iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 12 **Issue:** VII **Month of publication:** July 2024

DOI: https://doi.org/10.22214/ijraset.2024.63548

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue VII July 2024- Available at www.ijraset.com

Network Intrusion Detection and Classification System: A Supervised Machine Learning Approach

Kayode Akinlekan Akintoye[1], Michael Kolade Adu[2], Sunday Akinwamide[3]
[1, 2, 3]*Department of Computer Science, The Federal Polytechnic, Ado Ekiti, Nigeria*

Abstract: ***Intrusion detection systems (IDSs) are crucial for computer security, as they identify and counteract malicious activities within computer networks. Anomaly-based IDSs, specifically, use classification models trained on historical data to detect these harmful activities. This paper proposes an enhanced IDS based on 3-level training and testing of machine learning models, feature selection, resampling, and normalization using Decision Tree, Gaussian Naïve Bayes, K-Nearest Neighbours, Logistic Regression, Random Forest, and Support Vector Machine. In the first stage, the six models are trained and evaluated using the original datasets after pre-processing. In the second stage, the models are built and tested with a resampled version of the dataset using the Synthetic Minority Oversampling Technique (SMOTE). In the third stage, the models are trained and tested with a dataset that has been both resampled and normalized using the standard scaling method. We employ the feature importance technique using the random forest model to select the essential features from NSL-KDD and UNSW-NB15 datasets. The results of our study surpass previous related research, with the decision tree achieving an accuracy, precision, recall, and F1 score of 99.99% on the UNSW-NB15 dataset. Additionally, the decision tree recorded an accuracy of 99.98%, precision of 99.97%, recall of 99.97%, and F1 score of 99.99% on the NSL-KDD dataset.***
Keyword: Intrusion detection, Data analysis, UNSW-NB15 dataset, NSL-KDD dataset, Machine learning

I. INTRODUCTION

Over the past decade, the surge in interest and advancements in internet and communication technologies has made network security a critical research area. It employs tools such as firewalls, antivirus software, and intrusion detection systems (IDS) to protect the network and its associated assets in cyberspace. Among these tools, network-based intrusion detection systems (NIDS) are particularly vital for maintaining security by continuously monitoring network traffic for malicious and suspicious activities (Teresa, 1993). The concept of IDS was first introduced by Jim Anderson in 1980. Since then, numerous IDS products have been developed and refined to meet the evolving demands of network security (Herve et al., 1993). Significant technological advancements in recent years have led to a substantial increase in network size and the number of applications managed by network nodes. Consequently, a vast amount of critical data is generated and exchanged across various network nodes. Ensuring the security of this data and the nodes has become increasingly challenging due to the proliferation of new attacks, whether through the mutation of existing attacks or the emergence of novel ones. Nearly every node within a network is susceptible to security threats. For instance, a data node can be crucial for an organization, and any compromise of its information could severely impact the organization's market reputation and financial stability. Current IDSs have proven inadequate in detecting a variety of attacks, including zero-day attacks, and in reducing false alarm rates (FAR) (Hoque et al., 2012). This highlights the need for an efficient, accurate, and cost-effective NIDS to provide robust network security. To develop effective intrusion detection systems (IDS), researchers have explored the potential of machine learning (ML) and deep learning (DL) techniques. These techniques, which fall under the broader category of artificial intelligence (AI), focus on extracting valuable information from large datasets (Ramjee et al., 2020). Anomaly detection is crucial for identifying and preventing security attacks, as network traffic anomalies can indicate potential intrusions. Traditionally, early research and most commercially available IDS have predominantly relied on signature-based methods. However, the main challenge with these methods is the need for continuous updates to the signature database with new attack signatures, making them unsuitable for real-time network anomaly detection. Consequently, there has been a shift towards employing machine learning classification techniques for anomaly detection (Zaman & Lung, 2018a). In the past decade, ML and DL have gained popularity in network security, largely due to the development of powerful graphics processing units (GPUs) (Lew et al., 2019). Both ML and DL excel at identifying useful features from network traffic and distinguishing between normal and abnormal activities based on learned patterns. ML-based IDS heavily relies on feature engineering to extract relevant information from network traffic (Najafabadi et al., 2015), whereas DL-based IDS can automatically learn complex features from raw data without the need for feature engineering, thanks to their deep architectures (Dong & Wang, 2016).

In the past decade, researchers have proposed numerous ML- and DL-based solutions to enhance the efficiency of NIDS in detecting malicious attacks. However, the significant growth in network traffic and the accompanying security threats have created substantial challenges for NIDS in effectively identifying malicious intrusions. Research on applying DL methods to NIDS is expanding, and there remains considerable potential to further explore this technology to improve intrusion detection within networks.

In this research, we demonstrate that addressing class imbalance and applying data normalization, along with feature selection, can significantly enhance the accuracy of anomaly-based IDS. The reasoning for using these techniques is based on the observed success of similar architecture models in related research, which have demonstrated exceptional performance metrics for ML models.

Our contributions to the cyber-security domain are as follows:

1) We use the classic NSL-KDD and UNSW-NB15 as benchmark datasets and conduct detailed analysis and data cleaning
2) We employ the feature importance technique for feature selection to achieve a precise and accurate representation for the IDS problem. This approach acknowledges that not all features are equally significant or relevant in detecting intrusions.
3) The issue of data imbalance and data normalization were addressed to achieve better performance, generalization, interpretability, and computational efficiency, leading to more reliable and accurate predictions.

The remainder of the paper is organized as follows: Section 2 reviews previous related research. Section 3 provides a detailed methodology of the proposed anomaly-based IDS framework. Section 4 discusses the experimental results. Finally, conclusions are drawn in Section 5.

II. REVIEW OF RELATED WORK

Anomaly-based intrusion detection systems (IDS) were first introduced by Anderson in 1980. (Javaid et al., 2016). Since then, anomaly detection has been extensively covered in numerous surveys and review articles.

Researchers have employed various machine learning algorithms and utilized different publicly available datasets in their studies to improve detection outcomes. (Anjum & Chowdhury, 2024) proposed a model for developing a network intrusion detection system that classifies test data as either malicious or benign using a decision tree algorithm with the Change Control Identifiers (CCIDS) 2017 dataset. They utilized a label encoder to convert categorical features and applied Recursive Feature Elimination (RFE) for feature selection. The model achieved 99% accuracy, 99.9% precision, and a false positive rate (FPR) of 0.1%. In another development, (Mahfouz et al., 2020) introduced an ensemble and adaptive classifier model to improve accuracy and reduce the false alarm rate in intrusion detection systems (IDSs) by using the Game Theory and Cyber Security (GTCS) dataset to identify network intrusions. The dataset was normalized and encoded to transform categorical data into a format suitable for machine learning algorithms. Feature selection was conducted using the InfoGainAttributeEval algorithm. Data imbalance issues were addressed, and the experiments were performed using WEKA. Six machine learning classifiers were employed in the study: Naive Bayes (NB), logistic regression, multilayer perceptron (neural network), SMO (SVM), IBK (k-nearest neighbor), and J48 (decision tree). Additionally, a seventh classifier, which is a voting ensemble combining multilayer perceptron, IBK, and J48, was used. The experimental results were evaluated based on accuracy, true positive rate (TPR), false positive rate (FPR), precision, recall, F-measure, and receiver operating characteristic (ROC) area. The voting ensemble method demonstrated the highest performance, achieving an accuracy of 98.99%. (Khammassi & Krichen, 2017) proposed a model for developing a network intrusion detection system using the KDD99 and UNSW-NB15 datasets. They pre-processed the datasets and extracted important features using the GA-LR wrapper method. Classification was performed using three decision tree classifiers: C4.5, Random Forest, and Naïve Bayes Tree. For the KDD99 dataset, the optimal subset consisted of 18 features, achieving an accuracy of 99.90%, a detection rate (DR) of 99.81%, and a false alarm rate (FAR) of 0.105% with the Random Forest classifier. For the UNSW-NB15 dataset, the optimal subset consisted of 20 features, achieving an accuracy of 81.42% and a FAR of 6.39% with the C4.5 classifier. The author of (Çavuşoğlu, 2019) developed a hybrid and layered Intrusion Detection System (IDS) utilizing a combination of machine learning and feature selection techniques to achieve high performance in detecting various attack types using the NSL-KDD dataset. The process began with pre-processing operations, including transformation and normalization of the dataset. Feature selection was then performed using CfsSubsetEval and WrapperSubsetEval algorithms to identify the most important features. The machine learning techniques employed were: Random Forest (RF) algorithm for DOS and Probe attacks, a stacking method combining RF, J48, and Naive Bayes algorithms for R2L attacks, and J48 and Naive Bayes algorithms for detecting normal-traffic in U2R attacks. The experiments were conducted using Weka.

