A Multiclass Network Intrusion Detection System Using Stacking Ensemble Technique with Hybrid Feature Selection

Veena S Badiger * and Gopal K Shyam

School of Engineering, Presidency University, India
Email: veenasbadi@gmail.com (V.S.B.); gopalkrishna.shyam@presidencyuniversity.in (G.H.S.)
*Corresponding author

***Abstract*—Intrusion detection in network is essential in order to preserve the confidentiality, integrity and authentication in online community. Various research in earlier have used conventional machine learning classifiers and deep learning classifiers to detect intrusion in network but there is a need to achieve improved efficiency of classifiers. In this study we have proposed stacking ensemble machine learning model detect multiclass intrusion in network more effectively. Decision Tree, Random Forest (RF), XGBoost (XGB), AdaBoost, LightGBM (LGBM), Logistic Regression (LR) and Multilayer Perceptron (MLP) were used in ensemble learning for classification of intrusion. The novelty of the work lies in utilizing three techniques for feature selection, namely Gini index, correlation analysis, and random forest to select the most important features. To address class imbalance, Synthetic Minority Over-sampling Technique (SMOTE) was applied. In order to test the effectiveness of the proposed model it was evaluated on UNSWB-NB15, CSE-CIC-IDS2018 and HIKARI-2021 latest heterogeneous datasets. Our proposed method produces excellent results in terms of accuracy and precision, which exceeds 96% and 98%. For better validation other metrics such as Recall, F1-Score, AUC-ROC score, Log loss value and Cohen's Kappa score was considered. From the research findings it can be indicated that the proposed model significantly enhances the multiclass network intrusion detection through its outperforming performance and has potential to strengthen the networks and systems from cyber-attacks.**

***Keywords*—machine learning, ensemble learning, intrusion detection, cyber attacks**

I. Introduction

The successful functioning of a network has become essential due to the advent of the Internet and communication technologies. The growing number of users and their dependence on the Internet has attracted the intruders for attacking the data for misuse; hence it has led to a surge in network attacks that disrupt with the network's regular operation. Attacks of all kinds are becoming more prevalent which hamper the confidentiality, authencity and integrity of data. The network can no longer be protected by traditional techniques such as firewalls, safety measures, and antivirus software alone. An advanced network security system called a network intrusion detection system is made to defend against certain threats by continuously observing network traffic. Such systems analyze the data and try's to identify any unusual network traffic such as unauthorized access, alteration and intrusions.

Intrusion Detection System (IDS) is usually software which monitors for potentially harmful activity on the network and reports the system operation to the administrator. Based on the deployment the IDS can be classified as Host based Intrusion Detection System (HIDS) and Network Intrusion Detection System (NIDS). NIDS is frequently deployed or placed at critical network nodes to ensure that it monitors traffic that is more susceptible to attack whereas the HIDS is deployed with any networked device that has an Internet connection. Implementing an effective Network Intrusion detection system is one of the critical tasks when it comes to network security. The technology of the Network intrusion detection system is constantly evolved for nearly a decade [1]. Machine Learning (ML) algorithms have substantially been used in network security due to the algorithms nature of learning from features in data samples and find hidden patterns to make predictions in given new data instances [2]. Based on the nature of automatic feature selection and deep architecture, Deep Learning (DL) algorithms is been used to develop NIDS. Various standalone Machine learning algorithms, Deep learning algorithms are being proposed in the literature for detecting intrusion in the network. Hybrid techniques utilizing multiple Machine learning algorithms or Deep learning algorithms are being proposed to enhance the prediction of intrusion. Even though standalone Machine learning algorithms and Deep learning algorithms have proved good performance still there are challenges in achieving high detection rate, low false positive rate and minimal errors and hence a technique which overcomes these challenges has to be applied in order to have efficient NIDS which safeguards the resources from the attacks.

Manuscript received September 14, 2024; revised October 7, 2024; accepted December 2, 2024; published March 14, 2025.

An Ensemble based models have indeed shown promise in the realm of intrusion detection systems. By combining multiple individual models, ensemble model can achieve higher detection rates compared to single models. Ensemble models overcome the disabilities of standalone ML classifier. Each classifier in the ensemble may excel at detecting certain types of intrusions or patterns, and the combination leverages their strengths to enhance overall detection capabilities. Ensemble models are often more resilient to attacks compared to single models. This resilience is indeed from the fact that an attack might go undetected in one model but in the ensemble is less likely to be undetected. Therefore, the ensemble can still provide accurate predictions even when facing sophisticated attacks.

In the proposed work is on utilizing well known ML algorithms, Decision tree, Random Forest, Logistic Regression, XGBoost, AdaBoost, LightGBM to develop ensemble model for network intrusion detection. The main contribution of this paper is: i) Combining ML algorithms using two level stacking ensemble model; ii) Improving the performance of the stacking ensemble model compared to other existing work; iii) The proposed model can be effectively used on several latest benchmark datasets showing the models adaptability to heterogeneous datasets.

The remaining paper is organized as follows: Section II outlines the related work. Proposed stacking ensemble model for NIDS is described in Section III, while Section IV describes the experimental setup with the discussion of results obtained and comparative study with the state of art works. Finally, Section V concludes the paper and presents future work.

II. Literature Review

The field of intrusion detection is expanding quickly, and researchers constantly conducting investigations and inventing new ideas to boost the effectiveness of network intrusion detection systems. In this section, we give an overview of some of these works with emphasizes on the overall performance areas for improvement. Ding *et al.* [3] proposed a voting model based on ensemble learning is proposed to handle hard-to-classify data such as complex network datasets, and the model adopts a voting mechanism among seven base models which includes decision tree, LightGBM, linear discriminant analysis, logistic regression, random forest, gaussian naive bayes, and AdaBoost. Pandit *et al.* [4] introduced a novel ensemble method for intrusion detection in networks using a combination of decision tree, random forest, extra tree, and XGBoost algorithms, significantly improving detection accuracy. Dasari *et al.* [5] proposed an unique model named Intrusion Detection System using Machine Learning Analytics (IDSMLA), which uses Synthetic Minority Over-sampling Technique (SMOTE) oversampling technique to deal with class imbalance problem, it also uses Minimum Redundancy Maximum Relevance (MRMR) to perform feature selection to reduce time complexity by eliminating irrelevant features and achieved increasing accuracy of the model. In order to perform classification task, the proposed model IDSMLA uses Extra Trees (ET) bagging ensemble technique. Ramachandran *et al.* [6] proposed an intelligent intrusion detection system that can detect different networking attacks with an improved detection rate and overcomes the threat in faster manner. The authors proposed intelligent intrusion detection system applying ensemble deep learning algorithm approach. By using Convolutional Neural Networks (CNN) and Deep Neural Network (DNN) the model was created and tested on NSL-KDD dataset. Zhang *et al.* [7] proposed bagging ensemble learning to overcome asymmetry in feature and achieved improved accuracy. KNN as base estimator and Bayesian optimizer was applied. Extreme random tree was used to calculate weights of the features. Combination flow based features and payload features were used to detect intrusion in network by Kiflay *et al.* [8]. The classification results based on flow based features and that of payload based features were sent to soft voting ensemble classification for the final predictions. In both the feature categories random forest was applied for classification. A multimodal ensemble model using XGBoost and CNN was proposed by Lochanan *et al.* [9] utilizing network logs in tabular and image forms. Gaikwad *et al.* [10] proposed a stacking ensemble model for the intrusion detection system, three base classifiers Multilayer Perceptron, Ripple Down Rule learner, and RepTree Decision tree have been used. Logisitic regression with ridge estimator have been used as metclassifier and achieved accuracy of 99.64%. Otoom *et al.* [11] proposed an ensemble model based on Bagging with J48 Decision Tree for Network Intrusion Detection (NID), which achieved an average accuracy of 83.73%. An ensemble learning with expert system was proposed by Zewdu *et al.* [12]. Three ensemble techniques boosting, bagging, and random subspace was studied with three base learners namely random forest, Bayes net, and SMO. It was proved that ensemble learning techniques showed improved performance in network intrusion detection. AdaBoostM1 with random forest base learner achieved the better results. Hammood *et al.* [13] proposed an ensemble learning using voting technique to detect intrusion in network. Logistic regression, naive Bayes, decision trees, extra trees, Random forests and gradient boosting were applied on three datasets UNSW-NB15, IoTID20 and BoTNeTIoT-L01-v2 and achieved improved accuracy. Das *et al.* [14] proposed intrusion detection system based on machine classifiers. Ensemble learning was incorporated to enhance detection accuracy that combine multiple classifiers, specifically Balanced Bagging and XGBoost to improve the overall performance of the IDS. RF-HDDT was applied to overcome class imbalance. Some of the significant works and there limitations are summarized in Table I.

