

start_trt4 = time.time()
# Initialize Random Forest model
X5rf_model = RandomForestClassifier()

# Define the parameter grid for hyperparameter tuning
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [None, 5],
    'min_samples_split': [2, 5]
}
```

```python
# Initialize RandomizedSearchCV with 5-fold cross-validation
X5rf_random_search = RandomizedSearchCV(estimator=X5rf_model, param_distributions=param_grid, n_iter=5, cv=5, random_state=42)

# Fit RandomizedSearchCV to the training data
X5rf_random_search.fit(X5_train_selected, Y_train)

# Train Random Forest model with the best hyperparameters
X5best_rf_model = RandomForestClassifier(**X5rf_random_search.best_params_)
X5best_rf_model.fit(X5_train_selected, Y_train)
end_trt4 = time.time()

# Stack predictions from NB and RF models to use as additional features for XGBoost meta-model
X5nb_predictions = X5nb_model.predict(X5_train_selected)
X5rf_predictions = X5best_rf_model.predict(X5_train_selected)
XGBoost_input = np.column_stack((X5_train_selected, X5nb_predictions, X5rf_predictions))

# Split the data for meta-model training
X_xgb_train, X_xgb_test, Y_xgb_train, Y_xgb_test = train_test_split(XGBoost_input, Y_train, test_size=0.3, random_state=42)
```

```python
# Start the training timer
start_trt2 = time.time()

# Initialize XGBoost meta-model
xgb_meta_model = XGBClassifier()

# Define the parameter grid for hyperparameter tuning for XGBoost Model
param_grid_xgb = {
    'n_estimators': [100, 200],
    'max_depth': [3, 5],
    'learning_rate': [0.1, 0.01]
}

# Initialize RandomizedSearchCV with 5-fold cross-validation for XGBoost
xgb_random_search = RandomizedSearchCV(estimator=xgb_meta_model, param_distributions=param_grid_xgb, n_iter=5, cv=5, random_state=42)