The proposed system demonstrated an average accuracy of 99.86% across all attack types. (Yang et al., 2019) proposed an intrusion detection model that combines an improved conditional variational AutoEncoder (ICVAE) based on specific intrusion categories with a deep neural network (DNN), referred to as ICVAE-DNN, using the NSL-KDD and UNSW-NB15 datasets. The process begins with data pre-processing, feature mapping, and normalization, followed by training the ICVAE. The DNN is then used to generate and detect new attacks. The classification performance of ICVAE-DNN was evaluated, achieving an accuracy of 85.97% on NSL-KDD (KDDTest+), 75.43% on NSL-KDD (KDDTest-21), and 89.05% on UNSW-NB15. The work of (Gao et al., 2019) introduced a MultiTree algorithm and an adaptive ensemble learning model using the NSL-KDD dataset. The dataset underwent pre-processing through label and one-hot encoding, followed by standardization. Feature selection was performed using PCA. The MultiTree algorithm achieved an accuracy of 84.2%, while the adaptive voting algorithm achieved a final accuracy of 85.2%. (Fang et al., 2020) proposed a machine learning approach for detecting intrusion information using an Elman neural network and SVM for noise data elimination, utilizing the KDD Cup'99 dataset. The system is composed of four modules: machine learning, network packet capture, misuse rule processing, and data pre-processing. Results indicated that the neighbour classification algorithm based on robust SVM achieved a detection rate of 87.3% with a false alarm rate of 0. For data containing noise, the robust SVM-based neighbour classification algorithm achieved a detection rate of 70.9% with a false alarm rate of 0. In a related development, (Zaman & Lung, 2018b) introduced six commonly used machine learning techniques (K-Means, K-Nearest Neighbours – KNN, Fuzzy C-Means – FCM, Support Vector Machine – SVM, Naïve-Bayes – NB, Radial Basis Function – RBF, as well as a Voting Ensemble method combining these six algorithms for detecting network traffic anomalies using the Kyoto 2006+ dataset. The RBF classification technique demonstrated the best performance metrics, achieving 97.54% accuracy, a precision of 0.92, a recall of 0.9583, and an ROC value of 0.9741. The research of (Liu et al., 2021) investigated the application of machine learning and deep learning for detecting intrusions in imbalanced network traffic. It introduced a novel algorithm called Difficult Set Sampling Technique (DSSTE) aimed at addressing the class imbalance issue in NSL-KDD and CSE-CIC-IDS2018 datasets. The experimental results demonstrated that the DSSTE approach outperformed other sampling methods. Specifically, AlexNet achieved the highest performance on the NSL-KDD dataset with an accuracy of 0.8284, precision of 0.8394, recall of 0.8278, and F1 score of 0.8166. On the other hand, Mini-VGGNet achieved the best results on the CSE-CIC-IDS2018 dataset with an accuracy of 0.9699, precision of 0.9746, recall of 0.9697, and F1 score of 0.9704. (Tama et al., 2019) introduced an enhanced Intrusion Detection System (IDS) utilizing a hybrid feature selection technique and ensembles of two-level classifiers. The study evaluates the proposed system on several datasets: KDDTrainC (used for training), KDDTestC and KDDTest-21 (as separate test sets), along with UNSW-NB15$_{train}$ and UNSW-NB15$_{test}$. The classifiers were implemented using the RWeka library. Comparative analysis with existing methods demonstrated superior performance of the proposed system: KDDTestC achieves an accuracy of 85.797%, false positive rate (FPR) of 11.7%, recall of 86.8%, and precision of 88.0%. KDDTest-21 shows an accuracy of 72.52%, FPR of 18.00%, recall of 72.00%, and precision of 85.00%. UNSW-NB15test exhibits an accuracy of 91.27%, FPR of 8.90%, recall of 91.30%, and precision of 91.60%. The authors of (Binghao & Guodong, 2018) developed an Integrated Intrusion Detection Model (IDM) termed LA-GRU, combining imbalanced learning techniques with a Gated Recurrent Unit (GRU) neural network. They enhanced the model with a modified version of the Local Adaptive Synthetic Minority Oversampling Technique (LA-SMOTE) to address imbalanced traffic, followed by applying the GRU neural network based on deep learning principles for anomaly detection using the NSL-KDD dataset. Experimental results demonstrated that the LA-GRU model excelled not only in detecting high-frequency attacks such as DOS (99.16%) and Probe (99.20%), but also achieved significant performance in detecting low-frequency attacks. Specifically, detection rates of 98.34% for R2L and 98.61% for U2R were achieved, surpassing all other IDMs and establishing new benchmarks in experimental outcomes. Its best overall performance are: 99.04% accuracy, 98.92% detection rate and false alarm rate of 0.134%. (Sarker et al., 2020) introduced the IntruDTree, a machine-learning-based security model designed for intrusion detection. This model initially assessed and ranked the security features of the NSL-KDD dataset based on their importance. It then constructed a generalized tree-based intrusion detection model utilizing these key features. When comparing the results of this experiment to other studies that employed popular machine learning models, it is clear that IntruDTree outperformed previous efforts. The model achieved an accuracy, precision, recall, and F1 score of 98.00%. The work of (Verma & Ranga, 2020) explored the use of single and ensemble classifiers for protecting IoT systems against DoS attacks, utilizing the CIDDS-001, UNSWNB15, and NSL-KDD datasets to benchmark and evaluate the performance of machine learning classifiers for IDS in IoT. Hyperparameter tuning for all classifiers was conducted using RandomizedSearchCV. The machine learning models examined in this research included: Random Forest (RF), AdaBoost (AB), Gradient Boosted Machine (GBM), Extreme Gradient Boosting (XGB), Extremely Randomized Trees (ETC), Classification and Regression Trees (CART), and Multi-Layer Perceptron (MLP). These datasets were used to perform Hold Out and Cross Fold Validation tests on the classifiers.

While assessing the performance results from Hold Out Validation, it was observed that RF had the highest accuracy (94.94%) and specificity (91.6%). GBM excelled in sensitivity (99.53%), while XGB performed best in terms of the AUC metric, achieving 98.76%. It was noted that the performance of all classifiers improved with 10-Fold Cross Validation compared to their performance with Hold Out Validation. CART achieved the highest accuracy (96.74%), AB had the highest specificity (97.5%), and both RF and XGB showed the best sensitivity (97.31%). For the AUC metric, XGB was the top performer with a score of 98.77%. (Injadat et al., 2021) introduced a novel multi-stage, optimized ML-based NIDS framework designed to reduce computational and time complexity. It examined the impact of oversampling techniques, identified the minimal suitable sample size for training models, and compared two feature selection techniques: information gain and correlation-based methods. The study also investigated different ML hyper-parameter optimization techniques and assessed their effects while maintaining detection performance, using the CICIDS 2017 and UNSW-NB 2015 datasets. The classifiers used in the experiments were KNN and RF. Various hyper-parameter optimization methods were explored, including Random Search (RS), Particle Swarm Optimization (PSO), Genetic Algorithm (GA), and Bayesian Optimization algorithms (BO) with two types: Gaussian Process (GP) and Tree Parzen Estimator (TPE). The results demonstrated that the BO-TPE-RF method achieved the highest detection accuracy for both datasets and both feature selection algorithms, with detection accuracies of 99.99% and 100%, respectively. Lastly, (Agarwal et al., 2021) proposed an optimal algorithm capable of efficiently learning patterns of suspicious network activities in the shortest processing time to accurately classify attacks, using three machine learning classification algorithms—Naïve Bayes (NB), Support Vector Machine (SVM), and K-nearest Neighbour (KNN)—with the UNSW-NB15 dataset. The experimental results indicated that SVM achieved the highest performance accuracy of 97.78% following data pre-processing and model training.