TABLE I. A BRIEF DESCRIPTION OF SIGNIFICANT WORK

Reference	Dataset and Algorithm used	Results	Limitations
Ramachandran *et al.* [6]	NSL KDD. Classification: Ensemble DNN & CNN.	Accuracy = 93	Handling imbalanced data not performed. F1-Score, precision not measured.
Zhang *et al.* [7]	NSL-KDD. Classification: KNN as Bagging ensemble Bayesian Optimization for optimization objective function	Accuracy = 82.48%	Data balancing was not incorporated.
Kiflay *et al.* [8]	UNSWB-NB15. Classification: Soft voting using random forest	Accuracy = 98%	Lacks comparison analysis and details of feature selection techniques.
Gaikwad *et al.* [10]	NSL-KDD. Stacking ensemble using Multilayer Perceptron, Ripple Down Rule learner, RepTree Decision Tree and Logisitic regression.	Accuracy = 99.64%	Only accuracy parameter was considered and lacks data balancing
Otoom *et al.* [11]	NSL-KDD. Bagging ensemble model using J48 Decision Tree	Average accuracy = 83.73%	Data balancing was not incorporated.
Hammood *et al.* [13]	UNSW-NB15, IoTID20 and e BoTNeTIoT-L01-v2. Classification: Voting ensemble using logistic regression, naive Bayes, decision trees, extra trees, random forests, and gradient boosting.	Accuracy for UNSWB-NB15= 98.52%, Accuracy for IOTID20 = 88.41%, Accuracy for BoTNeTIoT-L01-v2 = 91.03%.	Lacks feature selection techniques explanation. Data balancing and comparative analysis not incorporated.
Das *et al.* [14]	UNSW-NB15. Classification: Balanced Bagging, XGBoost, and RF-HDDT	Accuracy = 97.8%, precision = 97.81%, recall = 97.75%, F1-Score = 97.88%.	Models errors was not calculated. Data balancing was not incorporated
Zhu *et al.* [15]	UNSWB-NB15. Framework based on subspace clustering and ensemble learning	Accuracy = 97.05%, precision = 96.33%, recall = 96.55%, F1-Score = 96.45%.	Models error detection was not incorporated.

From literature review and summary in Table I we can infer that the most of the dataset incorporated are a decade old and which does not include latest network attacks. The methodology employed does not take into consideration of all testing performance metrics and model errors. In order to achieve this, a robust and adaptable methodology that improves performance and addresses the problems with the current system is required. The proposed m Stacking ensemble machine learning model is a technique that is used to detect all types of intrusions (Multiclass) in network traffic.

III. PROPOSED METHODOLOGY

This section provides detailed description of the proposed model. The comprehensive architecture of the proposed stacking ensemble model to detect intrusion in network is shown in Fig. 1. The working of the model which is the pipeline of machine learning algorithms along with feature selection and balancing of the data is explained. The proposed model takes in account three latest heterogeneous dataset explained in Section IV.A. In order to overcome missing values, conversion of categorical values and overcome large values the datasets were preprocessed, data preprocessing is explained in Section III.A. The datasets were balanced to avoid bias and obtain improved performance of the model. Data balancing is explained in Section III.B. The most essential features required to train the model to enhance the performance of the model were selected using hybrid feature selection involving three different methods which is explained in Section III.C. The training of the base classifiers were done applying Stratified K-Fold cross validation explained in Section III.D. Six most popular Machine Learning (ML) classifiers which were used as base classifiers to create stacking ensemble model is explained in Section III.E. Finally, upon validation the model predicts non-intrusion and the multiple classes of intrusions in the incoming network traffic. Numerous performance metrics were considered to measure the efficiency of the proposed model. The detailed explanation of each section of the proposed model is as follows:

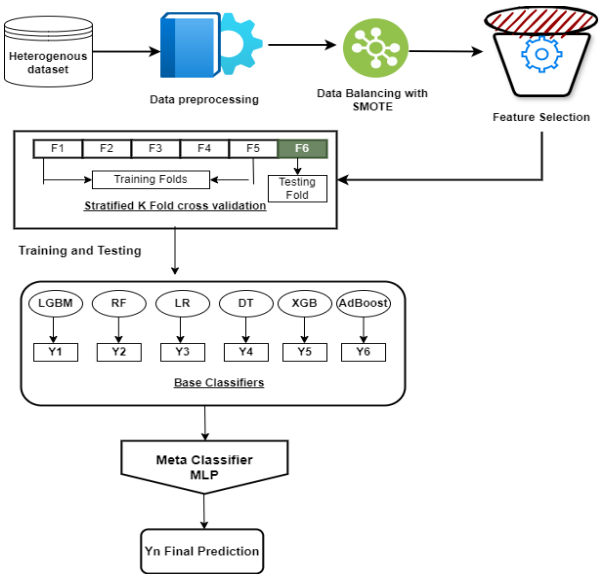


Fig. 1. Architecture of the proposed stacking ensemble model.

A. Data Preprocessing

In order to train the machine learning and deep learning models effectively, the dataset should be cleaned and processed. Data preprocessing is essential for data analysis [16]. Enhancing dataset quality is to ensure the dataset is free from missing values, undefined values, outliers and should be scaled to common range. Data becomes more manageable and easier to train the model after the data preprocessing step. Data preprocessing was carried in Colab which provides preprocessing modules such as standard_scaler, label_encoder for

implementation. In this proposed research work data preprocessing steps include checking for missing values and replacing them, finding large or infinite values and replacing them with NaN, removing rows containing NaN, converting categorical values to numerical, converting target values from categorical to discrete values. The remaining data processing steps after duplicate row removal is explained as follows:

1) Data imputation

In the first step dataset is checked for any missing values using isnull function of pandas. The function returns true for missing values and false for if values are existing. The missing values are treated as NaN by pandas. Missing values impacts model estimation performance [17]. The NaN is handled by applying statistical techniques such as mean, median or mode. In this research work features with NaN are replaced with mean values of features using SimpleImputer class of scikit learn.Numerical data provides meaningful information to train the model. Non numerical data called Categorical data are dtype with object represents qualitative data that don't have a natural ordering, such as types of objects, or categories like "yes" and "no". In our research work attack type, protocol name, source IP address, destination IP address are some of the categorical data's found in the dataset. LabelEncoder class of preprocessing module from sklearn is used to convert categorical data to numeric.

2) Data normalization

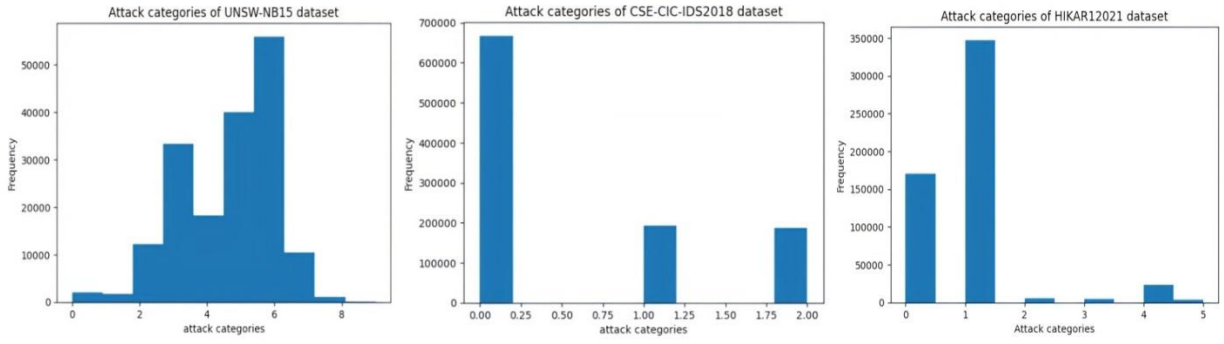
Numerical features in the dataset should be in common scale. Varying range of scales in features such as extremely higher number or extremely lower number leads to bias and hence will effect the performance of the machine learning model. Numerical values are rescaled to common scale typically between 0 to 1 or –1 to 1. In our work we have rescaled the features using StandardScaler class of scikit learn. StandardScaler normalizes the feature based on standardization technique were the values are subtracted from mean and brought to unit variance by dividing with standard deviation. The equation of standardization is given in Eq. (1).

$$Xnew = \frac{X-\mu}{M} \quad (1)$$

X is the original value of the feature, *Xnew* is the rescaled value of feature. μ is the mean value of the feature and *M* is the standard deviation.

B. Data Balancing

On examining the datasets it was found that the datasets are significantly imbalanced. The classes found in the dataset are skewed towards one category which significantly outnumbers the others. The imbalancing in dataset effects the model performance and can lead to skewed model predictions, where the intrusions are often misclassified or neglected posing a significant security risk to the network. Study on imbalance in dataset was performed in our research work and it is shown in graphical form in Fig. 2. Fig. 2(a) shows that the classes are imbalanced in UNSWB-NB15 were Class 6 has the highest instances, with over 50000 instances. Classes 3, 4, and 5 also have relatively high instances, each with tens of thousands. On the other hand, Classes 0, 1, 2, 7, and 8 have much lower instances, some with fewer than 10000. In Fig. 2(b) shows the Classes are imbalanced in CSE-CIC-IDS2018, were Class 0 has high number of instances with over 700000 and Classes 1,2 has lower instances compared to Class 0 of over 200000. Similarly Fig. 2(c) shows Class data imbalance in HIKARI-2021 dataset were Class 1 has higher number of instances compared to the other classes and Class 0 has moderately lower instances compared to the Class 1.



(a) (b) (c)

Fig. 2. Imbalance class distribution in (a) UNSWB-NB, (b) CSE-CICIDS2018 and (c) HIKARI-2021.

