FMDB *Transactions on Sustainable Computer Letters*

Enhancing Cyber Intrusion Detection through Ensemble Learning: A Comparison of Bagging and Stacking Classifiers

P. Dhinakaran[1,*], M.A. Thinesh[2], Mykhailo Paslavskyi[3]

[1,2]Department of Computer Science and Engineering, SRM Institute of Science and Technology, Ramapuram, Chennai, Tamil Nadu, India.
[3]Department of Computer Science, Ukrainian National Forestry University, Lviv City, Ukraine.
pp4417@srmist.edu.in[1], tm9045@srmist.edu.in[2], mykhailo.paslavskyi@nltu.edu.ua[3]

Abstract: Intrusions can interrupt network operations, steal critical data, and gain unauthorised access to network resources. Detecting and avoiding network breaches is critical to maintaining network security. Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) monitor network traffic for unusual behavior and respond to threats. Ensemble learning techniques are used to identify cyber intrusions to improve detection accuracy and resilience. The KDD CUP 99 dataset contains a variety of variables collected from cyber network traffic, including duration, byte transfer rates, protocol kinds, and more. Pre-processing stages include categorical variable encoding and feature selection to prepare the dataset for modeling. As a result, Random Forest classifiers serve as foundation learners for bagging and stacking ensemble techniques. The performance of these ensemble models is evaluated using a variety of measures, including accuracy, precision, recall, and F1 score. Furthermore, visualization approaches such as confusion matrices help to analyze categorization performance across different cyber incursion types. This paper uses empirical assessment to demonstrate the usefulness of ensemble learning paradigms in cyber intrusion detection, highlighting their value in improving network security against various cyber threats.

Keywords: Intrusion Detection Systems; Intrusion Prevention Systems; Ensemble Learning Techniques; Random Forest Classifiers; Categorical Variable Encoding; Feature Selection; Performance Evaluation; Cyber Incursion Types.

Received on: 02/04/2023, **Revised on:** 16/07/2023, **Accepted on:** 05/10/2023, **Published on:** 22/12/2023

Cite as: P. Dhinakaran, M.A. Thinesh, and M. Paslavskyi, "Enhancing Cyber Intrusion Detection through Ensemble Learning: A Comparison of Bagging and Stacking Classifiers," FMDB Transactions on Sustainable Computer Letters., vol. 1, no. 4, pp. 210 –227, 2023.

1. Introduction

In today's interconnected world's rapidly evolving digital landscape, the proliferation of technology has brought unprecedented convenience and efficiency. Advancements in technology have fostered a global network where people, organizations, and devices are interconnected. This interconnectedness transcends geographical boundaries, allowing instant communication, collaboration, and data exchange on a global scale. The world has become a closely woven web of digital interactions, enabling seamless connectivity and access to information across continents. The integration of technology into everyday life has brought about unparalleled convenience. Tasks that were once time-consuming and labour-intensive can now be accomplished with a few taps on a screen or clicking a button. Technology has streamlined and simplified numerous aspects of our lives, from online shopping and instant communication to remote work and digital entertainment. The adoption of technology has resulted in significant efficiency gains across various domains. Automation, data analytics, and smart technologies enhance productivity, reduce operational costs, and enable organizations to make informed decisions. Businesses can reach wider audiences, optimize processes, and deliver products and services more efficiently, contributing to economic growth and competitiveness. While the

[*]Corresponding author.

proliferation of technology brings numerous benefits, it also presents challenges, particularly in cybersecurity [13]. As technology becomes more integral to daily life and critical infrastructure, the surface area for potential cyber threats expands. Addressing security concerns and implementing robust measures to protect against cyberattacks and unauthorized access becomes crucial [14].

Cybercriminals are also expanding their strategies; no industry is safe from attacks. As of 2023, ransomware attacks impacted approximately 72% of enterprises globally, according to Statista [15]. This was by far the highest recorded amount, rising over the preceding five years. Since 2018, an average of over 50% of all poll participants have said that ransomware has affected their companies. Big companies are targeted not only by cyber criminals but also by individuals, as it is easier to steal data from them than from a well-organized company. Phishing attacks and ransomware attacks mostly cause cyberattacks on individuals. Attackers often pose as reputable companies like banks, governments, or well-known internet services to send false emails, texts, or phone calls [16]. The intention is to deceive people into divulging private information, such as credit card numbers, login passwords, or personal information. Malicious software is used in ransomware attacks to encrypt data or lock the victim's device, making it unusable [17]. After that, attackers demand money from the victim (often in cryptocurrency) to unlock the data or give them access to the device [18]. It is another common cyber-attack as people download cracked software instead of buying it, making it easier for criminals to steal data using ransomware. People should be cautious when revealing personal information online or answering unsolicited messages or requests [19]. They should also maintain vigilance and exercise basic cybersecurity hygiene. Using strong, one-of-a-kind passwords, staying up to speed on software updates, and being aware of frequent cyber threats can all help reduce the likelihood of becoming a cyberattack victim [20].

Machine learning is critical to cybersecurity in many areas. Machine learning algorithms are particularly good at anomaly detection [21]. They can detect changes in system logs, network traffic, or user behavior that could indicate a security breach. Machine learning also plays an important role in malware detection. It helps security systems and antivirus software detect and stop potential malware by classifying files. Also, machine learning that analyses email header, content, and sender information. Algorithms determine phishing attempts [22]. These algorithms can accurately detect and block suspicious emails using historical data from the phishing email. Also, machine learning is used in user behavior analytics to detect anomalies or suspicious activity within the enterprise network to help identify insider threats or deny access [23]. Moreover, machine learning helps address vulnerabilities by identifying various software and system vulnerabilities in the concept efforts. It supports security teams when prioritized. Machine learning is also useful in security information and event management (SIEM) design. SIEM systems can rapidly identify threats when used in conjunction with security events from multiple sources. Web traffic analysis uses machine learning techniques to analyse traffic patterns and identify abnormal activity, e.g., port scanning or denial of service (DoS) attacks are carried out [24]. These algorithms help prevent cyberattacks by detecting suspicious traffic patterns. In general, machine learning applications in cybersecurity reinforce and force automated and intelligent ways of detecting, assessing, and reacting to cyber threats, strengthening enterprises' security posture against the ever-changing cyber threat landscape [25]. Our project proposes an intrusion detection system (IDS) using machine learning, especially supervised learning processes [26].

The first step of the process involves collecting and pre-processing network traffic data, which involves partitioning the data into training and testing sets, dealing with missing values, and recording categorical variables [27]. Then, we engineer and select the most appropriate features to improve the model's performance. For this project, we used two ensemble learning techniques: a bagging classifier and a stacking classifier [28]. We used two base classifiers in our project. Random Forest classifier is used as a base classifier in the Bagging Classifier to create an ensemble model. Random Forest classifier as a base classifier, and Logistic Regression is the final estimator for the Stacking classifier [29]. The Bagging Classifier with Random Forests as the base classifier helps create an ensemble model that can better handle the complexities of intrusion detection. It involves training multiple instances of a base model (Random Forest model) on different subsets of the training data. The model then combines to make predictions, and averages or votes determine the final results. This approach helps improve overall model performance by reducing variance and increasing robustness, making it particularly useful in situations with complex decision boundaries or noisy data. The stacking classifier further improves the system's performance by combining the strengths of the two models [30]. The base models (random forests and logistic regression) each make individual predictions, which are then used as input features to the meta-model [31]. Then, the meta-model learns to combine these predictions to build the final decision. Stacking works well because it allows models to take advantage of the different strengths of different base models, which can improve performance compared to any one model [32].

Once the model is trained and tested, it can be deployed in a manufacturing environment to monitor network traffic continuously. Ensemble learning techniques such as bagging and boosting can be used to combine multiple models to improve performance [33]. In addition, stacking can be used to train meta-models on the results of a multi-basis model, further increasing performance. Our approach uses machine learning techniques to build a robust IDS that effectively detects and responds to cyber threats, enhancing network security [34]. By leveraging IDS with machine learning techniques, this project aims to enhance network security by developing a method to detect and respond to cyber threats [35]. Using the power of machine learning, this IDS can adapt to evolving threats, protect against unauthorized access and data breaches, and ultimately enhance the security posture of individuals and organizations in the digital age [36].

2. Objective

- The goal of developing an intrusion detection system (IDS) using machine learning is to detect and respond to cyber threats. This includes collecting and pre-processing network traffic data, selecting its technology components, training machine learning models, and evaluating their performance.
- The project aims to build a robust IDS that can accurately distinguish between normal and attacker network traffic, enhancing network security through a proactive approach to identify and mitigate cyber threats.
- Implementing an IDS at the manufacturing site will enable real-time detection of installations, enabling immediate response and mitigating cyber threats. Furthermore, the system's adaptability to new threats and its large enough size to handle a lot of web traffic will ensure effective protection against incoming attack patterns.

3. Literature Survey

Wang et al. [1] discussed the use of an early warning model in power systems to identify Cyber-Physical Cascading Attacks (CFCA). It gathers information from communication and control networks using a bypass mirror mode. The model is tested in a simulated environment with a power plant, smart substation, and IEEE 39-bus system.

Sommer and Paxson [2] discuss the challenges of applying machine learning in practical settings as you talk about how these approaches are utilized for network intrusion detection. They provide a framework for incorporating machine learning into intrusion detection systems and emphasize the importance of adapting machine learning models to account for the dynamic nature of network traffic.