However, there are some drawbacks noted in the reviewed studies:

1) *Utilization of only one ML model and Dataset:* Relying solely on a single machine learning model and dataset in intrusion detection systems is inadequate for achieving high accuracy in detecting harmful traffic and maintaining a low false positive rate across all types of attacks
2) *Using High-Dimensional Datasets Without Employing Feature Selection Techniques:* as a result of this situation, the processing times are prolonged and the desired performance may not be achieved
3) *False Positives and False Negatives:* NIDS can generate false alarms (false positives) or miss actual intrusions (false negatives), leading to inefficiencies and potential security risks.
4) *High Volume of Alerts:* NIDS often generates a large number of alerts, overwhelming security teams and making it challenging to prioritize and respond to real threats promptly.
5) *Complexity of Network Traffic:* Network traffic can be complex, especially in large and diverse networks, making it difficult for NIDS to accurately detect and classify malicious activities.
6) *Resource Intensive:* NIDS can be resource-intensive, requiring significant computational power and network bandwidth to analyse and monitor network traffic effectively.

To overcome several of these constraints, the proposed study implements and evaluates the effectiveness of six state-of-the-art ML models and utilized two prominent NIDS datasets for the network intrusion detection and classification system. This was achieved by tackling class imbalance, applying data normalization, and employing feature selection techniques to improve the accuracy of anomaly-based IDS.

III. METHODOLOGY

In this section, we begin by providing a conceptual overview of the proposed framework. Following that, we delve into the specifics of the techniques we have implemented, including label encoding, feature selection, one-hot encoding, data balancing/resampling, normalization, and the classification models used.

A. Conceptual Framework of IDS

The network intrusion detection datasets utilized in this research are NSL-KDD (training & testing) and UNSW-NB15 (training & testing). The pre-processing steps performed on these datasets include label encoding, one-hot encoding, and feature selection. The training and testing of the machine learning models were conducted in three distinct stages. In the first stage, the models were trained and tested using datasets that has been pre-processed and undergone one-hot encoding. In the second stage, the models were trained and tested with datasets that had been one-hot encoded and balanced using the Synthetic Minority Oversampling Technique (SMOTE) to address data imbalance. In the third stage, the models were trained and tested with datasets that had been one-hot encoded, balanced, and normalized using the standard scaling method as depicted in Figure 1.

Six state-of-the-art machine learning models were considered for this work: Decision Tree (DT), Gaussian Naïve Bayes (GNB), K-Nearest Neighbours (KNN), Logistic Regression, Random Forest (RF), and Support Vector Machine (SVM) for comparison of the efficacy of the three different stages of training and testing these models. The proposed conceptual framework performed binary classification, categorizing network signals as either "normal" or "attack".

B. Label Encoding

Label encoding is a machine learning technique used to convert categorical data, represented as labels or strings, into numerical format. In this process, each unique category is assigned a distinct integer, transforming categorical values into numerical ones. The primary goal is to make the data suitable for machine learning algorithms that need numerical input. Label encoding is particularly useful for categorical data with an inherent order or ranking, making it ideal for ordinal categorical data. Ordinal data is a type of categorical data where there is a clear order or ranking, indicating that one category is greater or smaller than another (Sunny, 2024). In this study, label encoding was used to transform all categorical features of the datasets into their numerical equivalents before performing feature selection. The label encoding algorithm is written as Algorithm 1.

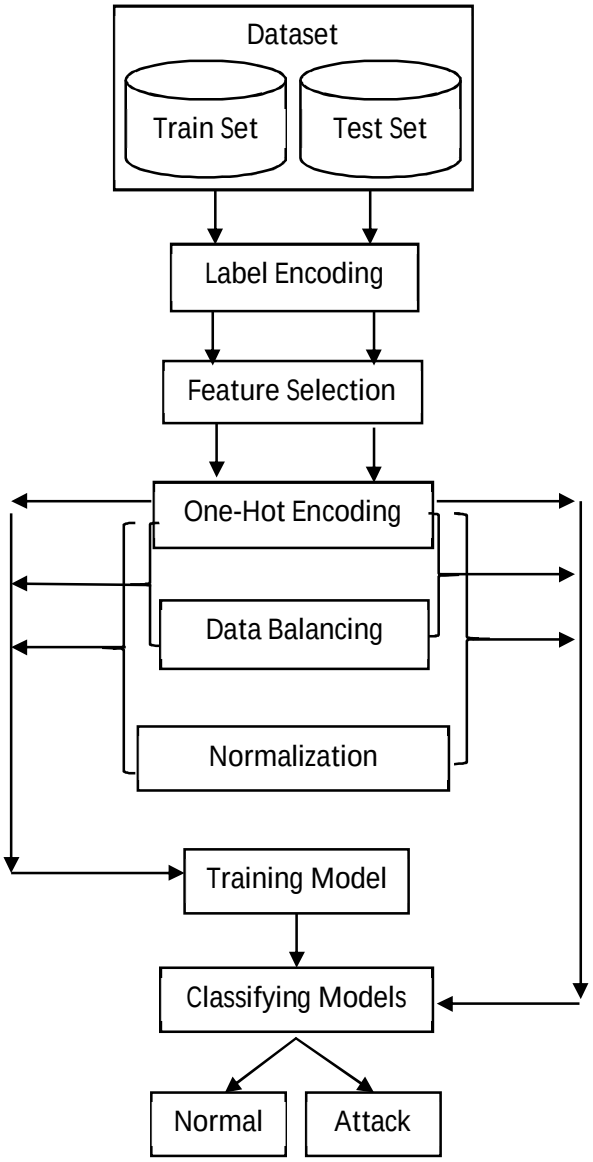


Figure 1: A conceptual framework for the proposed anomaly-based IDS

Algorithm 1: Label Encoding

```
Algorithm LabelEncoding(C):
    Input:
        C - a categorical dataset or column with n samples
    Output:
        L - an array of integer labels corresponding to the categorical values in C
    # Step 1: Identify unique categories
    U <- Unique(C)
    # Step 2: Create a mapping of unique categories to integer labels
    M <- Dictionary()
    for i from 0 to Length(U) - 1:
        M[U[i]] <- i
    # Step 3: Transform the data
    L <- []
    for v in C:
        L.append(M[v])
    # Step 4: Return the encoded labels
    return L
```

C. *Feature Selection*

Feature selection is a process of acquiring a subset from an original feature set according to a certain feature selection criterion, which selects the relevant features of the dataset (Cai et al., 2018). In other words, it reduces the number of columns or variables that are redundant and irrelevant in a dataset normally before constructing the predictive models. The purpose of implementing feature selection is to improve model's accuracy, save computational time and cost, as the machine learning methods will have difficulty in dealing with the large number of input features (Kumar, 2014) and causing overfitting problems. Here in this work, we utilize the feature importance or variable importance technique for feature selection to achieve a precise and accurate representation for the IDS problem. This method accounts for the fact that not all features are significant or relevant in detecting intrusions.