Addressing imbalanced class distribution is a crucial step before building any machine learning model, especially for tasks like intrusion detection where the occurrence of intrusions is typically a minority class compared to normal instances. In the proposed work we applied data augmentation technique, Synthetic Minority Over-sampling Technique (SMOTE) to address class imbalance.

C. SMOTE

The cost of misclassification of minority class is higher. A combination of increasing minority class and decreasing the majority class was proposed by Chawla *et al.* [18]. They declared this unique method as SMOTE and observed that model performance was superior. Increasing the minority class samples solves the

imbalanced class problem but does not improve the performance of the model. In order to increase the performance of the model randomly samples in the minority class are increased with random *k* nearest neighbor. This technique generates synthetic samples for minority classes to balance the class distribution, thereby improving the model's ability to learn from the minority class examples.

D. Feature Selection

Selecting the most reliable features is important in building machine learning model. Not all features in the dataset contribute equally in building the model. Irrelevant or redundant features can lead to overfitting which introduce noise and add unnecessary complexity, leading to decreased model accuracy. Feature selection helps the model to focus on the most important features and hence reduces training time, overfitting and leading to improved performance [19]. In the proposed work, network features were selected applying hybrid feature selection given in Fig. 3 which involves three different techniques Gini Index, Correlation analysis and Random Forest (RF). Description of these methods is given below:

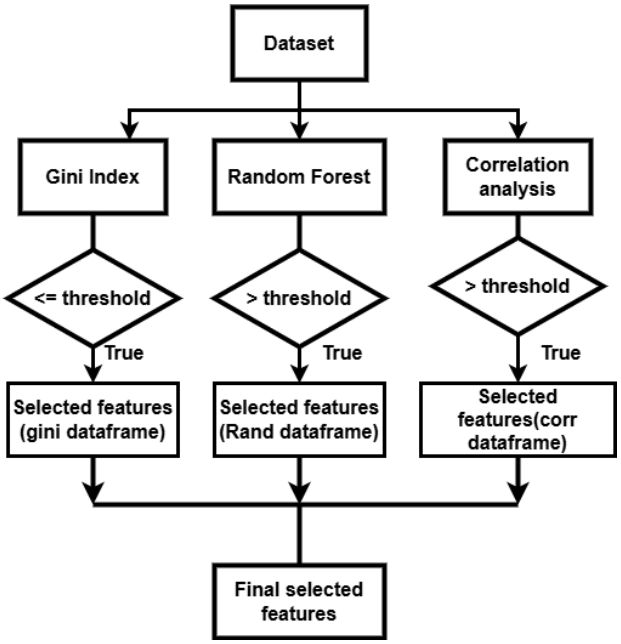


Fig. 3. Proposed hybrid feature selection approach.

1) Gini Index (GI)

Gini index is the important feature selection technique applied in classification and regression trees [20]. Gini index selects the features based on the feature importance. Features with lower Gini index values are considered better at separating the classes in the target variable by calculating probability of the feature. Gini index ranges from 0 to 1 where 0 is considered as the most pure feature of the dataset. Gini index is calculated applying the given Eq. (2).

$$Gini\ index = 1 - \sum_{i=1}^{n} pi \times pi \quad (2)$$

During the feature selection process threshold was set to 0.3 so that overfitting and underfitting of features are overcome. This is done by using the method "selectBestFeatures" which is given below in Eq. (3):

$$selected\_features = selectBestFeatures(dataset, scores, 0.3) \quad (3)$$

$$ginidataframe = selected\_features[0] \quad (4)$$

The features which are less than or equal to threshold value were selected and stored in data frame shown in Eq. (4) which is given for training the model. The resulting data frame contains the features which are suitable for training.

2) Correlation Analysis (CA)

Correlation analysis in feature selection involves calculating statistical relationship between features and the target variable. It helps in identifying strongly correlated features with other features referred as redundant features which cause overfitting and hence minimize model performance. Features should be highly correlated with target variable and uncorrelated with others in the same feature subset [21]. In our research highly correlated features are dropped and not used for prediction. Only relevant features are selected and used for training the model. The correlation between features and target column is calculated by applying Eq. (5).

$$correxy = \frac{\sum(x-x')(y-y')}{\sqrt{\sum\sqrt{(x-x')^2(y-y')^2}}} \quad (5)$$

correxy gives the correlation between feature *x* and target *y*. Correlation of feature *x* and target *y* is its covariance in numerator and sum of standard deviation in denominator where $x'$and y' are the mean value of feature *x* and target *y*. In our implementation threshold value was set to 0.5. Features whose correlation is more than threshold are selected and stored in the set highly correlated features given in Eq. (6). corr() function was used to calculate correlation. highcorr_features set contains the features whose correlation is greater than threshold. This is done by using index.tolist() function. These features are dropped and the others are used for model training.

$$highcorr\_features = target\_corr[target\_corr > 0.5].index.tolist() \quad (6)$$

$$corrdataframe = high\_corrfeatures[0] \quad (7)$$

3) Random Forest (RF)

Random forest is an ensemble classifier which can be used to select features by generating many decision trees based on attributes. Random forest is a classifier which performs feature selection as classification rule is built [22]. Random forest generates many decision trees and each tree is constructed by a different bootstrap sample from the original data using a tree classification algorithm. In our research work 100 decision trees are

generated to select features by using SelectFromModel package and by setting hyper parameter n_estimators to 100. The threshold value was set to 0.1 which is a boundary to select the features and features to be eliminated. Features whose score in less than threshold are eliminated and selected features are given to ensemble algorithm to classify intrusion. Random forest classifier to select features is given in Algorithm 1.

Algorithm 1 takes input training dataset which has *Y* target to be predicted. Initially in the algorithm number of decision trees to be generated is set and threshold value is determined. The selected features are voted in the trees. The results from all the trees are aggregated to check the features which have higher votes. The features selected from three different approach Gini index, Correlation analysis and Random forest classifier which are stored in data frame were combined into a single list. The final collection of features extracted has most important information which is more reliable and accurate. As a result models performance is enhanced to classify the intrusion in network. The selected features of each dataset are given in Table II.

Algorithm 1. Feature_selection_Random_forest

Input training set s:= (x1,y1),…..,(xn,yn), F features, Y target
Step1: Set the number of decision trees to be generated as N
Step2: Determine the threshold value
Step3: Construct N decision trees
Step4: Sort the value of features based on feature importance
Step4: Eliminate the features below the threshold
Step5: Select the M features from F features which have most votes from trees.
Output: M selected features

TABLE II. LIST OF FEATURES SELECTED FROM EACH DATASET

Dataset	Selected features for Multiclass detection
UNSW-NB15	dur, proto, service, state, spkts, sbytes, sttl, dttl, sload, sloss, tcprtt, synack, ackdat, smean, trans_depth, response_body_len, is_ftp_login, ct_ftp_cmd, ct_flw_http_mthd, is_sm_ips_ports
CSE-CIC-IDS2018	protocol, timestamp, flowduration, totfwdpkts, totbwd pkts, totlenfwd pkts. fwd pktlenmax, fwd pktlenstd, bwdpktlenmax, flowpkts/s, pshflag cnt, ackflagcnt, urgflagcnt, down/upratio, subflowfwdpkts, subflow fwdbyts, subflowbwdpkts, initfwdwinbyts, fwdactdatapkts, fwdsegsize min
HIKARI-2021	origin, fwd_pkts_tot, bwd_pkts_tot, fwd_data_pkts_tot, bwd_data_pkts_tot, fwd_header_size_tot, bwd_header_size_tot, bwd_header_size_max, fwd_psh_flag_count, bwd_psh_flag_count, flow_pkts_payload.avg, flow_pkts_payload.std, fwd_subflow_pkts, bwd_subflow_pkts, fwd_subflow_bytes, active.min, active.max, active.avg, bwd_init_window_size, label

E. Stratified K-Fold Cross Validation

K-fold cross-validation is an essential tool for model validation that helps in making more accurate and reliable predictions and enhances the model's performance, ensuring that it generalizes unseen data. In the context of stacking ensemble learning it improves model assessment. Initially, the training dataset is folded into k subgroups of equal size which include data that has been stratified and chosen at random in order to maintain the class ratio. The remaining fold is used as a validation set to evaluate the accuracy of the trained classifiers after k folds have been utilized to train base classifiers which learn from various subsets of training data and minimizing overfitting. This calculates the probability that each base classifier will be able to generalize to new information. When k iterations are completed, a total of k sets of predictions from each base classifier across the whole training dataset will be obtained. By utilizing the distinct advantages of each individual model, the stacking ensemble is able to improve predictive accuracy and robustness through the training of the meta-model using base classifier predictions.

F. Ensemble Learning

Ensemble models typically exhibit higher accuracy and robustness due to the diversity of individual classifiers within the ensemble and produce improved detection rate [23]. Ensemble methods often combine the predictions of multiple weaker learners leading to a stronger overall prediction. This is particularly advantageous in IDS, where no single model may excel at detecting all types of intrusions with high accuracy. In the proposed research work two level stacking ensemble learning is applied for intrusion detection in network. A two level stacking ensemble includes base classifier and meta classifier. Each classifier in the base classifiers work independently and takes into account the complete input dataset for training with different hyper parameters, hypothesis and algorithms. The combined output predictions of the base classifiers are cross validated and given to the metaclassifiers which provides the final prediction. The algorithm of proposed stacking ensemble learning is given in Algorithm 2.