Pu et al. [3] discuss utilizing the One-Class Support Vector Machine and Sub-Space Clustering-Based Hybrid Unsupervised Clustering-Based Anomaly Detection Method for efficient cyberattack detection. The approach performs better than current methods, as experimental findings on the NSL-KDD network attacks dataset show.

Al-Abassi and Karimipour [4] present an ensemble-based deep learning model for detecting industrial control systems (ICS) cyberattacks. Creating new balanced representations corrects the imbalance found in ICS datasets. In actual ICS datasets, the model performs better than conventional classifiers with higher F1 scores and accuracy.

Tufan et al. [5] discussed using machine learning models to identify probing assaults on a network used by a university and implement improved cybersecurity. Unlike popular public datasets, the models were trained using actual data from the institutional network, providing a more realistic approach. They used feature visualization methods such as saliency maps and t-SNE to understand the behavior of the data.

Loukas et al. [6] discuss a deep learning-based cloud-based cyber-physical intrusion detection system for cars. It emphasizes how improving intrusion detection skills through computing offloading might be beneficial. The authors illustrate the efficiency of deep learning in ongoing vehicle intrusion detection.

Alhakami et al. [7] discuss anomaly-based intrusion detection systems' shortcomings and suggest a nonparametric Bayesian method for enhancement. The approach incorporates feature selection to improve detection accuracy and lower false alarms. The paper aims to discuss the complexity of cyber threats in the context of smart cities. The model provides a more reliable intrusion detection solution by combining feature selection with Bayesian inference. This research facilitated the advancement of cybersecurity protocols in smart city applications.

Vinayakumar et al. [8] discuss the application of deep learning in developing an Intelligent Intrusion Detection System (IDS). It uses machine-learning approaches to address IDS development issues. The deep learning method improves cyberattack detection efficiency and accuracy. The study's results emphasize how critical it is to comprehend the phases of attacker activity.

Siddiqi and Pak [9] discuss an optimized framework for network intrusion detection based on image processing to enhance network security. The approach reduces features and turns non-image data into images for efficient anomaly detection by fusing augmented feature selection with image enhancement and transformation.

Tummala and Inapakurthi [10] present a two-stage Kalman Filter for Automatic Generation Control Systems Cyber-attack Detection. The filter is intended to improve cyber-security by identifying and reducing different types of cyber-attacks, including DoS, FDI, DRA, scaling, and ramp attacks. Benchmark power system model simulation results show how well the OTS-KF estimates cyberattacks and boosts system resilience. This paper addresses the increasing challenges cyber threats pose in modern power systems undergoing significant changes due to renewable energy integration.

Chen et al. [11] discuss a modified Q-learning approach for cyber-physical systems (CPS) to address unequal cost challenges in defense strategy selection. This paper aims to improve efficiency and reduce costs compared to traditional approaches that assume equal costs for misclassification errors.

Raghunath et al. [12] discuss the development of an XGBoost Regression Classifier (XRC) model for cyber-attack detection and classification using Inception V4. The study showcases the effectiveness of machine learning techniques in improving cyber-attack detection accuracy.

4. Proposed Methodology

4.1. Ensemble Techniques

Ensemble approaches are a machine learning strategy combining different models to increase system performance. The premise underlying ensemble approaches is based on the assumption that integrating numerous models may compensate for individual model deficiencies while maximizing their strengths. It uses the notion of "wisdom of the crowd," which states that collecting various viewpoints frequently results in better conclusions than depending on a single source. Homogeneous Ensembles are made up of numerous instances of the same basic model. Techniques like bagging and boosting come within this group. Heterogeneous ensembles integrate basic models, including decision trees, neural networks, and support vector machines. One important component of ensemble learning is the variety of the base models. Diversity guarantees that the models make diverse errors so that their strengths and weaknesses complement one another when coupled. [37] Diversity may be produced in various ways, including employing alternative methods, subsets of the training data, or feature representations [38]. Ensemble approaches aggregate basic model predictions using various aggregation strategies, including averaging, voting, and weighted averaging [39].

The aggregation technique chosen is determined by the type of the issue (classification, regression, etc.) and the base model parameters. Bagging is the process of training numerous instances of the same learning algorithm on different subsets of the training data (sampled with replacement) and then combining their predictions, frequently done by averaging or voting [40]. It reduces overfitting by decreasing the variance of the model. Random forests are a decision tree-based ensemble approach in which numerous trees are generated on various subsets of data and then blended via averaging or voting [41]. They provide randomization to the tree-building process to improve variety and avoid overfitting. Stacking entails developing a meta-model that learns to aggregate the predictions of numerous base models [42]. Unlike basic aggregation approaches, stacking determines the best way to aggregate predictions based on their performance on a validation set [43]. Ensemble approaches are frequently employed in many disciplines because they result in more resilient and accurate models than individual ones [44].

However, they add computational complexity and may necessitate more comprehensive hyperparameter tweaking. Ensemble approaches are more resistant to noise and outliers in data [45]. Because they consider several models, they are less susceptible to random fluctuations or inaccuracies in individual forecasts. Ensemble approaches can assist in preventing overfitting, particularly in complicated models [46]. Ensemble approaches can improve the generalization of new data by pooling predictions from several models and avoiding memorizing noise in training data. Ensemble approaches are often more stable than individual models [47]. They are less sensitive to tiny changes in the training data or model parameters, allowing for more consistent results across datasets. Ensemble techniques are capable of properly handling complicated data linkages and patterns. Ensemble approaches combine several models and may capture various properties and interactions, making them suited for jobs involving high-dimensional or nonlinear data [48]. Many ensemble methods are easy to install and need little adjustment. Techniques such as bagging and boosting have well-established algorithms and are supported by major machine-learning libraries, making them available to practitioners [49]. The machine learning model utilised in this paper is a hybrid of ensemble learning approaches, especially Bagging and Stacking, with Random Forest and Logistic Regression serving as basis classifiers [50].

4.2. Bagging Classifier

"Bagging" stands for Bootstrap Aggregating. It is an ensemble meta-algorithm that boosts the stability and accuracy of machine learning algorithms. It operates by randomly choosing and replacing sections of the original dataset. Then, a basic classifier is applied to each subgroup [51]. In this paper, the underlying classifier for bagging is a Random Forest Classifier, a decision tree-based ensemble learning approach. The Bagging process involves the following steps:

- Base Classifier Selection: First, a base classifier is chosen. In this situation, the basis classifier is RandomForestClassifier.
- Bagging Classifier Initialization: The Bagging Classifier is created using the basis classifier of your choice. In addition, parameters such as the number of estimators and the random state are defined. The number of estimators indicates the number of basic classifiers (Random Forests) to train.
- Training the Bagging Classifier: After initialization, the BaggingClassifier is trained using the training data (X_train, Y_train). During training, numerous subsets of training data are generated using random sampling with replacement (bootstrap sampling). Each subset trains an individual Random Forest classifier [52].
- Prediction: Following training, the BaggingClassifier generates predictions on test data (X_test). Each base classifier (Random Forest) predicts for each test data instance. A majority vote of all base classifier predictions then selects the final prediction [53].
- Finally, the performance of the BaggingClassifier is assessed using metrics such as accuracy, classification report, and confusion matrix. These metrics explain how successfully the BaggingClassifier classifies instances into distinct attack types [54].
- Overall, the Bagging process in this method consists of training several Random Forest classifiers on bootstrap samples of training data and aggregating their predictions to enhance overall classification performance.

4.3. Stacking Classifier

Stacking, or Stacked Generalisation, is another ensemble learning approach. Instead of simple majority voting, like Bagging, Stacking employs a meta-learner to determine best how to combine the basic learners' predictions. In this paper, the basic classifiers used for stacking are Random Forest and maybe more, as noted by the remark, “Add more base classifiers as needed”. The last meta-estimator for Stacking is Logistic Regression. The Stacking process involves the following steps:

- Base Classifier Selection: First, a set of base classifiers is established. In this paper, RandomForestClassifier is used as the basis classifier. You may add more base classifiers to this list.
- Stacking Classifier Initialization: The StackingClassifier is created using the list of basic classifiers as ‘estimators’ and a final estimator. The final estimator is another classifier that uses the basic classifiers’ outputs as input to create a final prediction. In this paper, Logistic Regression is utilised as the final estimator.
- Training the Stacking Classifier: After initialization, the StackingClassifier is trained using training data (X_train and Y_train). Each base classifier in the 'estimators' list is trained using the training data during training.
- Getting basic Classifier Predictions: After training the basic classifiers, predictions are created on the validation data (X_test) using each base classifier.
- Final Prediction: The basic classifier's predictions are utilised as features for the final estimator (Logistic Regression). The final estimator uses these predictions as input to produce a final prediction.
- Finally, the performance of the StackingClassifier is assessed using measures such as accuracy, classification report, and confusion matrix. These metrics explain how successfully the StackingClassifier classifies instances into various attack types [55].
- To summarise, the stacking method in this project consists of training several base classifiers, utilizing their predictions as features, and then training a final estimator to generate a final prediction [56]. This approach seeks to integrate the capabilities of several base classifiers and enhance overall classification performance [57].
- Bagging and stacking methods are used for classification problems. After training both classifiers, the algorithm assesses their performance using accuracy, precision, recall, and F1-score measures. It also displays both models' confusion matrices and classification reports, comparing their performance across various attack types [58]. In addition, the algorithm checks the accuracy of two classifiers to see which performs better [59].