D. *One-Hot Encoding*

Categorical variables are frequently used in machine learning and data analysis. Examples include colours, gender, or age groups, which have a limited number of distinct values. However, most machine learning algorithms require numerical input, and categorical variables are not numerical. Therefore, we need to encode these categorical variables into a numerical format. One common approach is one-hot encoding, which transforms categorical data into binary data, where each category is represented by a binary value. A binary vector, with a length equal to the number of categories, is created for the entire categorical variable. For example, if your categorical variable has three categories like "red", "green", and "blue", the one-hot encoding representation will have three dimensions, each corresponding to one of the categories. In one-hot encoding, the dimension that represents the category to which an observation belongs is assigned a binary value of 1, while all other dimensions are assigned a binary value of 0 (Tutorials Point, 2024). This study utilized the "pd.get_dummies()" function from the Pandas library in Python to transform categorical data into a format suitable for machine learning algorithms, enhancing their predictive performance.

E. *Data Balancing*

Imbalanced datasets are a common challenge for machine learning practitioners dealing with binary classification problems. This issue often occurs in real-world business applications such as fraud detection, spam filtering, rare disease discovery, network intrusion detection systems, and hardware fault detection. A widely used method to tackle this problem is the Synthetic Minority Oversampling Technique (SMOTE) (Swastik, 2024)**.** SMOTE is crafted to address imbalanced datasets by creating synthetic instances for the minority class. This study highlighted the importance of SMOTE in handling class imbalances, specifically emphasizing its role in enhancing classifier model performance. Through its ability to reduce bias and capture crucial features of the minority class, SMOTE leads to improved predictive accuracy and overall model performance.

F. Data Normalization

Dataset normalization plays a crucial role in pre-processing, particularly for classification tasks. It involves transforming dataset attributes into compatible values, facilitating faster operations and higher success rates. This study utilized Standard Scaler normalization to harmonize the dataset attributes (Çavuşoğlu, 2019). Standard Scaler is a feature scaling technique provided by the sklearn.preprocessing module in the Scikit-learn library in Python. It standardizes features by removing the mean and scaling to unit variance. This means that the resulting distribution of the features will have a mean of 0 and a standard deviation of 1. Standardizing the features is a common requirement for many machine learning algorithms. Algorithms that compute distances between data points, such as k-nearest neighbours (KNN) or support vector machines (SVM), can benefit from standardized data. It ensures that each feature contributes equally to the result, avoiding dominance by features with larger scales.

Mathematically, the transformation is given by:

$$X_{scaled} = \frac{X-\mu}{\sigma} \quad (1)$$

where μ is the mean of the feature values and σ is the standard deviation.

G. ML Models

Machine learning (ML) models are a subset of artificial intelligence (AI) that encompass various methods and algorithms enabling machines to learn automatically through mathematical models, extracting valuable information from large datasets (Teresa, 1993; Xin et al., 2018). The key ML models utilized in this study include Decision Tree (DT), Gaussian Naïve Bayes (GNB), K-Nearest Neighbors (KNN), Logistic Regression, Random Forest (RF), and Support Vector Machine (SVM).

1) Decision Tree (DT)

DT are fundamental supervised machine learning algorithms used for both classification and regression by making a series of decisions (rules). The model is structured like a conventional tree with nodes, branches, and leaves. Each node signifies an attribute or feature, each branch represents a decision or rule, and each leaf indicates a possible outcome or class label (Sahani et al., 2018). The DT algorithm automatically selects the optimal features to construct the tree and performs pruning to eliminate irrelevant branches, preventing overfitting. The most common DT models include CART, C4.5, and ID3 (Rai et al., 2016).

2) Gaussian Naïve Bayes (GNB)

Gaussian Naive Bayes is a variant of the Naive Bayes method that handles continuous attributes by assuming the data features follow a Gaussian distribution throughout the dataset. In the Sklearn library, Gaussian Naive Bayes is a classification algorithm for normally distributed continuous features based on the Naive Bayes algorithm. This classifier is rooted in Bayes Theorem from probability theory and excels at predicting the correct class for the given features. The term “naive” refers to the algorithm’s assumption that all features are independent of each other, an assumption that may not hold true in real-world scenarios. (Geeks for Geeks, 2024).

3) K-Nearest Neighbours (KNN)

KNN is a straightforward supervised machine learning algorithm that uses “feature similarity” to predict the class of a given data sample. It classifies a sample based on its proximity to its neighbours by measuring the distance between them. The performance of the KNN model is influenced by the parameter k. A very small k value can lead to overfitting, while a very large k value may cause misclassification of the sample instance (Zhang et al., 2019).

4) Logistic Regression

Logistic regression is a supervised machine learning algorithm designed for binary classification tasks, predicting the probability of an outcome, event, or observation. This model generates a binary outcome, limited to two possibilities such as yes/no, 0/1, or true/false. It examines the relationship between one or more independent variables and categorizes data into distinct classes. Logistic regression is commonly used in predictive modeling to estimate the mathematical probability of an instance belonging to a specific category. For example, a 0 might represent a negative class and a 1 a positive class.

This algorithm is especially useful for binary classification problems where the outcome variable falls into one of two categories (0 or 1) (Vijay, 2024).

5) Random Forest (RF)

The Random Forest Algorithm is a highly popular supervised machine learning algorithm used for both classification and regression tasks. Similar to how a forest is composed of numerous trees, a random forest's robustness increases with the number of trees it contains.

The greater the number of trees, the higher the algorithm's accuracy and problem-solving capability. Random Forest is a classifier that consists of several decision trees applied to different subsets of the dataset, and it averages their results to enhance predictive accuracy. This algorithm is based on the concept of ensemble learning, which combines multiple classifiers to tackle complex problems and improve model performance (Simplilearn, 2024).

6) Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised machine learning algorithm that relies on the concept of a maximum-margin hyperplane in an n-dimensional feature space. It can address both linear and nonlinear problems. For nonlinear problems, SVM employs kernel functions to map low-dimensional input vectors into a high-dimensional feature space. Then, it determines an optimal maximum-margin hyperplane, which serves as the decision boundary, using the support vectors (Chen et al., 2005), (Devi & Suganthe, 2018). For Network Intrusion Detection Systems (NIDS), the SVM algorithm can improve efficiency and accuracy by accurately distinguishing between normal and malicious classes (Yan & Han, 2018), (Ghanem et al., 2017).

IV. EXPERIMENT RESULT AND DISCUSSION

For this study, six advanced machine learning models were selected: Decision Tree (DT), Gaussian Naïve Bayes (GNB), K-Nearest Neighbours (KNN), Logistic Regression, Random Forest (RF), and Support Vector Machine (SVM). These models were used to compare the effectiveness of the three different stages of training and testing.

A. Benchmark Datasets

We chose NSL-KDD and UNSW_NB15 as the benchmark datasets for this experiment.

1) NSL-KDD Dataset

The NSL-KDD dataset is an improved version of the original KDD Cup 1999 dataset, widely used in network intrusion detection research. The version utilized here is in .csv format and can be accessed at (Kaggle, 2021). It includes 42 attributes (including the target variable) with 125,973 instances in the training set and 22,544 instances in the test set.

The target variable, referred to as "labels", in the training set consists of "normal" and 21 different attack types: ("normal", "neptune", "warezclient", "ipsweep", "portsweep", "teardrop", "nmap", "satan", "smurf", "pod", "back", "guess_passwd", "ftp_write", "multihop", "rootkit", "buffer_overflow", "imap", "warezmaster", "phf", "land", "loadmodule", "perl").

In the test set, the "labels" feature includes "normal" and 35 attack types: ("normal", "portsweep", "neptune", "smurf", "satan", "apache2", "teardrop", "guess_passwd", "ipsweep", "warezclient", "nmap", "warezmaster", "mscan", "back", "pod", "httptunnel", "processtable", "mailbomb", "snmpguess", "saint", "multihop", "snmpgetattack", "buffer_overflow", "xsnoop", "imap", "ps", "rootkit", "land", "xterm", "sendmail", "phf", "loadmodule", "perl", "xlock", "ftp_write", "named").