# Perform hyperparameter optimization for XGBoost
xgb_random_search.fit(X_xgb_train, Y_xgb_train)

# Use the best model from RandomizedSearchCV for XGBoost
best_xgb_meta_model = xgb_random_search.best_estimator_

# Train the XGBoost meta-model with the best hyperparameters
best_xgb_meta_model.fit(X_xgb_train, Y_xgb_train)

# End the training timer
end_trt2 = time.time()
```

Figure **4. 20**:Phases of Ensemble Modelling Scripting

CHAPTER 5: EVALUATION

This section promises to extensively analyse all conducted research experiments and comparatively identify the best models based on performance and computational efficiency. This evaluation section has been strategically divided into seven phases as highlighted in Chapter 3, Methodology. In the view of this, the ML modelling experiments were run several times for several objectives (Flowchart at Figure **3.1**), and the results were systematically documented.

5.1 Impact of Choice of Normalization Technique on Performance of Ensemble Baseline Models

From Figure **5.1** below, and Table **5.1** (Appendix **B**), RF and XGBoost performance accuracy remains approximately the same at 99.97% while on either z-score normalized or min-max normalized dataset. Likewise, NB remains constant for both techniques at 94.62%. This indifference suggests that the normalized datasets were able to retain enough important classification information to uphold same accuracy, as none of the two techniques had been able to distort the data attribute so poorly that the three base models begin losing accuracy (Alghushairy et al. 2024). However, owing to the statistical robustness of z-score technique to capture and withstand outliers, ensemble base models trained on z-score normalized datasets were introduce into the ensemble modelling (Santos, Miani and Silva 2024).

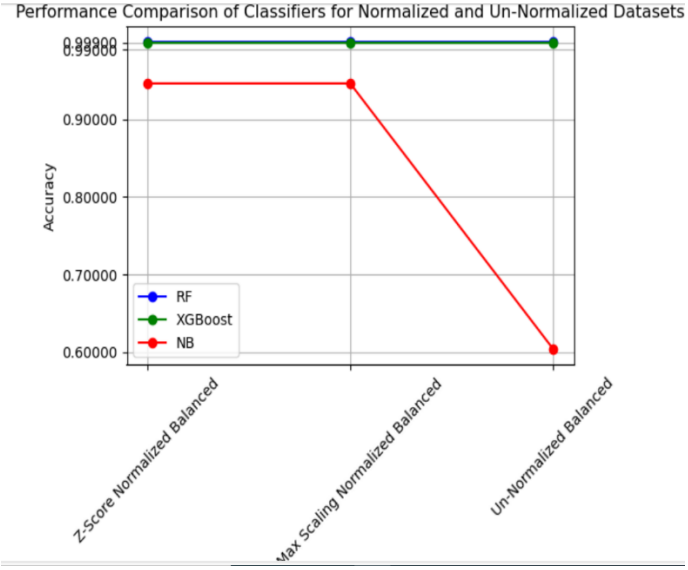


Figure **5. 1**: Performance Comparison of Ensemble Base Models for Normalized and Unnormalized Datasets

5.2 Impact of Normalization on Performance of Ensemble Baseline Model

From Figure **5.1** below, and Table **5.2** (Appendix **B**), The performance accuracy (99.97%) of RF and XGBoost ensemble base models remain indifferent to normalization of the training dataset. This further proves the robustness of both to variance and bias. However, NB became sensitive to the noise in the unbalanced dataset and its accuracy

dropped to 60.35%. This proves the significance of normalization when dealing with large datasets especially with simple classifier like NB systems (Jemili, Meddeb and Korbaa **2023**).

In addition, RF and NB appear to have benefited from the dataset normalization as the model training time dropped from 3753.611sec to 3667.838sec for RF and 4.034sec to 3.365sec for NB after normalization. This is expected as normalization reduces computation time in ML (Rashid et al. **2022)**. The reverse is the case for XGBoost as it increased from 136.168sec to 147.65sec. The forward error correction capability of Gradient boosting could be attributed to this, as normalization at times can reduce the feature discriminants and which model may have to compensate for, especially for a variance-bias balance seeking algorithm like XGBoost (Vijay et al. **2021)**. Testing times were lower for RF and NB under unnormalized condition, while the reverse is the case for XGBoost. RF benefits from parallelization and bagging for fast prediction, while NB only requires few discriminants in the dataset to make probabilistic decision even in the presence of noise in an unnormalized data. XGBoost is characterised with sequential boosting with queuing mechanism (Mohammed and Kora 2023). Figures **5.2** and **5.3** show the time bar plots.

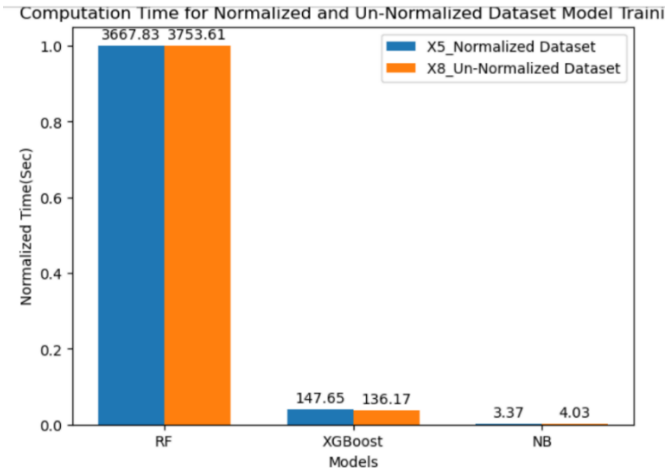


Figure **5. 2**: Training time for base models for normalized and unnormalized conditions

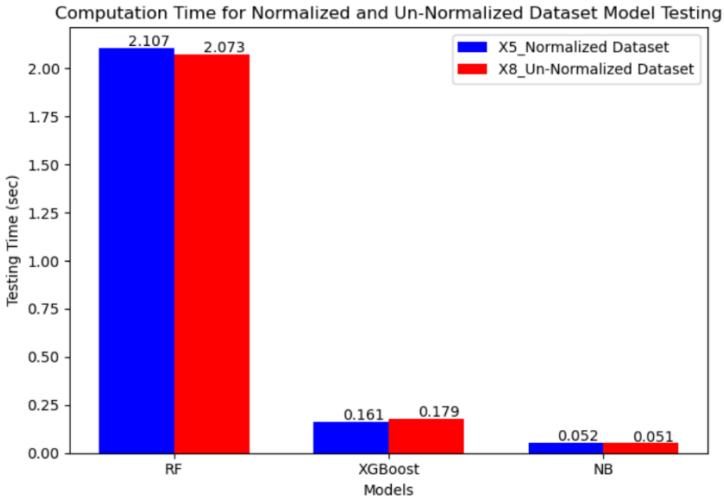


Figure **5. 3**: Testing time for base models for normalized and unnormalized conditions.

5.3 Impact of Class Imbalance on Ensemble Baseline Model Performance

From Figure **5.4** below, and Table **5.3** (Appendix **B**), ensemble base model performance accuracies were severely affected by class imbalance of the dataset with RF falling from 99.97% to as low as 99.37%; NB fell from 94.62% to 93.22%,

While XGBoost fell from approximately 99.90% to 99.32%. This proves that the best of all base models can be sensitive to class imbalance in large datasets, even when they are indifferent to normalization (Watthaisong et al. **2024**). Hence, the class-imbalance issue was handled with bagged undersampling.

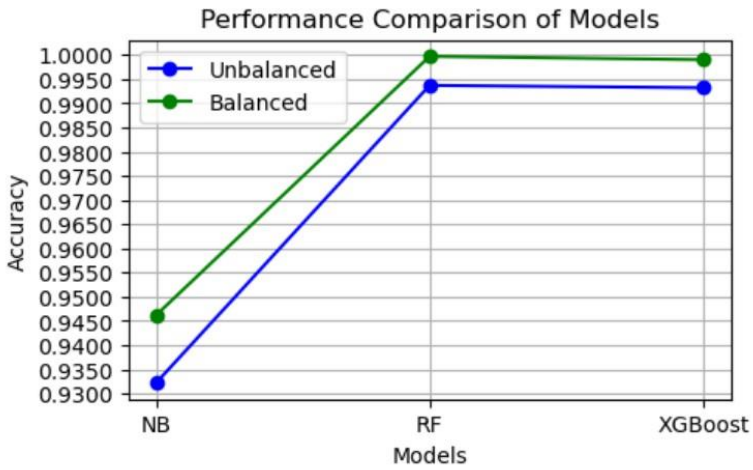


Figure **5. 4**: Impact of class imbalance on ensemble base model performance

5.4 Impact of Feature Selection Method on Performance of Ensemble Baseline Model

From Figures **5.5** to **5.7** below, and Table **5.4** (Appendix **B**), The Mutual Information-based feature selection method performed higher than the SelectKBest with ANOVA F-value scoring function method across the three base models and over a range of selected features, 14, 12 , 10, and 8. Figures **5.1** and **5.2** (in Appendix **B**) show the selected feature names by MI and selectKBest for 12 number of selected features. Other plots can be found in the additional submissions plus, scripts. The better performance of the MI method can be attributed to its feature-target variable mutual information importance scoring which should be of course richer than the Analysis of Variance comparison that SelectKBest uses (Rashid et al. 2022; Albulayhi et al. 2022).

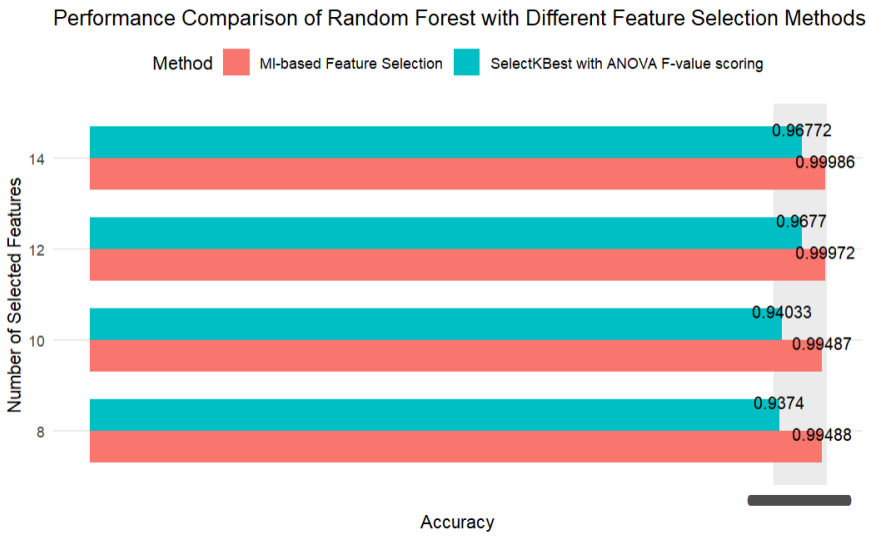


Figure **5. 5** Performance comparison of Feature Selection Methods on RF