Brief description of Base classifiers is as follows:

1) Decision Tree (DT) classifier

A decision tree is a prominent supervised machine learning technique for both regression and classification. The internal node represents a feature or attribute, each branch represents a decision rule based on that feature, and each leaf node represents the outcome or prediction. It can handle numerical and categorical data. A node with feature "x" is compared with the internal node and outcome of comparison is either true or false and based on result either left child or right child is traversed. In the proposed research work Gini index is applied to select features for decision tree.

2) Random Forest (RF) classifier

Random forest is a popular machine learning algorithm that operates by constructing multiple decision trees during training. To generate distinct training datasets for every tree, Random Forest employs bootstrapping. These decision trees are then used to make

predictions by aggregating the results from each tree. Each decision tree is trained on a random subset of the training data and a random subset of features at each split. A majority vote system determines the final prediction. The predicted class for the sample is determined by aggregating the votes cast in each tree.

Algorithm 2. Proposed stacking ensemble learning

Input: Training data 'D'
Step1: Select the best features 'B1' using gini index on dataset

Step2: Select the highly correlated data 'B2' with target using correlation analysis.
Step3: Select the best features 'B2' based on feature importance using random forest.
Step4: final features 'F'=B1 U B2 U B2
Step5: Train base classifiers
for i ←1 to n do
Train base classifiers Ci on 'D' based on 'F'
Where Ci are Decision Tree, Random Forest, Logistic Regression,
XGBoost, AdaBoost, LightGBM
end for
Step6: construct new training dataset based on the predictions of base classifier
for i ← 1 to n do
construct new feature matrix (xi,yi) where xi={C1(xi),C2(xi)….Cn(xi)}
end for
Step7: Train metaclassifiers Multilayer perceptron 'M' on newly constructed
Dataset
Output: final prediction of metaclassifiers
PM(x)= M(C1(x),C2(x)…..Cn(x))

3) Logistic Regression (LR) classifier

Logistic Regression is a statistical model commonly used for classification tasks, where the goal is to predict an outcome. It learns a linear relationship from labeled data and introduces nonlinearity through an activation function, often using the logistic function for binary classifications. A given input's likelihood of belonging into a specific class, either 0 or 1, is predicted through logistic regression. Next, using a threshold value of 0.5, the predicted probability is transformed into a binary result. The input is assigned to one class if the predicted probability is higher than the threshold; if not, it is assigned to the other class.

4) XGBoost classifier

Extreme Gradient Boosting, or XGBoost, is an effective machine learning algorithm that is currently well-known due to its effectiveness, scalability, and remarkable performance on a variety of tasks, such as classification and regression. It combines multiple weak learners to create a strong learner. It has been shown to outperform with other algorithms, such as logistic regression, Stochastic Gradient Descent, Radial Basis Function Classifier, Rotation Forest, Sequential Minimal Optimization and random forests. XGBoost has been found to be particularly effective for dealing with unbalanced large datasets and uses decision trees as base learners. The XGBoost adapts the concept of Gradient-based optimization which prevents overfitting and loss function by iteratively adding new trees to the ensemble that correct the errors made by the existing ensemble.

5) AdaBoost

Adaptive Boosting, or AdaBoost, is a popular ensemble learning method used in machine learning. It works by combining multiple weak learners to create a strong classifier. The algorithm assigns weights to the data points and adjusts them at each iteration to focus on the misclassified points from the previous round. By giving more weight to the misclassified data points. As a result, the final model produced by AdaBoost is a weighted combination of these weak classifiers, where each classifier contributes based on its performance in capturing different aspects of the data. The initial weights of the sample target is classified applying the Eq. (8)

$$w0 = (w10, w20, \ldots, wn0)$$

where $wi0$= 1 to n, n is the number of samples.

$$wi0 = i/n \tag{8}$$

At each boosting round t = 1, 2, ..., m weak classifier $ht(x)$ is trained using dataset $D = \{(xi, yi, wi^t)\}$ where wi^t is the weight of the instance i at round t.

Weights of the samples wi^t for weak classifiers $h(x)$ is updated in round t by first calculating loss value given in Eq. (9)

$$h(x) = \frac{1}{2} log\ (\frac{1-Et}{Et}) \tag{9}$$

where Et is the total error in classifying attacks at round t.

Weights of the each correctly classified sample wi^{t+1} for next iteration is updated based on the Eq. (10)

$$wit + 1 = wit \times eh(x) \tag{10}$$

Weights of the each incorrectly classified sample wi^{t+1} for next iteration is updated based on the Eq. (11).

$$wit + 1 = wit \times e - h(x) \tag{11}$$

AdaBoost helps to improve the overall performance of the model. AdaBoost enhances the accuracy of intrusion detection systems in detecting and categorizing harmful activity within computer networks.

6) LightGBM (LGBM)

LightGBM or Light Gradient Boosting Machine is a gradient boosting framework that works by sequentially adding decision trees as weak learners to an ensemble model, with each tree correcting the errors of the previous ones. Light Gradient Boosting is well known for its accuracy and speed during model training. It uses histograms to efficiently find optimal split points during tree construction. This histogram-based technique reduces computational costs, especially in datasets with numerous features and instances. Using a leaf-wise tree growth technique, LightGBM splits nodes first that result in the largest loss function decrease based on gradients.

IV. Experimental Setup

The experiment was carried out in Google Colaboratory an online platform for developing and testing machine learning and deep learning model in various domains with python 3 as language for implementation. The entire study was carried on multiple publically available datasets to test the adaptability of the model on balanced and unbalanced datasets. These datasets are discovered in the several previously published works. The various datasets using which experiment was conducted and there brief description is given in the following Section IV.A. The summary of the dataset is given in Table III.

TABLE III. Dataset Description

S. No	Dataset	Size	Features	Attacks	Attack classes
1	UNSWB-NB15	175341	45	9	Generic, Exploits, Fuzzers, DoS, Reconnaissance, Analysis, Backdoor, Shellcode, Worms
2	CSE-CIC-IDS2018	1048575	80	2	FTP-BruteForce, SSH-Bruteforce
3	HIKARI-2021	555278	88	5	Background, Probing, Bruteforce, Bruteforce-XML, XMRIGCC CryptoMiner

A. Dataset Description

Datasets are crucial for machine learning-based models because models learn to recognize patterns and make predictions based on the data they are trained. Exposure to variety of examples in dataset makes the model to become robust. In order to test the effectiveness of the proposed model it is evaluated on 3 publically available dataset which includes a range of attack categories, including the ones that organizations currently face. All attack category were classified as multiclass intrusion. Datasets used for training, testing and evaluating the model is as follows:

1) UNSWB-NB15

UNSW-NB15 dataset was generated in 2015 [24]. The dataset was generated using IXIA tool which generates network traffic. The features of the UNSW-NB15 are generated using Argus, Bro-IDS tools and 12 algorithms. The size of the dataset is more than 2.5 million out of which 221,876 contains normal records and 321,283 contains attacked records. In our study we have used refined dataset containing 175,341 records as training set and 85,332 as testing set for multiclass detection.

2) CSE-CIC-IDS2018

The dataset was developed in 2018 [25] as a result of collaborative project between the Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC) which generates cybersecurity datasets systematically using the notion of profiles. The information includes: the captures of the network flow, system logs of each machine. 1048575 records and 80 features were used in our research work to test the proposed model and to detect the attacks.

3) HIKARI-2021

HIKARI-2021 is publically available dataset which is developed by Ferriyan *et al.* [26], In lab which contains network traffic of the forms encrypted synthetic attack traffic and benign traffic. The dataset was published in 2022 which is the latest dataset containing new form of attacks in encrypted form. The dataset included 88 features where the uid is encrypted feature. There are 555278 records in dataset out of this 347,431 (62.5 %) are benign traffic and 207847 (37.5 %) are attack traffic.

