Ensemble Learning based Intrusion Detection System

Mahitha Sangem [a], Vineeth Balusani [b], Surya Kiran Suresh [c], U Aditya Karanth [d]

[a] Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada, mahithasangem@cmail.carleton.ca

[b] Department of Electrical and Computer Engineering, University of Ottawa, Ottawa, Canada, vbalu017@uottawa.ca

[c] Department of Computer Science, University of Ottawa, Ottawa, Canada, ssure044@uottawa.ca

[d] Department of Electrical and Computer Engineering, University of Ottawa, Ottawa, Canada, auppu062@uottawa.ca

Abstract

Over the years, several researchers have made efforts to develop intrusion detection systems for classifying network traffic as malicious or not using classical machine learning techniques including ensemble learning, as well as state-of-the-art methods using deep learning strategies like Multi-Layer Perceptron (MLP) and Long Short-Term Memory (LSTM). Despite having ensembles and state-of-the-art techniques, the intrusion detection systems still suffer from the false alarm rate, thereby encouraging the researchers to build more efficient systems for the classification task. The main goal of this project is to develop a novel ensemble learning-based intrusion detection system that correctly classifies the network traffic while keeping the false alarm rate low.

Keywords

Intrusion Detection; Machine Learning; Ensemble Learning; Deep Learning; False Alarm Rate

1. Introduction

In recent times, there has been a tremendous rise in computer network usage and an enormous increase in the number of applications running on it and especially with the COVID-19 outbreak, more and more businesses have invested in moving their operations online thereby creating more opportunities for the cyber-attackers to exploit the security flaws in the network systems and potentially escalating assaults that can cause both financial losses as well as information loss. Although many organizations use the firewall to block traffic based on some of the predetermined security rules defined by the organization, it merely acts as the first line of defense and the attackers can easily bypass the firewall. As a result, a more robust network intrusion detection system that can detect vulnerabilities in the incoming network traffic is necessary to prevent any kind of loss to the organizations.

An intrusion detection system (IDS) is a system that monitors the network traffic for suspicious activity and alerts the administrators or the other responsible teams upon the discovery of such suspicious activity. In general, there are two types of IDS – signature-based and anomaly-based. Before anomaly-based systems became popular, several organizations used signature-based systems that would require them to know the attack patterns which the attackers use, and they were able to detect only the known patterns of attack by monitoring the incoming network traffic (Yousefnezhad et al., 2021). However, because it is impossible to keep track of all the patterns of attack because new

patterns occur all the time, anomaly-based systems that employ Machine Learning have emerged as a solution. The developed system should have low generalization error, low false positive rate, high adaptability, and high scalability while developing an efficient IDS (Kumar et al., 2020). Furthermore, developing a system that can identify all threats with a single machine learning model is extremely challenging, necessitating the use of several machine learning models and integrating the predictive power of those various algorithms to develop the IDS system. Therefore, ensemble modeling is a Machine Learning method in which multiple distinct Machine Learning models are deliberately combined to forecast the outcome by using the information of these base learners. The system's false alarm rate decreases as long as the base learners are diversified and independent. Even though the ensemble model has numerous base learners, it functions and performs as a single model. In general, the ensemble models can be categorized as homogeneous, where all the base learners are of similar type, and heterogeneous, where all the base learners are of different kinds. Furthermore, the ensembles can also be categorized as Bagging, Boosting, Stacking, and Cascading models. In this research, we have come up with a novel strategy for using these ensembles that would improve the efficiency of the IDS in predicting malicious traffic while having a low false alarm rate.

2. Related Work

For the development of an Intrusion Detection System, several machine-learning paradigms, including ensemble learning like Stacking, Bagging, Boosting, and some Deep Learning models have been studied.

Different Machine Learning algorithms namely Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Logistic Regression, Naïve Bayes, Multi-Layer Perceptron, Random Forest, Extra Tree, and Decision Tree have been used in the research conducted by Abrar et al. (2020) to test and compare their performance on the NSL-KDD dataset (Canadian Institute for Cybersecurity | UNB, n.d.). They survey the results of the aforementioned machine learning models on 4 different datasets derived from a subset of attributes from the NSL-KDD dataset. The authors post their results based on metrics such as Precision, Recall, f1-score, and accuracy. The authors discuss the class imbalance in the NSL-KDD dataset which leads to poor results for a few classes. The classifiers generate promising results for Denial-of-service attack (DDoS) attacks whereas the least accurate results were produced by U2R. This paper gives us a good comparison between the most common ML models on the NSL-KDD dataset. However, the feature selection methodology to categorize the features into four different datasets used by the authors is vaguely discussed. Also, the authors haven't tried to use any optimization techniques to further improve the prediction power. An ensemble-based approach, which combines the output of various algorithms to predict the final results, could be a possible improvement.

The authors, Ahmad et al. (2018) have suggested an IDS for analyzing large amounts of traffic data utilizing well-known machine learning algorithms such as SVM with Linear, Radial Basis Function (RBF) Kernels, Random Forest (RF), and extreme learning machine (ELM) using the standard dataset NSL-KDD. The dataset has been separated into three sections by the authors: the entire dataset, the half dataset, and the quarter dataset. These datasets are then split into two distinct datasets that are 80% training and 20% testing and 90% training and 10% testing, respectively. The performance on both train-test splits is quite close. The strength of this research is the author's insights into how models' performance changes with different dataset sizes and data splitting. The author's failure to provide the

parameters considered for the models and the fact that the preprocessing stages solely include removing non-numeric characteristics are the paper's shortcomings. Despite a short training period, ELM takes a considerable time to evaluate. The effectiveness of ELM in feature selection and feature transformation approaches may thus be investigated further. As evaluation metrics, accuracy, precision, and recall are employed. In terms of accuracy, precision, and recall, ELM surpasses other techniques on the entire dataset of 65k samples. SVM (Linear) has higher values than SVM (RBF). Additionally, the SVM (Linear) showed superior outcomes in half and a quarter of the data samples, with RF showing the lowest results.

In an attempt to examine the Machine Learning ensembles for intrusion detection systems, Kumar et al. (2020) analyzed different tactics that have been proposed in this field of study and described the various methodologies employed and the results obtained. The authors of the research have also clearly demonstrated why ensembles should be considered while developing an efficient intrusion detection system. They have categorized the ensemble learning methods as Bagging (Random Forest and Wagging), Boosting, and Stacking. Furthermore, they have provided the categorization of ensemble integration methods such as Majority Voting, Threshold plurality voting, and Boolean combination method, and clearly explained how those integrations would work. Based on the research they have reviewed; the authors also summarized the results of the various results obtained by other researchers and made them available in the form of a comparative summary of homogeneous and heterogeneous classifiers. The dataset used and the evaluation metric for each of the methods are also clearly tabulated. Most of the researchers have used the KDD'99 dataset (Tavallaee et al., 2009) and used metrics like Accuracy, False Alarm Rate, F1-Score, Precision, and Recall for the evaluation. The main drawback of this paper is that the authors have not critically commented on the strengths, weaknesses, and limitations of the different approaches offered.

In the research conducted by Govindarajan (2016), the use of homogenous ensemble classifiers and heterogenous ensemble classifiers is discussed for building the IDS. For developing the system and evaluating against the known models the author has approached the research in three main parts, which are preprocessing, classification, and combining phases. Furthermore, the author has proposed new ensemble techniques under the homogenous category which uses the Bagging of Radial Basis Function (RBF) and Support Vector Machine (SVM), and under the heterogeneous category which uses the arcing classifier to develop a hybrid RBF-SVM model, and the evaluation for these models was done against some of the known standard homogeneous and heterogeneous which includes Error-correcting output codes (ECOC), Dagging, Weighted Majority Voting, and Stacking using a couple of datasets where NSL-KDD is one of them. Furthermore, the RBF-SVM bagged model has outperformed the ECOC and Dagging, while the hybrid RBF-SVM performed significantly better than voting and stacking schemes on RBF-SVM. Although the author covers the majority of aspects concerning the proposed models, he doesn't provide a discussion about the models' merits, shortcomings, or limitations.

Mkuzangwe et al. (2017) demonstrated that the performance bound of the IDS which is built using ensembles can be obtained even before building it. Hence, researchers would be able to estimate the system's performance without even implementing it based on the knowledge of the obtained bound. The performance bound is defined and calculated by the authors as the average information gain associated with the subset of features, which is calculated by employing

the Adaboosted decision stump as a weak learner in the ensemble. The researchers carried out the experiments using the NSL-KDD dataset which has been filtered out to consider only normal and Neptune attack types. It has been observed that the accuracy of the ensemble model using the Adaboosted decision stub is 90% if the features selected are such that the information gain value lies between 0.04561 and 0.25615. The biggest positive outcome of the research is that one would be able to estimate the performance even before building the system, however, the limitation is that one should consider several possible combinations of features for each of the decision stumps in the ensemble model, which is not efficient in terms of time.