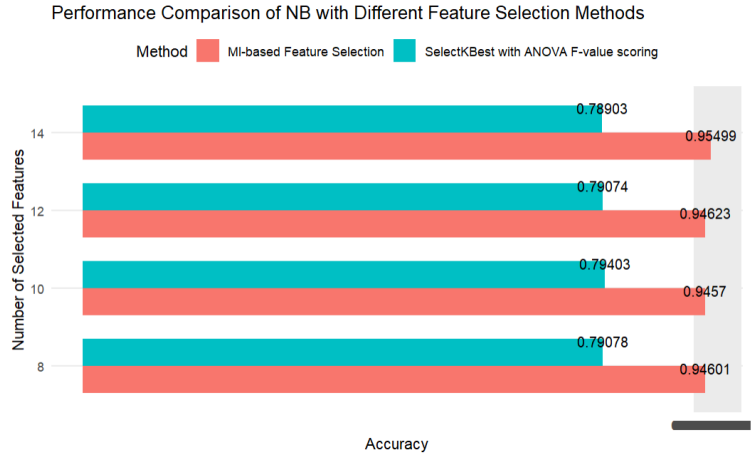


Figure **5. 6**: Performance comparison of Feature Selection Methods on NB

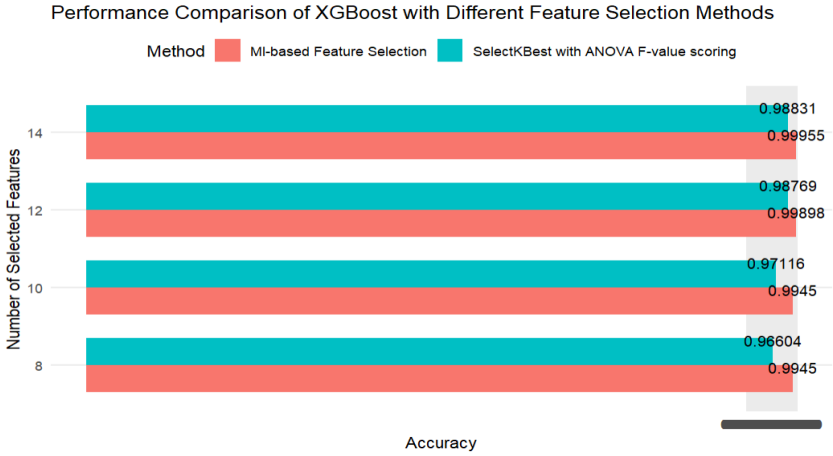


Figure **5. 7** Performance comparison of Feature Selection Methods on XGBoost

In addition, Figure **5.8** below and Table 5.2 (Appendix **B**) show base model performance using the MI selection method for the full features (79), 14, 12, and 10. Performance accuracies of the model, RF, XGBoost and NB at 12 and 8 selected features are 99.97%, 99.99%, 94.62% and 99.49%, 99.45%, 94.60% respectively.

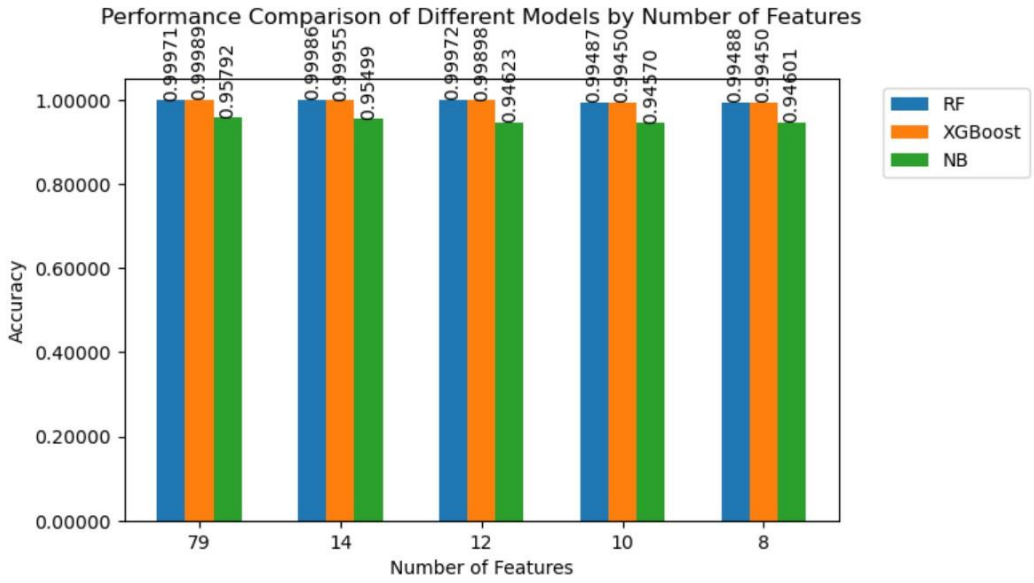


Figure **5. 8:** Performance comparison of Base Models across full features, 14, 10 and 8

5.5 Performance Analysis Nine (9) Developed Ensemble Models to Select the Ones Whose Accuracies are Greater than 99.90%.

Figure **5.9** below, and Table **5.5 (**Appendix **B)** below show performance accuracies of nine modelled and evaluated ensemble models. Stacking ensemble models, (RF) + XGBoost, (NB + RF) + XGBoost, (RF + XGBoost) + NB, and (RF)+ NB, with performance accuracies 0.9999, 0.9999, 0.9998, and 0.9998 at 12 selected number of features are best fit to explore with other performance metrics for the proposed ensemble model in this work with optimal performance and efficiency. All these four best accuracy ensemble models are NB-augmented tree-based stacking ensemble except (RF) + XGBoost. Note that the base models are sometimes put in bracket in this work to identify and separate them from the meta-model in the ensemble

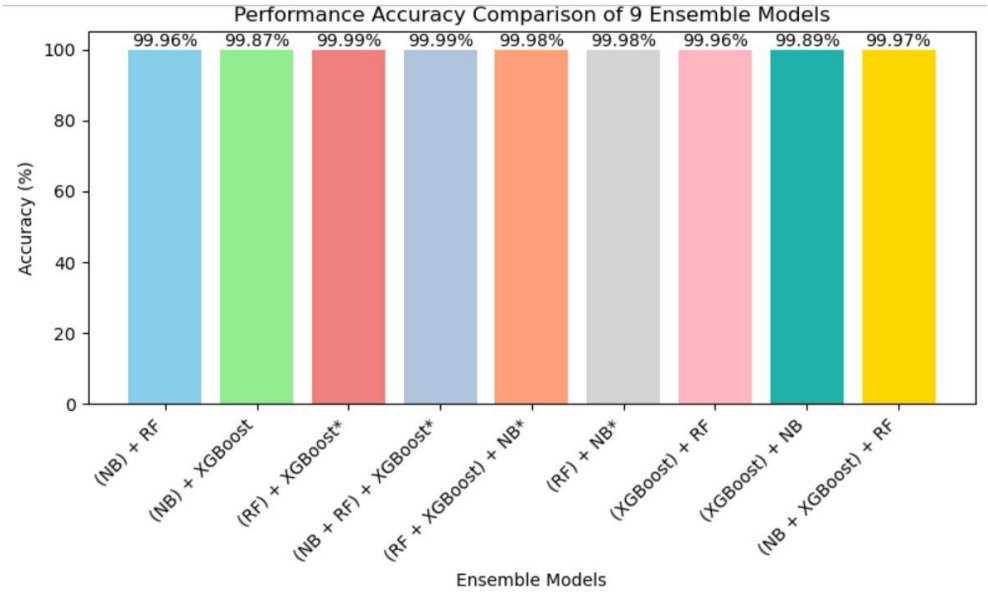


Figure **5. 9**: Performance Accuracies of Nine (9) Built Stacking Ensemble Models for Analysis

5.6 Performance analysis of selected Ensemble Models on Balanced and Unbalanced datasets.

Figure **5.10** and Table **5.6** (Appendix **B**) show the performance accuracy plot of the four best performing ensemble under balanced and unbalanced training datasets. This underscores the fact that the response of the ensemble base models to class imbalance was the same with the ensemble models built on them. Hence, ensemble learning would not correct the response to unbalanced dataset training. This is a glaring limitation subject to further research.

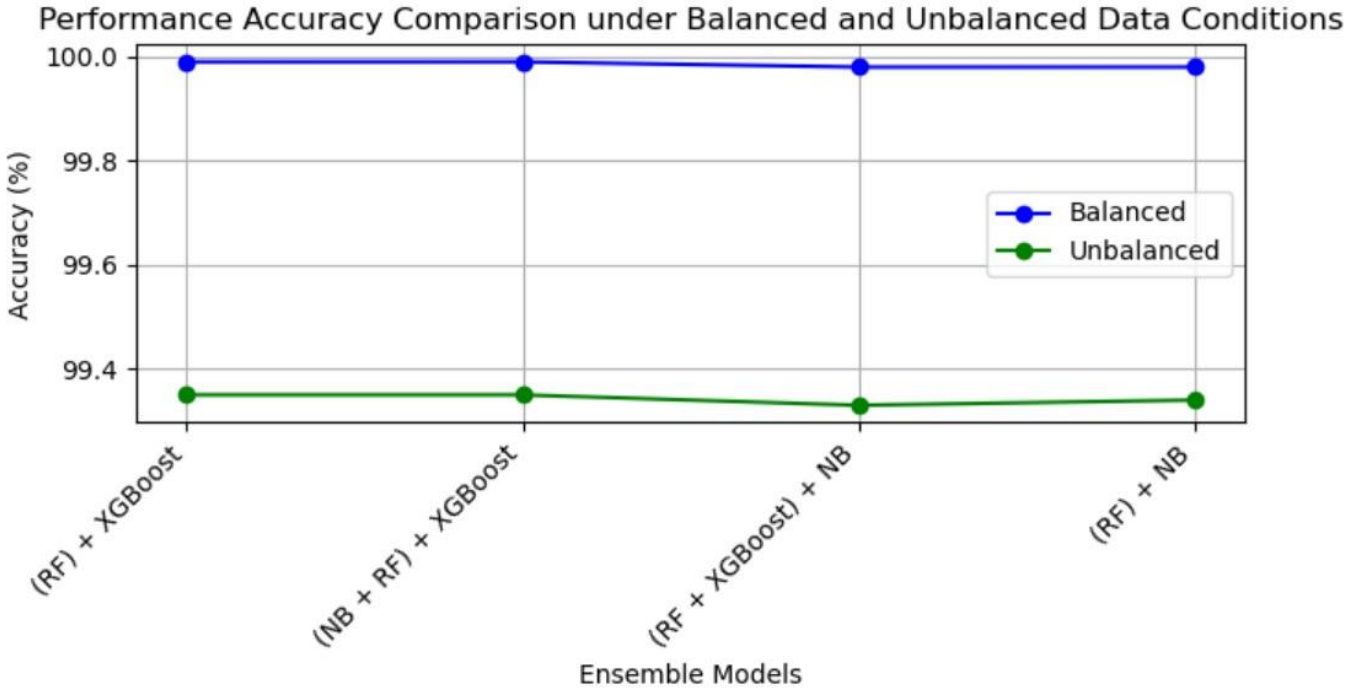


Figure **5. 10**: Performance of Four selected Ensemble Models on balanced and unbalanced datasets**.**

5.7 Evaluation of confusion matrix, accuracy, precision, recall, F1-score, False Alarm Rate (FAR), ROC-AUC curves, model training and testing times

- **Training and Testing Time Comparison**

Table **5.7** (Appendix **B**) and Figures **5.11** and **5.12** show the training and testing times of the four ensemble models. This is a **tie breaker** for the four best performing ensemble models, as **(NB + RF) + XGBoost** possess the least training time at 12 number of features while **(RF) + NB** possess the list training time at 8 features. They both have performance accuracy of 99.99% and 99.46% respectively. Since, both (NB + RF) + XGBoost and (RF) + XGBoost possess approximately same performance accuracy, likewise, (RF) + NB, and (RF + XGBoost) + NB. Figure 5.15 and Table 5.8 (Appendix **B**) breakdown the training time of the proposed ensemble model at 12 features. This proves that the introduction of NB into the RF + XGBoost tree-based ensemble lowers the learning time of the meta-model, XGBoost from 96.959sec to 88.467sec, hence, the total time of this ensemble becomes lower than the completely tree-based ensemble model

Figure **5. 11** : Training Time of Four Best Performing Ensemble Model at 12 Features

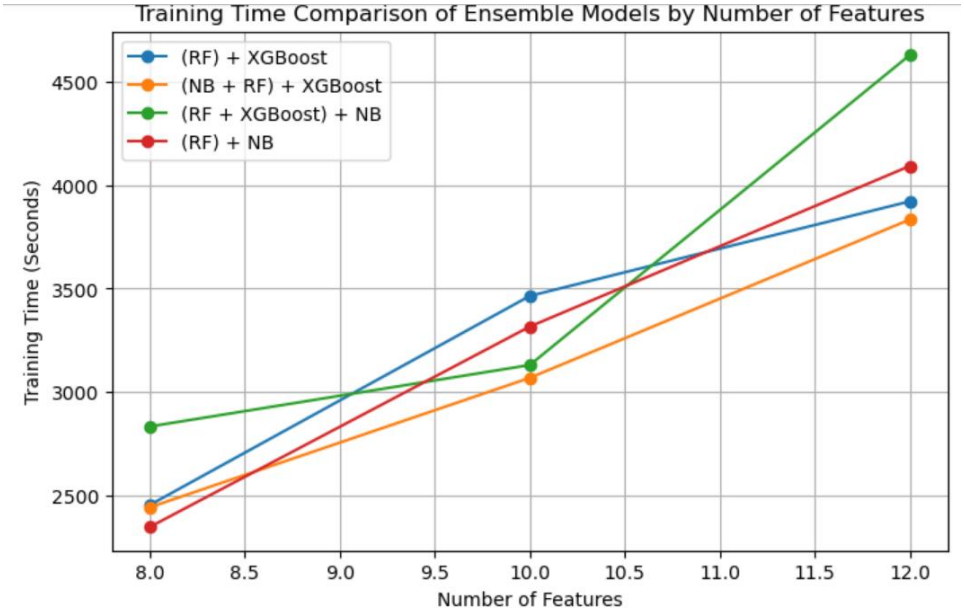


Figure **5. 12**: Training Time of Four Best Performing Ensemble Model

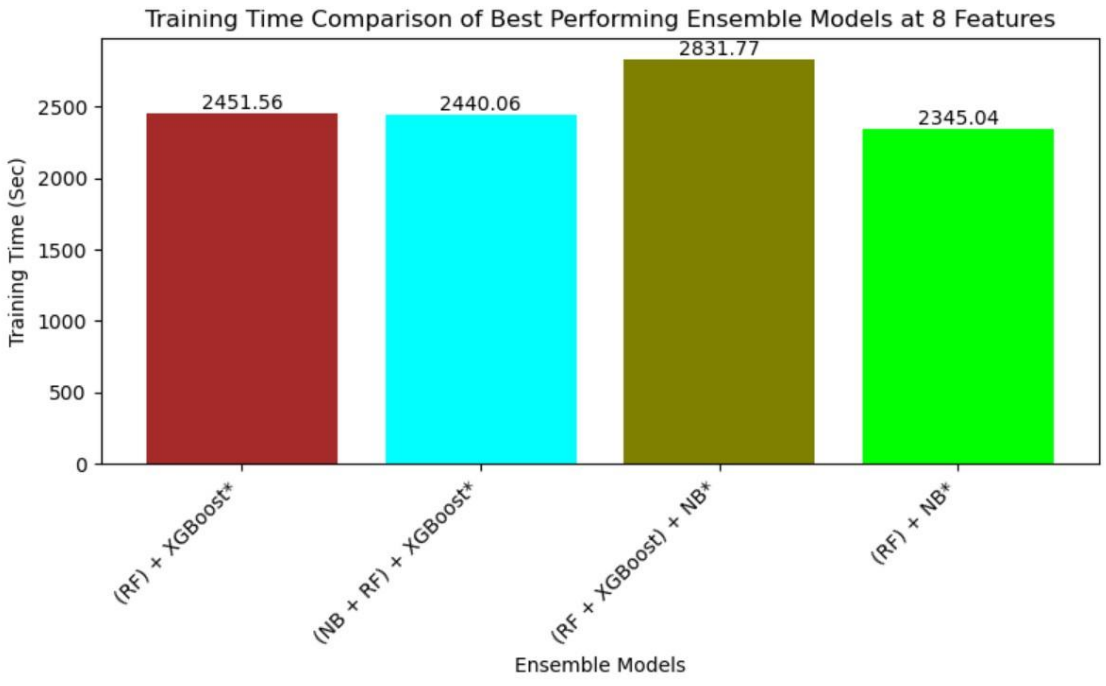


Figure **5. 13**: Training Time of Four Best Performing Ensemble Model at 8 Features

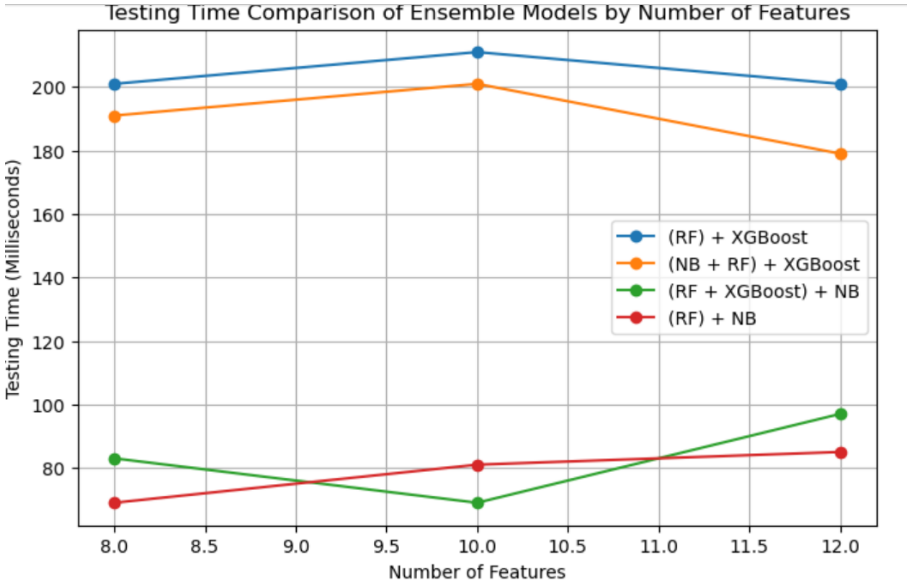


Figure **5. 14**: Testing Time of Four Best Performing

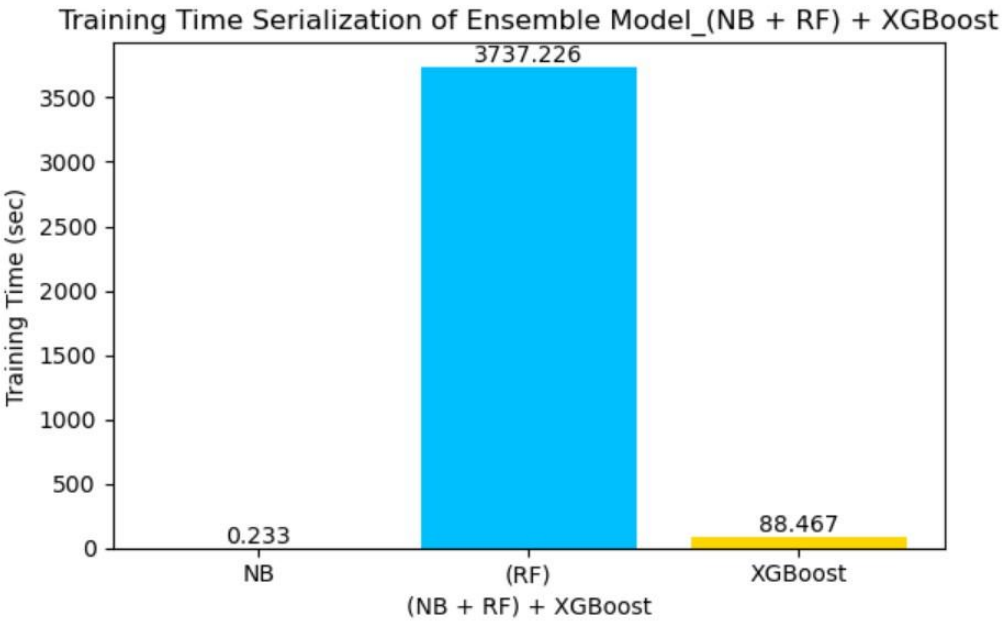


Figure **5. 15**: Time Serialization for the Best Performing Ensemble at 12 Features

- **Confusion Matrix**

The Figure **5.15** and Tables **5.1** and **5.2** below confusion matrices show the Fals Negatives (FN), False Positives (FP), True Positves (TP), and True Negatives (TN), Precision, Recall, FAR/FPR, and F1-score

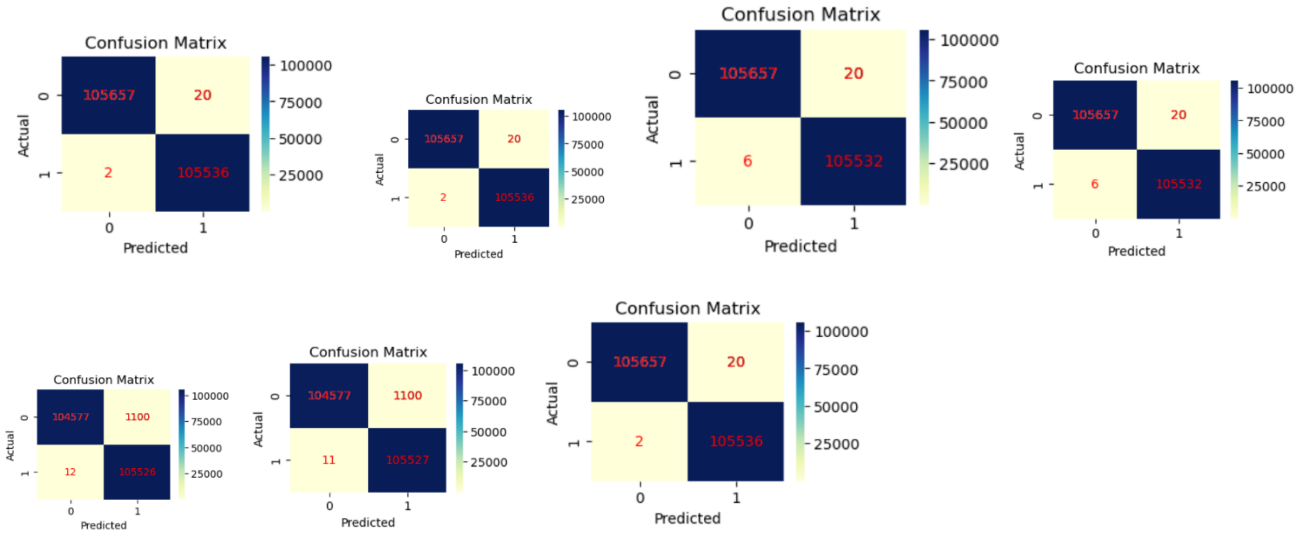


Figure **5. 16**: Confusion Matrix

Table **5. 1**: Confusion Matrix plus Precision , Recall, F1-score and FAR/FPR at 12 Features

Ensemble Model	Number of Selected Features	TP	TN	FP	FN	Precision	Recall /Sensitivity (DR/TPR)	F1 - Score	FAR/FPR
RF + XGBoost*	12	105536	105657	**20**	**2**	0.999811	0.999981	0.999896	0.000189
NB + RF + XGBoost*		105536	105657	**20**	**2**	0.999811	0.999981	0.999896	0.000189
RF + XGBoost + NB*		105532	105657	**20**	**6**	0.999811	0.999943	0.999877	0.000189
RF + NB*		105532	105657	**20**	**6**	0.999811	0.999943	0.999877	0.000189

Table **5. 2** : Confusion Matrix plus Precision , Recall, F1-score and FAR/FPR at 8 Features

Ensemble Model	Number of Selected Features	TP	TN	FP	FN	Precision	Recall /Sensitivity (DR/TPR)	F1 - Score	FAR/FPR
RF + XGBoost*	8	105526	104577	**1100**	**12**	0.989684	0.999886	0.994759	0.010409
NB + RF + XGBoost*		105526	104577	**1100**	**12**	0.989684	0.999886	0.994759	0.010409
RF + XGBoost + NB*		105516	104577	**1100**	**22**	0.989683	0.999792	0.994711	0.010409
RF + NB*		105522	104577	**1100**	**16**	0.989683	0.999848	0.99474	0.010409

- **ROC-AUC Curves:**

The below shoe the curves at 12 and 8 features.

The closeness of the curve up to the True Positive Rate (TPR) axis proves the efficacy of the proposed ensemble models at 12 and 8