4.4. Algorithm:

Data Pre-processing:

- Load the dataset.
- Define column names.
- Map attack types to their corresponding categories.
- Visualize the distribution of attack types and protocol types.
- Handle missing values and drop irrelevant columns.

- Encode categorical variables into numerical representations.

Model Training with Bagging Classifier:

- Split the dataset into training and testing sets.
- Initialize a Bagging Classifier with a Random Forest Classifier as the base estimator.
- Train the Bagging Classifier on the training data.
- Make predictions on the testing data.
- Evaluate the performance of the Bagging Classifier using metrics such as accuracy, classification report, and confusion matrix.

Model Training with Stacking Classifier:

- Split the dataset into training and testing sets.
- Define a list of base classifiers (Random Forest Classifier).
- Initialize a Stacking Classifier with the list of base classifiers and a final estimator (Logistic Regression).
- Train the Stacking Classifier on the training data.
- Make predictions on the testing data.
- Evaluate the performance of the Stacking Classifier using metrics such as accuracy, classification report, and confusion matrix.

Visualize Feature Relationships:

- Plot a pair plot of selected features to visualize their relationships.

Performance Comparison:

- Compare the performance of Bagging and Stacking classifiers using metrics such as precision, recall, F1-score, and accuracy.
- Visualize the performance metrics using bar graphs.

Summary:

- Provide a summary of the performance comparison between Bagging and Stacking classifiers.

4.5. Support Vector Machine

Support Vector Machine is a supervised machine learning system that performs classification and regression problems. SVM is successful in high-dimensional spaces and can handle linear and nonlinear datasets. SVM works by dividing classes using a decision boundary. The method of separation is determined by the kernel function used. After preparing the data and selecting the kernel, the SVM model is trained using labeled training data. During training, the algorithm determines the parameters (weights and bias) of the decision boundary that optimally divides the classes in the feature set [60]. Hyperparameter selection can affect SVM performance in areas such as the regularisation parameter, kernel type, and kernel-specific parameters. Choosing proper settings for these parameters is critical to getting acceptable performance, and it frequently necessitates thorough tweaking using techniques like grid search or cross-validation [61]. Ensemble methods may handle unbalanced datasets better than SVMs by assigning appropriate weights to distinct classes or employing strategies such as resampling during training [62]. SVM may be computationally costly, particularly for big datasets, because of its quadratic or cubic time complexity to sample count [63]. Ensemble approaches, while computationally costly, maybe more scalable with bigger datasets owing to parallelization.

4.6. Execution

Bagging Classifier:

- Initializes a base classifier (RandomForestClassifier) and a Bagging Classifier using this base classifier with ten estimators.
- The Bagging Classifier is trained on the training data (X_train, Y_train) using the fit() method.
- Predictions are made on the test data (X_test) using the trained Bagging Classifier.
- Accuracy and classification reports are computed for the Bagging Classifier using accuracy_score and classification_report.

Stacking Classifier:

- Like Bagging, Stacking initializes multiple base classifiers (RandomForestClassifier) and defines a Stacking Classifier using these base classifiers with a final estimator (LogisticRegression).
- The Stacking Classifier is trained on the training data (X_train, Y_train) using the fit() method.
- Predictions are made on the test data (X_test) using the trained Stacking Classifier.
- Accuracy and classification reports are computed for the Stacking Classifier using accuracy_score and classification_report.

4.7. Architecture Diagram

Figure 1 depicts the sequence of activities and steps for network intrusion detection using ensemble techniques. The dataset utilized in this study is known as the "KDD Cup 1999 Data," a benchmark dataset extensively used to evaluate intrusion detection systems. It simulates a computer network environment where several assaults are introduced into typical network traffic. The dataset has a predefined set of features that describe network connections [64]. These capabilities record different features of network traffic behavior, including duration, protocol type, service, flag, bytes transported, and more [65]. The heat map is used for feature selection, and certain features are dropped during pre-processing based on correlation analysis.

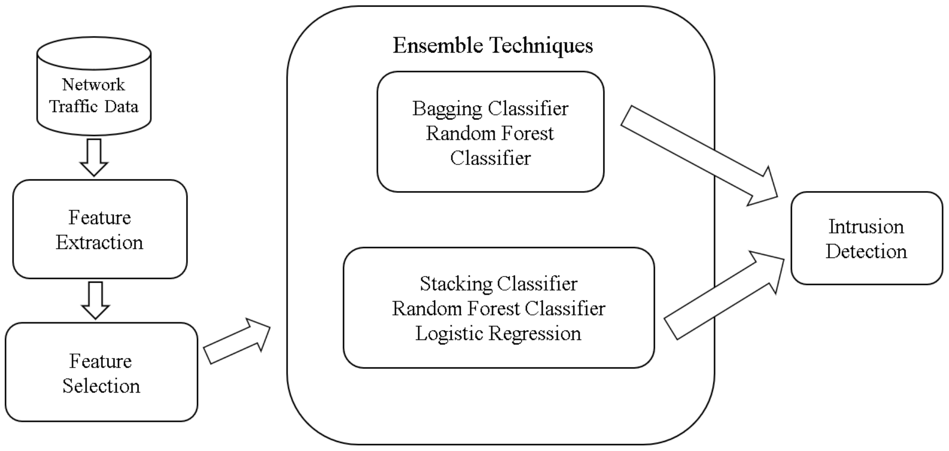
Network Traffic Data
Feature Extraction
Feature Selection
Ensemble Techniques
Bagging Classifier Random Forest Classifier
Stacking Classifier Random Forest Classifier Logistic Regression
Intrusion Detection

Figure 1: Architecture Diagram for Ensemble Techniques

Ensemble Techniques such as the bagging classifier and stacking classifier are used. A RandomForestClassifier is chosen as the base estimator for the bagging classifier. Multiple base classifiers (RandomForestClassifier) are defined as a list of tuples. These base classifiers are then used to train a stacking classifier with a final estimator (LogisticRegression). Overall, the paper's architecture follows a structured approach, starting from data pre-processing and feature engineering, followed by model training, evaluation, and visualization of results to gain insights into cyber intrusion detection.

5. Implementation

5.1. Data and Pre-processing

The data is imported from a CSV file into a Pandas DataFrame, and each column is properly labeled with specified column names. Next, basic data inspection is performed to understand the dataset's structure and distribution, which includes checking for missing values and analyzing the distribution of target classes. Data cleaning methods are then applied, such as deleting columns with missing values and preserving only columns with more than one unique value to assure data integrity. Furthermore, categorical variables are handled by translating them into numerical representations. Feature selection is used to remove strongly correlated features, hence reducing redundancy and improving model performance. Finally, the data is separated into input features and target variables, with the input features scaled using Min-Max scaling to guarantee that all characteristics contribute equally to model training. Overall, these pretreatment stages are critical for ensuring that the dataset is in a proper format for model training. This improves the efficacy and accuracy of the machine learning models used on the data.

5.2. Data Visualization

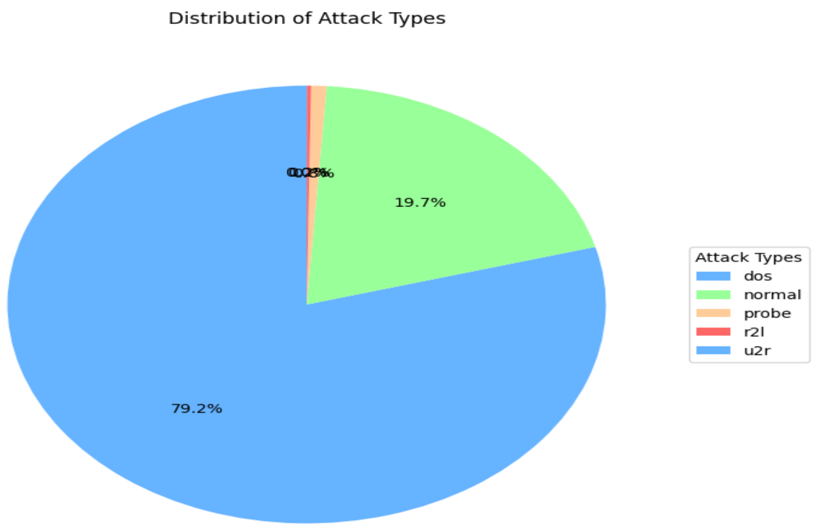

Figure 2: Pie Chart

Figure 2 depicts the distribution of attack types within the dataset. It begins by determining the number of attack types in the data. Each slice of the pie represents a distinct attack type, and its size indicates the fraction of that kind in the dataset. The chart's percentages also give a quantitative depiction of the relative frequency of each assault type. The chart is supplemented with stylistic features such as colour schemes and a starting angle for the first wedge. A legend is added to make the chart more interesting, and the attack-type labels provide context for each slice. For clarity, the legend is placed to the left of the chart. Overall, the pie chart provides a visual description of the distribution of attack types, making it easier to understand the dataset's composition.

