Network Intrusion Detection and Comparative Analysis Using Ensemble Machine Learning and Feature Selection

Article *in* IEEE Transactions on Network and Service Management · December 2021

DOI: 10.1109/TNSM.2021.3138457

CITATIONS
71

READS
888

7 authors, including:

Saikat Das
University of Memphis
27 PUBLICATIONS **558** CITATIONS
SEE PROFILE

Sajal Saha
Western University
32 PUBLICATIONS **402** CITATIONS
SEE PROFILE

Annita Priyoti
Western University
4 PUBLICATIONS **137** CITATIONS
SEE PROFILE

F.T. Sheldon
University of Idaho
239 PUBLICATIONS **2,777** CITATIONS
SEE PROFILE

Network Intrusion Detection and Comparative Analysis using Ensemble Machine Learning and Feature Selection

Saikat Das, Sajal Saha, Annita Tahsin Priyoti, Etee Kawna Roy, Frederick T. Sheldon, *Senior Member, IEEE,* Anwar Haque, and Sajjan Shiva, *Fellow, IEEE*

***Abstract*—Proper security solutions in the cyber world are crucial for enforcing network security by providing real-time network protection against network vulnerabilities and data exploitation. An effective intrusion detection strategy is capable of taking a holistic approach for protecting critical systems against unauthorized access or attack. In this paper, we describe a machine learning (ML) based comprehensive security solution for network intrusion detection using ensemble supervised ML framework and ensemble feature selection methods. In addition, we provide a comparative analysis of several ML models and feature selection methods. The goal of this research is to design a generic detection mechanism and achieve higher accuracy with minimal false positive rates (FPR). NSL-KDD, UNSW-NB15, and CICIDS2017 datasets are used in the experiment, and results show that our detection model can identify 99.3% of intrusions successfully with the lowest 0.5% of false alarms, which depicts better performance metrics compared to existing solutions.**

Keywords: Intrusion detection system, Ensemble machine learning, Ensemble feature selection, Comparative analysis

I. Introduction

The growing frequency of cyber-attacks is an eminent problem of the modern era. These attacks are detrimental for both individuals and enterprises, and are capable of impacting the confidentiality, integrity, and availability of critical information that is carried through the network. It is essential for enterprises to protect their networks against intruders, hackers, and network threats. Therefore, a network system must incorporate several security tools to protect important data and services from potential threats. The Intrusion Detection System (IDS) is a software or hardware implementation for screening vindictive movement or strategy infringement in the network. It monitors the malicious activity or security violation within the network and notifies the administrator of potential threats. Cyber-attacks are becoming pernicious day by day. In order to keep pace with the constantly changing cyber threats, modern networked business environments require a high level of security for safe and trusted communication of information between organizations. Traditional IDSs (Signature-based and Anomaly-based) are not adequately designed to adapt to the continuously changing patterns of network intrusion. Incorporating artificial intelligence within the IDS has the potential to cope with the new attack patterns and to ensure network security.

In the last few decades, machine learning (ML), a subset of artificial intelligence, has been used extensively to improve intrusion detection by enabling automation and prediction on an advanced level. ML techniques enable dealing with the modifiable, reproducible, and extensible datasets. These techniques help IDSs to become robust by learning and tackling seen and unseen attacks. In addition, a single ML model may not always predict accurately, whereas a combination of ML models (ensemble ML) detect more precisely. Dietterich et al., [1] mentioned that ensemble ML classifiers outperform individual classifiers in solving various classification problems. Besides, Feature Selection (FS) is a process of selecting a subset of appropriate and relevant features from a large feature space. It plays a vital role in achieving higher accuracy as well as minimizing the model training time. The challenges of feature optimization and designing IDSs with the best suitable ML techniques motivated us to investigate the performances of various FS methods. Also, we were motivated to explore the ensemble techniques for feature selection and model classification as it consider several candidates' decisions for optimization rather than individual choices.

In this research, we have extended our existing work [2] and implemented a supervised ensemble ML framework (SupEnML) by combining multiple ML classifiers from various classification families such as Decision Tree, Logistic regression, Naïve Bayes, Neural Network, Support Vector Machine for network intrusion detection. Furthermore, the ensemble feature selection (EnFS) technique has been incorporated inside the classification procedure of this framework. This EnFS technique combines three major FS method types (Filter-based, Wrapper-based and Embedded methods) for feature selection such as Chi-squared, Mutual Information, Pearson Correlation, SFPR, Logistic Regression with L1 penalty, Random Forests, ANOVA, Recursive Feature Elimination, and LASSO. The yielded features are ensembled with majority voting and resulted in an optimized set of features. After getting the feature set, both individual and ensemble classi-

S. Das, and S. Shiva are with the Department of Computer Science, University of Memphis, Memphis, TN, 38152 USA e-mail: {sdas1, sshiva}@memphis.edu.