2) UNSW_NB2015 Dataset

The second dataset considered is the University of New South Wales's network intrusion dataset (UNSW-NB2015) generated in 2015. The dataset is a hybrid of real modern network normal activities and synthetic attack behaviours. The data was collected through two different simulations conducted on two different days, namely January 22 and February 17, 2015. The resulting dataset consists of 2,540,044 instances and 49 features (1 class feature and 48 statistical features) representing the different characteristics of a network traffic request such as source and destination details, duration, protocol used, and packet size. These instances are labelled as follows: 2,218,761 normal instances and 521,283 attack instances.

The specific version used for this work is a partition from this dataset, configured as a training set and testing set, namely, UNSW_NB15_training-set.csv and UNSW_NB15_testing-set.csv respectively having a total of 45 features each. The number of records in the training set is 175,341 and the testing set is 82,332 records from different types of attack and normal (Moustafa & Slay, 2015).

B. Data Pre-processing

The details of the pre-processing operations performed on the two datasets are outlined below:

1) NSL-KDD Dataset

As detailed in Section 4.1.1, the NSL-KDD dataset comprises 42 attributes (including the target variable) with 125,973 instances in the training set and 22,544 instances in the test set. The target variable, known as "labels," includes "normal" and 21 distinct attack types in the training set, whereas the test set's "labels" feature consists of "normal" and 35 different attack types. All attacks not present in both the training and test sets were removed, leaving only 21 attacks common to both sets. Since the default structure of the "labels" attribute in NSL-KDD is multi-class classification, all the attacks were combined into a single feature named "attack". Consequently, the "labels" attribute now has only two classes: "normal" and "attack." This is further detailed in Table 1. The dataset contains no null values. Four attributes in the dataset are categorical: "protocol_type", "service", "flag" and "labels." The "labels" attribute, which has been merged into two classes ("normal" and "attack"), was converted into numeric variables (0 and 1) using a label encoder. The other three categorical attributes were converted using one-hot encoding.

The feature importance or variable importance feature selection technique was used to identify and retain only the important features of the dataset that contributed to the target variable. Irrelevant features were discarded prior to model training.

2) UNSW_NB2015 Dataset

In this case, no merging of attacks was necessary since the dataset was originally labelled in a binary manner. The dataset has no null values. Under the "service" attribute in both the training and test sets, the first sub-service option denoted by (-) without a valid name was removed. Regarding the "state" attribute, only sub-services common to both the training and test sets were retained, while sub-services not present in both sets were deleted. Four features of the dataset, namely "proto", "service", "state" and "attack_cat," which were categorical, were converted to numerical equivalents using one-hot encoding. Only important features were identified and used for model training.

Table 1: Pre-processed "labels" Attribute of NSL-KDD Dataset

NSL-KDD Dataset				
S/N	train['labels'].value_counts()		test['labels'].value_counts()	
1.	normal	67343	normal	11245
2.	neptune	41214	neptune	6654
3.	satan	3633	satan	698
4.	ipsweep	3599	smurf	540
5.	portsweep	2931	ipsweep	479
6.	smurf	2646	guess_passwd	411
7.	nmap	1493	portsweep	371
8.	back	956	warezmaster	279
9.	teardrop	892	back	227
10.	warezclient	890	nmap	206
11.	pod	201	warezclient	107
12.	guess_passwd	53	teardrop	104
13.	buffer_overflow	30	pod	35
14.	warezmaster	20	buffer_overflow	17
15.	land	18	multihop	9
16.	imap	11	rootkit	4
17.	rootkit	10	loadmodule	4
18.	loadmodule	9	imap	2
19.	ftp_write	8	land	2
20.	multihop	7	ftp_write	2
21.	phf	4	phf	1
22.	perl	3	perl	1
	TOTAL	125,971	TOTAL	21,398
	NORMAL	67,343	NORMAL	11,245
	ATTACK	58,628	ATTACK	10,153

C. Evaluation Metrics

This section details the evaluation metrics used to measure the performance of the ML models in this study. These metrics are derived from the various attributes of the Confusion Matrix, a two-dimensional matrix that provides information about the actual and predicted classes (Deng et al., 2016). The evaluation metrics include:

- True Positive (TP): Instances correctly identified as an attack by the classifier.
- False Negative (FN): Instances incorrectly predicted as normal.
- False Positive (FP): Instances incorrectly classified as an attack.
- True Negative (TN): Instances correctly classified as normal.

The diagonal elements of the confusion matrix represent correct predictions, while the off-diagonal elements indicate incorrect predictions made by the classifier. Table 2 illustrates these attributes of the confusion matrix. Additionally, the various evaluation metrics used in this study are:

1) Accuracy

Accuracy quantifies the ratio of correct predictions to the total number of predictions, calculated by dividing the number of correct predictions by the total predictions made.

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions} = \frac{TP+TN}{TP+FP+TN+FN} \quad (2)$$

2) Precision

Precision measures the accuracy of positive predictions by determining the proportion of correct positive results (True Positives, TP) divided by the total number of positive results (TP + False Positives, FP) predicted by the classifier.

$$Precision = \frac{Number\ of\ Correct\ Positive\ Results}{Total\ Number\ of\ Positive\ Results} = \frac{TP}{TP+FP} \quad (3)$$

3) Recall

Recall indicates the proportion of correctly predicted positive instances out of all the positives that the model could have identified. It's calculated by dividing True Positives (TP) by the sum of True Positives and False Negatives (FN) in the dataset. Unlike precision, which focuses on accurately predicted positives among all positive predictions, recall sheds light on missed positive predictions.

$$Recall = \frac{Number\ of\ Correct\ Positives}{Number\ of\ all\ Positives} = \frac{TP}{TP+FN} \quad (4)$$

4) F1 Score

The F1 Score aims to strike a balance between precision and recall by computing their harmonic mean. It serves as a measure of a test's accuracy, with a maximum value of 1 indicating perfect precision and recall alignment.

$$F1\ Score = \text{Harmonic Mean of Precision and Recall} = 2 \times \frac{Precision\ X\ Recall}{Precision + Recall} = \frac{2TP}{2TP+FP+FN} \quad (5)$$

Table 2: Confusion Matrix

		Predicted Class	
		Normal (0)	Attack (1)
Actual Class	Normal (0)	TN	FP
	Attack (1)	FN	TP

D. Results

We implemented all the methodologies using the Python programming language, running on Anaconda's Jupyter Notebook, utilizing Scikit-learn, a widely adopted machine learning library for predictive data analysis, and executed on a Windows PC. Here, we present our findings on the performance of each model across three training and testing stages: the original one-hot encoded datasets, one-hot encoded datasets with SMOTE, and one-hot encoded datasets with SMOTE and scaling. The results are summarized in Table 3, where the performance of these six ML models is evaluated using four key metrics: accuracy, precision, recall, and F1 score.

From Table 3, it is clear that both the decision tree and random forest exhibit identical performance, achieving 99.99% across all four metrics during each of the three stages of training and testing with the UNSW-NB15 dataset. The only exception is in the NSL-KDD dataset, where the decision tree slightly outperforms the random forest with an accuracy of 99.98% compared to the random forest's 99.97%. A close examination of Table 3 also reveals that all six ML classifiers in this experiment achieve their best performance in both datasets during the third stage, where the models are trained and evaluated on the dataset version that has undergone balancing (SMOTE) and scaling. Figures 2 and 3 respectively display the performances of the balanced and scaled versions of the two datasets on the six machine learning models.