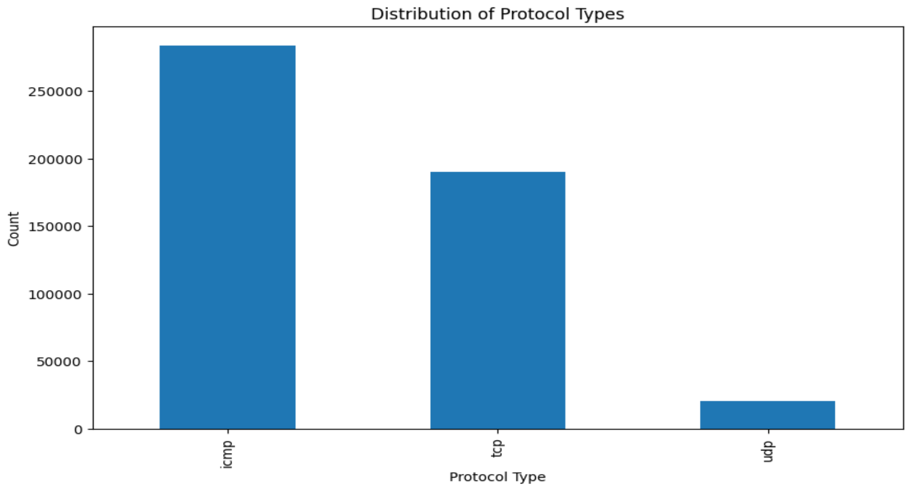

Figure 3: Bar Chart of Distribution of Protocol Types

Figure 3 depicts the distribution of protocol types. Initially, the script obtains the dataset's 'protocol_type' column and calculates the number of occurrences for each unique protocol type. This function is set with kind='bar' to generate a bar chart. The x-axis displays the numerous protocol types, while the y-axis shows the count of occurrences for each protocol. This visualization assists in determining the prevalence of various protocols within the dataset, provides insights into network traffic patterns, and allows for a quick assessment of protocol usage frequencies.

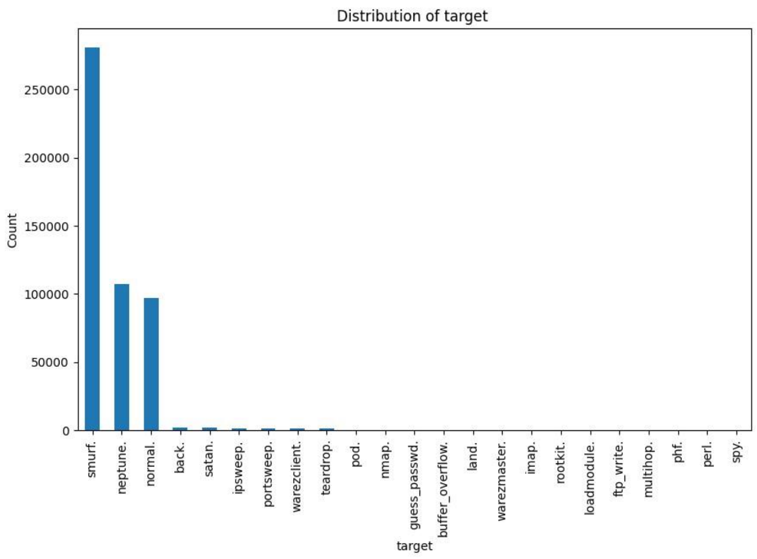

Figure 4: Bar Chart of Distribution of Target Types

Using a bar chart, figure 4 depicts the distribution of the target variable, which is likely to reflect distinct groups or categories in the dataset. Initially, the script retrieves the 'target' column from the dataset and computes the frequency of occurrence for each target value. A bar chart is created using the target counts as the data input. The function's kind='bar' parameter specifies the production of a bar chart. As a result, the x-axis displays the various target categories, while the y-axis shows the number of occurrences for each category. By producing this visualization, the script provides insight into the distribution of classes within the dataset, which aids in evaluating the balance or imbalance between various goal categories.

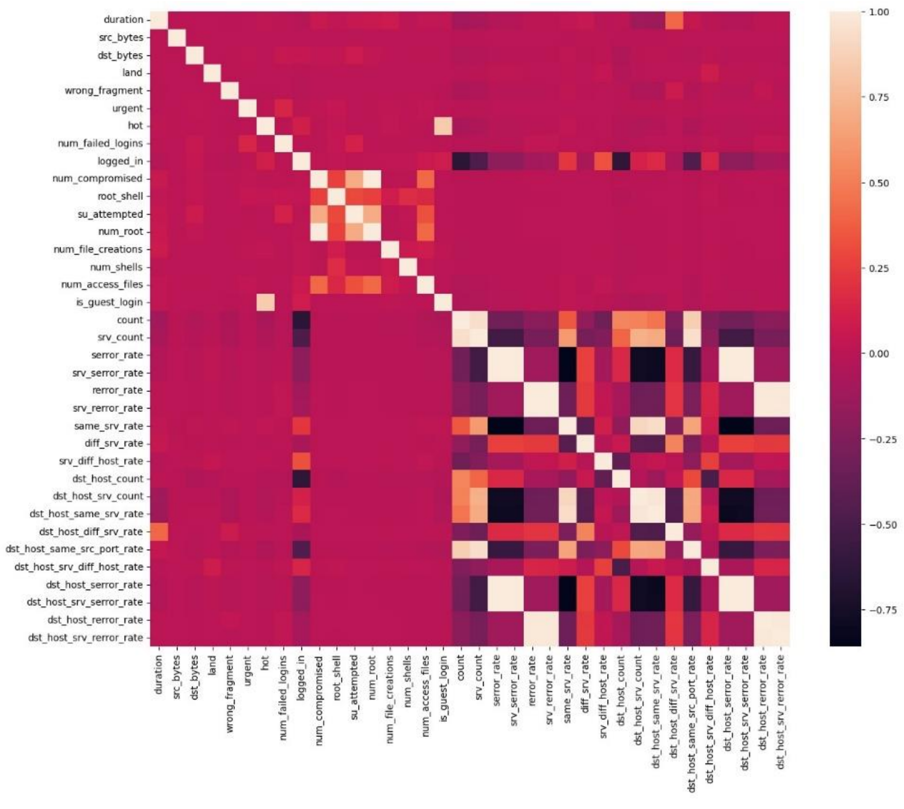

Figure 5: Heat Map

Figure 5 is created to visualize the dataset's correlation matrix. This heatmap provides a complete picture of the correlations between various pairs of characteristics by showing the correlation coefficients as colors.

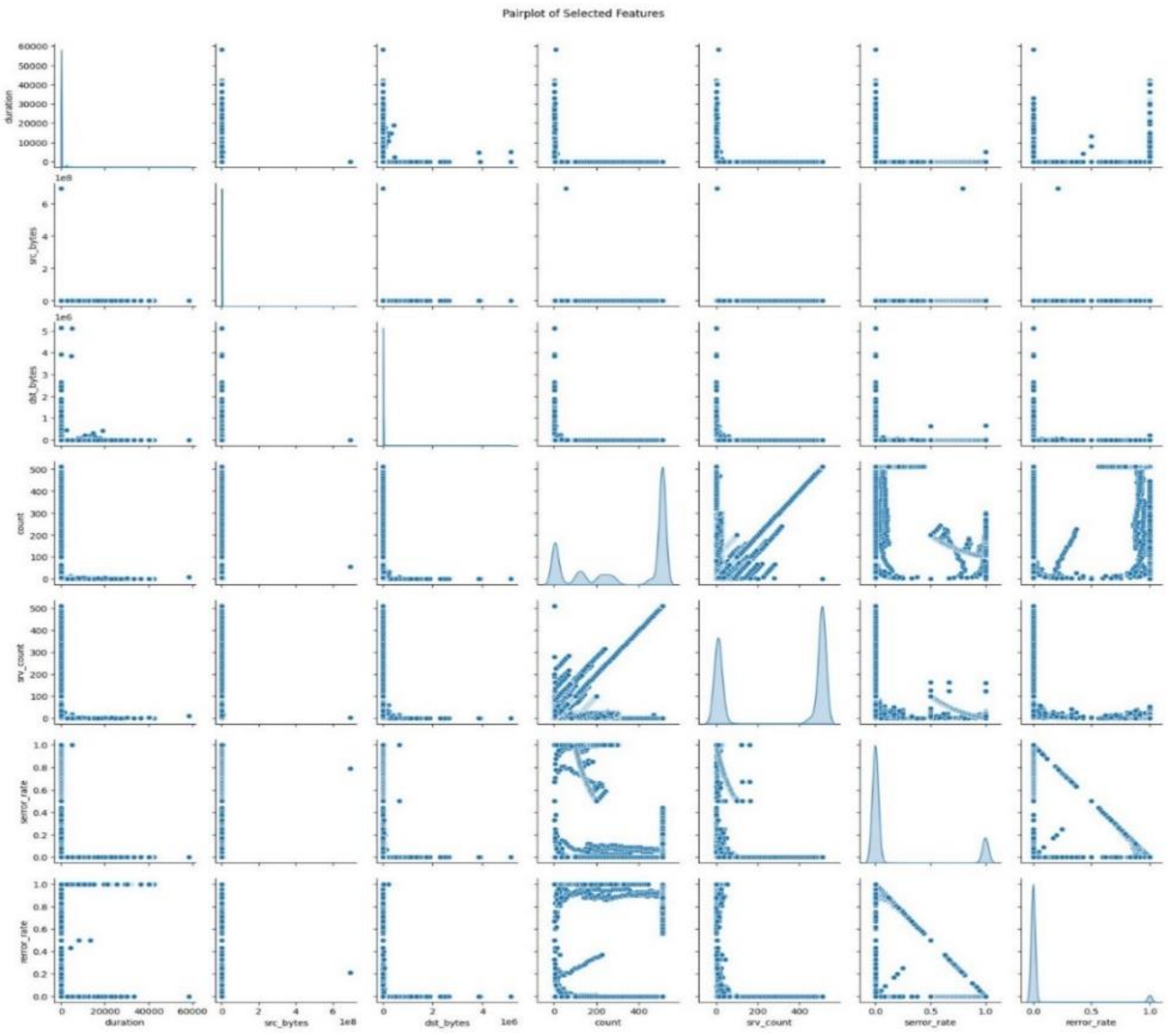

Figure 6: Pair Plot

Calculating Correlation Matrix: The script generates the correlation matrix using the Pandas DataFrame's corr() function, which computes the pairwise correlation coefficients between all numerical columns in the dataset.