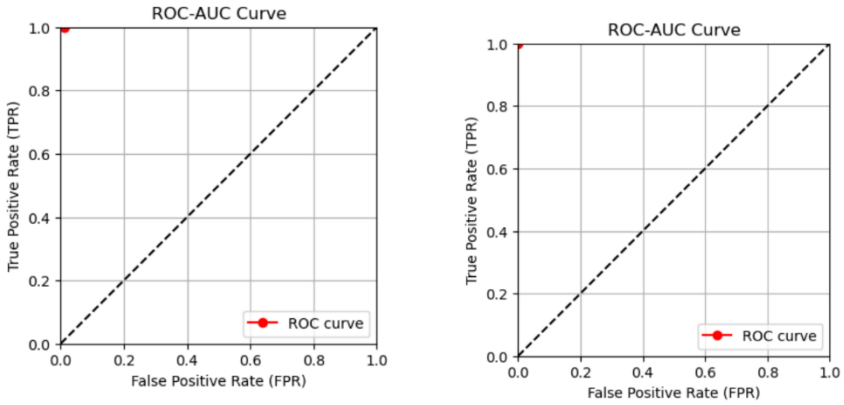


Figure **5. 17**: ROC-AUC Curves

CHAPTER 6: CONCLUSION AND FUTURE WORK

6.1 Conclusion

The scope of this study is to research into the machine learning pipeline of the ensemble of simple traditional ML models and optimize every phase towards producing an ensemble model with improved predictive performance accuracy over existing ones, while still benefitting from the lower computational cost and training times inherent in the chosen ensemble base models plus the robustness and adaptability that ensemble learning offers. The research experiments were conducted using the CICIDS benchmark dataset for NID on NB, RF and XGBoost ML models toward building and proposing best performing ensemble model. The evaluation was carried out in seven (7) phases and the results were systematically collected and presented. Following this, a comparative analysis of results was performed.

The work kicked-off with an investigation on the impact of normalization and choice of normalization technique (Z-score and Min-max) on the performance of ensemble base ML models (NB, RF and XGBoost). It was observed that even though ML models can be indifferent to normalization data-processing, their training and testing times are often impacted.

Subsequently, the impact of class imbalance in training dataset on base model performance accuracy was investigated. Results show that all three models dropped in performance accuracy with unbalanced data which was least pronounced with XGBoost. The forward error correction attribute of XGBoost could be attributed to this. Bagged undersampling technique was used for class balancing.

Furthermore, an exploration of the impact of choice of feature selection techniques (Mutual Information and SelectKBest with ANOVA) on performance of ensemble base model was conducted together with the investigation of performance across range of fewer number of selected features less than 16 (14, 12, 10 and 8). Results showed that Mutual Information (MI) technique proved superior to SelectKBest. This can be attributed with the feature-target information importance score on which operates, while SelectKBest mere work statistically based on analysis of variance between feature groups and classes. It was also observed that comparatively high predictive performance accuracy of 99.97%, 99.90%, and 94.62% with RF, XGBoost and NB respectively on 12 best selected features, and 99.49%, 99.45% and 94.60% respectively on eight (8) best selected features with MI were achieved. This met the dimensionality reduction without outright compromise on performance objective.

Following this, nine experimental ensemble models were built with the best performing base models of NB, RF and XGBoost with stacking ensemble method and meta-learning fusion method. All modelling of ensemble base models were done with hyperparameter tuning with cross-validation settings using RandomSearchCV.

Finally, models were evaluated using performance metrics such as confusion matrix, precision, recall, F1-score, False Alarm Rate (FAR), ROC-AUC Curve, model training and testing times. This led to the proposal of two NB-augmented stacking ensemble models for NIDS, (NB + RF) + XGBoost and (RF) + XGBoost which achieved performance accuracy of 99.99% and 94.60% respectively at 12 and 8 number of selected features respectively too.

These performance were achieved with comparatively low computational cost in terms of training and testing times, with best precision, recall, F1-score , lowest number of False Negatives (FN), False Positives (FP), and excellent ROC-AUC curves in their own categories.

6.2 Achievements

In comparison to recent works within the literature, the work presented in this study notable compete in accuracy, as shown in Figures **6.1** and **6.2,** and Tables **6.1** and **6.2.** It is worthy of note that the evaluation performance here can still vary among different studies depending on scenarios, etc.

The NB-augmented tree-based ensemble model at 12 number of used features, using MI-based feature selction method where NB and XGBoost are the base learners (models) and the XGBoost is the meta-model achieved the highest performance accuracy as seen in Table **6.1** and Figure **6.1**. Even, Abbas et al. (**2023**) which used an ensemble involving a multi-layer perceptron neural network mode recorded a lesser performance accuracy of 99.90%. This corroborates the proposal that well performing ensemble of traditional ML models like the one proposed in this study could compete well with neural network and deep learning solutions which are used to be computationally costly.

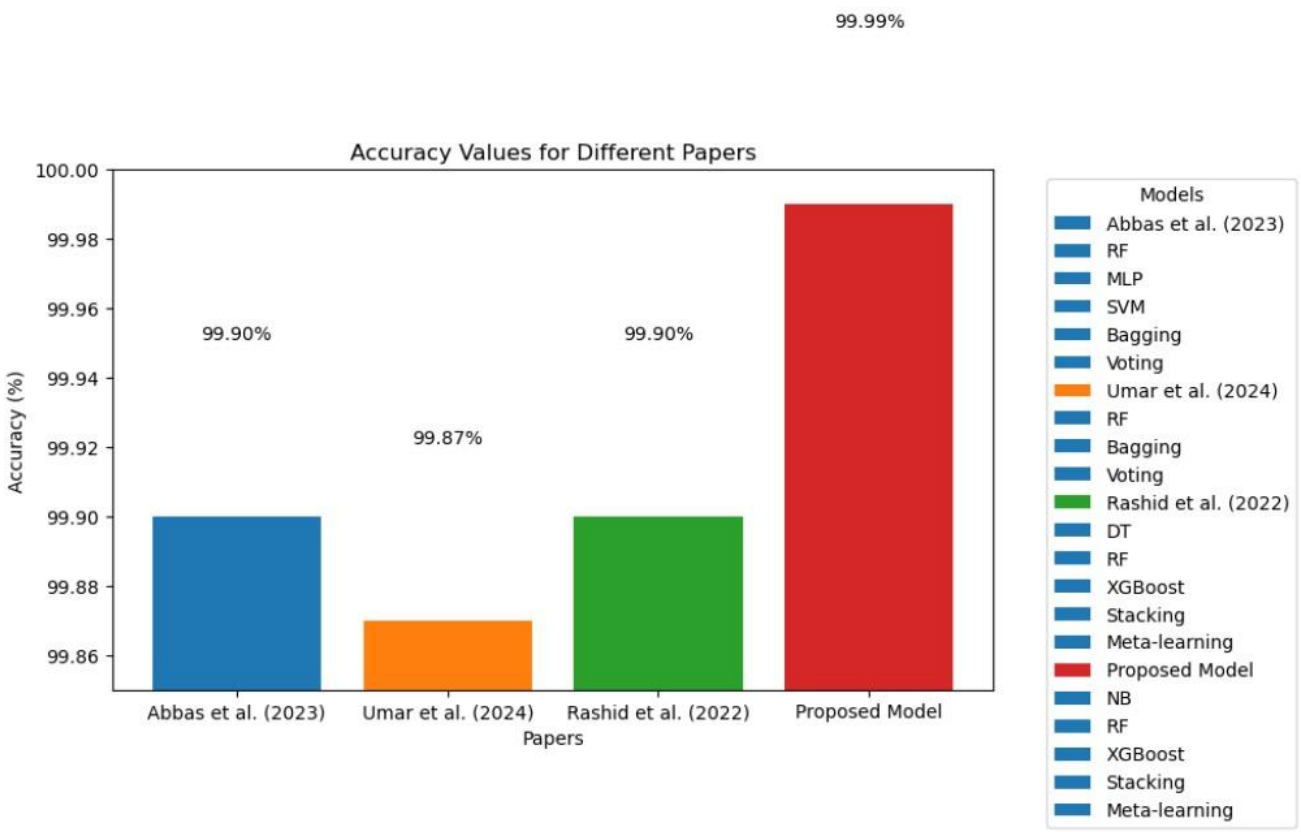


Figure **6. 1**: Existing Literature Comparison with the proposed ensemble mode at 12 used features

Table **6. 1**: Existing Literature Comparison with the proposed ensemble mode at 8 used features

Author & Year	Ensemble Model	Ensemble Method	Fusion Method	Dataset	Feature Selection Method	Number of Used Features	Maximum Achieved Accuracy
Abbas et al. (2023)*	RF, MLP* and SVM	Bagging	Voting	NSL-KDD,	Random Forest-Recursive Feature Elimination (RF-RFE)	15	99.90%
Umar et al. (2024)	Random Forest	Bagging	Voting	NSL-KDD	Decision tree wrapper-based approach	-	99.87%
Rashid et al. (2022)	(DT + RF) + XGBoost	Stacking	Meta-learning	NSL-KDD	SelectKBest	20	99.90%
Proposed Ensemble Model	**(NB + RF) + XGBoost**	**Stacking**	**Meta-learning**	**CICIDS 2017**	**Mutual Information-based**	**12**	**99.99%**