B. Performance Evaluation Metrices

The performance of the machine learning model for classification are evaluated using the most frequently used metrics accuracy, precision, recall, F1-Score [27] however these metrics are not sufficient to evaluate model when the dataset is huge and imbalanced. To study the performance of the proposed model for multiclass classification metrics such as ROC-AUC score, Multiclass Cross-Entropy Loss and Kappa score is employed along with frequently used metrices to test the effectiveness of the model. In order to calculate the performance metrics results the model is evaluated for the parameters such as True positive (TP), True negative (TN), False positive (FP), False negative (FN) from Confusion matrix. Further Confusion matrix, ROC curve are plotted to interpret the performance metrics. The details of the performance metrics in the context of network intrusion detection for classifying various types of intrusion traffic from network traffic is as follows:

- **Accuracy metric:** The proportion of correct predictions of intrusion made by the model out of the total number of predictions made is called accuracy. Micro average accuracy is applied for calculating accuracy which is the ratio of aggregate count of correct predictions of "*n*" classes of intrusion out of the total number of predictions of *n* classes. The micro average accuracy equation of n classes is given in Eq. (12).

$$Accuracy\ n\ = \frac{TP_n + TN_n}{TP_n + TN_n + FP_n + FN_n} \tag{12}$$

- **Precision:** The proportion of total number of correct predictions of intrusion traffic to the total number of predictions of intrusion predicted is called precision. Micro average precision is applied for calculating precision in multiclass classification which is the aggregate count of true positive predictions for n classes. The micro average precision of n classes is given in Eq. (13).

$$Precision\ n\ = \frac{TP_n}{TP_n + FP_n} \tag{13}$$

- **Recall**: The ability of the model to classify the correct instances is called recall. In multiclass classification micro average recall is applied for calculating recall which is the aggregate count of true positive predictions for n classes with the

total of correct predictions of *n* classes. The micro average recall for n classes is given in Eq. (14).

$$Recall\ n\ = \frac{TP_n}{TP_n+FN_n} \quad (14)$$

- **F1-Score:** It gives the harmonic mean of precision and recall providing the balance between the classes. In multiclass classification micro average F1-Score is applied. The micro average F1-Score of n classes is given in Eq. (15).

$$F1-score = 2 \times \frac{precision_n \times recall_n}{precision_n + recall_n} \quad (15)$$

- **ROC-AUC:** Receiver operating characteristics and area under curve is the plot between true positive rate and false negative rate. In multiclass classification for NIDS ROU-AUC is calculated and plotted by applying the approach One-vs-Rest (OvR). In this approach AUC is calculated by aggregating the AUC of all the classes. AUC for multiclass classification is given in the Eq. (16).

$$AUC = \frac{1}{n}\sum_{i=1}^{n} AUC_i \quad (16)$$

- **Multiclass Cross-Entropy loss**: The cross-entropy loss measures the performance of a classification model whose output is a probability value between 0 and 1. Lower the loss value better is the performance of the model. The cross entropy loss equation for multiclass classification is given in the Eq. (17).

$$Loss = -\frac{1}{k}\sum_{i=1}^{k}\sum_{n=1}^{n} y_{in}\ log\ (p_i, n) \quad (17)$$

"*k*" is the number of samples, "*n*" is the number of classes, y_{in} is the binary value 0 or 1, if *n* is the correct classification for sample *i*. P_{in} is the predicted probability that sample I belongs to class *k*.

- **Kappa score:** A statistical measure used to evaluate the agreement between two classifications systems were each classify items into mutually exclusive categories. In multiclass network intrusion detection, the Kappa score is used to assess how well the predicted classifications of network intrusions match the actual classifications across multiple classes. The Kappa score is calculated based on the Eq. (18).

$$k = \frac{P_o-P_e}{1-P_e} \quad (18)$$

where P_o is the is the observed agreement and P_e is the expected agreement. Kappa score more than 0.8 is a strong agreement indicating classifier performs well beyond the random chance. Score between 0.6–0.8 is a moderate agreement and 0.4–0.6 is a weak agreement. Less than 0.4 is a case of no agreement.

V. Results and Discussion

In this section we evaluate the performance of our Proposed Ensemble Model (PEM) for multiclass network intrusion detection system on 3 publically available dataset which has the modern day attacks. The results of PEM on each dataset was compared with 6 conventional algorithms such as DT, XGBoost, LR, AdaBoost, LGBM, RF. These algorithms were selected due to their extensive use in previous research. Default parameters were applied for ease to use. The results and discussion of our proposed model for various datasets incorporated in the study is as follows:

A. Experimental Outcome of UNSWB-NB15 Dataset

Table IV summarizes the experimental values achieved using 6 selected ML algorithms and proposed model using UNSWB-NB15 dataset.

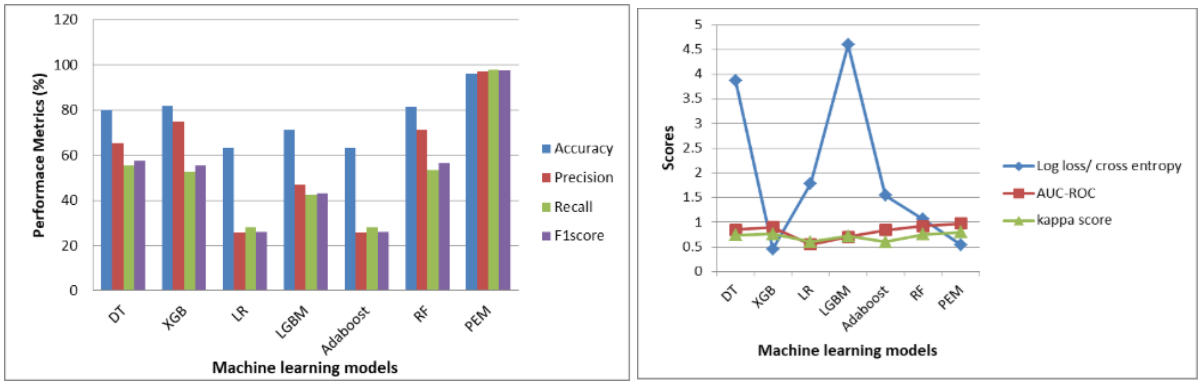
TABLE IV. Performance of ML Classifiers and PEM on UNSWB-NB15 Dataset

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Log loss	AUC-ROC	kappa score
DT	80	65.5	55.3	57.5	3.87	0.85	0.74
XGB	81.8	75	52.6	55.5	0.45	0.9	0.76
LR	63.4	25.5	28.2	26.1	1.78	0.55	0.6
LGBM	71.1	46.8	42.3	43.2	4.6	0.7	0.72
AdaBoost	63.4	25.5	28.2	26.1	1.54	0.84	0.6
RF	81.3	71.2	53.4	56.5	1.06	0.92	0.75
PEM	96	97	98	97.4	0.54	0.97	0.8

Table IV shows that proposed ensemble model has outperformed compared to conventional algorithms in terms Accuracy, Precision, Recall, F1score, Log loss, AUC-ROC and Kappa score. Accuracy of 96 indicates that model has good performance in classification. A precision of 97 which is higher indicates that the model classified 97% of intrusion classes as intrusion. High value of precision indicates very low false positive rates. A Recall of 98 was achieved which is higher compared to conventional algorithm indicates that model classified correctly 98% of instance as intrusions and achieved very low false negative rate. F1-Score of 97.4% indicates that there is a harmonic balance between recall and precision. The Area under the curve and Receiver operating characteristic Curve (AUC-ROC) measures the ability of the model to distinguish between classes. An AUC-ROC of 0.97 indicates a high level of distinction between the positive and negative classes, meaning the model performs well in differentiating between non-intrusion and intrusion instances. In order to check the error in model Log loss was calculated which measures the performance of a classification model by penalizing false classifications. Lower values indicate better performance. A log loss of 0.54 indicates the model's predictions are

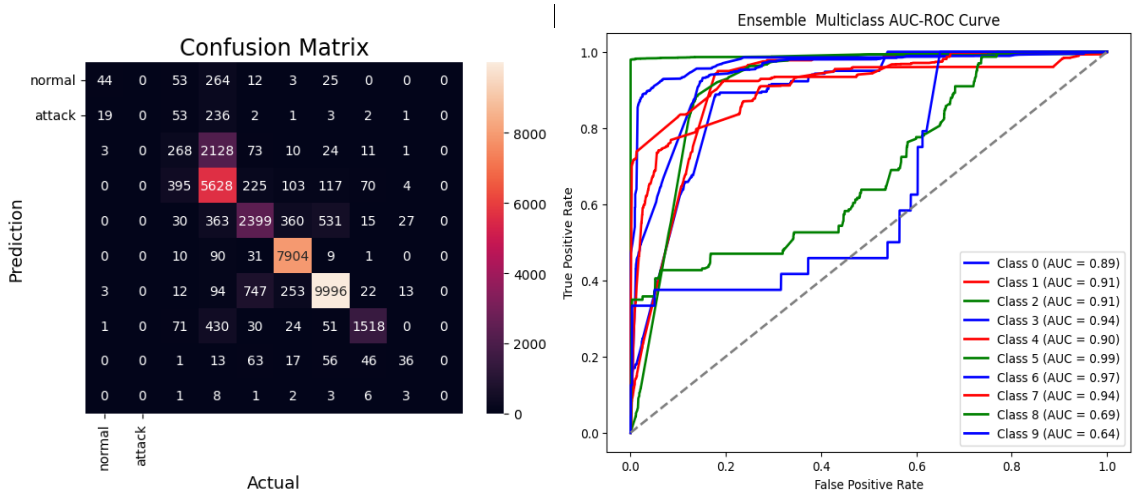
good. A kappa score of 0.8 achieved is a significant value in ML classification. It indicates that the model has a high level of agreement between the predicted classifications and the actual classifications of network intrusions. Fig. 4(a) gives the graphical representation of the performance metrics (accuracy, precision, recall, F1-Score) for the investigated ML algorithms and PEM. Fig. 4(b) gives the performance metrics which was used to test the effectiveness (Log loss, AUC-ROC score and kappa score) of investigated ML algorithm and PEM.

Fig. 5 depicts the confusion matrix and ROC-AUC curve achieved by PEM for multiclass classification. Fig. 5(a) provides the heat map for multiclass detection. The IDS performs well for several attack types, with high true positive counts. However, there are notable misclassifications, particularly between normal traffic and certain attack types, as well as among different attack types. From Fig. 5(b) it can be inferred that the model detection for multiclass intrusion detection appears to perform well overall, with most Classes 3, 5, 6, 7 having high AUC value of 0.90 and moderate value 0.89 to 0.91 for Classes 0, 1, 2, 4. However, for Classes 8 and 9 the model's performance could be improved as the AUC value is low with 0.69 and 0.64.