Creating the Heatmap:

- The correlation matrix provides input data for the heatmap.
- The heatmap function automatically converts the correlation coefficients to a color scale, with different colours denoting varying correlation strength.
- Warmer colours (e.g., red) often imply more positive correlations, whereas cooler colours (e.g., blue) represent higher negative correlations.

Interpretation:

- The generated heatmap visualizes the dataset's correlation structure.
- Darker tones indicate higher relationships, whereas lighter shades suggest weaker or absent associations.
- Examining the heatmap may determine whether pairings of attributes are favorable, adverse, or not connected.

This visualization helps with feature selection, multicollinearity detection, and comprehending possible variable interactions, which are important for model construction and interpretation in machine learning tasks.

Figure 6 is created to visualize the associations between selected feature pairs from the dataset, also known as a scatterplot matrix. Initially, a selection of characteristics is selected for visualisation, including 'duration,' 'src_bytes', 'dst_bytes', 'count','srv_count','serror_rate', and'rerror_rate'.This function creates a grid of scatterplots, with each feature plotted against every other feature. In addition, the grid's diagonal depicts each feature's distribution, commonly represented by kernel density estimation (KDE) curves or histogram. Examining these graphs provides insights into the dataset's relationships, trends, and probable outliers. Pair plots are useful tools for exploratory data analysis, helping to identify correlations between variables and directing further data pretreatment and modelling processes in machine learning workflows.

5.3. Training

Two ensemble classifiers, Stacking Classifier and Bagging Classifier, are trained for a network intrusion detection job.

Data Splitting: This example separates the dataset into features (X) and the target variable (Y), the 'Attack Type' column. The dataset is then split into training and testing sets using sci-kit-learn's train_test_split() method. This allows us to evaluate the trained models' performance on previously unknown data.

Bagging Classifier Training: A basic classifier (RandomForestClassifier) is selected. The Bagging Classifier is created using scikit-learn's BaggingClassifier() method, which combines many basic classifiers. The Bagging Classifier is trained on the training data via the fit() technique.

Stacking Classifier Training: Training the Stacking Classifier involves defining many base classifiers in a list. The Stacking Classifier is created using sci-kit-learn's Stacking Classifier() method, which mixes predictions from numerous basic classifiers. The Stacking Classifier is trained on the training data via the fit() function.

5.4. Model Evaluation

After both classifiers have been trained, performance measures such as accuracy, precision, recall, and F1-score are calculated to evaluate the classifiers' performance.

Visualization of Results: Confusion matrices and classification reports are visualised with Seaborn and matplotlib to offer a complete picture of the classifiers' performance on the test data. Heat maps of the confusion matrices may be used to show the distribution of accurate and wrong predictions across classes. Bar graphs or pie charts can compare the two classifiers' performance measures (e.g., accuracy, precision, recall, F1-score).

Comparison of Performance Metrics: Bar graphs or pie charts can compare the two classifiers' performance measures (e.g., accuracy, precision, recall, F1-score). This enables a visual comparison of the classifiers' performance categorizing various network intrusion assaults.

6. Result and Discussion

The proposed model was run and evaluated on Windows 11 with Ryzen 7 5800 X, 32GB RAM, and RTX 3060ti GPU and re-evaluated on the Google Colab. The dataset has been trained and tested using the Ensemble Techniques. The dataset is split into 80% and 20% for training and testing. Metrics like precision, recall, F1-score, and accuracy are used to evaluate the effectiveness of the Ensemble Techniques. The same dataset was trained with the SVM model, and the same metrics as the Ensemble Techniques were evaluated for the SVM model. The results are:

Table 1: Comparison of SVM and Ensemble Techniques

Model	Accuracy	Average_Precision	Average_Recall	Average_F1-score
SVM	99.87%	93%	89%	91%
Bagging Classifier	99.96%	98%	94%	96%
Stacking Classifier	99.97%	97%	94%	96%

Table 1 clearly illustrates the difference in metrics for the Ensemble Techniques and the SVM model. The same dataset was trained, but there is a difference between the metrics, which clearly shows that the Ensemble Technique is best suited and makes accurate predictions for the loaded dataset.

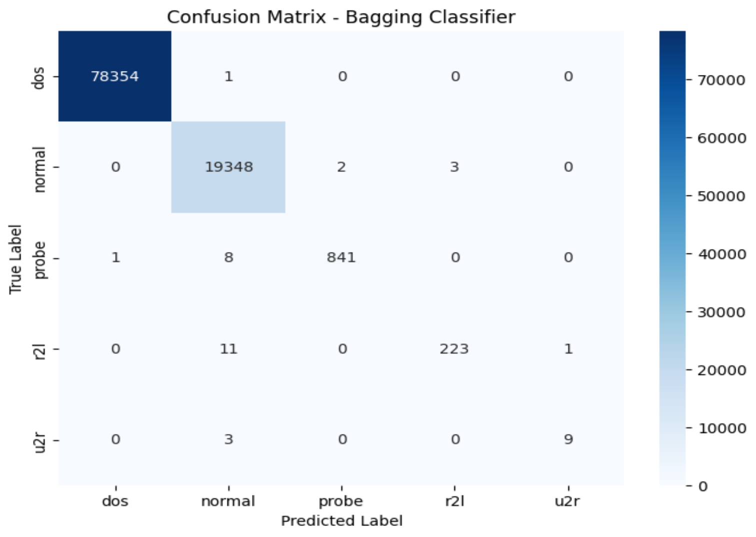

Figure 7: Confusion Matrix of Bagging Classifier

Figure 7 shows how well it predicts distinct classes (in this example, different sorts of network intrusion assaults) compared to the test dataset's actual labels. It consists of a grid with rows representing the actual classes and columns representing anticipated classes. The bagging classifier's confusion matrix visualizes how well the classifier performed in classifying network intrusion attacks into different categories, such as 'dos' (Denial of Service), 'normal' (No Attack), 'probe' (Probing), 'r2l' (Unauthorised Access from Remote to Local), and 'u2r' (Unauthorised Access to Root). The confusion matrix entries indicate the classifier's true positive (TP), false positive (FP), true negative (TN), and false negative (FN) predictions for each class.

- True Positive (TP): Instances where the classifier correctly predicted a certain attack type when it was present.
- False Positive (FP): Instances where the classifier incorrectly predicted a certain attack type when it was not present.
- True Negative (TN): Instances where the classifier correctly predicted the absence of a certain attack type.
- False Negative (FN): Instances where the classifier incorrectly predicted the absence of a certain attack type when it was present.

One may evaluate the classifier's accuracy, precision, recall, and other classification metrics by reviewing the confusion matrix values. It aids in determining which attack types the classifier frequently misclassifies and provides insights into model development opportunities. Furthermore, visualizing the confusion matrix provides rapid and straightforward insight into the classifier's performance across classes.

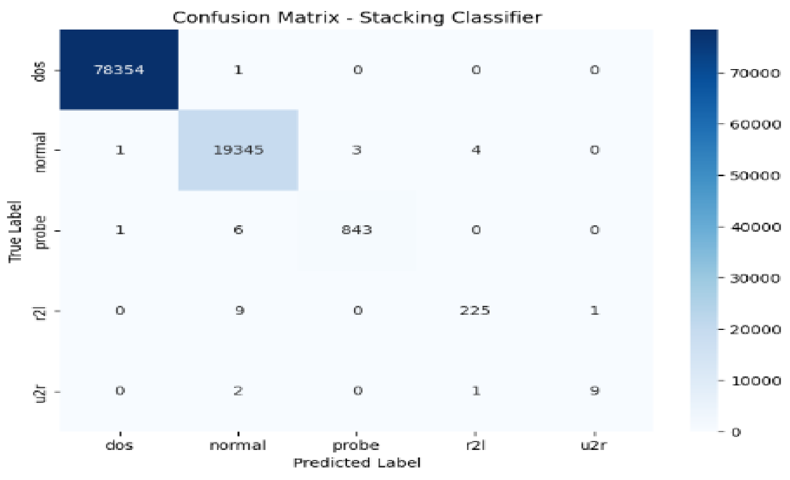

Figure 8: Confusion Matrix of Stacking Classifier

Figure 8 shows how well it predicts different classes (in this case, different types of network intrusion attacks) compared to the actual labels in the test dataset. It is a grid, similar to the confusion matrix of the bagging classifier, with rows representing the actual classes and columns representing the anticipated classes. The stacking classifier's confusion matrix visualises how well the classifier performed in classifying network intrusion attacks into different categories, such as 'dos' (Denial of Service), 'normal' (No Attack), 'probe' (Probing), 'r2l' (Unauthorised Access from Remote to Local), and 'u2r' (Unauthorised Access to Root). Similarly, the confusion matrix entries indicate the number of true positive (TP), false positive (FP), true negative (TN), and false negative (FN) predictions given by the stacking classifier for each class. Analyzing the numbers in the confusion matrix allows one to assess the classifier's accuracy, precision, recall, and other classification metrics. It assists in determining which attack types the stacking classifier frequently misclassifies and gives insights into model development opportunities. Furthermore, visualising the confusion matrix enables a rapid and intuitive assessment of the classifier's performance across multiple classes, which aids decision-making and model modification.