In Figure **6.2**, and Table **6.2,** there is Jemili, Meddeb and Korbaa **(2023)** which achieved a performance accuracy of 96.36% on completely tree-based stacking ensemble with meta-learning of RF and XGBoost evaluated with the same dataset (CICIDS2017) as the one in this work and with feature importance-based feature selection method. The second proposed NB-augmented tree-based stacking ensemble of RF as the ensemble base models of DTs, and NB as the meta-model performed better with accuracy of 99.46% at even same number of selected features (12) but using Mutual Information-based feature selection techniques. This underscores the proposal that not only completely tree-based stacking ensemble can achieve optimum performance in NIDS as NB-augmented ones can do the same and even better.

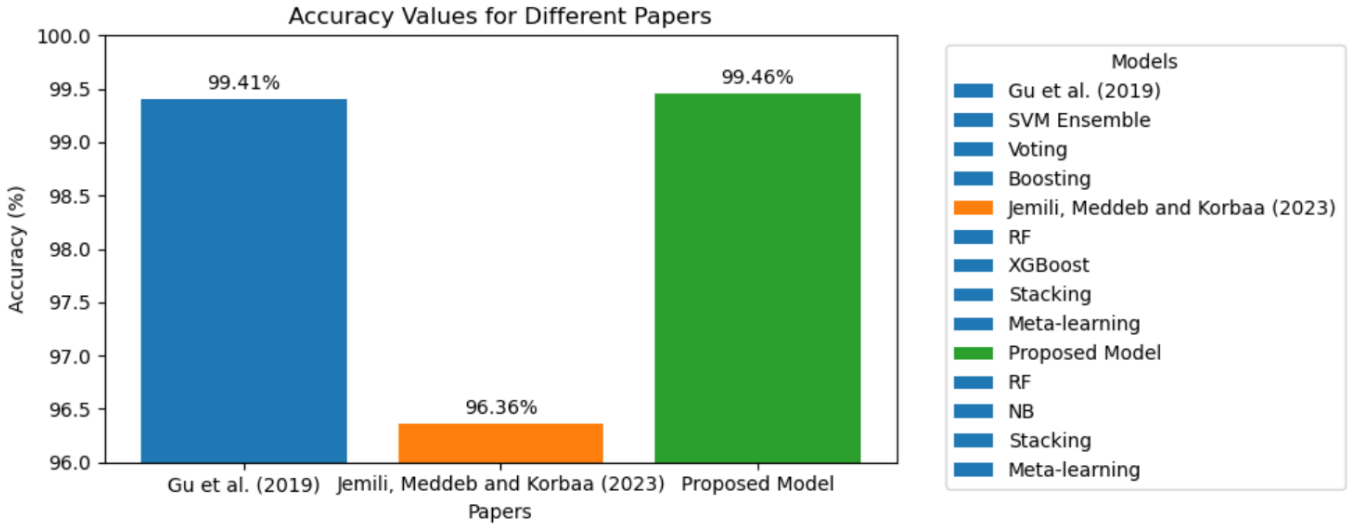


Figure **6. 2**: Existing Literature Comparison with the proposed ensemble mode at 8 used features

Table **6. 2:** Existing Literature Comparison with the proposed ensemble mode at 8 used features

Author & Year	**Ensemble Model**	**Ensemble Method**	**Fusion Method**	**Dataset**	**Feature Selection Method**	**Number of Used Features**	**Maximum Achieved Accuracy**
Gu et al. (2019)	SVM Ensemble	Boosting	Voting	-	Feature Data Density-ratio Transformation (DT)	-	99.41%
Jemili, Meddeb and Korbaa (2023)	Random Forest, XGBoost	Stacking	Meta-learning	CICIDS 2017,	Feature Importance	12	96.36%
Proposed Ensemble Model	**(RF) + NB**	**Stacking**	**Meta-learning**	**CICIDS 2017**	**Mutual Information-based**	**8**	**99.46%**

The only glaring limitation of this work from the results gathered is the response of the ensemble base models and the ensemble models built on them which have the same negative or low performance with class imbalanced datasets. This imply that ensemble learning may not be able to correct this or improve performance on this.

Finally, it is believed that this work has been able to successfully contribute to the knowledge domain of network intrusion detection using ensemble learning.

6.3 Future works

The scope of this research is limited to evaluation with only one dataset, CICIDS 2017 due to time constraints. Further study could be done to better validate this work with more public dataset like the NSL-KDD, etc and even real-time network traffic datasets.

Also, only two feature selections method were explored in this study. This work could be extended by evaluating with more other techniques like the RF-RFE, RFFI, etc., which are Random Forest Model embedded-based.

This work could be further extended with transfer learning technique, online learning, reinforcement learning and even deep learning and multiclassification . Finally integrating the performance model with an Intrusion Detection System (IDS) like Suricata could provide a promising path for future research. This integration promises to provide new opportunities for enhancing network security and intrusion detection capabilities.

References

ABBAS, Q. et al., 2023. Optimization of predictive performance of intrusion detection system using hybrid ensemble model for secure systems. *PeerJ*, 9(9), pp. e1552–e1552.

ADMKIE, K. and TEKLE, K., 2020. Efficient Data Mining Algorithm Network Intrusion Detection System for Masked Feature Intrusions. In: *6th African Conference on Information Systems and Technology (ACIST)*. Ethiopia: DIGITACOMMONS at Kennesaw State University. p. 11.

ALBULAYHI, K. et al., 2022. IoT Intrusion Detection Using Machine Learning with a Novel High Performing Feature Selection Method. *Applied Sciences*, 12(10), p. 5015.

ALDUAILIJ, M. et al., 2022. Machine-Learning-Based DDoS Attack Detection Using Mutual Information and Random Forest Feature Importance Method. *Symmetry*, 14(6), p. 1095.

ALGHUSHAIRY, O. et al., 2024. An Efficient Support Vector Machine Algorithm based Network Outlier Detection System. *IEEE Access*, 12, pp. 1–1.

ARASS, M., 2019. Smart SIEM: From Big Data logs and events to Smart Data alerts. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, 8(8), pp. 2278–3075.

AV-TEST, 2024. *Malware Statistics & Trends Report | AV-TEST*. [online]. Av-test.org. Available from: https://www.av-test.org/en/statistics/malware/ [Accessed 3 Apr 2024].

BELOUCH, M., EL HADAJ, S. and IDHAMMAD, M., 2018. Performance evaluation of intrusion detection based on machine learning using Apache Spark. *Procedia Computer Science*, 127(2), pp. 1–6.

BLAGUS, R. and LUSA, L., 2013. SMOTE for high-dimensional class-imbalanced data. *BMC Bioinformatics*, 14(1).

BONG, K. and KIM, J., 2022. Analysis of Intrusion Detection Performance by Smoothing Factor of Gaussian NB Model Using Modified NSL-KDD Dataset. In: *13th International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE ACCESS. pp. 1471–1476.

BRIDGES, R.A. et al., 2020. A Survey of Intrusion Detection Systems Leveraging Host Data. *ACM Computing Surveys*, 52(6), pp. 1–35.

CARLEY, K.M., 2020. Social cybersecurity: an emerging science. *Computational and Mathematical Organization Theory*, 26(4), pp. 365–381.

CHAND, N. et al., 2016. *A comparative analysis of SVM and its stacking with other classification algorithm for intrusion detection*. IEEE Xplore.

CHEN, A. et al., 2022. An efficient network behaviour anomaly detection using a hybrid DBN-LSTM network. *Computers & Security*, 114(114), p. 102600.

DIVINA, F. et al., 2018. Stacking Ensemble Learning for Short-Term Electricity Consumption Forecasting. Energies, 11(4), p. 949.

ENISA, 2023. Artificial Intelligence and Cybersecurity Research: ENISA Research and Innovation Brief, June 2023, pp. 32

EUROPEAN COMMISSION, 2021. Ethics guidelines for trustworthy AI | Shaping Europe's digital future. [online]. European Commission. Available from: https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai [Accessed 22 Apr 2024].