Nguyen et al. (2021) proposed an ensemble feature selection method for Machine learning-based Intrusion Detection Systems. They leverage the properties of Bagging and Boosting algorithms to improve the accuracy of the machine learning models using a lesser number of features leading to an improved training and prediction time. They use Random Forest and AdaBoost to perform feature selection and prediction on NSL- KDD, UNSW-NB15, and CICIDS-2017 datasets. It is reported that there is a drastic increase in accuracy and training time for both the UNSW-NB15 dataset and NSL-KDD dataset using their feature selection technique, whereas the CICIDS-2017 dataset doesn't produce great results in terms of accuracy. They claim to have used 20, 14, and 14 features to obtain an accuracy of 99.26%, 77.74%, and 98.9% for CICIDS-2017, UNSW-NB15, and NSL-KDD datasets respectively. This proposed feature selection technique has significantly reduced the training time and the number of features required by the algorithm to make equally accurate results. On average the features were reduced to one-third of the original number of features.

Seth et al. (2021) focused on building a new ensemble learning for a multi-class intrusion detection model to detect several attack categories efficiently. The authors have chosen the latest CIC IDS 2018 dataset, which reflected the latest attack trends to conduct the research. As the data is highly skewed, researchers to address the data imbalance issue the SMOTE statistical technique is used for optimizing the training. The methodology employs the ranking of several classifiers like Random Forest, KNN, XGBoost, Light gradient boosting machine (LGBM), Decision Tree, Extra Tree, and Histogram-based Gradient Descent (HBGM) by their f1-scores on detection ability of specific attack categories. The LGBM and HBGB algorithms are shown to perform better than the other algorithms in terms of attack detection as well as detection time. When compared to the performance of base learners and other techniques such as Deep Learning and Voting ensemble, the results produced have an accuracy of 96.97%, and a high recall rate of 97.4%. The biggest strength of this research is that it utilizes novel attack patterns for building an efficient ranking-based ensemble and it does not limit to a single ML algorithm, for effective attack detection.

A feature-selection and ensemble model-based IDS was proposed by Pham et al. (2018). For feature selection, the Naive Bayes classifier and "leave-one-out" methods were used to obtain 25 features and the Gain Ratio (GR) approach was used to obtain 35 attributes from the NSL-KDD dataset. Using the baseline classifiers from J48, Random Forest, Decision Stump, and REPTree, two ensemble models were constructed based on bagging and boosting. The classification accuracy and FAR are the metrics employed for measuring the models' performance and the results have outperformed the previously existing methodologies. The authors could have analyzed the strategy on more datasets, and since all the parameters of each model were left at their default configurations, they could have tried

hyperparameter tuning. The bagging model, which employed J48 as the base classifier and operated on a subset of 35 features, generated the best results, with 84.25% accuracy and 2.79% FAR.

In an attempt to build an efficient IDS using an ensemble-based classifier, Yousefnezhad et al. (2021) in their research have proposed the combination of several different configurations of SVM and KNN models as base learners for creating diversity. Furthermore, the authors have used a deep learning-based autoencoder network which generally consists of three layers namely input, output, and hidden layer for extracting features from the data and thereby obtaining compact features from the original features. The KNN classifiers are trained independently with the number of nearest neighbors as 3, 5, 8, and 10, while the SVM classifiers are trained individually with RBF (degree = 1 and 3) and Hermit (degree = 8 and 10) kernels. The classifiers are made to output the probability scores by using a heuristic function based on the posterior probability estimation and sigmoid function on KNN and SVM respectively. In the last stage, the Dempster-Shafer technique, which is considered to be the most powerful data integration method, is used to aggregate the probability outputs of the base learners to form the ensemble's final prediction. For evaluating the designed scheme, the authors have considered various benchmarking datasets and NSL-KDD is one of them. Accuracy, Precision, Recall, and F-measure are the metrics that are considered for evaluating the ensemble and the performance of the designed strategy in most classes of attacks is higher than the previously existing methods. One of the major limitations of the system is that the time taken for obtaining the probability output for each classifier is more while the performance of the sigmoid and heuristic methods is low. Furthermore, there was no discussion made on why probabilistic models like Gaussian Naïve Bayes, which can inherently output a probability score, were not preferred as base learners.

Zhou et al. (2019) propose a novel ensemble system based on modified adaptive boosting with the area under the curve (M-AdaBoost-A) to identify intrusions in networks efficiently. The proposed algorithm has 2 flavors that are M-AdaBoost-A-SMV and M-AdaBoost-A-PSO based on simple majority voting and particle swarm optimization respectively and how the expert learners are ensembled differs in both flavors. The authors claim themselves to be the first ones to use the M-AdaBoost-A algorithm which is an enhanced version of the AdaBoost-A concept that was developed to perform well on binary classification on smaller datasets. AdaBoost-A is adapted to perform well on multiclass classification using the one-versus-rest methodology. The authors compare the performance of their proposed algorithm with AdaBoost-A, SAMME, and M-AdaBoost-A-EL (expert learners) algorithms based on various metrics such as Accuracy, TPR/FPR ratio, G-Mean, and Macro averages of Precision, Recall, and f1-score on AWID and NSL-KDD datasets. It is identified that the proposed algorithm outperforms all the other algorithms, and they achieve this without performing any of the data preprocessing steps. The algorithm inherently handles class imbalance very well and produces excellent results.

Rani et al. (2022) intended to develop an IDS utilizing ML algorithms for multi-class classification on complicated data while assessing network traffic patterns in the IoT environment in their research. On openly available TON-IoT datasets, the researchers have implemented ensemble learning algorithms using Bagging (Random Forest) and Boosting (Gradient boosting, XGBoost, LGBM, HistGradient classifiers) techniques to focus on the improvement of two main metrics which are accuracy of prediction and speed of prediction. Along with accuracy and speed, this paper

presents the performances and their comparison of all the mentioned classifiers based on the metrics True Positive Rate (TPR), False Positive Rate (FPR), precision, recall, and f1-score. Furthermore, the algorithm tuning helped in achieving the best accuracy for all the classifiers implemented on all datasets. While the Random Forest has shown the highest accuracy of 94-96% on all datasets, it is slower with run-time and requires high computational power. Furthermore, the LGBM classifier is faster with a runtime of 5 to 7 seconds, it has the highest ROC_AUC score, and it is comparatively less accurate than the Random Forest. Since the authors valued the execution time as the most important metric, the LGBM classifier is concluded as the most efficient. Unlike most of the researchers, the authors of this research have not only focused on accuracy but also on the speed of the execution. The researchers also focus on multi-class classification rather than the mere binary classification even on complex data.

Gangula et al. (2022) suggested an IDS based on the improved flower pollination algorithm (EFPA) and ensemble classification methodology to address problems with previous methods that solely depend on human traffic characteristics, increasing system complexity and lowering detection rates on big traffic datasets. The authors used EFPA to choose the best collection of features from the UNSW-NB15 and NSL-KDD datasets. For improved convergence and optimal feature selection, the standard FPA in the EFPA employs a scaling factor. The ensemble classifier seeks to train a variety of classifiers, including support vector machine, decision tree, and random forest classifiers and then selects the top outcomes. The proposed ensemble-based EFPA model achieved an accuracy of 99.32% and 99.67% respectively, and these results are better than those of the conventional methods. In this work, two preprocessing approaches called data cleaning and the min-max normalization technique are utilized to enhance the integrity of raw data. Precision, false acceptance rate (FAR), recall, accuracy, area under the curve (AUC), false omission rate (FOR), and F-measure are the performance measures used. Some of the study's advantages include the proposed and current models being compared by deploying a unique intrusion detection model based on enhanced genetic algorithms and DBN. According to the investigation, the suggested ensemble-based EFPA model outperformed DBN with enhanced genetic algorithms by almost 0.67%–7.49% in terms of intrusion categorization for each class (DoS, probe, R2L, and U2R). Some limitations are as RF requires a lot of training time and DT is generally unstable when compared to other decision predictors, additional ensemble models can also be employed in place of RF, DT, and SVM. Further solutions can be found for optimization problems.