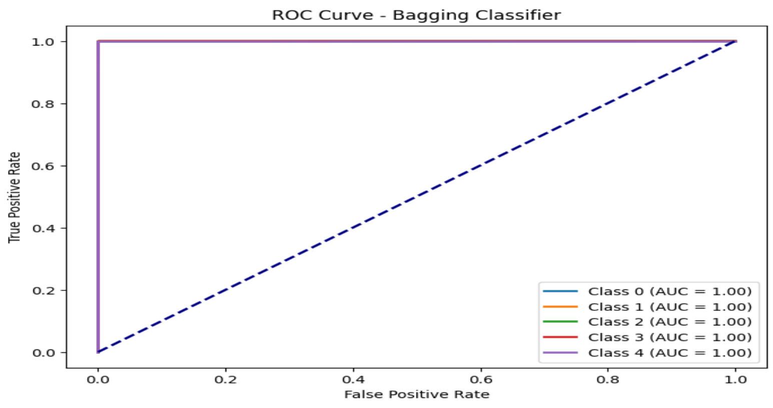

Figure 9: ROC curve for Bagging Classifier

Figure 9 is a graphical depiction of the classifier's performance at various thresholds for binary classification issues. However, because the problem requires multi-class classification, the ROC curve may not be immediately relevant. In binary classification, the ROC curve compares the true positive rate (sensitivity) against the false positive rate (1-specificity) at different thresholds. Each point on the curve indicates a sensitivity-specificity pair assigned to a specific threshold. The area under the ROC curve (AUC-ROC) measures the classifier's overall performance, with a greater AUC suggesting better discrimination between positive and negative classifications. In multi-class classification, alternative evaluation measures like accuracy, recall, F1-score, and confusion matrices are commonly used to analyze the classifier's performance across several classes. These metrics give information about the classifier's ability to accurately categorise examples within each class and highlight areas for development. As a result, while the ROC curve is a useful tool for assessing binary classifiers, it may not be easy to apply to multi-class classification issues. Instead, various multi-class classification-specific evaluation metrics and methodologies are more suited to analyzing the bagging classifiers's performance.

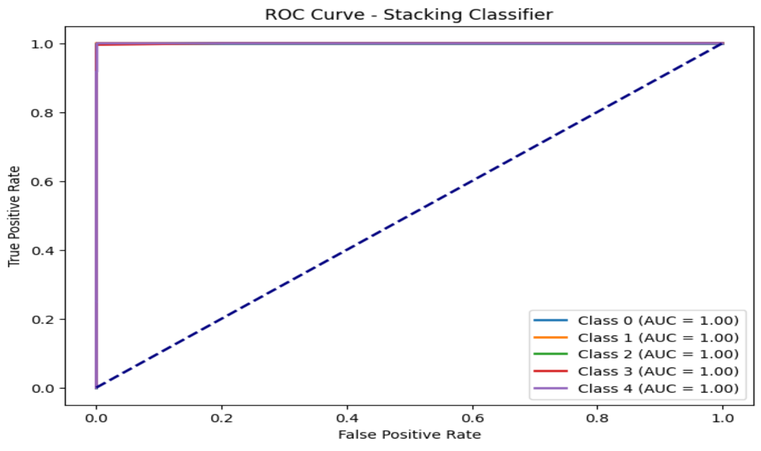

Figure 10: ROC curve for Bagging Classifier

Figure 10 is a graphical depiction of the classifier's performance at various thresholds for binary classification issues because producing a ROC curve for a multi-class classifier such as the stacking classifier may not be simple. One technique of visualising the performance of a multi-class classifier, such as stacking, is to employ a One-vs-Rest strategy. Here's a step-by-step approach to visualize the ROC curve for the stacking classifier using the OvR strategy:

Binary Transformation: Using the OvR technique, break down the multi-class issue into several binary classification subproblems. Consider each class as positive and all other classes as negative.

Predict Probabilities: Using the stacking classifier, forecast the probability of being in the positive class for each binary classification subproblem.

Compute ROC Curve: Compute the ROC curve for each binary classification subproblem using the predicted probabilities.
Plot ROC Curves: Plot each ROC curve on the same plot to see how the stacking classifier performs across classes.

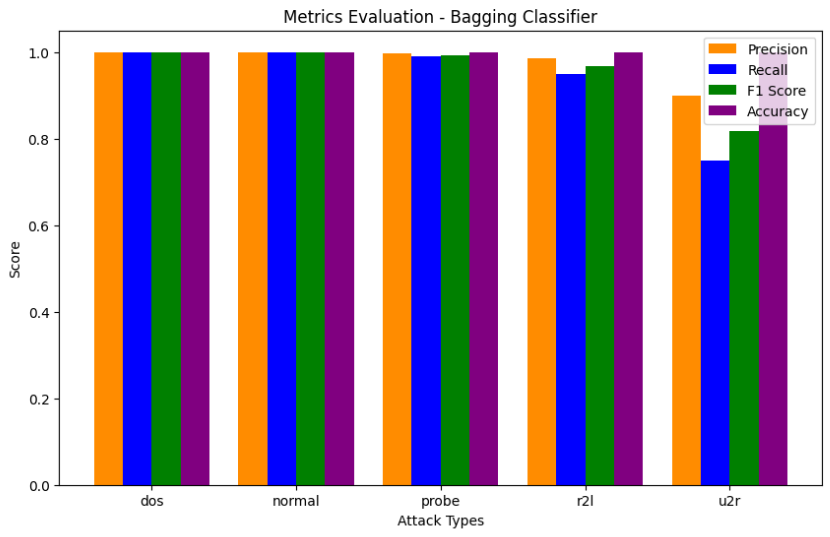

Figure 11: Metrics Evaluation of Bagging Classifier

In a multi-class classification task, Figure 11 usually comprises bar graphs that indicate several evaluation metrics, including precision, recall, F1 score, and accuracy for each class.

Precision refers to the fraction of real positive predictions among all instances projected as positive. It is the ratio of genuine positives to the sum of true positives plus false positives. In a multi-class problem, accuracy is calculated separately for each class.
Recall: Recall, also known as sensitivity or true positive rate, calculates the proportion of genuine positive forecasts among all real positive cases. It is the ratio of genuine positives to the sum of true positives plus false negatives. In a multi-class situation, recall is calculated separately for each class, much like accuracy.

F1 Score: The F1 score is the harmonic mean of accuracy and recall, resulting in a balance of the two measures. It is computed as 2 × (precision * recall) divided by (precision + recall). In a multi-class context, the F1 score is calculated independently for each class, just as accuracy and recall.

Accuracy: Accuracy assesses the overall accuracy of the classifier's predictions across all classes. It is determined as the proportion of properly categorized examples to the total number of instances.

The metrics evaluation graph for the bagging classifier depicts each evaluation metric as a bar graph, with each bar representing a distinct class in the multi-class issue. The x-axis normally depicts the various classes, and the y-axis indicates the value of the evaluation measure. Each bar's height represents the metric's value for the related class. This graph presents a full summary of the bagging classifier's performance across many classes, allowing for easy comparison of each class's precision, recall, F1 score, and accuracy. It aids in determining which classes the classifier excels at and which require more work.

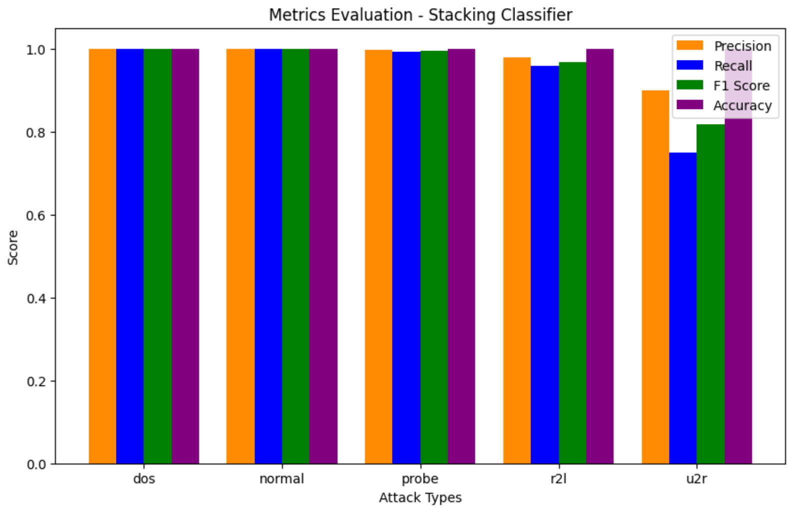

Figure 12: Metrics Evaluation of Stacking Classifier

Figure 12 depicts important evaluation metrics such as precision, recall, F1 score, and accuracy over several classes in a multi-class classification situation. Each measure is depicted as a bar graph, with each bar representing a distinct class in the classification task. The x-axis shows the various classes, while the y-axis reflects the values of the assessment metrics. Higher precision, recall, and F1 scores imply superior performance, but accuracy values close to 1.0 suggest correct classification across all classes. Examining the heights of the bars for each statistic allows one to analyze the classifier's performance for each class rapidly. Comparing metrics across classes sheds light on the classifier's strengths and limitations, indicating where performance may be poor and regions where it excels. This visualization helps you see the classifier's overall performance and ability to categorise cases across classes accurately. It also aids in detecting any imbalances or biases in the classifier's predictions across classes, allowing for targeted modifications that increase overall classification accuracy and efficiency.