S. Saha, A T. Priyoti, and A. Haque are with the Department of Computer Science, Western University, London, ON N6A 5B7 Canada e-mail: {ssaha59, apriyoti, ahaque32}@uwo.ca

E K. Roy is with the Department of Electrical Engineering, Arkansas State University, Jonesboro, AR, 72401 USA email: {eteekawn.roy}@smail.astate.edu.

F.T. Sheldon is with the Department of Computer Science, University of Idaho, Moscow, ID, 83843 USA email: {sheldon}@ieee.org.

fication have been performed using the SupEnML framework. This study shows comparative performance analysis of single models (built from individual classifications) and ensemble models (built from ensemble classifications). Three benchmark intrusion detection datasets: NSL-KDD [3], UNSW-NB15 [4], and CICIDS2017 [5] are used for the experimentation, where the outcome displays significant improvement of performance in terms of the accuracy and false-positive rate for our ensemble-based approach. The proposed approach detects network intrusion more accurately and efficiently than existing ML-based intrusion detection methods mentioned in the literature [6], [7], [8], [9]. The key contributions of this research are listed as

1) A grid search approach has been implemented to find the best performing FS methods from a significant number of well-known FS methods.
2) We proposed an ensemble feature selection technique (EnFS) which combines the features obtained from selected best performing FS methods.
3) Our proposed ensemble FS technique extracts an optimized feature set which significantly improves our classification results.
4) Supervised ensemble ML framework (SupEnML) [2] has been extended to detect network anomaly over DDoS.
5) To provide better performance, we've integrated our proposed EnFS technique with the extended version of the SupEnML framework.
6) A comparative performance analysis of individual FS methods with our proposed EnFS method as well as individual ML models with ensemble ML models has been done.
7) To validate the performance of our framework, and implement a robust & generic IDS, we've extensively experimented with three well-known intrusion datasets.

The rest of the paper is organized as follows: In Section II, we discuss the state-of-the-art IDS that uses ensemble learning and distinguish our contribution from other approaches. We provide a threat model that describes well-known cyber threats and the attackers' motivation in Section III. We propose an IDS framework in Section IV. In Section V, we show and elaborate our experiments which were carried out with three datasets and resulted in the most relevant and minimal features sets with best performing models. Finally, in Section VI, we discuss the pros and cons of our approach and conclude the paper with the future research direction. The detailed results of our full-stack experiments are listed in this URL, https://github.com/CSMLGroup/NIDS-SupEnML-EnFS/, which is a supplementary document of this paper.

II. Related Work

Machine Learning (ML) algorithms focus on the development of computer programs with the system's ability to automatically acquire and improve functionality without human intervention. Tsai et al., [10] reviewed 55 papers on intrusion detection using ML and performed a comparative analysis of algorithms and datasets used. Mukkamala et al., [11] performed a comparative analysis on artificial neural network, support vector machine, and the ensemble of these two models. They concluded that the ensemble classifier outperformed these single classifiers in detecting intrusions. Chebrolu et al., [12] proposed a lightweight intrusion detection technique using two feature selection methods and ensemble learning. Amiri et al., [13] studied the effect of feature selection and proposed a modified mutual information-based feature selection method (MMIFS) using the KDD cup 99 data set. Deep learning based detection system is becoming promising now a days. Vinayakumar, Ravi, et al., [14] proposed a highly scalable and hybrid DNNs framework which is capable of detecting real-time cyberattacks by monitoring network traffic and host level events. These authors have also examined and evaluated the effectiveness of various shallow and deep networks for Network Intrusion Detection Systems (NIDS) in a different research [15]. Both research have been evaluated with the KDDCup' 99 dataset. Moreover, Vinayakumar, Ravi, et al., [16] have proposed a deep learning-based botnet detection system for detecting and classifying domain names, and this system can be deployed at the ISP level for monitoring IoT devices.

Gomes et al., [17] proposed a taxonomy of ensemble methods for data stream classification, identified open-source tools, and discussed the upcoming trends in ensemble learning. Sagi, Omer, and Lior Rokach [18] reviewed the major approaches and techniques for ensemble learning and discussed the future trends, including refinement of popular algorithms for big data compatibility, model transformation into a more straightforward form, and integration with deep neural networks. Gao et al., [6] designed an ensemble adaptive voting algorithm to improve the detection accuracy using the NSL-KDD dataset and compared it with their constructed MultiTree algorithm. Tu Pham et al., [7] proposed an improved IDS using limited FS (25 subsets and 35 subsets of features) and ensemble models based on bagging and boosting techniques which depicted high accuracy for the NSL-KDD dataset. Das et al., [2] designed an ensemble framework for supervised classifiers, which accomplishes an immense increase in accuracy for detecting DDoS attacks based on four ML classifiers. Kittler et al., [19] developed a theoretical framework for classifier combination schemes, and analyzed the sensitivity of these schemes to the estimated errors, especially for sum rule.