Choi et al. (2019) have used an unsupervised learning algorithm autoencoder to create and test a network intrusion detection system. The model outperformed earlier research that used cluster analysis techniques to reach an accuracy of 80% and obtained 91.70% accuracy. The findings offer a useful roadmap for creating autoencoder-based network intrusion detection systems. The authors have clearly explained the two types of network intrusion detection systems (NIDS) and further briefed about the two-anomaly detection-based NIDSs along with difficulties in acquiring attack-labeled data for the development of supervised learning-based NIDS. Further strengths of this study are the authors have also performed data preprocessing and feature engineering. Using the preprocessed NSL KDD data set, the training data set and test data set were built, with 85% of the training data set and 15% of the test data set each being used. To mimic actual network circumstances, the training data set was resampled so that the fraction of anomalous data was 1%, 5%, and 10%. They utilized three hidden layers with 32, 16, and 32 units each in the stacked autoencoder and variational autoencoder models and also specified hyperparameters. Because the NSL-KDD data set is the only

one used to train models, this model's validity is confined to this dataset and employs only a single type of autoencoder-based NIDS for unsupervised learning. Therefore, it is still to be determined if alternative unsupervised learning methods can be used.

A comparison study of Machine Learning (ML) and Deep Learning (DL) algorithms for binary classification, based on attack detection rate is done by Rawat et al. (2022). As part of the classical machine learning classifiers, Decision Trees, Extra Tree, Ensemble Extra Trees, and LightGBM models are used, while Deep Neural Network is used as part of the Deep Learning algorithms. The algorithms use the implementation of an unsupervised feature selection method. The NSL-KDD dataset is considered to carry out the research and the authors also obtained the minimized subset of features from this dataset using the strategy of Software Defined Networking Variables (SDN). While other classifiers performed better with the original data, the DNN did well with the minimized feature set. When PCA is used on all the data features and fed into the DNN, the performance was better comparatively. The DNN accuracy without PCA and PCA is 77.2% and 79.3% respectively. Hence, the authors conclude that DNN performs better with PCA-based feature selection. The biggest strength is that the DNN is even able to detect new types of intrusions in the test data that are not part of the train data. However, the limitation is that the authors haven't employed hyperparameter tuning of Classical Machine Learning models or the DNN, hence it is not appropriate to compare ML and DL algorithms without it.

3. Dataset

The NSL-KDD dataset (Canadian Institute for Cybersecurity | UNB, n.d.) is known to overcome the problems of data redundancy and bias towards certain sorts of attacks that are prominent in the KDD'99 dataset (Tavallaee et al., 2009). Furthermore, with the detailed categorizing of the attack types in the anomalous data, the difficulty level in detecting anomalies was reduced, hence, this dataset is considered to be a good benchmarking dataset and has been widely used by several researchers to develop Intrusion Detection Systems for real-world applications. As a result, we used the NSL-KDD dataset in the creation and evaluation of the Intrusion Detection System.

The dataset comes with both train data and test data. There are 125973 samples in the train data whereas the test data comprises 22544 samples. Each of the records in the train and test data consists of 42 attributes where the first 41 attributes correspond to the independent variables (features) and the last attribute corresponds to the classification of the input traffic data. These 41 features can be further categorized into the following distribution - 4 Categorical, 23 Discrete, 10 Continuous, and 6 Binary features. The categorical features have to be converted to binary using one-hot encoding before one starts with the modeling of the data.

Furthermore, these features can also be categorized as Intrinsic, Content, Time-based, and Host-based features.

Intrinsic Features (Features 1 to 9) – Intrinsic Features are the features that are derived from the header of the packet, and they hold the basic information about the packet.

Content Features (Features 10 to 22) – Content features are the features that consist of the information which is mandatory to access the payload information.

Time-based Features (Features 23 to 31) – Time-based features are the features that contain information about counts rather than content in the traffic input, like the number of connection attempts made to a particular host over a 2-second window.

Host-based Features (Features 32 to 41) – Host-based features are kind of similar to Time-based features, instead of analyzing the connection attempts over the 2-second window, these features are derived by analyzing the number of requests made to the particular host over n number of connections.

The training data is imbalanced, with 67343 normal instances and the other 58630 instances belonging to different kinds of attacks, where the majority of the attacks are of type Neptune. All the attack instances are combined into the same class, hence converting the multi-class classification problem to a binary classification problem where the designed algorithm would predict the packet as an attack or normal. As a result of grouping all occurrences of an attack into one class, the multi-class classification issue is now a binary classification problem, with the developed algorithm predicting whether a packet obtained from network traffic is an attack or not.

4. Methodology and Experimental Setup

In this project, the NSL-KDD train data is used for training using 5-fold cross validations, while the NSL-KDD test data is used for evaluating the performance of the developed models.

As shown in Figure 1, before we began modeling the data, we performed data cleaning and pre-processing on the train data by checking for duplicates and missing values and then dealing with outliers. Furthermore, because the categorical features flag and service have 11 and 70 unique values, respectively, the target encoding is used to convert these categorical features into numerical features, whereas one-hot encoding is used on the categorical feature protocol type, which has only three unique values. In addition to the aforementioned methods, the data is normalized, and a few features are filtered out using variance thresholding and Pearson's correlation coefficient (PCC) with a threshold of 0.9 using the following logic - if A and B are two features and have PCC(A, B) greater than 0.9, as follows: if A and B are two features and have PCC(A, B) greater than 0.9, filter out feature A if PCC(A, Target label) is less than PCC(B, Target label), otherwise filter out feature B. Following the logic, we retained only one of the pair of highly correlated features; the feature retained is the one that is highly correlated with the Target label.

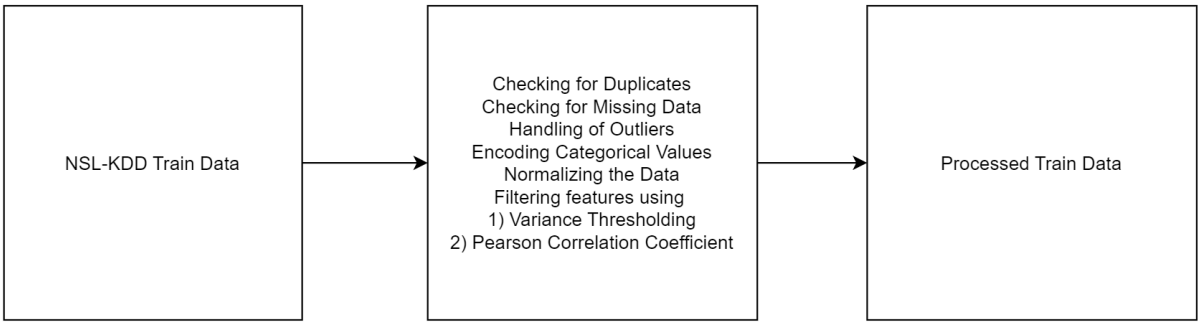

Fig 1. Data Cleaning, Pre-Processing, and Transformation of Train Data

After the training data is processed using the discussed steps, the methodology shown in Figure 2 has been used to train each of the four base learners that are commonly found from the literature review, which are Decision Tree, Naïve Bayes, Logistic Regression, and K-Nearest Neighbor models (Yousefnezhad et al., 2021; Abrar et al., 2020; Seth et al., 2021; Pham et al., 2018; Gangula et al., 2022).

Decision Tree - Decision trees are often used in intrusion detection systems because they can automatically learn from data and provide interpretability. This means that they can identify complex patterns in network behavior and provide insight into the reasons behind a decision, which can help security analysts understand and address potential threats. To make a prediction, a new set of network data is fed through the tree, following the decision path at each node until a leaf node is reached. The prediction is then based on the majority class of the data at that leaf node.

Naïve Bayes - Naive Bayes is a machine learning algorithm that is commonly used for intrusion detection because it is simple and easy to implement. It can make predictions based on known probabilities of different types of network behavior, which can be useful for identifying potential threats. However, it assumes that the presence of one feature is independent of the presence of any other feature, which is often not the case in practice. This assumption can limit the predictive power of the algorithm.

Logistic Regression - Logistic regression is another commonly used machine learning algorithm for intrusion detection because it is simple and efficient, and it provides a probabilistic interpretation of the predictions. However, it assumes a linear relationship between the dependent and independent variables, which is not always the case in reality. This can limit the predictive power of the algorithm.

K-Nearest Neighbors – K Nearest Neighbors (KNN) is another machine-learning approach that is often used for intrusion detection. It makes predictions based on the known labels of the "nearest neighbors" to a new data point,

which can provide a level of interpretability. However, it can be computationally expensive, and it can be sensitive to the choice of the value of k, which can affect the accuracy of the prediction.