GADAL, S. et al., 2022. Machine Learning-Based Anomaly Detection Using K-Mean Array and Sequential Minimal Optimization. *Electronics*, 11(14), p. 2158.

GANDHI, K. et al., 2023. Ensemble Machine Learning-Based Network Intrusion Detection System. Smart Innovation, Systems and Technologies, 370, pp. 135–144.

GHARIB, A. et al., 2016. An Evaluation Framework for Intrusion Detection Dataset. *2016 International Conference on Information Science and Security (ICISS)*.

GU, J. et al., 2019. A novel approach to intrusion detection using SVM ensemble with feature augmentation. *Computers & Security*, 86, pp. 53–62.

GUEZZAZ, A. et al., 2021. A Reliable Network Intrusion Detection Approach Using Decision Tree with Enhanced Data Quality. *Security and Communication Networks*, 2021, pp. 1–8.

GUPTA, N., JINDAL, V. and BEDI, P., 2021. CSE-IDS: Using Cost-Sensitive Deep Learning and Ensemble algorithms to handle class imbalance in Network-based Intrusion Detection Systems. *Computers & Security*, p. 102499.

HANSEN, L.K. and SALAMON, P., 1990. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10), pp. 993–1001.

HOSPEDALES, T.M. et al., 2021. Meta-Learning in Neural Networks: A Survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(9), pp. 1–1.

HUSSAIN IQBAL, M. and RAHIM SOOMRO, T., 2015. Big Data Analysis: Apache Storm Perspective. *International Journal of Computer Trends and Technology*, 19(1), pp. 9–14.

HUSSEIN, S. et al., 2019. Lung and Pancreatic Tumor Characterization in the Deep Learning Era: Novel Supervised and Unsupervised Learning Approaches. *IEEE Transactions on Medical Imaging*, 38(8), pp. 1777–1787

IEEE, 2019. ETHICALLY ALIGNED DESIGN A Vision for Prioritizing Human Well-being with Autonomous and Intelligent Systems First Edition. Available from: https://standards.ieee.org/wpcontent/uploads/import/documents/other/ead1e.pdf [Accessed 16 Apr 2024].

ILLY, P. et al., 2019. Securing Fog-to-Things Environment Using Intrusion Detection System Based On Ensemble Learning. *2019 IEEE Wireless Communications and Networking Conference (WCNC)*.

JEMILI, F., MEDDEB, R. and KORBAA, O. 2023. Intrusion detection based on ensemble learning for big data classification. *Cluster Computing*.

JEONG, D.-H. et al., 2022. A Comparative Study on the Influence of Undersampling and Oversampling Techniques for the Classification of Physical Activities Using an Imbalanced Accelerometer Dataset. *Healthcare*, 10(7), p. 1255.

JIE , G. and SHAN , L., 2021. An effective intrusion detection approach using SVM with naïve Bayes feature embedding. *Computers & Security*, 103(4048), p. 102158.

JUNIPER RESEARCH, 2023. *Online Payment Fraud: Market Forecasts, Emerging Threats & Segment Analysis 2023-2028*. [online]. Hampshire, UK: Juniper Research Ltd. Available from: https://www.juniperresearch.com/research/fintech-payments/fraud-identity/online-payment-fraud-research-report [Accessed 7 Feb 2024].

KASONGO, S.M. and SUN, Y., 2020. Performance Analysis of Intrusion Detection Systems Using a Feature Selection Method on the UNSW-NB15 Dataset. *Journal of Big Data*, 7(1).

KRISHNAVENI, S. et al., 2021. Efficient feature selection and classification through ensemble method for network intrusion detection on cloud computing. *Cluster Computing*, 24, pp. 1761–1779.

KUMAR, S., GUPTA, S. and ARORA, S., 2021. A comparative simulation of normalization methods for machine learning-based intrusion detection systems using KDD Cup'99 dataset. *Journal of Intelligent & Fuzzy Systems*, pp. 1–18.

KURUVAYIL, S. and PALANISWAMY, S., 2021. Emotion recognition from facial images with simultaneous occlusion, pose and illumination variations using meta-learning. Journal of King Saud University - Computer and Information Sciences, 34(9), pp. 7271–7282.

LIU, H. and LANG, B., 2019. Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey. *Applied Sciences*, 9(20), p. 4396.

MA, Z. et al., 2018. Ensemble of machine learning algorithms using the stacked generalization approach to estimate the warfarin dose. Edited by Maciej Huk. PLOS ONE, 13(10), p. e0205872.

MOHAMMED, A. and KORA, R., 2023. A Comprehensive Review on Ensemble Deep Learning: Opportunities and Challenges. *Journal of King Saud University - Computer and Information Sciences*, 35(2).

OBAID, H., DHEYAB, S. and SABRY, S., 2019. The Impact of Data Pre-Processing Techniques and Dimensionality Reduction on the Accuracy of Machine Learning. IEEE Xplore.

OMAR, M., 2023. Harnessing the Power of Decision Trees to Detect IoT Malware. *ArXiv (Cornell University)*.

OZA, N.C. and TUMER, K., 2008. Classifier ensembles: Select real-world applications. *Information Fusion*, 9(1), pp. 4–20.

PANIGRAHI, R. and BORAH, S., 2018. A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems. *International Journal of Engineering & Technology*, 7(3), pp. 479–482.

PHAM, N.T. et al., 2018. Improving performance of intrusion detection system using ensemble methods and feature selection. *Proceedings of the Australasian Computer Science Week Multiconference*, 18(2), pp. 1–6.

PRASAD, M., TRIPATHI, S. and DAHAL, K., 2022. A probability estimation-based feature reduction and Bayesian rough set approach for intrusion detection in mobile ad-hoc network. *Applied Intelligence*, 53, pp. 7169–7185.

PRODUCT PLAN, 2024. What is MoSCoW Prioritization? | Overview of the MoSCoW Method. [online]. Available from: https://www.productplan.com/glossary/moscow-prioritization/ [Accessed 17 April 2024].

RAJADURAI, H. and GANDHI, U.D., 2020. A stacked ensemble learning model for intrusion detection in wireless network. *Neural Computing and Applications*, 34(5).

RAJAGOPAL, S., KUNDAPUR, P.P. and HAREESHA, K.S., 2020. A Stacking Ensemble for Network Intrusion Detection Using Heterogeneous Datasets. *Security and Communication Networks*, 2020(4586875), pp. 1–9.