To improve the ML algorithm's efficiency, feature selection (FS) plays a vital role during the preprocessing phases by removing unwanted and irrelevant features. Chandrashekhar and Sahin [20] provided a thorough study on the FS technique that demonstrates improved performance in analyzing standard and RF generator datasets with supervised learning. Sheikhpour et al., [21] investigated the semi-supervised FS methods that use both labeled and unlabeled data and illustrated two taxonomies based on a hierarchical structure. Khalid et al., [26] performed a detailed survey on feature selection and extraction to reduce dimensionality for improved data analysis. Luis et al., [27] proposed an evaluation technique to determine the correct usage of feature selection algorithms based on a scoring measure and explained the particularities of relevance, irrelevance, and redundancy. Adams and Beling [28] performed a survey on FS methods for Gaussian

TABLE I: A comparative analysis of the study of literature review

Classification Method	Feature selection method	Pre-processing	Pros	Cons	Dataset
Ensemble [11]	Not mentioned	Not mentioned	Comparative analysis and proposed two ensemble methods	Experiment is done on one dataset only.	DARPA, KDDCup'99
Supervised [12]	Not mentioned	Not mentioned	Suitable for specific intrusion detection	No preprocessing and not as a general purpose IDS	DARPA, KDDCup'99
Supervised [13]	Modified Mutual Information	FS methods for preprocessing	Proposed two FS method and performed comparative analysis	Experiment is done on one dataset only.	KDDCup' 99
Deep Learning [14], [15], [16]	Deep learning	Levenshtein distance function	They examine the effectiveness of shallow and deep networks to NIDS. Highly scalable with real time data processing	Training was not done using advanced hardware through distributed approach	KDDCup 99, NSL-KDD, UNSW-NB15, Kyoto, WSN-DS, and CICIDS 2017
Ensemble [6]	Not mentioned	One hot encoding, PCA, scaling	Ensemble model effectively improves detection accuracy.	No FS method is applied	NSL-KDD
Ensemble [7]	Leave-one-out techniques and NB, Gain Ratio	FS methods for preprocessing	Achieved high accuracy and low false alarm rate (FAR)	Experiment is done on one dataset only.	NSL-KDD
Ensemble [2]	Existing methods	MinMax, transformed, normalized	DDoS attack detection with higher accuracy	only one dataset was used to evaluate the built classifiers.	NSL-KDD
Supervised [20]	Filter, Wrapper, Embedded method	PCA, FFT	Providing overall view for supervised feature selection method with comparison.	Unsupervised learning methods are not covered.	Breast cancer, Diabetes, Ionosphere, Liver disorder,Medical,Fault mode
Supervised [21]	Filter, Wrapper, Embedded method	PCA	Two taxonomies of semi-supervised feature selection methods presented	The have'nt provided the solution for regression problems.	WDBC, WBCD, Diabetes
Supervised [22]	Combination of SVM, DT, and SA	Not mentioned	Obtained decision rules for detecting new attacks	Experiment is done on one dataset only.	KDDCup'99
Supervised [23]	Info gain Gain ratio Chi-squared ReliefF	EMFS	Ensemble-based multi-filter feature selection method provides better accuracy with less complexity.	Needs to incorporate more dataset and algorithm.	NSL-KDD
Ensemble [24]	EnFS	MinMax,sanitized, transformed, normalized	Not mentioned	NSL-KDD	
Ensemble, Unsupervised [25]	Not mentioned	Non numeric to numeric conversion	Detects DDoS attack using ensemble classifiers and a reduced feature dataset.	Not mentioned	NSL-KDD

mixture models (GMM) and hidden Markov models (HMM), demonstrating that FS methods developed for GMM can be adapted for HMM and vice versa. Lin et al., [22] performed research on intrusion detection using the KDD'99 dataset and proposed an intelligent algorithm with the FS method. Osanaiye et al., [23] came up with an ensemble-based multi-filtered feature selection technique combining four filter-based FS methods. The technique generated important features from the selected algorithm with higher accuracy for the NSL-KDD dataset. Das et al., [24] proposed an ensemble framework for FS methods to produce an optimal set of features and performed a comparative analysis of the existing approaches in terms of accuracy and false alarms using the NSL-KDD dataset.