Table 3: Comparison results of the three stages of training and testing the six models

Model	NSL-KDD Dataset				UNSW-NB15 Dataset			
	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1
DT (Ogiginal Dataset)	99.95	99.94	99.95	99.94	99.95	99.94	99.95	99.97
DT+SMOTE	99.97	99.95	99.96	99.97	99.98	99.97	99.98	99.96
DT+SMOTE+Scaled	99.98	99.97	99.97	99.99	99.99	99.99	99.99	99.99
GNB (Ogiginal Dataset)	52.34	52.48	98.57	68.49	79.26	99.82	71.58	83.37
GNB+SMOTE	49.86	49.93	98.54	66.28	85.43	99.18	71.45	83.06
GNB+SMOTE+Scaled	99.92	99.98	99.85	99.92	99.99	99.95	99.96	99.96
KNN (Ogiginal Dataset)	96.17	94.08	98.94	96.45	89.10	88.56	97.61	92.86
KNN+SMOTE	96.02	93.50	98.92	96.14	83.69	78.40	92.99	85.07
KNN+SMOTE+Scaled	96.87	95.74	98.11	96.91	99.92	99.96	99.89	99.92
LR (Ogiginal Dataset)	85.79	83.66	90.66	87.02	77.42	83.37	86.09	84.71
LR+SMOTE	85.00	83.39	88.61	85.92	82.00	79.18	86.87	82.85
LR+SMOTE+Scaled	99.93	99.92	99.96	99.95	99.96	99.94	99.97	99.96
RF (Ogiginal Dataset)	99.94	99.94	99.94	99.93	99.95	99.94	99.95	99.97
RF+SMOTE	99.96	99.95	99.96	99.96	99.98	99.97	99.98	99.96
RF+SMOTE+Scaled	99.97	99.97	99.97	99.99	99.99	99.99	99.99	99.99
SVM (Ogiginal Dataset)	52.60	52.58	55.52	68.92	80.76	79.73	98.57	88.16
SVM+SMOTE	50.24	90.91	00.53	01.06	85.00	96.82	72.38	82.83
SVM+SMOTE+Scaled	99.74	99.92	99.56	99.74	99.86	99.79	99.94	99.86

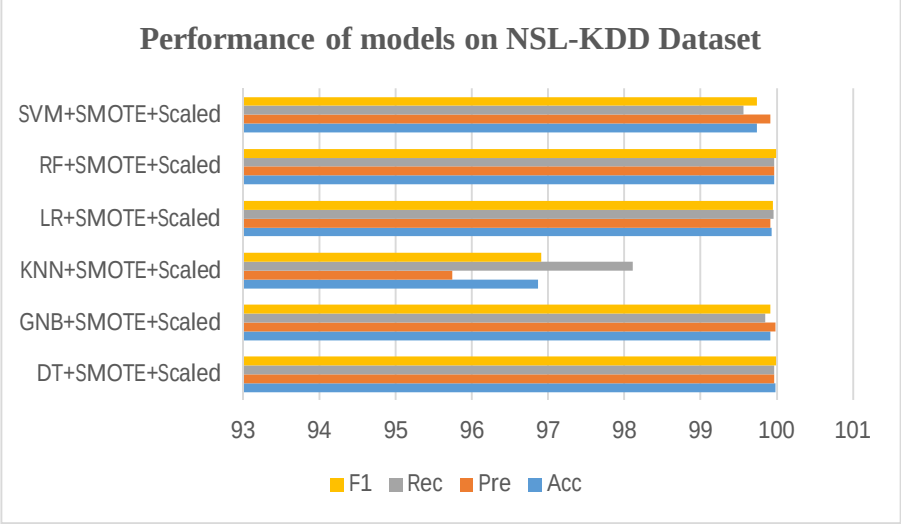


Figure 2: Performance of models on NSL-KDD Dataset

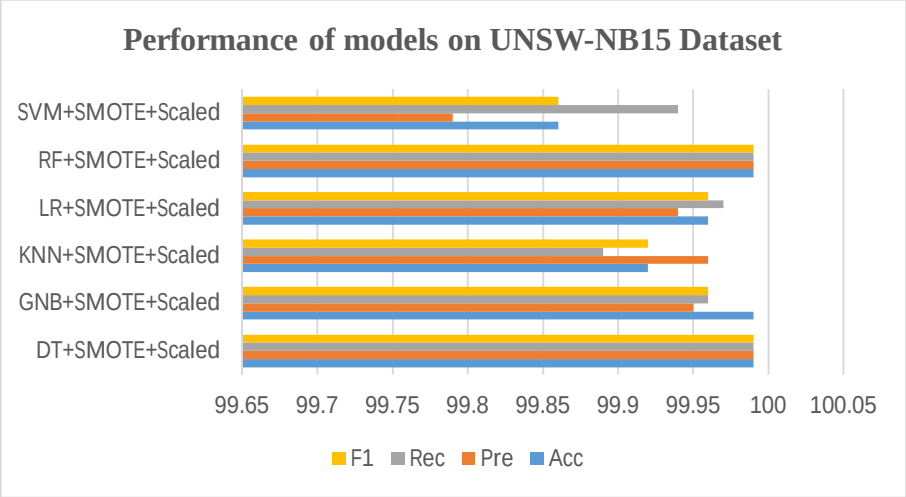


Figure 3: Performance of models on UNSW-NB15 Dataset

Table 4: Performance benchmark with some of the existing approaches on NSL-KDD$_{test}$

Reference	Feature Selection	Acc	Pre	Rec	F1
(Çavuşoğlu, 2019)	CfsSubsetEval and WrapperSubsetEval	99.86	-	99.96	-
(Yang et al., 2019)	-	85.97	97.39	77.43	86.27
(Gao et al., 2019)	PCA	85.20	86.50	85.20	84.90
(Liu et al., 2021)	-	82.84	83.94	82.78	81.66
(Yan & Han, 2018)	-	99.04	-	-	-
(Sarker et al., 2020)	Random Forest Algorithm	98.00	98.00	98.00	98.00
(Tavallaee et al., 2009)	-	82.02	-	-	-
(Ma & Shi, 2021)	SMOTE	82.09	84.11	82.09	82.43
(Bedi et al., 2021)		80.00	96.80	97.40	90.70
(Tama et al., 2019)	Hybrid	91.27	96.60	91.30	-
Proposed Study	Random Forest Algorithm	99.98	99.97	99.97	99.99

Table 5: Performance benchmark with some of the existing approaches on UNSW-NB15$_{test}$

Reference	Feature Selection	Acc	Pre	Rec	F1
(Yang et al., 2019)	-	89.08	86.05	98.90	90.61
(Tama et al., 2019)	Hybrid	91.27	91.60	91.30	-
(Khammassi & Krichen, 2017)	GA-LR Wrapper Method	81.42	-	-	-
(Agarwal et al., 2021)	-	97.78	-	-	-
(Zong et al., 2018)	Information Gain	85.78	-	-	-
(Souhail Et. Al., 2019)	Recursive Feature Elimination & Random Forests	86.00	-	-	-
(Bagui et al., 2019)	*k*-means clustering & correlation-based	99.59	-	-	-
(Jing & Chen, 2019)	-	85.99	-	-	-
(Kasongo & Sun, 2020)	XGBoost algorithm	90.85	83.91	98.38	90.00
Proposed Study	Random Forest Algorithm	99.99	99.99	99.99	99.99

As demonstrated in Tables 4 and 5, our proposed study exhibits higher accuracy compared to other related research studies. Consequently, our proposed method is more generalizable to imbalanced network traffic.

V. CONCLUSION

This paper introduces a novel method for an anomaly-based intrusion detection system, which involves a three-level process of training and testing machine learning models, including feature selection, resampling, and normalization. The models utilized in this study are Decision Tree, Gaussian Naïve Bayes, K-Nearest Neighbors, Logistic Regression, Random Forest, and Support Vector Machine. The effectiveness of this approach is evaluated using two intrusion datasets: NSL-KDD and UNSW-NB15. Compared to other related research, our findings indicate that the proposed approach outperforms existing state-of-the-art methods. It demonstrates superior results in accuracy, precision, recall, and F1-score when validated against predefined testing sets, namely NSL-KDDtest and UNSW-NB15test. Future research should aim to validate the proposed method for addressing multi-class classification problems, which involve categorizing incoming network traffic as normal or into various attack groups.