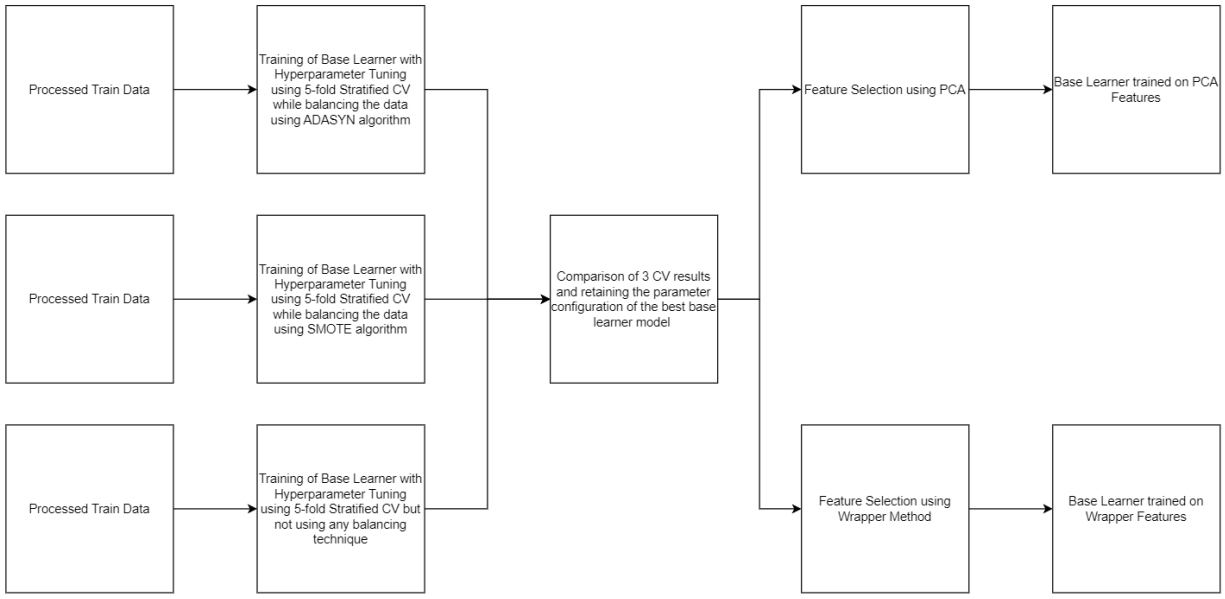

Fig 2. Strategy for obtaining the base learners' results

To perform the training of each of the base learners' we have used the following steps.

Step I: Obtain the processed train data and use it to train the base learner while tuning the hyperparameters with the 5-fold CV and handling the imbalance using the ADASYN algorithm in each of the folds.

Step II: Repeat Step I using SMOTE algorithm instead of the ADASYN algorithm in all the 5-folds.

Step III: Repeat Step I without using any balancing technique in all the 5-folds.

Step IV: Evaluate and compare the cross-validation results of the models' performance obtained in Steps I, II, and III.

Step V: Retain the best base learner's parameter configuration for the next steps.

Step VI: Perform the Feature Selection using Principal Component Analysis (PCA) and obtain the base learner trained on PCA features

Step VII: Repeat Step VI using the Wrapper Method instead of PCA and obtain the base learner trained on Wrapper features.

As shown in Figure 3, once all the base learners are obtained, these are used as the input to several independent meta-classifiers and obtain predictions for each of the different stacking classifiers. The stacking classifiers used are Random Forest, AdaBoost, XGBoost, and Gradient-Boosted Decision Trees (GBDT) as these are the ones that have been commonly used as the stacking classifiers found during the literature review (Kumar et al., 2021; Abrar et al., 2020; Ahmad et al., 2018; Mkuzangwe et al., 2017; Nguyen et al., 2021; Seth et al., 2021; Zhou et al., 2019; Rani et al., 2022).

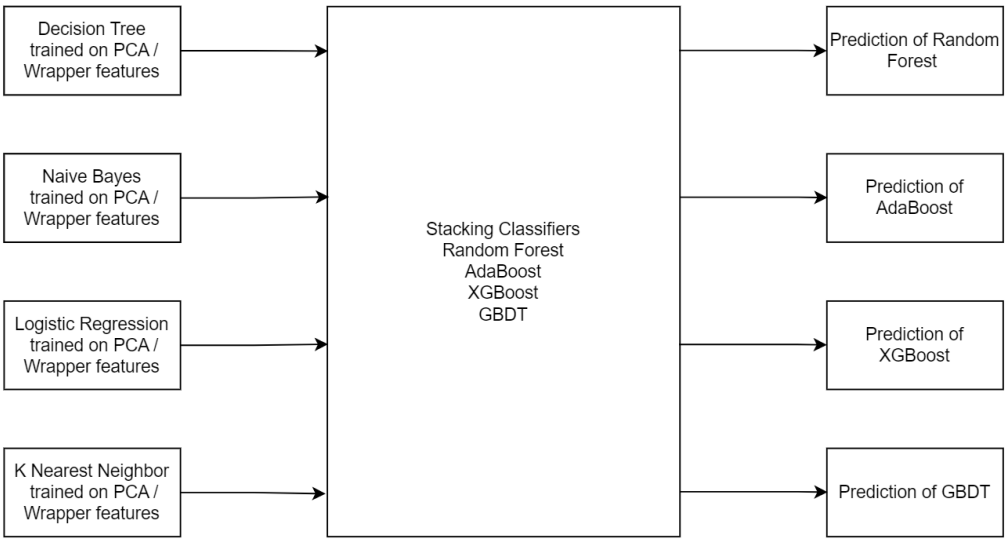

Fig 3. Stacking Classifiers on the Base learners

Now, once the predictions from the stacking classifiers are obtained, a pair of stacking predictions are considered at a time and OR Gate logic is applied to those predictions to obtain a new prediction.

Prediction of Stacking Classifier x (0 – normal, 1 – attack)	Prediction of Stacking Classifier y ($y \neq x$) (0 – normal, 1 – attack)	Output (0 – normal, 1 – attack)
0	0	0
0	1	1
1	0	1
1	1	1

Table 1: OR Gate Logic between the two Stacking Classifier Predictions

As shown in Table 1, the OR logic is applied to every pair of the Stacking Classifier predictions to obtain additional nC2 predictions, where n is the number of stacking classifier predictions. If n=4, that is if we are having 4 different

stacking classifier predictions then we will have an additional 6 predictions generated from those stacking classifiers' predictions.

Now, we would horizontally stack the predictions from 4 base learners, 4 stacking classifiers, which were trained on PCA features, and 6 predictions obtained by OR gate logic between the pairs of stacking classifier predictions, to make the dataset of 14 columns. Now a new column is created by summing up all 14 columnar values corresponding to that particular row of predictions. In the last step, this sum is compared with a threshold value to obtain the final prediction by using the logic that if the sum is greater than the threshold then predicts the output as 1 (attack) or else as 0 (normal). The threshold can be thought of as a hyperparameter, that would be tuned between the range of 1 and 14 to obtain the best possible performance from this novel voting scheme design. Similarly, the same procedure is followed for the base learners and stacking classifiers that were trained on Wrapper-based features to compare with the PCA-based features.

In addition, for both PCA and Wrapper-based feature-trained base learners and stacking classifiers, we repeated the method with AND Gate Logic as shown in Table 2 instead of OR Gate Logic between pairs of Stacking classifier predictions to compare the results between AND Gate and OR Gate Logic.

Prediction of Stacking Classifier x (0 – normal, 1 – attack)	Prediction of Stacking Classifier y ($y \neq x$) (0 – normal, 1 – attack)	Output (0 – normal, 1 – attack)
0	0	0
0	1	0
1	0	0
1	1	1

Table 2: AND Gate Logic between the two Stacking Classifier Predictions

5. Evaluation Metrics

It is critical to employ metrics that account for the imbalanced class distribution when assessing a machine learning model on an imbalanced dataset. For instance, simply using accuracy as a metric, might be deceptive because a classifier that always predicts the majority class would have a high accuracy even if it is not producing relevant predictions. Some metrics that are commonly used for evaluating imbalanced datasets include precision, recall, F1 score, the area under the receiver operating characteristic curve (AUC-ROC), and the area under the precision-recall curve (AUC-PR).