RASHID , M. et al., 2021. Performance Enhancement of Intrusion Detection System Using Bagging Ensemble Technique with Feature Selection. In: *2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*. Gold Coast, Australia, 28 April 2021. IEEE.

RASHID, M. et al., 2022. A tree-based stacking ensemble technique with feature selection for network intrusion detection. *Applied Intelligence*, 51(2022), pp. 9768–9781.

SAEED, M.M., 2022. A real-time adaptive network intrusion detection for streaming data: a hybrid approach. *Neural Computing and Applications*, 38(8), pp. 1–14.

SAHU, A. et al., 2020. *Data Processing and Model Selection for Machine Learning-based Network Intrusion Detection*. IEEE Xplore.

SALEH, H. et al., 2022. Heterogeneous Ensemble Deep Learning Model for Enhanced Arabic Sentiment Analysis. Sensors, 22(10), p. 3707.

SANTOS, K., MIANI, R. and SILVA, F., 2024. Evaluating the Impact of Data Preprocessing Techniques on the Performance of Intrusion Detection Systems. *Journal of Network and Systems Management*, 32(2).

SARKER, I.H. et al., 2020. IntruDTree: A Machine Learning Based Cyber Security Intrusion Detection Model. *Symmetry*, 12(5), p. 754.

SAXENA, A.Ku., SINHA, S. and SHUKLA, P., 2017. *General study of intrusion detection system and survey of agent-based intrusion detection system*. IEEE Xplore.

SHARAFALDIN, I., LASHKARI, A. and GHORBANI, A., 2018. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. *Proceedings of the 4th International Conference on Information Systems Security and Privacy*.

SINGH, P. and RANGA, V., 2021. Attack and intrusion detection in cloud computing using an ensemble learning approach. *International Journal of Information Technology*, 13(2), pp. 565–571.

TAMA, B.A., COMUZZI, M. and RHEE, K.-H., 2019. TSE-IDS: A Two-Stage Classifier Ensemble for Intelligent Anomaly-Based Intrusion Detection System. *IEEE Access*, 7(7), pp. 94497–94507.

UMAR , M. et al., 2024. Effects of Feature Selection and Normalization on Network Intrusion Detection. Elsevier, 3.

VIBHUTE, A.D. et al., 2024. Towards Detection of Network Anomalies using Machine Learning Algorithms on the NSL-KDD Benchmark Datasets. *Procedia Computer Science*, 233, pp. 960–969.

VIJAY, R. et al., 2021. AUGMENTING NETWORK INTRUSION DETECTION SYSTEM USING EXTREME GRADIENT BOOSTING(XGBOOST). *IJCRT - International Journal of Creative Research Thoughts (IJCRT)*, 9(6).

VOLKERDON, 2024. MoSCoW prioritization technique | Volkerdon. [online]. Available from: https://www.volkerdon.com/pages/moscow-prioritisation [Accessed 17 April 2024].

WANG, W. et al., 2009. *Attribute Normalization in Network Intrusion Detection*. IEEE Xplore.

WATTHAISONG, T. et al., 2024. Comparative Evaluation of Imbalanced Data Management Techniques for Solving Classification Problems on Imbalanced Datasets. Statistics, Optimization & Information Computing, 12(2), pp. 547–570.

XIONG, Y., YE, M. and WU, C.-R., 2021. Cancer Classification with a Cost-Sensitive Naive Bayes Stacking Ensemble. Computational and Mathematical Methods in Medicine, 2021, pp. 1–12.

XU, W. et al., 2021. Improving Performance of Autoencoder-based Network Anomaly Detection on NSL-KDD dataset. *IEEE Access*, 9(9), pp. 140136–140146.

ZHANG, X. et al., 2022. RANet: Network intrusion detection with group-gating convolutional neural network. *Journal of Network and Computer Applications*, 198(2), p. 103266.

APPENDICES

Appendix A : Project Plan

Task No.	Task Description	Duration	Start Date	End Date
1	• Developing research scope, aim and objectives of anomaly-based Network Intrusion Detection using Ensemble machine learning. • Create a detailed project plan with timelines and milestones. • Deciding on coding and simulation tools and platform for the research experiments (Python or/and R)	8 days	02/02/2024	09/02/2024
2	• Critical survey of reviewed literatures on the chosen project title with likely research gaps in mind. • Deeper understanding and analysis of the selected publicly available Network Intrusion Detection-based Dataset (CICIDS 2017).	7 days	10/02/2024	16/02/2024
3	• Researching and analysis of literature on different types of dataset class balancing techniques as related to the project scope. • Python libraries and code implementation of the same.	7 days	17/02/2024	23/02/2024
4.	• Researching and analysis of literature on different types of dataset Normalization techniques as related to the project scope. • Python libraries and code implementation of the same.	7 days	24/02/2024	01/03/2024
5	• Researching and analysis of literature on different types of feature selection techniques as related to the project scope. • Python libraries and implementation of the same.	7 days	02/03/2024	08/03/2024
6	• Researching and analysis of literature on different types of ensemble learning methods, ensemble fusion techniques, automated hyperparameter search and tuning methods plus cross-validation as related to the project scope. • Python libraries plus code implementations of the same.	7 days	09/03/2024	15/03/2024

Task No. (Continued)	Task Description	Duration	Start Date	End Date