(a) (b)

Fig. 4. (a) Comparative analysis of Accuracy, Precision, Recall, F1score on UNSWB-NB15 dataset and (b) Comparative analysis of Log loss, AUC-ROC, Kappa score on UNSWB-NB15 dataset.



(a) (b)

Fig. 5. (a) Confusion matrix for UNSWB-NB15 dataset and (b) AUC- ROC curve for UNSWB-NB15 dataset.

B. *Experimental Outcome of CSE-CIC-IDS2018 Dataset*

In this section we discuss the results obtained on testing the model on CSE-CIC-IDS2018 dataset. The Table V provides the summary of evaluated results obtained by ML algorithms and PEM on CSE-CIC-IDS2018 dataset.

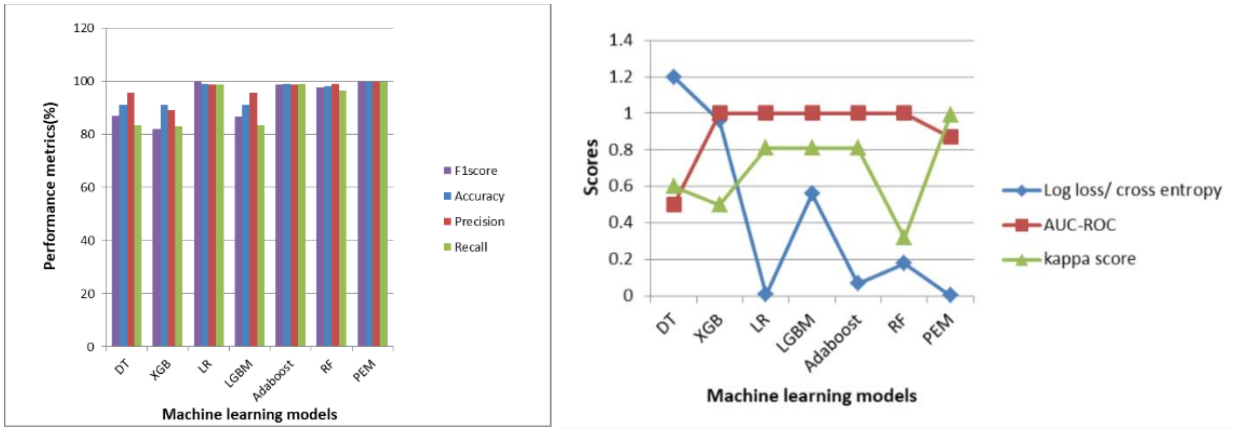
From Table V performance metrics, it can be interpreted that accuracy, precision, recall, F1-Score and AUC-ROC of the LR, AdaBoost and RF has very good performance compared to other conventional algorithm applied however, PEM has performed brilliantly as all the metrics (accuracy, precision, recall and F1-Score) are 99.9 and close to ideal value. Log loss value of PEM is 0.003 which is lowest compared to all the classifiers indicating very less error. AUC-ROC of 0.87 indicates that the model is able to discriminate the positive (intrusion) classes and negative (non-intrusion) classes. A Kappa score of 0.99 obtained is a high agreement between predicted value and expected value. All the obtained performance metrics values of PEM indicate that the ensemble model is performing ideally compared to conventional algorithm.

TABLE V. PERFORMANCE OF ML CLASSIFIERS AND PEM ON CSE-CIC-IDS2018 DATASET

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Log loss	AUC-ROC	kappa score
DT	91.1	95.8	83.4	86.8	1.2	0.5	0.6
XGB	91	89	83	82	0.96	1	0.5
LR	98.9	98.8	98.8	99.8	0.01	1	0.81
LGBM	91	95.7	83.3	86.7	0.56	1	0.81
AdaBoost	98.9	98.8	98.9	98.8	0.07	1	0.81
RF	98.1	99	96.6	97.7	0.18	1	0.32
PEM	99.9	99.8	99.9	99.9	0.003	0.87	0.99

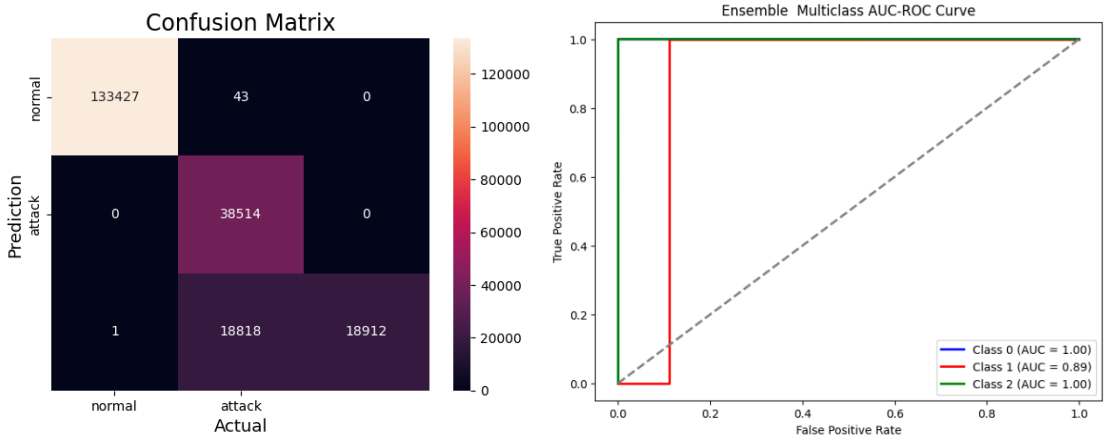
Fig. 6(a) gives the graphical representation of the performance metrics (accuracy, precision, recall, F1-Score) for the investigated ML algorithms and PEM. Fig. 6(b) gives the performance metrics which was used to test the effectiveness (Log loss, AUC-ROC score and kappa score) of investigated ML algorithm and PEM.

In order to show that the model performed effectively on CSE-CIC-IDS2018 dataset we have shown the confusion matrix in Fig. 7(a) and AUC-ROC curve in Fig. 8(b) which is obtained on testing the PEM model for multiclass classification. From Fig. 8(a) it indicates that the model is success in identifying true attacks as there are high TP in contrast a high number of TN was achieved which indicates model's ability to correctly identify normal traffic. There is no false alarm as FP is zero and relatively high FN indicates that model is moderately able to classify different types of attacks. The ROC-AUC from Fig. 7(b) achieved it can be inferred that model was able to perfectly discriminate for Class 0 and Class 2 with an AUC of 1.0 and high discrimination for Class 1 with an AUC of 0.89. Overall it highlights the strengths of model in identifying Class 0 and Class 2 while suggesting potential improvements for Class 1.



(a) (b)

Fig. 6. (a) Comparative analysis of Accuracy, Precision, Recall, F1score on CSE-CIC-IDS2018 dataset and (b) Comparative analysis of Log loss, AUC-ROC, Kappa score on CSE-CIC-IDS2018 dataset.



(a) (b)

Fig. 7. (a) Confusion matrix for CSE-CIC-IDS2018 dataset and (b) AUC- ROC curve for CSE-CIC-IDS2018 dataset.

C. HIKARI-2021 Dataset Result Discussion

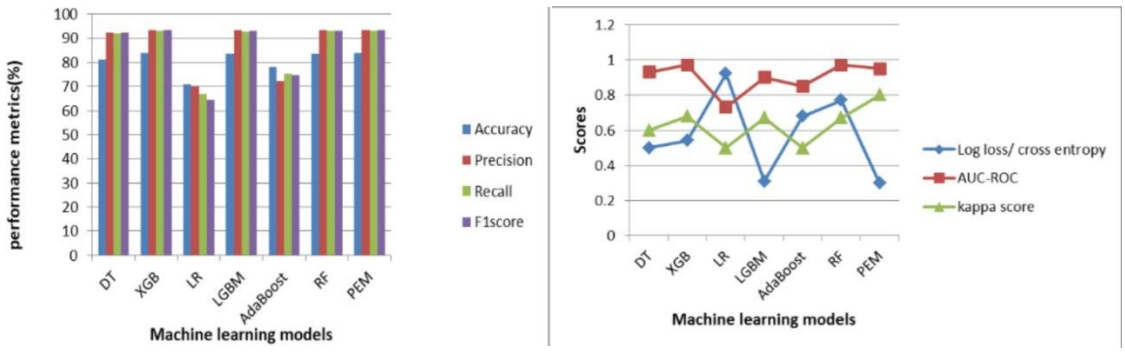
Table VI provides the summary of evaluated results obtained by ML algorithms and PEM on HIKARI-2021 dataset.