Precision: Precision is the ratio of true positives to the sum of the true positives and false positives. It measures the proportion of positive predictions that are correct.

$$\text{Precision} = \frac{TP}{TP+FP}$$

Recall: Recall is the ratio of true positives to the sum of the true positives and false negatives. It measures the proportion of actual positive instances that were correctly predicted by the model

$$\text{Recall} = \frac{TP}{TP+FN}$$

F1 score: The F1 score is a measure of a model's performance that combines precision and recall in a way to balance both precision and recall. It is calculated as the harmonic mean of precision and recall.

$$\text{F1 score} = \frac{2*Precision*Recall}{Precision+Recall}$$

AUC-ROC: The area under the receiver operating characteristic curve (AUC-ROC) is a metric that is commonly used to evaluate the performance of binary classifiers. It is a probability curve that shows the TPR vs the FPR at various threshold levels and then computing the area under the curve.

AUC-PR: The area under the precision-recall curve (AUC-PR) is a metric used to evaluate the performance of a binary classification model. It is calculated by plotting the precision and recall values for different thresholds, and then computing the area under the curve.

All of these metrics will have a value between 0 and 1, 0 indicates that the model is performing worst and 1 indicates that the model is performing the best. These metrics allowed us to evaluate the performance of classifiers at different stages on imbalanced data and get a more complete picture of their performance. It's important to use a combination of these metrics, rather than just relying on a single metric, to get the best understanding of a model's performance at different stages.

6. Results and analysis

By following the approach described in the methodology section of the report for training the base learners, the 5-fold cross-validation f1 scores obtained by the base learners on the training data from hyperparameter tuning along with data balancing in each fold are shown in Table 3.

Baseline Model / Imbalance Technique	Decision Tree	Naïve Bayes	Logistic Regression	K Nearest Neighbors
None	0.998032453837149	**0.957912581903690**	**0.979742684458016**	0.997418003327534
SMOTE	**0.998145648843404**	0.957659435981625	0.979625124538544	**0.997418096426344**
ADASYN	0.997281935050659	0.951157077909532	0.968801930724261	0.996987424403884

Table 3: 5-Fold Cross-validation f1-scores on Train Data from Hyperparameter tuning

Although the ADASYN imbalance technique hasn't been the best for all the base learners, we continued to use the ADASYN balancing technique and the corresponding base learners' parameter configuration for further steps because we wanted to handle the imbalance and ADASYN is known to be better than the SMOTE algorithm.

The 5-fold Cross-validation (CV) f1-score results on train data from the selection of features using PCA, Wrapper methods are shown in Table 4

Baseline Model	Feature Selection Method	Number of Features	f1-score
Decision Tree	PCA	35	0.9938
Naïve Bayes	PCA	19	0.924
Logistic Regression	PCA	26	0.949
K Nearest Neighbors	PCA	31	0.998
Decision Tree	RFE	25	0.9969
Naïve Bayes	SFS	23	0.9354
Logistic Regression	RFE	12	0.9519
K Nearest Neighbors	RFE	28	0.998

Table 4: 5-Fold Cross-validation f1-scores on Train Data from feature Selection

As part of the feature selection process using Wrapper methods, the Recursive Feature Elimination (RFE) method is used for Decision Tree, Logistic Regression as both the models would output feature importance scores which would be used the RFE method to filter the features, however, Naïve Bayes and KNN models do not produce the feature importance scores hence Sequential Feature Selection (SFS) is used for Naïve Bayes, however for KNN, as SFS method is slower, the RFE method with feature importance scores given by the decision tree is used by KNN for feature selection. As seen in Table 4, out of all the base learners, when the CV f1-scores are compared, the KNN model irrespective of PCA or Wrapper features comparatively performed better on the train data than the other base learners. So far the performance of the base learners on the training data is analyzed, Table 5 shows the performance metrics obtained when the base learners that are trained on ADASYN data with feature selection using PCA and Wrapper methods are used on the unseen (test) data.

Classifier	Feature Selection (Number of Features)	F1-Score	Precision	Recall	AUC-ROC	AUC-PR
Decision Tree	PCA (35)	0.7726	0.9588	0.647	0.81	0.82
Naïve Bayes	PCA (19)	0.8779	0.8931	0.863	0.86	0.85
Logistic Regression	PCA (26)	0.7984	0.9196	0.705	0.81	0.82
KNN	PCA (31)	0.8086	0.9635	0.697	0.83	0.84
Decision Tree	RFE (25)	0.8375	0.9252	0.765	0.84	0.84
Naïve Bayes	SFS (23)	0.8026	0.9114	0.717	0.81	0.81
Logistic Regression	RFE (12)	0.8015	0.9152	0.713	0.81	0.82
KNN	RFE (28)	0.805	0.9642	0.691	0.83	0.84

Table 5: Performance metrics obtained by base learners when tested on Test Data

When the results are analyzed, all the base learners irrespective of training using PCA or Wrapper-based features have high precision and low recall. Since, f1-score is the harmonic mean of precision and recall, the values of the f1-score are a bit high around 0.8 as the precision is high. Similarly, the AUC-ROC and the AUC-PR are high because of the high precision values. These base learners which are either trained on PCA or Wrapper-based features are used to train the stacking classifiers which are mentioned in the earlier parts of the report, the results obtained by the stacking classifiers are shown in Table 6.

Stacking Classifier	Feature Selection on Base Learners	F1-Score	Precision	Recall	AUC-ROC	AUC-PR
Random Forest	PCA	0.8149	0.9615	0.707	0.83	0.85
AdaBoost	PCA	0.8111	0.9608	0.702	0.83	0.84
XGBoost	PCA	0.8122	0.9612	0.703	0.83	0.84
GBDT	PCA	0.8135	0.9599	0.706	0.83	0.84
Random Forest	Wrapper	0.85	0.9623	0.761	0.86	0.87
AdaBoost	Wrapper	0.8394	0.9597	0.746	0.85	0.86
XGBoost	Wrapper	0.8477	0.9613	0.758	0.86	0.87
GBDT	Wrapper	0.8506	0.9604	0.763	0.86	0.87

Table 6: Performance metrics obtained by stacking classifiers when tested on Test Data

When using stacking classifiers, the precision and recall values have slightly improved from base learners, but the precision values continue to outweigh the recall, which is why the f1-score, AUC-ROC, and AUC-PR are also over 0.80. Moreover, the stacking classifiers that have been trained using the base learners which utilized Wrapper-based features have higher precision and recall scores than the ones trained on PCA features, hence the higher values of f1-score, AUC-ROC, and AUC-PR for the former than the latter. The logical OR is now used to combine the predictions

from the pairs of stacking classifiers, yielding 6 new predictions each for PCA and Wrapper-based features, respectively. Table 7 presents the results of these.

Stacking Classifier 1	Stacking Classifier 2	Feature Selection on Base Learners	F1-Score	Precision	Recall	AUC-ROC	AUC-PR
Random Forest	AdaBoost	PCA	0.8178	0.9604	0.7121	0.84	0.85
Random Forest	XGBoost	PCA	0.8186	0.9612	0.7128	0.84	0.85
Random Forest	GBDT	PCA	0.8189	0.9597	0.7141	0.84	0.85
AdaBoost	XGBoost	PCA	0.8147	0.9604	0.7073	0.83	0.85
AdaBoost	GBDT	PCA	0.8135	0.9598	0.7058	0.83	0.84
XGBoost	GBDT	PCA	0.8154	0.9598	0.7087	0.83	0.85
Random Forest	AdaBoost	Wrapper	0.8636	0.9597	0.785	0.87	0.88
Random Forest	XGBoost	Wrapper	0.856	0.9606	0.7719	0.87	0.87
Random Forest	GBDT	Wrapper	0.8654	0.9603	0.7877	0.87	0.88
AdaBoost	XGBoost	Wrapper	0.859	0.9595	0.7775	0.87	0.87
AdaBoost	GBDT	Wrapper	0.8545	0.9597	0.7701	0.86	0.87
XGBoost	GBDT	Wrapper	0.8649	0.9602	0.7869	0.87	0.88

Table 7: Performance metrics obtained when Logical OR is used on stacking classifiers

After combining the predictions using the Logical OR strategy, the precision values remained the same however, the recall values have increased slightly for both types of stacking classifiers (PCA and Wrapper-based features). Moreover, the precision and recall values are again higher for Wrapper-based Stacking Classifiers than the PCA based.

As stated in the methodology, the 14 different predictions from base learners, stacking classifiers, and Logical OR between stacking classifier predictions on the train set are summed up, and the threshold value is used as the hyperparameter to find the best threshold using cross-validation on the training dataset.

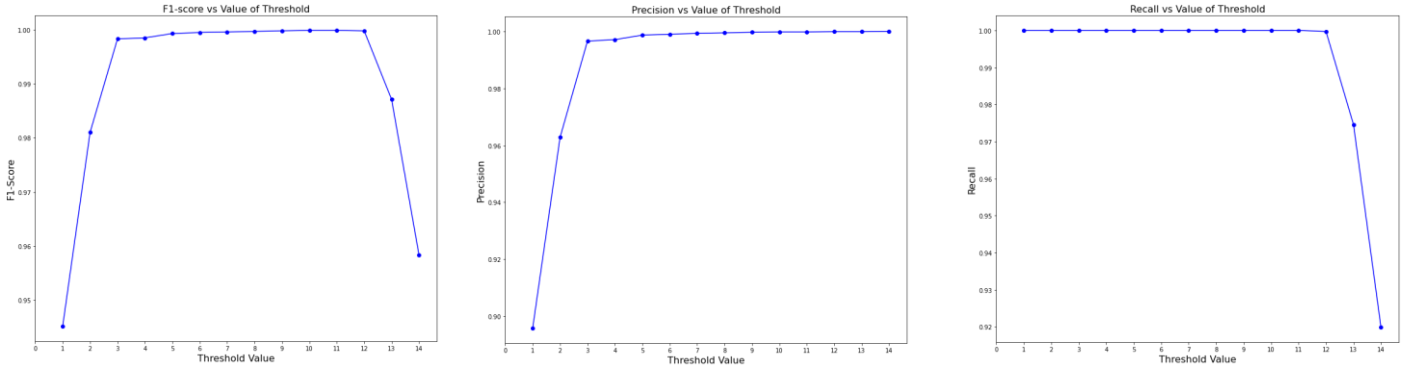

Fig 4. F1 score, precision, recall at various values of threshold (Logical OR PCA-based features)

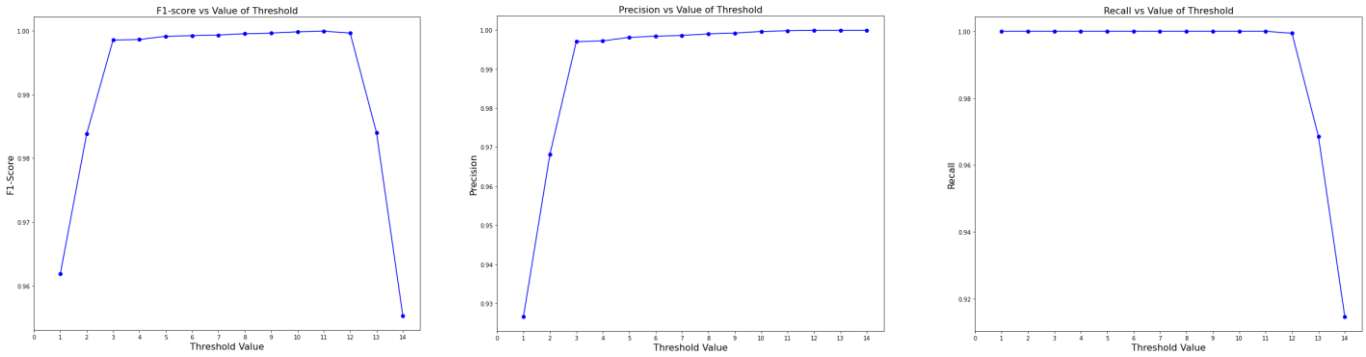

Fig 5. F1 score, precision, recall at various values of threshold (Logical OR Wrapper-based features)

As seen in Figures 4, and 5, as the threshold value increased from 1 to 14, the f1-score increased first and then later decreased, whereas the precision values kept increasing and the recall value kept decreasing. Table 8 below provides the threshold values where the best values for f1-score, precision, and recall are obtained when the 5-fold CV is used on the train data.

Features	Metric	Metric Value	Threshold Value
PCA	Recall	1	1
PCA	F1 Score	0.9999	10
PCA	Precision	1	14
Wrapper	Recall	1	1
Wrapper	F1 Score	0.9999	11
Wrapper	Precision	0.9999	13

Table 8: Threshold values at which best values of f1-score, precision, and recall are obtained (Logical OR)

So far we have seen that the precision values are high whereas recall values are very low, which means the models while predicting the class labels are predicting fewer false positives but more false negatives. But in designing the IDS system, the cost of False Negatives is more than the False Positives because there is no harm in predicting the normal traffic packet as an attack but there would be more harm caused if the system predicts the malicious traffic as normal traffic. Hence the recall should be optimized for the system. As shown in Table 8, regardless of PCA or Wrapper features, the optimal threshold value for recall obtained is 1, where the value of recall is maximum. The same threshold value is used on the test (unseen) dataset for evaluating the performance of the designed scheme, at first, all 14 predictions for each of the PCA and Wrapper-based models are obtained on the test dataset, and then the threshold value which is obtained from the training dataset is used on the summation of these 14 predictions to make the final prediction, the results obtained are provided in Table 9.

Features	Threshold Value	F1-Score	Precision	Recall	AUC-ROC	AUC-PR	Accuracy
PCA	**1**	**0.8851**	**0.8882**	**0.8821**	**0.87**	**0.85**	**0.8697**
PCA	10	0.8131	0.9617	0.7043	0.83	0.85	0.8157
PCA	14	0.7356	0.9704	0.5923	0.78	0.81	0.7576
Wrapper	**1**	**0.8791**	**0.9104**	**0.8365**	**0.86**	**0.85**	**0.8601**
Wrapper	11	0.8281	0.9619	0.727	0.84	0.85	0.8282
Wrapper	13	0.7839	0.9627	0.6611	0.81	0.83	0.7925

Table 9: Performance metrics obtained on Test Data using the Thresholding on Logical OR strategy

By using the threshold value of 1, the maximum recall is obtained on test data, hence the designed thresholding concept generalizes well even on the unseen data. Furthermore, the threshold value of 1 works better for PCA features than Wrapper features since the f1-score, recall acquired from PCA features is somewhat higher than that obtained from Wrapper features. However, with the use of this threshold value, there is a compromise in Precision value as the precision value falls below 0.9 which was not the case earlier, but as discussed earlier since the recall is more important we can compromise a bit on precision value.

To compare the results with Logical AND, we modified the methodology by substituting Logical AND with Logical OR and repeated the thresholding strategy with Logical AND. Table 10 presents the results of using Logical AND between the predictions of stacking classifiers.

Stacking Classifier 1	Stacking Classifier 2	Feature Selection on Base Learners	F1-Score	Precision	Recall	AUC-ROC	AUC-PR
Random Forest	AdaBoost	PCA	0.8082	0.962	0.6967	0.83	0.84
Random Forest	XGBoost	PCA	0.8085	0.9615	0.6975	0.83	0.84
Random Forest	GBDT	PCA	0.8094	0.9618	0.6987	0.83	0.84
AdaBoost	XGBoost	PCA	0.8086	0.9617	0.6976	0.83	0.84
AdaBoost	GBDT	PCA	0.8111	0.9609	0.7016	0.83	0.84
XGBoost	GBDT	PCA	0.8103	0.9614	0.7002	0.83	0.84
Random Forest	AdaBoost	Wrapper	0.8251	0.9624	0.7221	0.84	0.85
Random Forest	XGBoost	Wrapper	0.8416	0.963	0.7474	0.85	0.86
Random Forest	GBDT	Wrapper	0.8346	0.9624	0.7368	0.85	0.86
AdaBoost	XGBoost	Wrapper	0.8277	0.9615	0.7266	0.84	0.85
AdaBoost	GBDT	Wrapper	0.8354	0.9604	0.7391	0.85	0.86
XGBoost	GBDT	Wrapper	0.8328	0.9615	0.7345	0.85	0.86

Table 10: Performance metrics obtained when Logical AND is used on stacking classifiers

The f1-scores achieved by using logical AND between the predictions of the stacking classifiers are lower than those obtained using logical OR. Further examination reveals that when Logical OR is swapped out for Logical AND to derive the new predictions from a pair of stacking classifiers predictions, the precision values only marginally improved while the recall values significantly reduced. By repeating the procedure by treating the threshold value as the hyperparameter and using 5-fold cross-validation on the training dataset, the best values of the thresholds are determined.

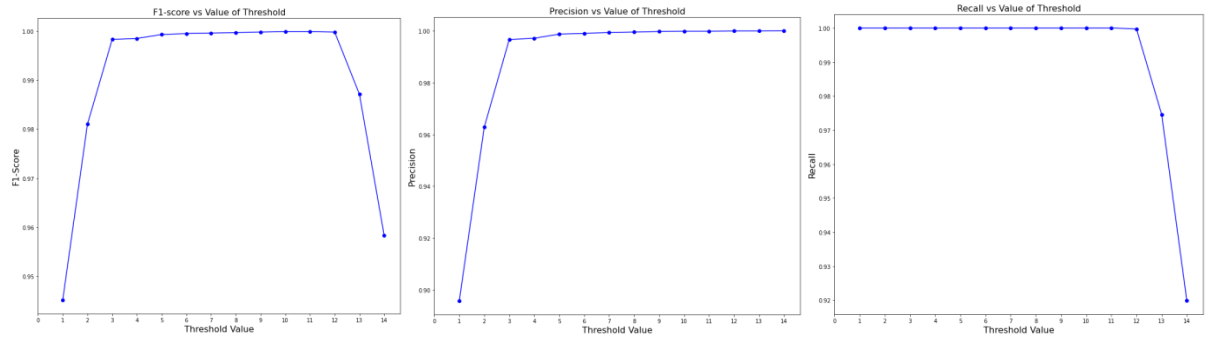

Fig 6. F1 score, precision, recall at various values of threshold (Logical AND PCA-based features)

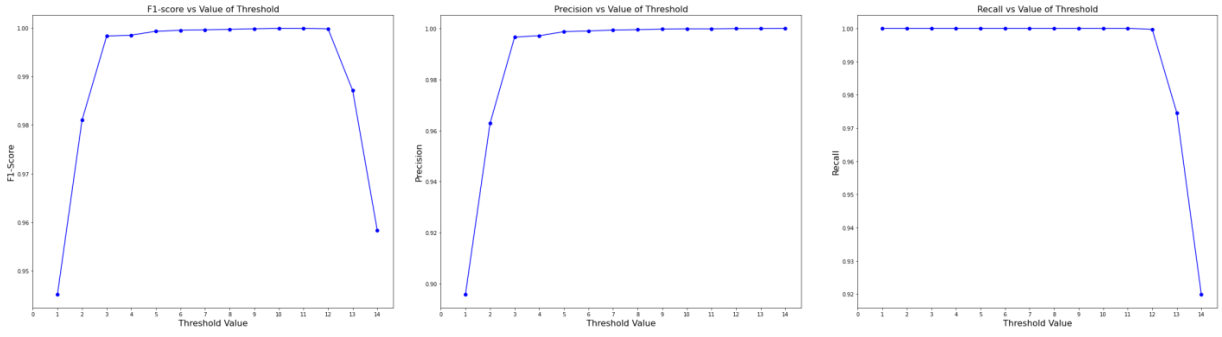

Fig 7. F1 score, precision, recall at various values of threshold (Logical AND Wrapper-based features)

As seen in Figures 6, and 7, as the threshold value rose from 1 to 14, the precision values continued to increase but the recall values continued to fall. The f1-score increased initially before dropping. The pattern is similar to the one observed for Logical OR. Table 11 shows the threshold values at which the best f1-score, accuracy, and recall values are obtained when the 5-fold CV is applied to training data.

Features	Metric	Metric Value	Threshold Value
PCA	Recall	1	1
PCA	F1 Score	0.9999	6
PCA	Precision	1	14
Wrapper	Recall	1	1
Wrapper	F1 Score	0.9999	8
Wrapper	Precision	0.9999	13

Table 11: Threshold values at which best values of f1-score, precision, and recall are obtained (Logical AND)

Irrespective of PCA or Wrapper features, the ideal threshold value for recall attained is 1, where the value of recall is maximum, as demonstrated in Table 11. The threshold values obtained on the train data are the same as that of the threshold values obtained by using the Logical OR technique except for the f1-score metric because we will have a lesser number of 1s in predictions when Logical AND is used (Logical AND will produce 1 only when the two stacking classifiers predict 1). The performance of the designed scheme is evaluated using the same threshold value on the test dataset, which is used to obtain all 14 predictions for the PCA and Wrapper-based models, respectively, before using the training dataset's threshold value to make the final prediction. The results are shown in Table 9.

Features	Threshold Value	F1-Score	Precision	Recall	AUC-ROC	AUC-PR	Accuracy
PCA	**1**	**0.8851**	**0.8882**	**0.8821**	**0.87**	**0.85**	**0.8697**
PCA	6	0.8132	0.9617	0.7044	0.83	0.85	0.8157
PCA	14	0.7356	0.9704	0.5923	0.78	0.81	0.7576
Wrapper	**1**	**0.8791**	**0.9104**	**0.8365**	**0.86**	**0.85**	**0.8601**
Wrapper	8	0.8281	0.9619	0.727	0.84	0.85	0.8282
Wrapper	13	0.7839	0.9627	0.6611	0.81	0.83	0.7925

Table 12: Performance metrics obtained on Test Data using the Thresholding on Logical AND strategy

The performance metrics values obtained by Logical AND for threshold value 1 on the test data are the same as that of the Logical OR technique for both PCA and Wrapper-based features. Hence, using Logical AND instead of Logical OR hasn't produced any different results. When features are PCA and the threshold value is 1, regardless of whether the Logical OR or the Logical AND is employed, the f1-score, precision, and recall values on the test data are around 0.8851, 0.8882, and 0.8821, respectively, and the detailed results such as confusion matrix, the ROC curve, and the PR curve are shown in Figure 8.

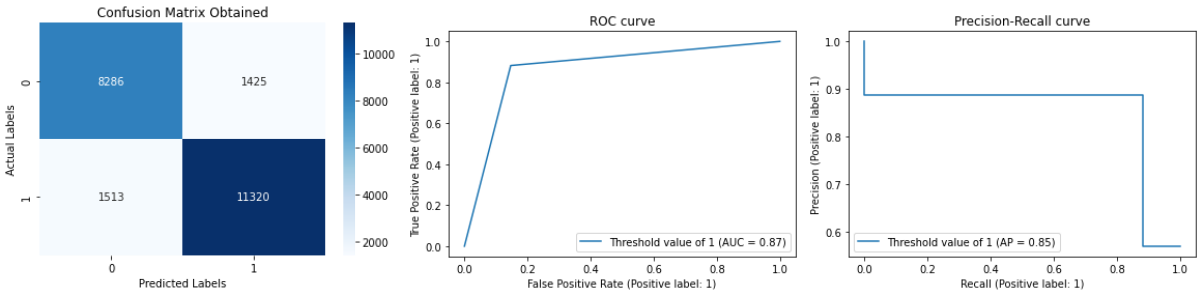

Fig 8. Confusion Matrix, ROC curve, and PR curve for threshold value = 1

From the confusion matrix, we can observe that the number of false positives and false negatives is nearly equal, as shown in Figure 8, but as previously mentioned, we need to focus on the recall value more in order to develop the IDS as cost of false negatives is usually more than the cost of false positives, so the false negatives (1513) should be further reduced in order to enhance the recall. However, based on the literature review we have seen that some researchers have focused on precision values along with recall values, in order to compare our algorithm's performance with the existing solutions, we obtained performance metrics on the test dataset by using the threshold value that is optimal for the precision obtained from the training dataset. Tables 9 and 12 illustrate the results for these as well.

7. Comparison with existing strategies

We compared and examined the findings of the literature review using our proposed technique. Table 13 shows a comparison of the outcomes.

Research Reference	Methodology Used	F1-score	Precision	Recall	Accuracy
Abrar et al. (2020)	Logistic Regression - Attribute Set 1	N/A	N/A	N/A	0.8705
Abrar et al. (2020)	Logistic Regression - Attribute Set 2	N/A	N/A	N/A	0.8603
Abrar et al. (2020)	Logistic Regression - Attribute Set 3	N/A	N/A	N/A	0.8664
Abrar et al. (2020)	Logistic Regression - Attribute Set 4	N/A	N/A	N/A	0.8666
Govindarajan (2016)	Proposed Bagged RBF	N/A	N/A	N/A	0.864
Govindarajan (2016)	Proposed Bagged SVM	N/A	N/A	N/A	0.9392
Zhou et al. (2019)	AdaBoost.M1	0.8777	0.9484	0.8188	0.9952
Proposed Methodology	PCA features with Threshold = 1	0.8851	0.8882	0.8821	0.8697
Proposed Methodology	PCA features with Threshold = 14	0.7356	0.9704	0.5923	0.7576

Table 13: Comparison of the results obtained from the proposed methodology with existing strategies

The accuracy gained on the test dataset with the PCA features and the threshold value of 1 is greater than what the researchers acquired in Abrar et al. (2020) when Logistic Regression is applied with the different features set.

Moreover, the obtained accuracy is also more than what was obtained by Govindarajan (2016) using the strategy of bagged RBF. However, the accuracy obtained by our methodology falls below the bagged SVM model. Zhou et al. (2019) employed AdaBoost to create the IDS system; their recall and f1-score values are lower than what we achieved using the threshold value of 1, but our precision is lower than their precision value at the threshold value of 1. However, at the threshold value of 14, the precision value obtained by our methodology is more than the precision value obtained by them, although at the compromise of recall and f1-score.

8. Advantages and Limitations

By using the proposed methodology, we were able to control the precision and recall by only changing the threshold value. The novel algorithm developed by us has produced results better than some of the existing schemes. The stacking classifiers use base learners which have been trained on different features, hence there is a lot of diversity induced while training the stacking classifiers. However, training the base learners and stacking classifiers takes significantly longer than training the existing schemes. For instance, in order to use the SFS method for feature selection as part of training the KNN, the estimated time consumption was more than 60hrs despite using a high-end laptop that comes with 16GB of RAM and 4GB of GPU. Another limitation of the approach is the time taken to predict the network traffic as normal or the attack. The prediction time is longer since the base learners and stacking classifiers should produce output results for computing the Logical AND or Logical OR, as well as summing up the predictions to apply the desired threshold value, but the IDS system is intended to detect attacks faster. Although we achieved better results than some of the existing methodologies, there is still room for improvement in terms of the f1-score without compromising the recall values. Besides that, since the predictions are accumulated from different stages in the training process, attackers would find it difficult to learn the weights of the models because they wouldn't know how many models are contributing to the decision, so the IDS system is more secure from attackers breaching it and exploiting it to their advantage, but the interpretability and the error analysis is more complicated for IDS developers. Furthermore, our suggested technique works well as a dynamic model because if we detect a decrease in the performance of the strategy on unseen data, we may improve the performance by altering the threshold values first before retraining the system with the most recent data. The other limitation is that the developed approach has only been tested on the NSL KDD dataset, in order to increase trust in the methodology, the strategy should also be validated on different datasets.

9. Conclusion and Future Work

In this research, we attempted to develop an intrusion detection system using a novel idea in Ensemble Learning. For this purpose, we used the NSL-KDD dataset and independently trained four different base learners – Decision Tree, Naïve Bayes, Logistic Regression, and K Nearest Neighbors by using the ADASYN data balancing technique and PCA for feature selection. On top of these base learners, four different stacking classifiers – Random Forest, AdaBoost, XGBoost, and GBDT are trained using the base learners. The Logical OR is used between the pairs of stacking classifier predictions to obtain additional six predictions. These fourteen predictions are summed, and a threshold value obtained from the training data is used to get the final predictions. In order to compare the results of the PCA, we have also trained the base learners by using wrapper-based feature selection methods and found that the PCA features produce better results than the wrapper-based features. For further comparison, the Logical AND is used instead of the Logical OR, but there was no difference in the results obtained for both PCA and wrapper-based features. The base learners when tested on the test data produced precision values above 0.9 while most of the recall values were less than 0.75. The precision values further got increased with the use of stacking classifiers but there wasn't any noticeable change to recall values. The usage of Logical AND or Logical OR between the pair of stacking classifier predictions has no effect on precision or recall. However, by aggregating all of these predictions and employing the appropriate threshold value, we were able to enhance the recall values and attain f1-score, accuracy, and recall values as 0.8851, 0.8882, and 0.8821 respectively. As far as the training time is concerned the base learners which were trained on PCA features took lesser time for training when compared to the base learners that were trained on wrapper-based features. Moreover, the PCA features produced better results than the Wrapper-based features for the optimal threshold value obtained for recall. Although the results obtained using PCA features with a threshold value of 1 are promising and outperform some of the methodologies found in the literature review, there is still room for improvement. In the future, we would like to improvise this methodology by trying different models for base learners and stacking classifiers. Although it is complex, we would also want to perform a detailed error analysis on the existing results which would help us determine where the errors are coming from. Furthermore, we would test this technique on other datasets to guarantee that the intended methodology worked on any type of data. Because we have only tested our methodology on static data, we would like to test our strategy on data streams to see how we can respond to concept drifts in data by altering thresholds before contemplating retraining with the most recent data from data streams. Overall, the proposed methodology, which is a combination of predictions from base learners and predictions

from stacking classifiers, has shown good results in developing an intrusion detection system using the NSL-KDD dataset.

10. References

Abrar, I., Ayub, Z., Masoodi, F., & Bamhdi, A. M. (2020). A machine learning approach for intrusion detection system on NSL-KDD dataset. *In 2020 International Conference on Smart Electronics and Communication (ICOSEC)*, 919-924. https://doi.org/10.1109/ICOSEC49089.2020.9215232

Ahmad, I., Basheri, M., Iqbal, M. J., & Rahim, A. (2018). Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection. *IEEE Access*, 6, 33789-33795. https://doi.org/10.1109/ACCESS.2018.2841987

Canadian Institute for Cybersecurity | UNB. (n.d.). NSL-KDD | Datasets | Research. Retrieved from https://www.unb.ca/cic/datasets/nsl.html

Choi, H., Kim, M., Lee, G., & Kim, W. (2019). Unsupervised learning approach for network intrusion detection system using autoencoders. *The Journal of Supercomputing*, 75(9), 5597-621. https://doi.org/10.1007/s11227-019-02805-w

Gangula, R., Mohan, V. M., & Kumar, M. R. (2022). Network intrusion detection system for Internet of Things based on enhanced flower pollination algorithm and ensemble classifier. *Concurrency and Computation: Practice and Experience*, 34(21). https://doi.org/10.1002/cpe.7103

Govindarajan, M. (2016). Evaluation of ensemble classifiers for intrusion detection. *World Academy of Science, Engineering and Technology, Open Science Index 114, International Journal of Computer and Information Engineering*, 10(6), 9. https://doi.org/10.5281/zenodo.1124579

Kumar, G., Thakur, K., & Ayyagari, M. R. (2020). MLEsIDSs: Machine learning-based ensembles for intrusion detection systems a review. *Journal of Supercomputing*, 76(11), 8938-8971. https://doi.org/10.1007/s11227-020-03196-z

Mkuzangwe, N. N. P., Nelwamondo, F., Mkuzangwe, N. N. P., & Nelwamondo, F. (2017). Ensemble of classifiers based network intrusion detection system performance bound. *In 2017 4th International Conference on Systems and Informatics (ICSAI)*, 970-974. https://doi.org/10.1109/ICSAI.2017.8248426

Nguyen, P. C., Nguyen, Q. T., & Le, K. H. (2021). An ensemble feature selection algorithm for machine learning based intrusion detection system. *In 2021 8th NAFOSTED Conference on Information and Computer Science (NICS)*, 50-54. https://doi.org/10.1109/NICS54270.2021.9701577

Pham, N. T., Foo, E., Suriadi, S., Jeffrey, H., & Lahza, H. F. M. (2018). Improving performance of intrusion detection system using ensemble methods and feature selection. *In Proceedings of the Australasian Computer Science Week Multiconference Brisbane Queensland Australia*, 1-6. https://doi.org/10.1145/3167918.3167951

Rani, D., Gill, N. S., Gulia, P., & Chatterjee, J. M. (2022). An ensemble-based multiclass classifier for intrusion detection using Internet of Things. *Computational Intelligence and Neuroscience*, 2022, 1668676. https://doi.org/10.1155/2022/1668676

Rawat, S., Srinivasan, A., & Ravi, V. (2022). Intrusion detection systems using classical machine learning techniques vs integrated unsupervised feature learning and deep neural network. *Internet Technology Letters*, 5(1). https://doi.org/10.1002/itl2.232

Seth, S., Chahal, K. K., & Singh, G. (2021). A novel ensemble framework for an intelligent intrusion detection system. *IEEE Access*, 9, 138451-138467. https://doi.org/10.1109/ACCESS.2021.3116219

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. *In 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 1-6. https://doi.org/10.1109/CISDA.2009.5356528

Yousefnezhad, M., Hamidzadeh, J., & Aliannejadi, M. (2021). Ensemble classification for intrusion detection via feature extraction based on Deep Learning, *Soft Computing*, 25(20), 12667-12683. https://doi.org/10.1007/s00500-021-06067-8

Zhou, Y., Mazzuchi, T. A., & Sarkani, S. (2019). M-AdaBoost-A based ensemble system for network intrusion detection. *Expert Systems with Applications*, 162, 113864. https://doi.org/10.1016/j.eswa.2019.113864