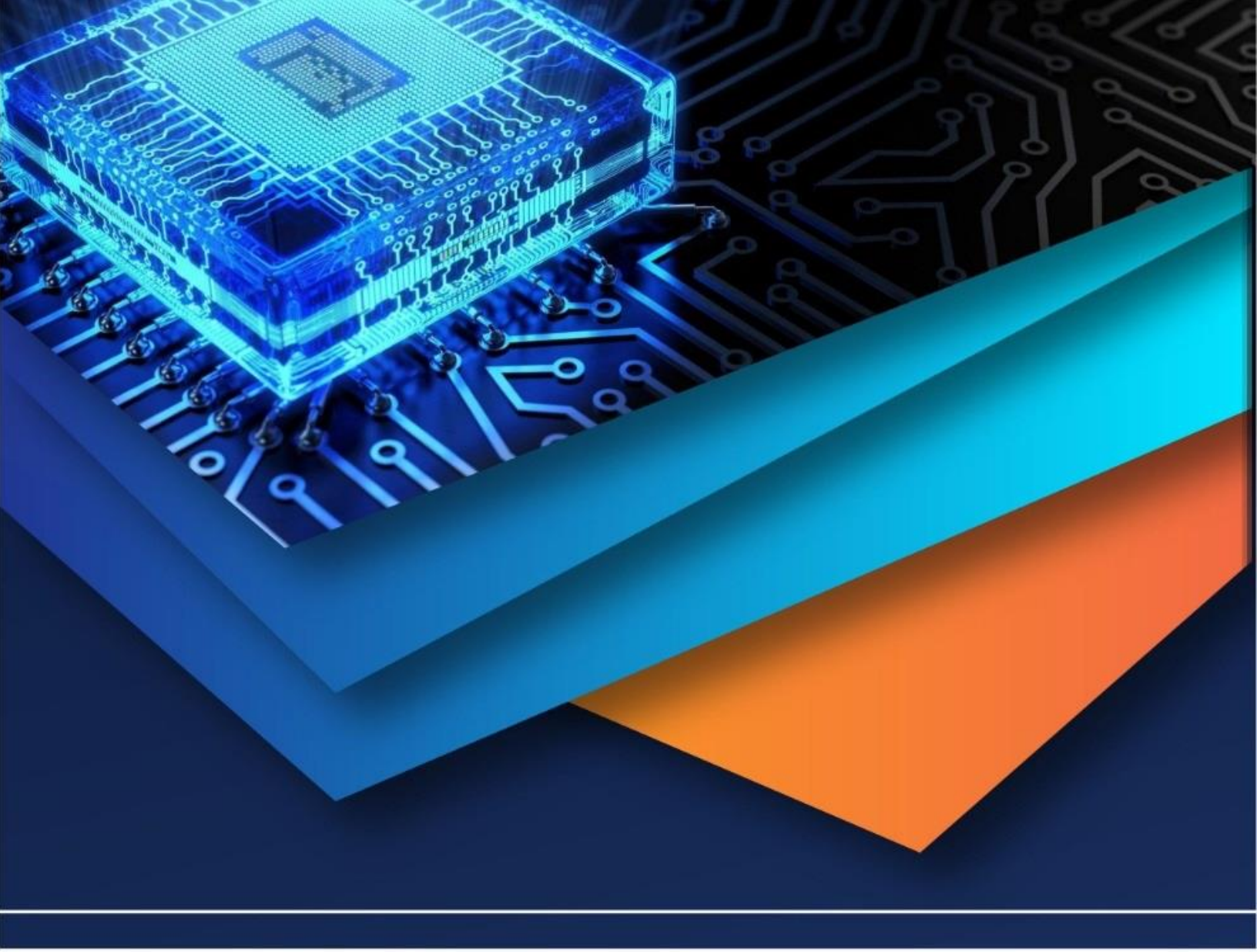
VI. ACKNOWLEDGMENT

The authors wish to thank TetFund Nigeria for sponsoring this research work. The Directorate, Centre for Research, Innovation and Development of the Federal Polytechnic, Ado-Ekiti is highly appreciated for their cooperation and understanding.

REFERENCES

[1] Agarwal, A., Sharma, P., Alshehri, M., Mohamed, A. A., & Alfarraj, O. (2021). Classification model for accuracy and intrusion detection using machine learning approach. PeerJ Computer Science, 7, e437. https://doi.org/10.7717/peerj-cs.437
[2] Anjum, N., & Chowdhury, M. R. (2024). International Journal of Advanced Research in Computer and Communication Engineering. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.4847308
[3] Bagui, S., Kalaimannan, E., Bagui, S., Nandi, D., & Pinto, A. (2019). Using machine learning techniques to identify rare cyber-attacks on the UNSW-NB15 dataset. SECURITY AND PRIVACY, 2(6), e91. https://doi.org/10.1002/spy2.91
[4] Bedi, P., Gupta, N., & Jindal, V. (2021). I-SiamIDS: An improved Siam-IDS for handling class imbalance in network-based intrusion detection systems. Applied Intelligence, 51(2), 1133–1151. https://doi.org/10.1007/s10489-020-01886-y
[5] Binghao, Y., & Guodong, H. (2018). Building Combined Intrusion Detection Model Based on Imbalanced Learning and Gated Recurrent Unit Neural Network. Security and Communication Networks, 2018, 13 Pages. https://doi.org/10.1155/2018/6026878
[6] Cai, J., Luo, J., Wang, S., & Yang, S. (2018). Feature selection in machine learning: A new perspective. Neurocomputing, 300, 70–79. https://doi.org/10.1016/j.neucom.2017.11.077
[7] Çavuşoğlu, Ü. (2019). A new hybrid approach for intrusion detection using machine learning methods. Applied Intelligence, 49(7), 2735–2761. https://doi.org/10.1007/s10489-018-01408-x
[8] Chen, W.-H., Hsu, S.-H., & Shen, H.-P. (2005). Application of SVM and ANN for intrusion detection. Computers & Operations Research, 32(10), 2617–2634. https://doi.org/10.1016/j.cor.2004.03.019
[9] Deng, X., Liu, Q., Deng, Y., & Mahadevan, S. (2016). An improved method to construct basic probability assignment based on the confusion matrix for classification problem. Information Sciences, 340–341, 250–261. https://doi.org/10.1016/j.ins.2016.01.033
[10] Devi, E. M. R., & Suganthe, R. C. (2018). Enhanced transductive support vector machine classification with grey wolf optimizer cuckoo search optimization for intrusion detection system. John Wiley & Sons, Ltd., 11 Pages. https://doi.org/DOI: 10.1002/cpe.4999
[11] Dong, B., & Wang, X. (2016). Comparison deep learning method to traditional methods using for network intrusion detection. 2016 8th IEEE International Conference on Communication Software and Networks (ICCSN), 581–585. https://doi.org/10.1109/ICCSN.2016.7586590
[12] Fang, W., Tan, X., & Wilbur, D. (2020). Application of intrusion detection technology in network safety based on machine learning. Safety Science, 124, 104604. https://doi.org/10.1016/j.ssci.2020.104604
[13] Gao, X., Shan, C., Hu, C., Niu, Z., & Liu, Z. (2019). An Adaptive Ensemble Machine Learning Model for Intrusion Detection. IEEE Access, 7, 82512–82521. https://doi.org/10.1109/ACCESS.2019.2923640
[14] Geeks for Geeks. (2024). Gaussian Naive Bayes. Available Online: https://www.geeksforgeeks.org/gaussian-naive-bayes/
[15] Ghanem, K., Aparicio-Navarro, F. J., Kyriakopoulos, K. G., Lambotharan, S., & Chambers, J. A. (2017). Support Vector Machine for Network Intrusion and Cyber-Attack Detection. 2017 Sensor Signal Processing for Defence Conference (SSPD), 1–5. https://doi.org/10.1109/SSPD.2017.8233268
[16] Herve, D., Marc, D., & Andreas, W. (1993). Towards a taxonomy of intrusion-detection systems. Computer Networks 31, 805–822.
[17] Hoque, M. S., Mukit, M. A., & Bikas, M. A. N. (2012). An Implementation of Intrusion Detection System Using Genetic Algorithm. International Journal of Network Security & Its Applications, 4(2). https://doi.org/DOI : 10.5121/ijnsa.2012.4208
[18] Injadat, M., Moubayed, A., Nassif, A. B., & Shami, A. (2021). Multi-Stage Optimized Machine Learning Framework for Network Intrusion Detection. IEEE Transactions on Network and Service Management, 18(2), 1803–1816. https://doi.org/10.1109/TNSM.2020.3014929
[19] Javaid, A., Niyaz, Q., Sun, W., & Alam, M. (2016). A Deep Learning Approach for Network Intrusion Detection System. Proceedings of the 9th EAI International Conference on Bio-Inspired Information and Communications Technologies (Formerly BIONETICS). 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS), New York City, United States. https://doi.org/10.4108/eai.3-12-2015.2262516
[20] Jing, D., & Chen, H.-B. (2019). SVM Based Network Intrusion Detection for the UNSW-NB15 Dataset. 2019 IEEE 13th International Conference on ASIC (ASICON), 1–4. https://doi.org/10.1109/ASICON47005.2019.8983598
[21] Kaggle. (2021). NSL-KDD Dataset. NSL-KDD Dataset. https://www.kaggle.com/datasets/kiranmahesh/nslkdd
[22] Kasongo, S. M., & 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), 105. https://doi.org/10.1186/s40537-020-00379-6