7	• Simple learning modelling for comparative analysis of the effect of class imbalance on ensemble baseline model performance (NB, RF and XGBoost). • Simple learning modelling for comparative analysis of the effect of choice of different normalization techniques on ensemble baseline model performance (NB, RF and XGBoost). • Simple learning modelling across a range of number of selected features and comparative analysis of the effect of choice of different feature selection methods on ensemble baseline model performance (NB, RF and XGBoost). • Evaluation and analysis for best performing ensemble baseline models. • Building of nine (9) different stacking ensemble models across range of number of selected features. • Evaluation and Analysis of stacking ensemble models for balanced and unbalanced training dataset scenarios using performance evaluation metrics (Confusion Matrix, Precision, Recall, F1-score, FAR and ROC-AUC curves). • Evaluation and Analysis of best performing stacking ensemble models for training and testing times.	21 days	16/03/2024	05/04/2024
8	• Report writing and presentation of project research findings, recommendation and future works.	16 days	06/04/2024	21/04/2024
9	• Poster design, report fine-tuning and submissions.	4 days	22/04/2024	25/04/2024

Appendix B : Evaluation Data Extras

- Table **5.1** : Impact of Choice of Normalization Technique on Ensemble Base Model Performance.

	Z-Score Normalized Balanced Dataset, X5	**Min-Max Normalized Balanced Dataset, X7**
Base Models	**Accuracy**	
RF	**0.99972**	0.99971
XGBoost	**0.99898**	0.99898
NB	**0.94623**	0.94624

- Table **5.2:** Impact of Normalization on Performance of Ensemble Baseline Model

	Z-Score Normalized Balanced Dataset, X5 with Bagged Undersampling Technique			**Un-Normalized Balanced Dataset, X8** with Bagged Undersampling Technique		
Base Model	**Accuracy**	**Training Time (s)**	**Testing Time (s)**	**Accuracy**	**Training Time (s)**	**Testing Time (s)**
RF	**0.99972**	3667.828	2.107	**0.99969**	3753.611	2.073
XGBoost	**0.99898**	147.653	0.161	**0.99898**	136.168	0.179
NB	**0.94623**	3.365	0.052	**0.60347**	4.034	0.051

- Table **5.3**: Impact of Class Imbalance on Ensemble Baseline Model Performance:

Base Models	**Unbalanced Dataset, X4**	**Balanced Dataset, X5**
RF	0.9937	**0.99972**
XGBoost	0.9932	**0.99898**
NB	0.9322	**0.94623**

- Table **5.4** : Base Model Performance Comparison on Feature Selection Methods with Balanced Dataset

Model	**RF**		**XGBoost**		**NB**	
Feature Selection method	**MI-based**	**SelectKBest with ANOVA F-value Scoring Function**	**MI-based**	**SelectKBest with ANOVA F-value Scoring Function**	**MI-based**	**SelectKBest with ANOVA F-value Scoring Function**
No. of Selected Features	**Accuracy**		**Accuracy**		**Accuracy**	
79	**0.99971**		**0.99989**		**0.95792**	
14	**0.99986**	0.96772	**0.99955**	0.98831	**0.95499**	0.78903
12	**0.99972**	0.9677	**0.99898**	0.98769	**0.94623**	0.79074
10	**0.99487**	0.94033	**0.9945**	0.97116	**0.9457**	0.79403
8	***0.99488**	0.9374	**0.9945**	0.96604	**0.94601**	0.79078

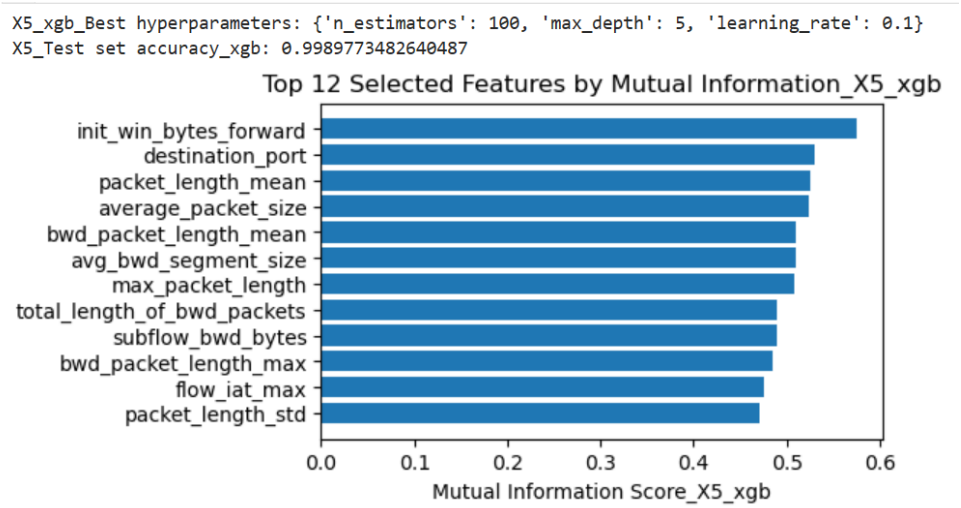


Figure **5.1:** Mutual Information-based selection method for 12 features

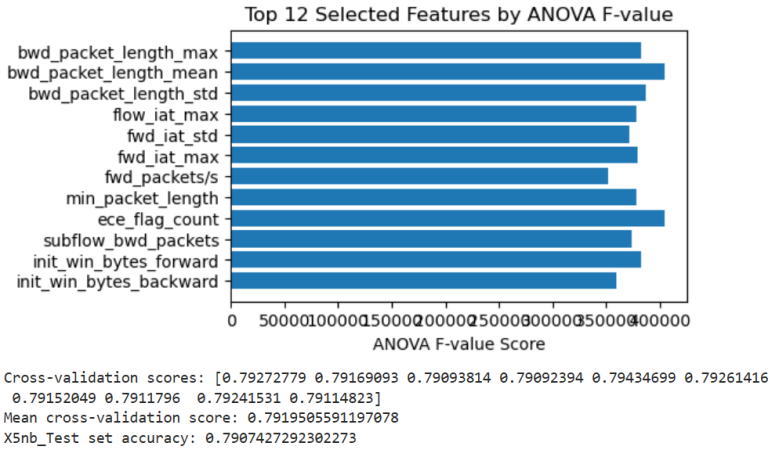


Figure **5.2:** SelectKBest with ANOVA-based selection method for 12 features

Table **5.5:** Performance Accuracies of Nine (9) Built Stacking Ensemble Models for Analysis

Model	Number of Selected Features	**Z-Score Normalized Balanced Dataset, X5** with Bagged Undersampling Technique		
		Accuracy	**Training Time (s)**	**Testing Time (s)**
RF	12	**0.9997**	3667.828	2.107
XGBoost		**0.9989**	141.536	0.169
NB		**0.9462**	0.238	0.0625
NB + RF		0.9996	1955.55	4.465
NB + XGBoost		0.9987	107.567	0.181
RF + XGBoost*		**0.9999**	**4286.164**	**0.359**
XGBoost + RF		0.9996	2358.0839	5.00883
NB + RF + XGBoost*		**0.9999**	**3812.445**	**0.201**
RF + XGBoost + NB *		**0.9998**	**4631.756**	**0.097**
RF + NB*		**0.9998**	**4094.055**	**0.085**
XGBoost + NB		0.9989	166.859	0.098
NB + XGBoost + RF		0.9997	1895.448	2.16

Table **5.6:** Performance Accuracies of Selected Four Ensemble in Unbalanced Dataset

S/N	Model	Number of Selected Features	Z-Score Normalized Un-Balanced Dataset, X4		
			Accuracy	Training Time (s)	Testing Time (s)
*	**RF**	12	**0.9937**		
**	**XGBoost**		**0.9932**		
***	**NB**		**0.9322**		
3*	**RF + XGBoost**		**0.9935**	2786.035	0.163
5*	**NB + RF + XGBoost**		**0.9935**	3214.168	0.149
6*	**RF + XGBoost + NB**		**0.9933**	2973.939	0.0832
7*	**RF + NB**		**0.9934**	3188.356	0.069

Table **5.7** : Training and Testing Times for Four Best Performing Ensemble Models

Number Features	**12**	**10**	**8**
Ensemble Model	**Training Time (sec)**		
RF + XGBoost*	3922.806	3464.508	2451.559
NB + RF + XGBoost*	3834.487	3067.408	2440.061
RF + XGBoost + NB *	4631.756	3129.649	2831.768
RF + NB*	4094.055	3315.963	2345.041

Number Features	**12**	**10**	**8**
Ensemble Model	**Testing Time (sec)**		
RF + XGBoost*	0.201	0.211	0.201
NB + RF + XGBoost*	0.179	0.201	0.191
RF + XGBoost + NB *	0.097	0.069	0.083
RF + NB*	0.085	0.081	0.069

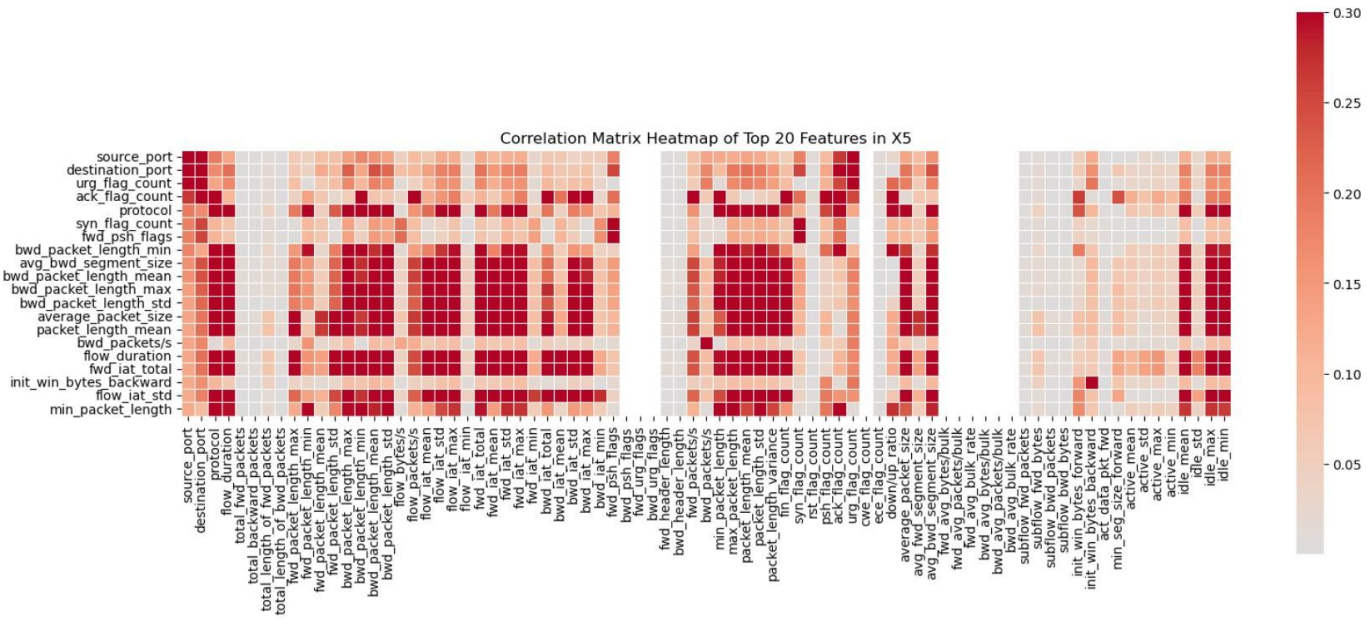


Figure **5.3**: Correlation Matrix Heat Map for the Features in the CICIDs Dataset showing top 20

Table **5.8:** Time Serialization of the Two Best Performing Ensemble at 12 features- Breaking the Tie

	Z-Score Normalized Balanced Dataset, X5 with Bagged Undersampling Technique			**Time Serialization for Best Ensemble Model**		
Model	**Accuracy**	**Training Time (s)**	**Testing Time (s)**	**XGB_TRT (s)**	**RF_TRT (s)**	**NB_TRT (s)**
RF + XGBoost*	**0.9999**	**3922.806**	**0.201**	**96.959**	**3817.501**	**Nill**
NB + RF + XGBoost*	**0.9999**	**3834.487**	**0.179**	**88.467**	**3737.226**	**0.233**

Appendix C : Additional Submissions

The coding files and all other collected data have been uploaded to the campus Moodle, and they are accessible within the additional materials section. The upload includes the following:

1. Python and R Coding files: **2213400_Data and Python Codes (Zipped file)**

2. All of the gathered data and corresponding graphs, plots are also provided for reference**: 2213400_Evaluation Plot_Ensembles (Zipped file)**