7. Conclusion

Using random forest classifiers and stacking classifiers for intrusion detection has shown promising results in detecting and classifying network intrusions, including cyber-attacks. The Random forest classifier showed high accuracy in detecting fraudulent transactions. We were able to develop an effective IDS capable of detecting network attacks. We used ensemble techniques such as bagging and stacking classifiers. After evaluation with accuracy, average precision, average F1-score, and average recall. The bagging classifier got results (accuracy=99.96%, average precision=98%, average f1-score=96% and average recall=94%) and the stacking classifier got results (accuracy=99.97%,average precision=97%, average f1-score=96% and average recall=94%).The use of ensemble techniques proved to improve the overall performance of the IDS. Further research and development in machine learning and intrusion detection systems will continue to advance the industry, providing better security solutions to protect communications from cyber-attacks and threats.

Acknowledgment: The support of all my co-authors is highly appreciated.

Data Availability Statement: The research used the KDD CUP 99 dataset. The dataset contains network traffic data that includes various features related to network connections.

Funding Statement: No funding has been obtained to help prepare this manuscript and research work.

Conflicts of Interest Statement: No conflicts of interest have been declared by the author(s). Citations and references are mentioned in the information used.

Ethics and Consent Statement: The consent was obtained from the organization and individual participants during data collection, and ethical approval and participant consent were received.

References

1. Y. Wang, Y. Liu, and J. Li, "Deducing cascading failures caused by cyberattacks based on attack gains and cost principle in cyber-physical power systems," in Journal of Modern Power Systems and Clean Energy, vol. 7, no. 6, pp. 1450-1460, 2019.
2. R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," 2010 IEEE Symposium on Security and Privacy, Oakland, CA, USA, pp. 305-316, 2010.
3. G. Pu, L. Wang, J. Shen and F. Dong, "A hybrid unsupervised clustering-based anomaly detection method," in Tsinghua Science and Technology, vol. 26, no. 2, pp. 146-153, 2021.
4. A. Al-Abassi, and H. Karimipour, A. Dehghantanha and R. M. Parizi, "An Ensemble Deep Learning-Based Cyber-Attack Detection in Industrial Control System," in IEEE Access, vol. 8, pp. 83965-83973, 2020.
5. E. Tufan, C. Tezcan and C. Acartürk, "Anomaly-Based Intrusion Detection by Machine Learning: A Case Study on Probing Attacks to an Institutional Network," in IEEE Access, vol. 9, pp. 50078-50092, 2021.
6. G. Loukas, T. Vuong, R. Heartfield, G. Sakellari, Y. Yoon and D. Gan, "Cloud-Based Cyber-Physical Intrusion Detection for Vehicles Using Deep Learning," in IEEE Access, vol. 6, pp. 3491-3508, 2018.
7. W. Alhakami, A. ALharbi, S. Bourouis, R. Alroobaea and N. Bouguila, "Network Anomaly Intrusion Detection Using a Nonparametric Bayesian Approach and Feature Selection," in IEEE Access, vol. 7, pp. 52181-52190, 2019.
8. R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," in IEEE Access, vol. 7, pp. 41525-41550, 2019.
9. M. A. Siddiqi and W. Pak, "Tier-Based Optimization for Synthesized Network Intrusion Detection System," in IEEE Access, vol. 10, pp. 108530-108544, 2022.
10. A. S. L. V. Tummala and R. K. Inapakurthi, "A Two-stage Kalman Filter for Cyber-attack Detection in Automatic Generation Control System," in Journal of Modern Power Systems and Clean Energy, vol. 10, no. 1, pp. 50-59, 2022.
11. X. Chen, J. Cheng, L. Jiang, Q. Li, T. Wang and D. Li, "Q-learning based strategy analysis of cyber-physical systems considering unequal cost," in Intelligent and Converged Networks, vol. 4, no. 2, pp. 116-126.
12. K. M. K. Raghunath, V. V. Kumar, M. Venkatesan, K. K. Singh, T. R. Mahesh and A. Singh, "XGBoost Regression Classifier (XRC) Model for Cyber Attack Detection and Classification Using Inception V4," in Journal of Web Engineering, vol. 21, no. 4, pp. 1295-1322, 2022.
13. A. A. Alfaifi and S. G. Khan, “Utilizing data from Twitter to explore the UX of ‘Madrasati’ as a Saudi e-learning platform compelled by the pandemic,” Arab Gulf Journal of Scientific Research, vol.39, no.3, pp. 200–208, 2022.
14. A. Ahmed Chhipa et al., “Adaptive Neuro-fuzzy Inference System Based Maximum Power Tracking Controller for Variable Speed WECS”,” Energies, vol. 14, no.19, p.6275, 2021.
15. A. Jafar, O. A. Alzubi, G. Alzubi, and D. Suseendran, “+ A Novel Chaotic Map Encryption Methodology for Image Cryptography and Secret Communication with Steganography,” International Journal of Recent Technology and Engineering, vol. 8, no. IC2, pp. 1122- 1128, 2019.
16. A. K. Sharma et al., “Classification of Indian Classical Music with Time-Series Matching using Deep Learning”,” IEEE Access, vol. 9, pp. 102041–102052, 2021.
17. A. K. Sharma et al., “Dermatologist-Level Classification of Skin Cancer Using Cascaded Ensembling of Convolutional Neural Network and Handcrafted Features Based Deep Neural Network”,” IEEE Access, vol. 10, pp. 17920–17932, 2022.
18. A. K. Sharma, A. Panwar, P. Chakrabarti, and S. Viswakarma, “Categorization of ICMR Using Feature Extraction Strategy and MIR with Ensemble Learning”,” Procedia Computer Science, vol. 57, pp. 686–694, 2015.
19. A. Kumar, S. Singh, K. Srivastava, A. Sharma, and D. K. Sharma, “Performance and stability enhancement of mixed dimensional bilayer inverted perovskite (BA2PbI4/MAPbI3) solar cell using drift-diffusion model,” Sustain. Chem. Pharm., vol. 29, p. 100807, 2022.
20. A. Kumar, S. Singh, M. K. A. Mohammed, and D. K. Sharma, “Accelerated innovation in developing high-performance metal halide perovskite solar cell using machine learning,” Int. J. Mod. Phys. B, vol. 37, no. 07, 2023.
21. A. L. Karn et al., “B-lstm-Nb based composite sequence Learning model for detecting fraudulent financial activities,” Malays. J. Comput. Sci., vol.1, no.1, pp. 30–40, 2022.
22. A. L. Karn et al., “Designing a Deep Learning-based financial decision support system for fintech to support corporate customer’s credit extension,” Malays. J. Comput. Sci., pp. 116–131, 2022.
23. A. Magare, M. Lamin, and P. Chakrabarti, “Inherent Mapping Analysis of Agile Development Methodology through Design Thinking”,” Lecture Notes on Data Engineering and Communications Engineering, vol. 52, pp. 527–534, 2020.
24. B. Senapati and B. S. Rawal, “Quantum communication with RLP quantum resistant cryptography in industrial manufacturing,” Cyber Security and Applications, vol.1, p. 100019, 2023.
25. D. K. Sharma, B. Singh, M. Anam, K. O. Villalba-Condori, A. K. Gupta, and G. K. Ali, “Slotting learning rate in deep neural networks to build stronger models,” in 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC), Tamil Nadu, India, 2021.