[23] Khammassi, C., & Krichen, S. (2017). A GA-LR wrapper approach for feature selection in network intrusion detection. Computers & Security, 70, 255–277. https://doi.org/10.1016/j.cose.2017.06.005
[24] Kumar, V. (2014). Feature Selection: A literature Review. The Smart Computing Review, 4(3). https://doi.org/10.6029/smartcr.2014.03.007
[25] Lew, J., Shah, D. A., Pati, S., Cattell, S., Zhang, M., Sandhupatla, A., Ng, C., Goli, N., Sinclair, M. D., Rogers, T. G., & Aamodt, T. M. (2019). Analyzing Machine Learning Workloads Using a Detailed GPU Simulator. 2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 151–152. https://doi.org/10.1109/ISPASS.2019.00028
[26] Liu, L., Wang, P., Lin, J., & Liu, L. (2021). Intrusion Detection of Imbalanced Network Traffic Based on Machine Learning and Deep Learning. IEEE Access, 9, 7550–7563. https://doi.org/10.1109/ACCESS.2020.3048198
[27] Ma, X., & Shi, W. (2021). AESMOTE: Adversarial Reinforcement Learning With SMOTE for Anomaly Detection. IEEE Transactions on Network Science and Engineering, 8(2), 943–956. https://doi.org/10.1109/TNSE.2020.3004312
[28] Mahfouz, A., Abuhussein, A., Venugopal, D., & Shiva, S. (2020). Ensemble Classifiers for Network Intrusion Detection Using a Novel Network Attack Dataset. Future Internet, 12(11), 180. https://doi.org/10.3390/fi12110180
[29] Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). 2015 Military Communications and Information Systems Conference (MilCIS), 1–6. https://doi.org/10.1109/MilCIS.2015.7348942
[30] Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., & Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. Journal of Big Data, 2(1), 1. https://doi.org/10.1186/s40537-014-0007-7
[31] Rai, K., Devi, M. S., & Guleria, A. (2016). Decision Tree Based Algorithm for Intrusion Detection. 07(04).
[32] Ramjee Prasad, Vandana Rohokale. (2020). Cyber Security: The Lifeline Of Information And Communication Technology. Springer.
[33] Sahani, R., Shatabdinalini, Rout, C., Chandrakanta Badajena, J., Jena, A. K., & Das, H. (2018). Classification of Intrusion Detection Using Data Mining Techniques. In P. K. Pattnaik, S. S. Rautaray, H. Das, & J. Nayak (Eds.), Progress in Computing, Analytics and Networking (Vol. 710, pp. 753–764). Springer Singapore. https://doi.org/10.1007/978-981-10-7871-2_72
[34] Sarker, I. H., Abushark, Y. B., Alsolami, F., & Khan, A. I. (2020). IntruDTree: A Machine Learning Based Cyber Security Intrusion Detection Model. Symmetry, 12(5), 754. https://doi.org/10.3390/sym12050754
[35] Simplilearn. (2024). Random Forest Algorithm. Available Online: https://www.simplilearn.com/tutorials/machine-learning-tutorial/random-forest-algorithm
[36] Souhail Et. Al., M. (2019). Network Based Intrusion Detection Using the UNSW-NB15 Dataset. International Journal of Computing and Digital Systems, 8(5), 477–487. https://doi.org/10.12785/ijcds/080505
[37] Sunny, K. (2024, June 18). What is label encoding? Application of label encoder in machine learning and deep learning models. Available Online: https://medium.com/@sunnykumar1516/what-is-label-encoding-application-of-label-ncoder-in-machine-learning-and-deep-learning-models-593669483ed
[38] Swastik, S. (2024). SMOTE for Imbalanced Classification with Python. Available Online: https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/
[39] Tama, B. A., Comuzzi, M., & Rhee, K.-H. (2019). TSE-IDS: A Two-Stage Classifier Ensemble for Intelligent Anomaly-Based Intrusion Detection System. IEEE Access, 7, 94497–94507. https://doi.org/10.1109/ACCESS.2019.2928048
[40] Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, 1–6. https://doi.org/10.1109/CISDA.2009.5356528
[41] Teresa, F. L. (1993). A survey of intrusion detection techniques. Elsevier Science Publishers Ltd., 405–418.
[42] Tutorials Point. (2024). One Hot Encoding and Label Encoding Explained. Tutorials Point. Available Online: https://www.tutorialspoint.com/one-hot-encoding-and-label-encoding-explained
[43] Verma, A., & Ranga, V. (2020). Machine Learning Based Intrusion Detection Systems for IoT Applications. Wireless Personal Communications, 111(4), 2287–2310. https://doi.org/10.1007/s11277-019-06986-8
[44] Vijay, K. (2024). What is Logistic Regression? Equation, Assumptions, Types, and Best Practices. Available Online: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/
[45] Xin, Y., Kong, L., Liu, Z., Chen, Y., Li, Y., Zhu, H., Gao, M., Hou, H., & Wang, C. (2018). Machine Learning and Deep Learning Methods for Cybersecurity. IEEE Access, 6, 35365–35381. https://doi.org/10.1109/ACCESS.2018.2836950
[46] Yan, B., & Han, G. (2018). Effective Feature Extraction via Stacked Sparse Autoencoder to Improve Intrusion Detection System. IEEE Access, 6, 41238–41248. https://doi.org/10.1109/ACCESS.2018.2858277
[47] Yang, Y., Zheng, K., Wu, C., & Yang, Y. (2019). Improving the Classification Effectiveness of Intrusion Detection by Using Improved Conditional Variational AutoEncoder and Deep Neural Network. Sensors, 19(11), 2528. https://doi.org/10.3390/s19112528
[48] Zaman, M., & Lung, C.-H. (2018a). Evaluation of machine learning techniques for network intrusion detection. NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium, 1–5. https://doi.org/10.1109/NOMS.2018.8406212
[49] Zaman, M., & Lung, C.-H. (2018b). Evaluation of machine learning techniques for network intrusion detection. NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium, 1–5. https://doi.org/10.1109/NOMS.2018.8406212
[50] Zhang, Y., Cao, G., Wang, B., & Li, X. (2019). A novel ensemble method for k-nearest neighbor. Pattern Recognition, 85, 13–25. https://doi.org/10.1016/j.patcog.2018.08.003
[51] Zong, W., Chow, Y.-W., & Susilo, W. (2018). A Two-Stage Classifier Approach for Network Intrusion Detection. In C. Su & H. Kikuchi (Eds.), Information Security Practice and Experience (Vol. 11125, pp. 329–340). Springer International Publishing. https://doi.org/10.1007/978-3-319-99807-7_20



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY