

Tripartite optimized framework for network based intrusion detection system

Poonkody Iruthayaraj, Poornima Vivekanandhan

Department of Computer Science, Vels Institute of Science, Technology and Advanced Studies, Chennai, India

Article Info

Article history:

Received Aug 7, 2025

Revised Jul 16, 2026

Accepted Aug 12, 2026

Keywords:

Hyper parameter tuning
Network-based intrusion
detection system
Optimized decision tree
Optimized soft voting classifier
Optimized support vector machine
Unbalanced dataset

ABSTRACT

An unsecured network pays the way for various illegal access. Therefore, an efficient security system is needed to safe guard the data from attackers. Even though various network security methods such as authentication, authorization, and firewall have been used, the system still suffers from various types of security issues. An optimized soft voting classifier (OSOVOC) algorithm is proposed for network-based intrusion detection system (NIDS). It performs the hybridization of two algorithms namely: optimized decision tree (ODT) and optimized support vector machine (OSVM). All the three algorithms are optimized using hyper parameter tuning. The final proposed model produces a more accurate result in an unbalanced dataset. The primary focus of the proposed OSOVOC algorithm is to eradicate the false alarms, which significantly increases the true positive count and true negative count. The proposed OSOVOC algorithm produces negotiable false alarms, with performance reaching a F1-score of 100% during training phase and 99.87% during testing phase.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Poornima Vivekanandhan

Department of Computer Science, Vels Institute of Science, Technology and Advanced Studies

Pallavaram, Chennai, Tamil Nadu, India

Email: poornimasudhaagar@gmail.com

1. INTRODUCTION

Intrusion is an unauthorized activity taken place in a computer network. It can access, modify, and delete data and it also leads to destabilizing network functioning and installs malicious software. Network based intrusion detection system (NIDS) are developed to identify these unauthorized activities. Network unauthorized activities can take several forms. Malware attacks involve in the installation of the malicious software on the system with the intention to steal data, take over control or disrupt the normal operations. Phishing attacks aims to collect sensitive information from users through suspicious links. Distributed and denial of service attacks deny the service to legitimate user by overloading the network or server with heavy traffic. An unauthorized access attacks occur when a network is accessed without legitimate authorization resulted due to very weak passwords and misconfigured systems. Structured query language (SQL) injection attacks manipulate the database queries by exploiting web applications and gaining unauthorized access.

Due to the increase in online threats, an effective NIDS is necessary to safeguard network or system from malicious activities. The following factors should be considered, while evolving a NIDS. Detecting all attacks accurately and efficiently is complex because of the changing nature of cyber-attack. Everyday huge amount of network traffic is generated and identifying these abnormal patterns could be time consuming. Eradicating, false alarm completely, requires a suitable algorithm that significantly increases the true alarm rates. False positive leads to unnecessary protection and waste of time, where else false negative crashes the

system which leads to potential loss. Even though, various networks-based intrusion detection methods have been developed, optimal detection is still not achieved.

Talukder *et al.* [1] introduced a machine learning (ML) based NIDS that address the skewed dataset using the random oversampling method, the dimension reduction using principal component analysis (PCA) method and clustering result using stacking feature embedding method. Thanh [2] used resampling and feature selection techniques for training the skewed dataset. Nassreddine *et al.* [3] deployed synthetic minority over-sampling technique (SMOTE) Tomek technique for an imbalanced data set. Neto and Gomes [4] applied mean decrease in impurity (MDI) technique and reduced the features to 51 from 78. Birahim *et al.* [5] selected the relevant features for intrusion detection. This was done by means of correlation, mutual information (MI), and PCA. Mokbal *et al.* [6] used extreme gradient boost (XGBoost) and an embedded feature selection. Mhawi *et al.* [7] proposed an ensemble learning based algorithm for NIDS. Hybrid way of feature selection was done using the combined approach of correlation and forest panelized attributes. Taher *et al.* [8] indicates that artificial neural network (ANN)-based ML along with wrapper feature selection out performs the support vector machine (SVM).

Hossain and Islam [9] proposed a random forest (RF) ensemble method, which out performs the other ensemble methods bagging and simple stacking. Bibers *et al.* [10] framework was tested using the various ensemble methods known as bagging, stacking, blending, and boosting on the base classifiers. Alalwany and Mahgoub [11] proposed intrusion detection system (IDS), to classify the real time bus attacks using controller area network, combining ensemble and kappa architecture. Sathya [12] proposed an ensemble ML method. It can detect and prevent various attacks of network. Ahsan *et al.* [13] proposed a voting classifier that combines the decisions from nine ML algorithms. Uppal *et al.* [14] employed the max voting technique for Gaussian naive Bayes (NB), decision tree (DT), and XGBoost classifiers to obtain desired result. Wani *et al.* [15] proposed a model that uses three base classifiers and an ensemble method named voting is used for classification on the original and an optimized dataset. Jabbar [16] comparative analysis of individual ML models like K-nearest neighbors (KNN), DT, and gradient boosting were employed using voting classifiers. Yoheswari [17] proposed an optimized support vector machine (OSVM) algorithm for intrusion detection.

Chua and Salam [18] evaluated the performance of six popular ML algorithms and they state that ANN and SVM algorithms were unaffected by overfitting. Banu and Sangeetha [19] used integration of whale optimization algorithm and RF algorithm. Latif *et al.* [20] investigated the following ML algorithms: DT, SVM, RF, NB, neural network (NN), and adaptive boosting (AdaBoost) using network security laboratory - knowledge discovery and data mining (NSL-KDD) dataset. Almutairi *et al.* [21] was done evaluation of NIDS using the ML algorithms SVM, J48, and RF. Bachar *et al.* [22] proposed artificial intelligence technique for intrusion detection using ML. Polynomial kernel and Gaussian kernel were used for SVM classification model. Mills *et al.* [23] was designed hybrid type of IDS that combined supervised and unsupervised learning models through an ensemble stacking model. Tiwari *et al.* [24] proposed tree-based model that analyzed the traffic behavior in vehicular ad-hoc networks. Ali *et al.* [25] developed a dataset CIPMAIDS2023-1. A stacking-based ensemble approach was applied and tested on the topology of network called GNS-3. Aeraj and Cherkaoui [26] was used SVM in conjunction with the Snort IDS.

Bearing these factors, a suitable NIDS using ML has been constructed to identify the possible attacks with maximum accuracy and thus reducing false alarm rates. Hybridization is done for combining the optimized algorithms: SVM and DT. Optimization was done through hyper parameter tuning. The proposed optimized soft voting classifier (OSOVC) algorithm can strengthen the safety of computer networks. The experimental observations, proves the potency of the proposed system with the accuracy reaching 100% during training and 99.91% during testing.

2. PROPOSED FRAMEWORK

The hybridization of OSVM and optimized decision tree (ODT) models using the proposed OSOVOC algorithm is illustrated in Figure 1. The required network attack data is obtained from KDDCup99 dataset. It contained missed and duplicate values. So, the dataset ought to be prepped to remove missed and duplicate values [2]. Then, the preprocessed dataset is separated into a training set (75%) and a testing set (25%). Hyperparameter tuning is done to the two models chosen: SVM, DT. Then, hybridization technique has been applied to integrate the two models and again optimized with hyperparameter tuning for detecting the possible network attacks [7], [23]. The results obtained from each model were compared for accuracy.

2.1. Proposed optimized soft voting classifier algorithm

The OSOVOC algorithm was introduced to detect the various threats existing in a network and to improve the performance metrics. This algorithm focuses on eradicating the false alarm rate and thereby improving the model's accuracy. Since this model is optimized, the computational overhead is significantly

reduced. Moreover, multiple individual models were integrated to enhance the robustness of the model. The following steps describes the development of OSOVOC algorithm:

Step 1: import the necessary libraries for data preprocessing, ML, and visualization.

Step 2: employ KDDCup99 data set for both training and testing.

Step 3: do all the necessary preprocessing to obtain the final pre-processed dataset.

Step 4: partition the dataset into two parts: 75% for the training dataset and 25% for the testing dataset.

Step 5: construct a model that uses OSVM and ODT as classifier.

Step 6: apply hyper parameter tuning to individual models for better performance.

Step 7: construct the OSOVOC algorithm that does the hybridization by combining both the OSVM and ODT models.

Step 8: apply hyper parameter tuning to the OSOVOC algorithm for optimized performance

Step 9: execute the model, by training using the training dataset and evaluating it on the testing dataset.

Step 10: calculate the F1-score, false positive rate (FPR), and false negative rate (FNR).

Step 11: tabulate the classifier performance based on precision, F1-score, recall, FPR, FNR, and accuracy.

Step 12: visualize confusion matrices for each classifier.

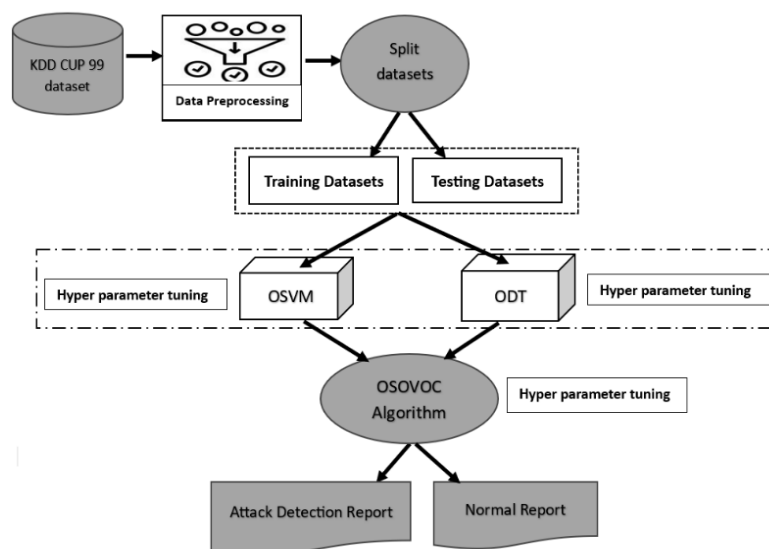


Figure 1. Proposed tripartite optimized framework

2.2. Implementation

The software required to execute the tripartite optimized framework is python. It is one of the programming languages to implement the code. It is a suitable choice for implementing NIDS because of its simplicity, extensive libraries, and data visualization. Automating repetitive tasks like scanning networks, analyzing traffic, and detecting abnormalities in the network can be seamlessly implemented by using the python programming language.

2.2.1. Importing the necessary python libraries

The experiment was conducted using the python programming environment. The following python libraries were imported for the execution of various tasks involved in the proposed methodology. These libraries support feature extraction, data preprocessing, hyperparameter optimization, model training, and performance evaluation [2], [3]. They also provide optimized implementations of core algorithms, enabling efficient experimentation and reliable outputs. Standard classification metrics such as precision, recall, F1-score, false alarm rate and accuracy were also employed for performance evaluation. This section outlines the Python libraries used for implementation and explains their specific roles.

- NumPy: this package is used for performing mathematical computations. Operations such as multi-dimensional arrays and matrices can also be done.
- Pandas: this package is used for importing and managing datasets with data manipulation capabilities. It is used mainly used for data analysis.
- Matplotlib: this package is used for visual presentation of various 2D plots. It is mainly used for the creation of different types of charts, figures for easy visualization of the data.

- Seaborn: this package is used for data exploration and visualization of informative statistical graphics.
- Train_test_split: this method separates the data set into training and testing part.
- CountVectorizer: it is used to convert a given text document into a vector, based on the frequency of each word.
- StandardScaler: a technique that transforms the data, that ranges from 0 to 1.
- Hstack: it stacks arrays column wise. Also, performs concatenation along the second axis.
- Datetime: it is used for manipulating dates and times.
- accuracy_score: it compares the original and output label and returns an accuracy score between 0 and 1.
- Confusion_matrix: a tabular representation of original and output values. Also, used for evaluating the performance of individual classifier.
- precision_score: it identifies predicted positive instances. The score ranges from 0 to 1, where 0 represents the worst score and 1 represents the best score.
- recall_score: it finds out all the positive samples. The score ranges from 0 to 1, where 0 represents the worst score and 1 represents the best score.
- F1_score: combines precision and recall by weighted average. The score ranges from 0 to 1, where 0 represents the worst score and 1 represents the best score.
- DecisionTreeClassifier: it measures the quality of a split by choosing different depths of the tree [20].
- Support vector classifier: hyperplane separating data points into various classes accurately. It is used for classification, regression, and, outliers' detection [22].

2.2.2. Dataset description

First create a folder and make it a working directory. Import the KDDcup99 dataset and store it in the working directory. Save the python file also in the same directory. Perform the execution to obtain the results. The tripartite optimized framework uses the KDDcup99 dataset. It is a simulated network containing 41 features that represents the various aspects of a network connection [20]. There are 22 different attack classes. Identifying these predefined attack types accurately is done by two ML classifiers: SVM and DT.

2.3. Data preprocessing

Null data values in the dataset may produce a biased output. To prevent this the dataset should be checked for any null values and should be removed. Also, removing duplicate values from a dataset improves the data quality and accuracy of ML models [2]. The pre-processed dataset is separated into two parts: the first part, nearly 75%, is used for training, and the second part, nearly 25%, is used for testing. The 41 features have been selected for network-based IDS. Out of the 41 features, three features namely protocol, service and flag have been identified as categorical features. So, one hot encoding has been performed on these features. Then, standard scaling is done for the remaining 38 features for normalization. Finally, all these features have been merged again for analysis.

2.4. Dataset exploration

Exploratory data analysis is a technique for analyzing the dataset. It is the visual exploration of the data. Python libraries like matplotlib, pandas, and seaborn were used for performing exploratory data analysis. The diagrams projected from the exploratory data analysis are useful for spotting errors, outliers, and inconsistencies in the data. It is also used to examine feature distributions, class imbalance, and any anomalies [1], [13]. Snapshot of normal instances and various attack classes found in the network-based IDS is shown in Figure 2.

The graph clearly indicates an unbalanced dataset with a class called "Normal" has a higher count than the remaining classes. The next class named "Neptune" has a count of 51,820 and "Back", "Tear drop", "Satan" has a count of 968, 918, 906 respectively, which indicates a heavy drop-down count. Classes "Warezcclient", "Ipsweep", "Smurf", "Portsweep", "Pod", and "Nmap" has a count of 893, 651, 641, 416, 206, and 158 respectively, which indicates a major decline while compared to the "Normal" class, the remaining classes "Quess_passwd", "Land", "Imap", "Buffer_overflow", "Warezmater", "Rootkit", "Loadmodule", "Phf", "Perl", "Spy", "Ftp_write", and "Multihop" has a negotiable count of less than 100.

Since the dataset is unbalanced, F1-score is used to evaluate the performance of the model [13]. It maintains balance among precision and recall in such a way that if false positive increases, precision decreases. On the other hand, if the false negative increases, recall decreases. Thus, F1-score is improved only if the classification is balanced. So, we need to construct an algorithm that works efficiently in classifying the different classes accurately. OSOVOC algorithm is proposed for detecting possible attack on the networks.

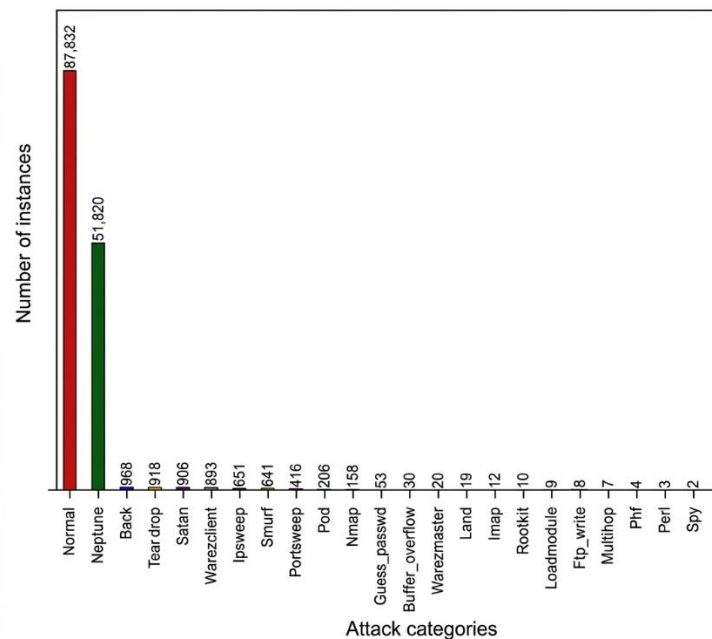


Figure 2. Class distribution of normal instances and different network attack types in the dataset

3. CONSTRUCTING THE TRIPARTITE OPTIMIZED HYBRIDIZATION FRAMEWORK

Hybridization methods combine the predictions from two models, making it working robustly and effectively in an unbalanced dataset [7], [23]. The results from the individual classifiers namely OSVM, ODT were combined through the OSOVOC algorithm to obtain a final classifier which will be used to test the real data. Hyper parameters increase the performance of model. They are selected, on testing the model with various parameter values. The values or parameters that produces the most favorable outcome will be chosen. Thus, optimization is done to obtain more accurate results. Then, k fold cross validation is also done by selecting one partition (k) for testing and remaining partition (k-1) for training. The validation procedure is replicated with various k and k-1 partition for training and testing.

The proposed model is limited for two cross validation sets. This enables the model to consume less time for training and testing. A comparative analysis is done to choose the best classifier. The two classifiers namely DT and SVM were not as accurate as standalone alone algorithms. So, hybridization strategy is employed to combine the both models using OSOVOC algorithm to produce an effective network-based IDS. The final proposed model produces a more accurate results with FPR and FNR reducing notably.

3.1. Optimized decision tree classifier

DT performs classification and regression task. It breaks the complex decisions into smaller steps for easy understanding and implementation. The DT classifier works by splitting the data in the best selected feature. The process starts from the entire dataset that is root node. A split at a node happens for each possible value of the selected feature. Repeat the process until a maximum depth is reached or end nodes or leaf nodes has been reached [20]. Hyper parameter is tuned to a minimum sample split of 200 and a maximum depth of 20 to get an optimized result. The output obtained from the recall, precision, and F1-measure of ODT classifier is compared with OSOVOC algorithm output.

3.2. Optimized support vector machine classifier

SVM is a supervised ML algorithm primarily used for regression and classification purpose. It classifies a new incoming data pattern into the correct side, by creating a decision boundary named hyper plane [22]. The SVM works from the pre-processed dataset. Then, an appropriate gamma value is chosen for classifier. Finally, a suitable decision boundary is chosen based on the hyper parameter of 0.01 to get an optimized result. The output obtained from the recall, precision, and F1-measure of OSVM classifier is compared with OSOVOC algorithm output.

3.3. Optimized soft voting classifier algorithm

The OSOVOC algorithm combines the predictions obtained from the ODT classifier and OSVM classifier. It works on the principle of weighted average prediction from each class [15], [16]. This algorithm

is hyper tuned to the weight parameter of (1, 3), which means, the SVM classifier takes the lowest weightage of one. While, DT classifier takes the highest weightage of three. Thus, it helps to prioritize each model based on their strengths for improving the overall accuracy.

4. RESULTS AND DISCUSSION

The experimental results obtained from the proposed model and a detailed discussion of its performance is presented in this section. The results obtained from the various classifiers are evaluated using the standard evaluation metrics. By examining the output, we can identify the areas for improvement, to obtain the maximum benefit from the model. The performance of the proposed system is evaluated using various metrics such as recall, precision, F1-score, and accuracy. The proposed system was found to be very effective in detecting intrusions based upon the performance.

4.1. Outcome of optimized decision tree classifier

Figure 3 illustrates the multi class classification results of the ODT classifier. The model's performance is evaluated from the test dataset by comparing the predicted and actual output generated. The results shows that the training accuracy reaching 98.04% and the testing accuracy reaching 97.94%. Observations from the prior network-based intrusion detection studies indicate that the outcome of the ODT classifier showed improved baseline performance when compared to non-ODT classifier [13]. Thus, ODT is necessary for improving the efficiency of the proposed model.

Actual Class	back	butter_overflow	loadmodule	guess_passwd	imap	ipsweep	warezmaster	rootkit	multihop	neptune	hmap	normal	phf	perl	pod	portsweep	ftp_write	satan	smurf	teardrop	warezclient	land
back	242	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
butter_overflow	0	3	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
loadmodule	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
guess_passwd	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
imap	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ipsweep	0	0	0	0	0	157	0	4	0	2	0	0	0	0	0	0	0	0	0	0	0	0
warezmaster	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rootkit	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
multihop	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0
neptune	0	0	0	0	25	34	0	0	0	238	0	0	0	0	0	23	0	0	0	36	0	0
hmap	0	0	0	0	0	0	0	0	0	0	34	0	0	0	0	0	0	0	6	0	0	0
normal	0	46	15	5	0	17	0	28	22	2	50	124	1	4	0	2	4	87	0	10	0	348
phf	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
perl	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
pod	0	0	0	1	0	0	0	0	0	0	0	0	0	0	51	0	0	0	0	0	0	0
portsweep	0	0	0	0	0	0	3	0	1	0	0	1	0	0	0	98	0	1	0	0	0	0
ftp_write	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
satan	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	220	0	0	0	3
smurf	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	160	0	0	0
teardrop	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
warezclient	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	230	0	0
land	0	0	2	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	213	5
	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
Predicted Class	back	butter_overflow	loadmodule	guess_passwd	imap	ipsweep	warezmaster	rootkit	multihop	neptune	hmap	normal	phf	perl	pod	portsweep	ftp_write	satan	smurf	teardrop	warezclient	land

Figure 3. ODT Multi classification result

4.2. Outcome of optimized support vector machine classifier

Figure 4 illustrates the multi class classification results of the OSVM classifier. The model's performance is evaluated from the test dataset by comparing the predicted and actual output generated. The results shows that the training and testing accuracy reaching 97.85% and 97.82% respectively. The previous network-based intrusion detection studies indicate that the outcome of the optimized support vector classifier showed an improved baseline performance when compared to non-optimized support vector classifier [13]. Since, the classifier can construct effective decision boundaries even in high dimensional feature space, optimization improves the detection rate across all attack categories. So, OSVM is used as base classifier in the proposed model.

4.3. Outcome of optimized soft voting classifier algorithm

As shown in Figure 5, the proposed OSOVOC algorithm illustrates the effective multi class classification results across all attack categories. The model's performance is evaluated from the test dataset by comparing the predicted and actual output generated. Averaging the class wise predictions by applying the weightage to each base classifier makes the OSOVOC algorithm to produce a highly reliable discrimination

across all the attack categories. As a result, the OSOVOC algorithm reduced the FPR and the FNR when compared to individual base classifiers.

Actual Class	back	butter_overflow	loadmodule	guess_passwd	imap	ipsweep	warezmaster	rootkit	multihop	neptune	nmap	normal	phf	perl	pod	portsweep	ftp_write	saturn	smurf	teardrop	warezclient	land
Predicted Class	back	butter_overflow	loadmodule	guess_passwd	imap	ipsweep	warezmaster	rootkit	multihop	neptune	nmap	normal	phf	perl	pod	portsweep	ftp_write	saturn	smurf	teardrop	warezclient	land
back	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
butter_overflow	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
loadmodule	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
guess_passwd	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
imap	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ipsweep	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
warezmaster	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rootkit	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
multihop	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
neptune	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
nmap	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
normal	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
phf	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
perl	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
pod	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
portsweep	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ftp_write	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
saturn	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
smurf	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
teardrop	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
warezclient	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
land	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4. OSVM multi classification result

Actual Class	back	butter_overflow	loadmodule	guess_passwd	imap	ipsweep	warezmaster	rootkit	multihop	neptune	nmap	normal	phf	perl	pod	portsweep	ftp_write	saturn	smurf	teardrop	warezclient	land
Predicted Class	back	butter_overflow	loadmodule	guess_passwd	imap	ipsweep	warezmaster	rootkit	multihop	neptune	nmap	normal	phf	perl	pod	portsweep	ftp_write	saturn	smurf	teardrop	warezclient	land
back	242	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
butter_overflow	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
loadmodule	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
guess_passwd	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
imap	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ipsweep	0	0	0	0	0	163	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
warezmaster	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rootkit	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
multihop	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
neptune	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0
nmap	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
normal	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
phf	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
perl	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
pod	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
portsweep	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ftp_write	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
saturn	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
smurf	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
teardrop	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
warezclient	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
land	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5. OSOVOC multi classification result

The binary classification result obtained from the OSOVOC algorithm during the training and testing phase is shown in Tables 1 and 2 respectively. Experimental results indicate that, the proposed OSOVOC algorithm achieved a training accuracy of 100% and a testing accuracy of 99.91%. In an unbalanced dataset, combining heterogeneous base classifiers significantly reduces the false alarm rate, as reported in prior intrusion detection studies [15]. While existing literature comparing hard and soft voting strategies reported a higher accuracy for hard voting (100%) than for soft voting (99.967%) on a balanced dataset [16], the proposed OSOVOC algorithm provides a superior false alarm control with improved accuracy on an unbalanced dataset. These findings confirm that incorporating weighted average prediction for each class in a soft voting classifier significantly reduces the false alarm rate.

Table 1. Training phase result

Confusion matrix (training phase)		Actual class	
Predicted class	True	Positive	Negative
	False	65,873	43,315
		0	1

Table 2. Testing phase result

Confusion matrix (training phase)		Actual class	
Predicted class	True	Positive	Negative
	False	21,940	14,424
		15	18

4.4. Evaluation report

As illustrated in Table 3, the proposed model outperforms the baseline algorithms across multiple evaluation metrics. From the performance evaluation report, it is found that the OSOVOC algorithm produces a highest F1-score of 100% during training phase and a F1-score of 99.86% during testing phase. Next, ODT algorithm produces a F1-score of 98.23 during training phase and 98.16 during testing phase. Finally, OSVM algorithm produces a F1-score of 96.97% in training and testing phases. The visual representation of performance evaluation of the three algorithms is illustrated in Figure 6.

Table 3. The performance evaluation of various algorithms

Algorithm	Precision (%)	Recall (%)	Training F1-score (%)	Testing F1-score (%)
ODT	99.11	97.49	98.23	98.16
OSVM	96.67	97.72	96.97	96.97
OSOVOC	99.87	99.86	100	99.86

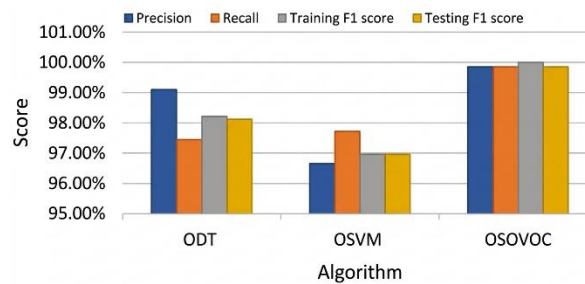


Figure 6. Performance evaluation chart

Since the primary focus of the proposed OSOVOC algorithm is the reduction of false alarms, generating a false alarm report is essential. Lowering the false alarm count leads to an increase in both true positive and true negative counts. The proposed model achieves a significantly reduced false alarm rate, indicating improved discrimination between normal and attack traffic. Table 4 demonstrates the false alarm rate generated by the algorithms ODT, OSVM, and OSOVOC.

The findings specify that, OSOVOC algorithm succeeds in reducing the false alarm and exponentially increases the true positive rate of 100% during training phase and 99.92% during testing phase. These results were obtained in a controlled experimental setup using the KDDCup99 dataset. Future work will be focused on validating the proposed model using the real-world network traffic. The visual representation of FPR and FNR generated from the three algorithms is shown in Figure 7.

Table 4. Evaluation of false alarm report

Algorithm	Training		Testing	
	FPR (%)	FNR (%)	FPR (%)	FNR (%)
ODT	0	3	0	3
OSVM	5	0	5	0
OSOVOC	0	0	0	0

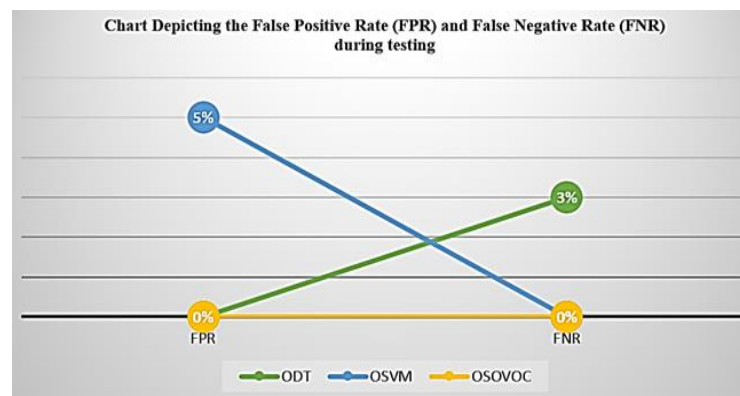


Figure 7. False alarm chart

5. CONCLUSION

This paper focus on the hybridization of ODT and OSVM algorithms using OSOVOC algorithm. From performance evaluation of the two classifiers, ODT is performing better when compared to OSVM. So, ODT has been given higher weightage than the OSVM during hybridization. Hyper parameter tuning promoted the model to achieve the desired results. Even though the two optimized classifiers DT and SVM performing individually well, OSOVOC hybridization algorithm enhanced the framework to achieve the maximum accuracy by eradicating the false alarm rate notably. Thus, an effective network-based IDS has been proposed for detecting the possible threats in the network. Though the OSOVOC algorithm consumes more time compared to the other two classifiers execution time, the security it provides to the network through its highest threat detection rate makes it a suitable choice. Another concern is the detection of new attacks. Further research will be focused on with the latest dataset available in network-based IDS, for detection of any new attacks.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Poonkody Iruthayaraj	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	
Poornima	✓	✓			✓	✓	✓			✓	✓	✓	✓	
Vivekanandhan														

C : Conceptualization

M : Methodology

So : Software

Va : Validation

Fo : Formal analysis

I : Investigation

R : Resources

D : Data Curation

O : Writing - Original Draft

E : Writing - Review & Editing

Vi : Visualization

Su : Supervision

P : Project administration

Fu : Funding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The data that support the findings of this study are openly available in Stolfo, Salvatore, Fan, Wei, Lee, Wenke, Prodromidis, Andreas, and Chan, Philip. (1999). KDD Cup 1999 Data. UCI ML Repository. <https://doi.org/10.24432/C51C7N>.

REFERENCES

- [1] M. A. Talukder *et al.*, "Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction," *Journal of Big Data*, vol. 11, no. 1, 2024, doi: 10.1186/s40537-024-00886-w.
- [2] H. N. Thanh, "The data preprocessing in improving the classification quality of network intrusion detection systems," *EAI Endorsed Transactions on Context-aware Systems and Applications*, vol. 9, no. 1, Sep. 2023, doi: 10.4108/eetcasa.v9i1.3778.
- [3] G. Nassreddine, M. Nassereddine, and O. Al-Khatib, "Ensemble learning for network intrusion detection based on correlation and embedded feature selection techniques," *Computers*, vol. 14, no. 3, Feb. 2025, doi: 10.3390/computers14030082.
- [4] M. G. D. S. Neto and D. G. Gomes, "Network intrusion detection systems design: a machine learning approach," in *Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2019)*, Sociedade Brasileira de Computação, May 2019, pp. 932–945, doi: 10.5753/sbrc.2019.7413.
- [5] S. A. Birahim *et al.*, "Intrusion detection for wireless sensor network using particle swarm optimization based explainable ensemble machine learning approach," *IEEE Access*, vol. 13, pp. 13711–13730, 2025, doi: 10.1109/ACCESS.2025.3528341.
- [6] F. Mokbal, W. Dan, M. Osman, Y. Ping, and S. Alsamhi, "An efficient intrusion detection framework based on embedding feature selection and ensemble learning technique," *The International Arab Journal of Information Technology*, vol. 19, no. 2, 2022, doi: 10.34028/iajit/19/2/11.
- [7] D. N. Mhawi, A. Aldallal, and S. Hassan, "Advanced feature-selection-based hybrid ensemble learning algorithms for network intrusion detection systems," *Symmetry*, vol. 14, no. 7, Jul. 2022, doi: 10.3390/sym14071461.

- [8] K. A. Taher, B. M. Y. Jisan, and M. M. Rahman, "Network intrusion detection using supervised machine learning technique with feature selection," in *2019 International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*, 2019, pp. 643–646, doi: 10.1109/ICREST.2019.8644161.
- [9] M. A. Hossain and M. S. Islam, "Ensuring network security with a robust intrusion detection system using ensemble-based machine learning," *Array*, vol. 19, Sep. 2023, doi: 10.1016/j.array.2023.100306.
- [10] I. Bibers, O. Arreche, and M. Abdallah, "A comprehensive comparative study of individual ML models and ensemble strategies for network intrusion detection systems," 2024, *arXiv: 2410.15597*.
- [11] E. Alalwany and I. Mahgoub, "An effective ensemble learning-based real-time intrusion detection scheme for an in-vehicle network," *Electronics*, vol. 13, no. 5, Feb. 2024, doi: 10.3390/electronics13050919.
- [12] R. Sathya, "Ensemble machine learning techniques for attack prediction in NIDS environment," *Iraqi Journal for Computer Science and Mathematics*, pp. 78–82, Mar. 2022, doi: 10.52866/ijcs.2022.02.01.008.
- [13] R. Ahsan, W. Shi, and J. Corriveau, "Network intrusion detection using machine learning approaches: addressing data imbalance," *IET Cyber-Physical Systems: Theory & Applications*, vol. 7, no. 1, pp. 30–39, Mar. 2022, doi: 10.1049/cps2.12013.
- [14] M. Uppal, Y. Gulzar, D. Gupta, J. Uppal, M. Kumar, and S. Saini, "Enhancing accuracy through ensemble based machine learning for intrusion detection and privacy preservation over the network of smart cities," *Discover Internet of Things*, vol. 5, no. 1, Feb. 2025, doi: 10.1007/s43926-025-00101-z.
- [15] A. A. Wani, J. Iqbal, and M. Makhdoomi, "Modelling an intrusion detection system using ensemble approach based on voting to improve accuracy of base classifiers," *Journal of Algebraic Statistics*, vol. 13, no. 2, pp. 1844–1865, 2022.
- [16] H. G. Jabbar, "Advanced threat detection using soft and hard voting techniques in ensemble learning," *Journal of Robotics and Control (JRC)*, vol. 5, no. 4, pp. 1104–1116, 2024, doi: 10.18196/jrc.v5i4.22005.
- [17] S. Yoheswari, "Optimized intrusion detection model for identifying known and innovative cyber attacks using support vector machine (SVM) algorithms," *Journal of Science Technology and Research*, vol. 5, no. 1, pp. 401–407, 2024.
- [18] T.-H. Chua and I. Salam, "Evaluation of machine learning algorithms in network-based intrusion detection system," 2022, *arXiv: 2203.05232*.
- [19] A. N. Banu and S. K. B. Sangeetha, "Ensemble machine learning algorithm methods for detecting the attacks using intrusion detection system," *Journal of Theoretical and Applied Information Technology*, vol. 102, no. 5, pp. 2207–2229, 2024.
- [20] S. Latif, F. F. Dola, M. M. Afsar, I. J. Esha, and D. Nandi, "Investigation of machine learning algorithms for network intrusion detection," *International Journal of Information Engineering and Electronic Business*, vol. 14, no. 2, pp. 1–22, Apr. 2022, doi: 10.5815/ijieeb.2022.02.01.
- [21] Y. Almutairi, B. Alhazmi, and A. Munshi, "Network intrusion detection using machine learning techniques," *Advances in Science and Technology Research Journal*, vol. 16, no. 3, pp. 193–206, Jul. 2022, doi: 10.12913/22998624/149934.
- [22] A. Bachar, N. E. Makhfi, and O. E. Bannay, "Machine learning for network intrusion detection based on SVM binary classification model," *Advances in Science, Technology and Engineering Systems Journal*, vol. 5, no. 4, pp. 638–644, 2020, doi: 10.25046/aj050476.
- [23] G. A. Mills, D. K. Acquah, and R. A. Sowah, "Network intrusion detection and prevention system using hybrid machine learning with supervised ensemble stacking model," *Journal of Computer Networks and Communications*, vol. 2024, no. 1, Jan. 2024, doi: 10.1155/2024/5775671.
- [24] P. K. Tiwari *et al.*, "A secure and robust machine learning model for intrusion detection in internet of vehicles," *IEEE Access*, vol. 13, pp. 20678–20690, 2025, doi: 10.1109/ACCESS.2025.3532716.
- [25] M. Ali *et al.*, "Effective network intrusion detection using stacking-based ensemble approach," *International Journal of Information Security*, vol. 22, no. 6, pp. 1781–1798, Dec. 2023, doi: 10.1007/s10207-023-00718-7.
- [26] O. E. Aeraj and L. Cherkaoui, "Intelligent intrusion detection system snort and SVM," *Revue d'Intelligence Artificielle*, vol. 37, no. 6, pp. 1629–1635, Dec. 2023, doi: 10.18280/ria.370627.

BIOGRAPHIES OF AUTHORS



Poonkody Iruthayaraj    is a Ph.D. research scholar in Computer Science at Vels Institute of Science, Technology and Advanced Studies, Chennai. She obtained her M.C.A from Madurai Kamaraj University in 2002. She is working as an assistant professor in the Department of Computer Applications, and having more than 10 years of teaching experience. She has published nearly 3 technical papers in journals and presented a paper in international conference. Her research interests are network security and ML. She can be contacted at email: poonkody@gmail.com.



Dr. Poornima Vivekanandhan    was awarded a Ph.D. from Bharathiar University in 2020. She is currently serving as an associate professor at the Department of Computer Science, Vels Institute of Science, Technology and Advanced Studies, Chennai, India. With over 15 years of teaching experience at both undergraduate and postgraduate levels, she has demonstrated a strong commitment to academic excellence. She has published more than 15 technical papers in various Scopus-indexed journals and several additional publications in other reputed indexed journals. Her areas of interest include ANN, the internet of things (IoT), and ML. She can be contacted at email: poornimasudhaagar@gmail.com.