26. E. Geo Francis and S. Sheeja, “Towards an Optimal Security Using Multifactor Scalable Lightweight Cryptography for IoT," 2022 3rd International Conference on Communication, Computing and Industry 4.0 (C2I4), Bangalore, India, pp. 1-6, 2022.
27. E. Geo Francis, S. Sheeja and E. F. Antony John, “IoT Intrusion Detection Using Two-Tier-Convolutional Deep-Learning Model," 2023 International Conference on IoT, Communication and Automation Technology (ICICAT), Gorakhpur, India, pp. 1-7, 2023.
28. E. Geo Francis, S. Sheeja and Joseph Jismy, “A Three-layer Convolution Neural Network Approach for Intrusion Detection in IoT," 2023 Eleventh International Conference on Intelligent Computing and Information Systems (ICICIS), Cairo, Egypt, pp. 261-268, 2023.
29. G. A. Ogunmola, M. E. Lourens, A. Chaudhary, V. Tripathi, F. Effendy, and D. K. Sharma, “A holistic and state of the art of understanding the linkages of smart-city healthcare technologies,” in 2022 3rd International Conference on Smart Electronics and Communication (ICOSEC), Tamil Nadu, India, 2022.
30. G. Kannan, M. Pattnaik, G. Karthikeyan, Balamurugan, P. J. Augustine, and Lohith, “Managing the supply chain for the crops directed from agricultural fields using blockchains,” in 2022 International Conference on Electronics and Renewable Systems (ICEARS), Tuticorin, India, pp. 908-913, 2022.
31. H. Sharma and D. K. Sharma, “A Study of Trend Growth Rate of Confirmed Cases, Death Cases and Recovery Cases of Covid-19 in Union Territories of India,” Turkish Journal of Computer and Mathematics Education, vol. 13, no. 2, pp. 569–582, 2022.
32. I. Nallathambi, R. Ramar, D. A. Pustokhin, I. V. Pustokhina, D. K. Sharma, and S. Sengan, “Prediction of influencing atmospheric conditions for explosion Avoidance in fireworks manufacturing Industry-A network approach,” Environ. Pollut., vol. 304, p. 119182, 2022.
33. J. A. Alzubi, O. A. Alzubi, A. Singh, and T. Mahmod Alzubi, “A blockchain-enabled security management framework for mobile edge computing,” Int. J. Netw. Manage., vol. 33, no. 5, pp.1-14, 2023.
34. J. A. Alzubi, O. A. Alzubi, M. Beseiso, A. K. Budati, and K. Shankar, “Optimal multiple key-based homomorphic encryption with deep neural networks to secure medical data transmission and diagnosis,” Expert Syst., vol. 39, no. 4, 2022.
35. J. A. Alzubi, R. Jain, O. Alzubi, A. Thareja, and Y. Upadhyay, “Distracted driver detection using compressed energy efficient convolutional neural network,” J. Intell. Fuzzy Syst., vol. 42, no. 2, pp. 1253–1265, 2022.
36. J. J. Lohith, A. Abbas, and P. Deepak, “A Review of Attacks on Ad Hoc On Demand Vector (AODV) based Mobile Ad Hoc Networks (MANETS),” International Journal of Emerging Technologies and Innovative Research, vol. 2, no. 5, pp. 1483–1490, 2015.
37. K. Kaliyaperumal, A. Rahim, D. K. Sharma, R. Regin, S. Vashisht, and K. Phasinam, “Rainfall prediction using deep mining strategy for detection,” in 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2021.
38. K. Peddireddy and D. Banga, “Enhancing Customer Experience through Kafka Data Steams for Driven Machine Learning for Complaint Management,” International Journal of Computer Trends and Technology, vol. 71, no.3, pp. 7–13, 2023.
39. K. Peddireddy, “Effective Usage of Machine Learning in Aero Engine test data using IoT based data driven predictive analysis,” International J. Adv. Res. Comput. Commun. Eng., vol. 12, no. 10, pp.18-25, 2023.
40. K. Shah, P. Laxkar, and P. Chakrabarti, “A hypothesis on ideal Artificial Intelligence and associated wrong implications”,” Advances in Intelligent Systems and Computing, vol. 989, pp. 283–294, 2020.
41. K. Verma, P. Srivastava, and P. Chakrabarti, “Exploring structure oriented feature tag weighting algorithm for web documents identification”,” Communications in Computer and Information Science, vol. 837, pp. 169–180, 2018.
42. L. J, A. Manoj, G. Nanma, and P. Srinivasan, “TP-Detect: trigram-pixel based vulnerability detection for Ethereum smart contracts,” Multimed. Tools Appl., vol. 82, no. 23, pp. 36379–36393, 2023.
43. Lohith J J and Bharatesh Cahkravarthi S B, “Intensifying the lifetime of Wireless Sensor Network using a centralized energy accumulator node with RF energy transmission,” in 2015 IEEE International Advance Computing Conference (IACC), Bangalore, India, pp. 180-184, 2015.
44. Lohith, K. Singh, and B. Chakravarthi, “Digital forensic framework for smart contract vulnerabilities using ensemble models,” Multimed. Tools Appl., 2023, Press.
45. M. F. Alajmi, S. Khan, and A. Sharma, “Studying Data Mining and Data Warehousing with Different E-Learning System,” International Journal of Advanced Computer Science and Applications, vol. 4, no. 1, pp. 144–147, 2013.
46. M. S. Rao, S. Modi, R. Singh, K. L. Prasanna, S. Khan, and C. Ushapriya, “Integration of Cloud Computing, IoT, and Big Data for the Development of a Novel Smart Agriculture Model,” in the 2023 3rd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), Greater Noida, India, 2023.
47. M. Tiwari, P. Chakrabarti, and T. Chakrabarti, “Novel work of diagnosis in liver cancer using Tree classifier on liver cancer dataset (BUPA liver disorder)”,” Communications in Computer and Information Science, vol. 837, pp. 155–160, 2018.

48. N. Al-Najdawi, S. Tedmori, O. A. Alzubi, O. Dorgham, and J. A. Alzubi, “A Frequency Based Hierarchical Fast Search Block Matching Algorithm for Fast Video Video Communications,” International Journal of Advanced Computer Science and Applications, vol. 7, no. 4, pp.448-455, 2016.
49. N. Priyadarshi, A. K. Bhoi, A. K. Sharma, P. K. Mallick, and P. Chakrabarti, “An efficient fuzzy logic control-based soft computing technique for grid-tied photovoltaic system”,” Advances in Intelligent Systems and Computing, vol. 1040, pp. 131–140, 2020.
50. O. A. Alzubi, I. Qiqieh, and J. A. Alzubi, “Fusion of deep learning based cyberattack detection and classification model for intelligent systems,” Cluster Comput., vol. 26, no. 2, pp. 1363–1374, 2023.
51. P. Chakrabarti and P. S. Goswami, “Approach towards realizing resource mining and secured information transfer”,” International Journal of Computer Science and Network Security, vol. 8, no. 7, pp. 345–350, 2008.
52. P. P. Dwivedi and D. K. Sharma, “Application of Shannon entropy and CoCoSo methods in selection of the most appropriate engineering sustainability components,” Cleaner Materials, vol. 5, p. 100118, 2022.
53. P. S. Venkateswaran, F. T. M. Ayasrah, V. K. Nomula, P. Paramasivan, P. Anand, and K. Bogeshwaran, “Applications of artificial intelligence tools in higher education,” in Advances in Business Information Systems and Analytics, IGI Global, USA, pp. 124–136, 2023.
54. R S Gaayathri, S. S. Rajest, V. K. Nomula, R. Regin, “Bud-D: Enabling Bidirectional Communication with ChatGPT by adding Listening and Speaking Capabilities,” FMDB Transactions on Sustainable Computer Letters., vol. 1, no. 1, pp. 49–63, 2023.
55. R. Singh et al., “Smart healthcare system with light-weighted blockchain system and deep learning techniques,” Comput. Intell. Neurosci., vol. 2022, pp. 1–13, 2022.
56. S. Abukharis, J. A. Alzubi, O. A. Alzubi, S. Alamri, and T. O. Tim O\’Farrell, “Packet error rate performance of IEEE802.11g under Bluetooth interface,” Res. J. Appl. Sci. Eng. Technol., vol. 8, no. 12, pp. 1419–1423, 2014.
57. S. Khan, “Artificial Intelligence Virtual Assistants (Chatbots) are Innovative Investigators,” International Journal of Computer Science Network Security, vol. 20, no. 2, pp. 93–98, 2020.
58. S. Khan, M. Altayar, , “Industrial internet of things: Investigation of the applications, issues, and challenges,” Int. J. Adv. Appl. Sci., vol. 8, no. 1, pp. 104–113, 2021.
59. S. Parthasarathy, A. Harikrishnan, G. Narayanan, L. J., and K. Singh, “Secure distributed medical record storage using blockchain and emergency sharing using multi-party computation,” in 2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Paris, France, 2021.
60. S. Samadi, M. R. Khosravi, J. A. Alzubi, O. A. Alzubi, and V. G. Menon, “Optimum range of angle tracking radars: a theoretical computing,” Int. J. Electr. Comput. Eng. (IJECE), vol. 9, no. 3, p. 1765, 2019.
61. Sholiyi A., O’Farrell T., Alzubi O., and Alzubi J., “Performance Evaluation of Turbo Codes in High Speed Downlink Packet Access Using EXIT Charts”, International Journal of Future Generation Communication and Networking, vol. 10, no. 8, pp.1-14, 2017.
62. Sudheesh et al., “Analyzing sentiments regarding ChatGPT using novel BERT: A machine learning approach,” Information (Basel), vol. 14, no. 9, p. 474, 2023.
63. Sudheesh et al., “Bidirectional encoder representations from transformers and deep learning model for analyzing smartphone-related tweets,” PeerJ Comput. Sci., vol. 9, no. 5, p. e1432, 2023.
64. T. Chen, J. Blasco, J. Alzubi, and O. Alzubi “Intrusion Detection”. IET Publishing, vol. 1, no. 1, pp. 1-9, 2014.
65. V. K. Nomula, R. Steffi, and T. Shynu, “Examining the Far-Reaching Consequences of Advancing Trends in Electrical, Electronics, and Communications Technologies in Diverse Sectors,” FMDB Transactions on Sustainable Energy Sequence, vol. 1, no. 1, pp. 27–37, 2023.