Comparing PEM against the ML classifiers, Table VI shows that PEM has done well in terms of performance metrics. The model correctly classified the instances into intrusion and non-intrusion by achieving 83.97. The precision of 93.5% achieved indicates that the model has low false positive rate and correctly classified instances of intrusion. Additionally, the model obtained a recall of 9.32% and detected 93.2% of intrusions with few false negatives. An F1-Score of 93.4% indicates a good balance between precision and recall was achieved. The model has moderate errors in prediction as it achieved lower log loss value of 0.3. An AUC score of 0.95 produced by the model indicates a very high ability to distinguish between intrusion and non-intrusion classes. Kappa score of 0.8 means there is a high agreement between predicted value and exact value. Fig. 8(a) gives the graphical representation of the performance metrics accuracy, precision, recall, F1-Score of 6 conventional ML algorithms and PEM. Fig. 8(b) gives the performance metrics which was used to test the effectiveness (Log loss, AUC-ROC score and kappa score) of investigated 6 ML algorithm and PEM.

In Fig. 9, we provide the confusion matrix and AUC-ROC curve obtained on examining the HIKARI-2021 dataset. From Fig. 9(a), high values of TP and TN indicate that the IDS is quite effective at correctly identifying both normal and attack instances. A significant number of FP and FN indicates there are a considerable number of instances where the IDS misclassifies non-intrusion traffic as intrusion and vice versa. In Fig. 9(b), it indicates that classes 2, 3, 4, and 5 have AUC score of 1.0 which means the model significantly discriminated these intrusion classes from the normal traffic. Classes 0 and 1 have AUC score of 0.86 and 0.87 indicating good performance for these classes.

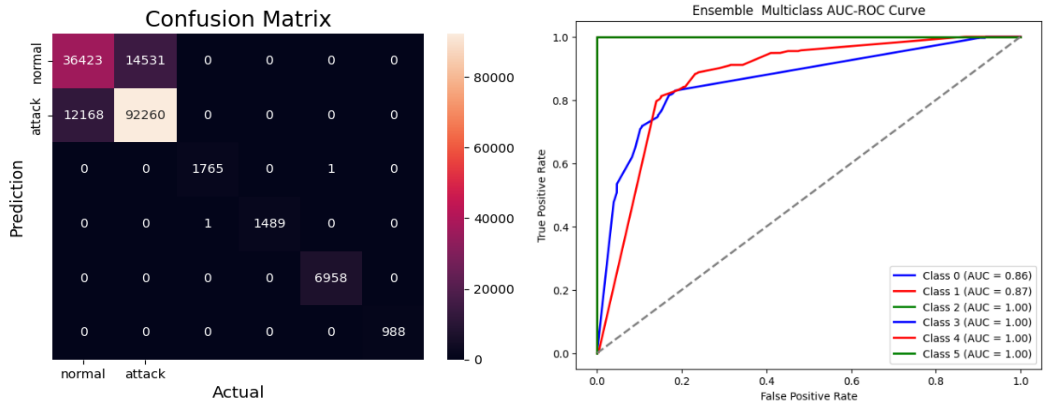
TABLE VI. PERFORMANCE OF ML CLASSIFIERS AND PEM ON HIKARI-2021 DATASET

Algorithms	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Log loss	AUC-ROC	kappa score
DT	81.2	92.4	92.2	92.3	0.5	0.93	0.6
XGB	84	93.6	93.1	93.3	0.54	0.97	0.68
LR	71.1	69.8	66.8	64.5	0.921	0.73	0.5
LGBM	83.7	93.6	92.7	93.1	0.31	0.9	0.67
AdaBoost	78	72.2	75.5	74.6	0.68	0.85	0.5
RF	83.4	93.3	93	93.1	0.77	0.97	0.67
PEM	83.97	93.5	93.2	93.4	0.3	0.95	0.8



(a) (b)

Fig. 8. Comparative analysis of Accuracy, Precision, Recall, F1score on HIKARI-2021 dataset and (b) Comparative analysis of Log loss, AUC-ROC, Kappa score on HIKARI-2021 dataset.



(a) (b)

Fig. 9. (a) Confusion matrix for HIKARI-2021 dataset and (b) AUC- ROC curve for HIKARI-2021 dataset.

D. Comparison of Studied Dataset with the Investigated PEM

In this section we provide the comparison of the experimental results of the ML classifiers and proposed stacking ensemble model (PEM) on the 3 datasets used for study. Table VII provides the comparative results of datasets.

Table VII shows that few ML algorithms have performed well in more than two datasets. But PEM has outperformed when compared with top performing ML classifiers in all datasets. All performance metrics value is relatively good which a model should achieve.

It is very important to compare our PEM results with the state-of-art (SOTA) works. The comparison of SOTA with PEM is given in Table VIII. The performance values which were not evaluated in SOTA are tabulated as Not reported.

TABLE VII. COMPARISON OF TOP PERFORMING ML CLASSIFIERS AND PEM ON 3 DATASETS

Dataset	Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Log loss	AUC-ROC	kappa score
Performance of top performing classifiers on UNSWB-NB15 dataset	XGB	81.8	75	52.6	55.5	0.45	0.9	0.76
	RF	81.3	71.2	53.4	56.5	1.06	0.92	0.75
	PEM	**96**	**97**	**98**	**97.4**	**0.54**	**0.97**	**0.8**
Performance of top performing classifiers on CSE-CIC-IDS2018 dataset	LR	98.9	98.8	98.8	99.8	0.01	1	0.81
	AdaBoost	98.9	98.8	98.9	98.8	0.07	1	0.81
	RF	98.1	99	96.6	97.7	0.18	1	0.32
	PEM	**99.9**	**99.8**	**99.9**	**99.9**	**0.003**	**0.87**	**0.99**
Performance of top performing classifiers on HIKARI-2021 dataset	XGB	84	93.6	93.1	93.3	0.54	0.97	0.68
	LGBM	83.7	93.6	92.7	93.1	0.31	0.9	0.67
	RF	83.4	93.3	93	93.1	0.77	0.97	0.67
	PEM	**83.97**	**93.5**	**93.2**	**93.4**	**0.3**	**0.95**	**0.8**

TABLE VIII. COMPARISON OF SOTA WORKS WITH PEM (X INDICATES NOT REPORTED)

Dataset	References	Methodology	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Log loss	AUC-ROC	kappa score
Comparison of SOTA works with PEM on UNSWB-NB15 dataset	Al-Ambusaidi *et al.* [27]	Improved Random Forest	92.12	92	91	91	X	97.1	X
	Ding *et al.* [3]	Random forest	99.7	99.7	99.7	99.7	X	X	X
	Bukhar *et al.* [28]	stacking ensemble	98.8	97	X	96	X	X	X
	Tahri *et al.* [29]	SVM	92.7	X	X	X	X	X	X
	Ahmed *et al.* [30]	Random forest	95	94.7	95.7	95.1	X	X	X
	Proposed	**PEM**	**96**	**97**	**98**	**97.4**	**0.54**	**0.97**	**0.8**
Comparison of SOTA works with PEM on CSE-CIC-IDS2018 dataset	Wang *et al.* [31]	CNN and LSTM	98.85	98.85	98.85	98.83	X	X	X
	Wang *et al.* [32]	OCSVM and GMM	95.1	96.1	95.65	95.8	X	X	X
	Andrecut [33]	Nearest Neighbour	98.58	96.67	97.15	96.21	X	X	X
	Proposed	**PEM**	**99.9**	**99.8**	**99.9**	**99.9**	**0.003**	**0.87**	**0.99**
Comparison of SOTA works with PEM on HIKARI-2021 dataset	Gu [34]	SPAFIS	92.4	X	X	96.03	X	X	X
	Noori [35]	DFA-GPE-MetaHeuristic	99.09	99.13	99.03	99.07	X	X	X
	Proposed	**PEM**	**83.97**	**93.5**	**93.2**	**93.4**	**0.3**	**0.95**	**0.8**

Table VIII shows the various existing intrusion detection system along with the proposed ensemble stacking model on three datasets which was used in this study. The comparison was done on used metrics such as accuracy, precision, recall, F1-Score, AUC-ROC score, Log loss and Kappa score. From the comparative study for UNSWB-NB15 dataset it can be seen that the proposed model has higher performance compared to the SOTA work [3, 27–30], even though the performance values are high in work [28] but the performance values Log loss, AUC-ROC and Kappa score were not reported which is essential to check the efficiency of the model and PEM performs competitively across various metrics, showing strong performance in precision, recall, F1-Score, AUC-ROC, and Kappa score. The proposed model has excellent performance compared to existing works in literature [31–33] with accuracy, precision, recall, F1-Score 99.9%. The Log loss valve is 0.003 which is very low error indicating models high efficiency. The methodology [34, 35] has outperformed PEM across accuracy, precision, recall, and F1-Score, demonstrating superior performance. While PEM does not achieve the highest scores in these metrics, but it provides additional metrics such as Log Loss, AUC-ROC, and Kappa Score. The additional metrics indicate PEM's reliability and performance, though its overall performance is lower compared to the other methods in terms of accuracy and F1-Score.

From Table VIII, it can be inferred that the proposed stacking ensemble model shows improved performance across all the metrics in most of the cases on studied datasets. A very low Log loss and perfect scores for AUC-ROC and Kappa Score achieved on all datasets indicates that proposed model is robust, reliable and versatile method for detecting network intrusion in comparison to others. The model shows promising results and has potential to detect intrusion in network.

VI. CONCLUSION AND FUTURE WORK

In the field of Network security, the potential of network threats has been increased. Network Intrusion Detection System is essential in detecting these treats. Considering the increased treats, developing a robust NIDS has attracted academicians, researchers due to its ability to detect intrusion efficiently. In this study a

stacking ensemble machine learning model is proposed to detect intrusion in network more effectively. The experiment was carried out on UNSWB-NB15, CSE-CIC-IDS2018 and HIKARI-2021 three latest heterogeneous datasets to validate model performance. Decision tree, XGBoost, Logistic regression, Light GBM, AdaBoost and Random forest were used as base classifiers of the model and Multi layer perceptron was used as Metaclassifiers. After data preprocessing, feature selection was done in three stages. First list of the initial features were selected using Gini index, next list of the features are selected by applying correlation analysis. Random forest was used to select the last list of features. Final features were selected after combining the three list of features. Various performance metrics were used to measure the performance of the proposed model. The model was tested for its performance with six machine learning base classifiers. From simulation outcome, we observed that proposed stacking ensemble model performed more effectively. For CSE-CIC-IDS2018 99.9% accuracy was achieved which is higher result indicating the models ability to classify intrusion. An accuracy of 96% for UNSW-NB15 shows that the model has performed good even in these dataset. Even though the achieved accuracy of HIKARI-2021 is 83.97% which is lower compared to other datasets but the precision of 93.5%, recall of 93.2% and AUC-ROC score of 0.95 indicates strongly that model detects the intrusions precisely. The model was compared with the existing methodologies in state-of-art and shows critical validation. The findings of the study signifies that the model has shown improved performance in detecting intrusion in network by reducing false positive rate and false negative rate. From the different types of dataset applied which has demonstrated excellent results signifies adaptability of the proposed model on various types of datasets. The proposed stacking ensemble model can be considered as an helpful Network Intrusion Detection System to protect the systems from network attacks.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

V.S.B conducted the investigations; V.S.B collected the dataset: V.S.B designed the methodology; V.S.B performed the visualization V.S.B wrote the complete paper. G.H.B performed analysis of data; G.H.B performed validation on the results. All authors have read and approved the final version of the manuscript.

REFERENCES

[1] V. Hnamte and J. Hussain, "DCNNBiLSTM: An efficient hybrid deep learning-based intrusion detection system," *Telematics and Informatics Reports*, vol.10, June 2023.

[2] A. Almomani, I. Akour, A. M. Manasrah, O. Almomani, M. A. Esra, A. Al-Shwait, and R. Al-Sharaa, "Ensemble-based approach for efficient intrusion detection in network traffic," *Intelligent Automation & Soft Computing*, vol. 37, no. 2, June 2023.

[3] W. Ding and S. Popoola, "An ensemble learning voting technique for network intrusion detection systems," Research Square Preprint, June 2023.

[4] P. V. Pandit, S. Bhushan, and P. V. Waje, "Implementation of intrusion detection system using various machine learning approaches with ensemble learning," in *Proc. International Conference on Advancement in Computation & Computer Technologies (InCACCT)*, Gharuan, India, May 2023.

[5] A. K. Dasari, S. K. Biswas, S. Sanyal, and B. Purkayastha, "Intrusion detection system using ensemble learning analytics," in *Proc. 2023 IEEE 8th International Conference for Convergence in Technology (I2CT)*, Lonavla, India, April 2023.

[6] V. Ramachandran, Y. Anuhya, V. Lakshmi, R. Pravallika, V. Makke, V. Venkata, and L. S. Srikar, "Intrusion detection using an ensemble deep learning approach," *International Journal of Food and Nutritional Sciences*, vol. 11, pp. 1872–1880, Dec 2023.

[7] Z. Zhang, S. Kong, T. Xiao, and A. Yang, "A network intrusion detection method based on bagging ensemble," *Symmetry*, vol. 16, no. 7, 2024.

[8] A. Kiflay, A. Tsokanos, M. Fazlali, and R. Kirner, "Network intrusion detection leveraging multimodal features," *Array*, vol. 22, 2024.

[9] M. A. Lochanan, I. Chebolu, and C. A. S. Murty, "Network intrusion detection using multi-modal ensemble modeling," in *Proc. Second International Conference on Emerging Trends in Information Technology and Engineering (ICETITE)*, 2024.

[10] D. P. Gaikwad, V. Nagrale, and M. P. Bauskar, "Ensemble of learner for network intrusion detection system," *Journal of Network Security Computer Networks*, vol. 9, no. 1, 2023.

[11] M. M. Otoom, K. N. A. Sattar, and M. Al-Sadig, "Ensemble model for network intrusion detection system based on bagging using J48," *Advances in Science and Technology Research Journal*, vol. 17, no. 2, pp. 322–329, 2023.

[12] A. C. Zewdu and H. K. Kumssa, "An ensemble method for supervised learning for intrusion detection and network forensics," in *The Role of Cybersecurity in the Industry 5.0 Era*, IntechOpen. 2024.

[13] B. A. K. Hammood and A. T. Sadiq, "Ensemble machine learning approach for IOT Intrusion detection systems," *Iraqi Journal for Computers and Informatics*, vol. 99, no. 2, 2023.

[14] A. Das and B. S. Sunitha, "Anomaly-based network intrusion detection using ensemble machine learning approach," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 2, 2022.

[15] J. Zhu and X. Liu, "An integrated intrusion detection framework based on subspace clustering and ensemble learning," *Computers and Electrical Engineering*, 115, 2024.

[16] C. Fan, M. Chen, X. Wang, J. Wang, and B. Huang, "A review on data preprocessing techniques toward efficient and reliable knowledge discovery from building operational data," *Frontiers in Energy Research*, vol. 9, March 2021.

[17] J. Huang, Y.-F. Li, and M. Xie, "A systematic analysis of data preprocessing for machine learning-based software cost estimation," *Information and Software Technology*, 67, pp. 108–127, 2021.

[18] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research (JAIR)*, vol. 16, 2002.

[19] R.-C. Chen, C. Dewi, S.-W. Huang, and R. E. Caraka, "Selecting critical features for data classification based on machine learning methods," *Journal of Big Data*, 52, July 2020. https://doi.org/10.1186/s40537-020-00327-4

[20] H. Liu, M. Zhou, X. S. Lu, and C. Yao, "Weighted gini index feature selection method for imbalanced data," in *Proc. 15th International Conference on Networking, Sensing and Control (ICNSC)*, March 2018. https://doi: 10.1109/ICNSC.2018.8361371

[21] F. Wang, Y. Yang, X. Lv, J. Xu, and L. Li, "Feature selection using feature ranking, correlation analysis and chaotic binary particle swarm optimization," in *Proc. IEEE 5th International Conference on Software Engineering and Service Sciences (ICSESS)*, 2014. https://doi.org/10.1109/ICSESS.2014.6933569

[22] Y. Qi, "Random forest for bioinformatics," in *Ensemble Machine Learning*, Springer, US, 2012, pp. 307–323.

[23] N. Martindale, N. Martindale, and N. Martindale, "Ensemble-based online machine learning algorithms for network intrusion

detection systems using streaming data," *Information*, vol. 11, no. 6, June 2020. https://doi.org/10.3390/info11060315
[24] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. 2015 Military Communications and Information Systems Conference (MilCIS)*, Canberra, Australia, 2015.
[25] A Realistic Cyber Defense Dataset (CSE-CIC-IDS2018). [Online]. Available: https://registry.opendata.aws/cse-cic-ids2018
[26] A. Ferriyan, A. H. Thamrin, K. Takeda, and J. Murai, "HIKARI-2021: Generating network intrusion detection dataset based on real and encrypted synthetic attack traffic," *Applied Sciences*, vol. 11, 17, 2022.
[27] M. Al-Ambusaidi, Z. Yinjun, Y. Muhammad, and A. Yahya, "ML-IDS: An efficient ML-enabled intrusion detection system for securing IoT networks and applications," *Neural Networks*, vol. 28, pp. 1765–1784, 2023.
[28] O. Bukhar *et al*., "Anomaly detection using ensemble techniques for boosting the security of intrusion detection system," *Procedia Computer Science*, vol. 218, pp. 1003–1013, 2023.
[29] R. Tahri1, Y. Balouki, A. Jarrar, and A. Lasbahani, "Intrusion detection system using machine learning algorithms," in *Proc. ITM Web of Conferences*, EDP Sciences, 2022, vol. 46.
[30] H. A. Ahmed, A. Hameed, and N. Z. Bawany, "Network intrusion detection using oversampling technique and machine learning algorithms," *PeerJ Computer Science*, 2022.
[31] Y.-C. Wang, Y.-C. Houng, H.-X. Chen, and S.-M. Tseng, "Network anomaly intrusion detection based on deep learning approach," *Sensors*, vol. 3, 2023.
[32] C. Wang, Y. Sun, S. Lv, C. Wang, H. Liu, and B. Wang, "Intrusion detection system based on one-class support vector machine and gaussian mixture model," *Electronics*, vol. 12, no. 4, 2023.
[33] M. Andrecut, "Attack vs benign network intrusion traffic classification," arXiv preprint, arXiv:2205.07323, 2022.
[34] X. Gu, G. Howells, and H. Yuan, "A soft prototype-based autonomous fuzzy inference system for network intrusion detection," *Information Sciences*, 2024.
[35] M. S. N. *et al*., "Feature drift aware for intrusion detection system using developed variable length particle swarm optimization in data stream," *IEEE Access*, 2023.