In a nutshell, the literature study indicates that the ensemble technique can be broadly applied in feature selection and model classification to obtain better outcomes in intrusion detection. We have found that the researchers were focused on detecting specific cyber-attack from a particular dataset. Here, the goal of our research is to provide a generic intrusion detection framework with higher accuracy using various anomalous datasets, like NSL-KDD, UNSW-NB15 and CICIDS2017. As the critical features are important in model classification, we have increased the number of FS methods in comparison to earlier research. We also ensemble filter-based, wrapper-based, and embedded FS methods in our EnFS framework. Finally, we have incorporated ensemble feature selection (EnFS) with ensemble classification (SupEnML) to get better accuracy with minimal false alarms.

III. Threat Model

Secure Development Lifecycle (SDL) [29] is a well-known methodology and is an integration of security and privacy. In order to build a secure software system, it is important to perceive the attacker's way of thinking. For example, an attacker may plan an attack and make the system to compromise by exploiting the design flaws and by building the necessary defense mechanisms within the system. In this

respect, threat modeling is crucial in that SDL provides an integrated systematic approach to secure the system using STRIDE.

According to Uzunov and Fernandez [30], threat modeling is a process used in analyzing possible attacks or threats and is supported by threat libraries or attack taxonomies. In other words, threat modeling "provides a structured way to secure software design, which involves understanding an adversary's goal in attacking a system based on system's assets of interest". Xiong, Wenjun [31] categorized the threat modeling methods as either graphical or formal. Carnegie Mellon Software Engineering Institute [32] has discussed a variety of threat modeling methods, like STRIDE, PASTA, LINDDUN, CVSS, hTMM, Trike, VAST, and OCTAVE. Among these, STRIDE threat modeling method is considered to be the best mature one. This method was proposed by Loren Kohnfelder and Praerit Garg in 1999 and was adopted by Microsoft in 2002 [33]. Deng Mina, et al., [34] provided a detailed survey on threat modeling methods and their validation processes.

In this research, we followed the STRIDE threat modeling method and addressed most of the nine high-level steps of STRIDE [34]. Let's consider that the attacker plans to perform any of the activities as summarized below:

- malfunctions the system operation by compromising it or sending traffic from several different compromised hosts or botnets.
- floods the victim through spoofing IP addresses.
- escalates his user access to gain control of the system.
- replicates the malicious code inside the target with/without backdoor access.
- exploits the system's vulnerability by using malformed software.
- intrudes through any layer of the OSI framework.

We assume these following attacks can be occurred as our datasets consist of these attack data samples. They are, User to Root Attack, Remote to Local Attack, Probing Attack, Fuzzers, Analysis, Backdoors, Exploits, Generic, Reconnaissance, Shellcode, Worms, Brute Force Attack, Heartbleed Attack, Botnet, Distributed Denial of Service Attack (DDoS), Web Attack, Infiltration Attack. Our proposed method can only protect the system from the attack occurrence.

The validation system is one of the fundamental stages of any threat model to evaluate the performance. The threat model validation process can be categorized into two groups: theoretical and empirical validation. Theoretical validation is based on the existing theories, hypothesis, and simulations, while empirical proof is based on the verification through experiments, experiences, and observation [31]. In our proposed threat model, we applied the empirical validation approaches, which are more specifically experiment-based. The performance of our threat has been measured by collecting and analyzing data through comprehensive experiments. We validated our ultimate result by comparing different experimental models on a different dataset.

IV. Methodology

This section provides an overview of our proposed methodology. The detailed architecture illustrating the process flow is given in Fig. 1 and 2. The methodology is divided into five main phases, namely a) data collection, b) data preprocessing, c) ensemble feature selection, d) model classification, and e) anomaly detection.

A. Data Collection

The performance of a machine learning model largely depends on the quality of the data used for model training. The data contain relevant information of the problem domain and need to be cleaned for further analysis. Here, we use three well-known datasets from different research organizations to train, test, and verify our proposed framework, and they are NSL-KDD, CICIDS2017, and UNSW-NB15 from the Canadian Institute of Cybersecurity and the University of New South Wales. Apart from dividing the dataset into train and test data, we have a third portion of the dataset that we call the 'verification data'. This verification data mimics the real time data and is used to verify our proposed model within the IDS.

B. Data Preprocessing

The crude datasets need to be cleaned, sanitized, transformed, normalized, and feature reduced before feeding the ML classifiers. To clean and sanitize the dataset, we remove the rows containing ',' or 'inf' values. For the CICIDS2017 dataset, the single space before each of the feature names was stripped. We also deleted the column 'id' in the UNSW-NB15 dataset. As the ML models can handle numeric input, we converted the non-numeric data into numeric using the data encoding technique for all three datasets. Though level encoding and OneHot encoding are two well known data transformation techniques, we haven't used them as they increase the data dimensionality and do not provide feature names. Instead, we performed this conversion using our algorithm. To normalize the data, we use the MinMax scaling technique that fits the wide range of values within a scale ranging from 0 to 1.

We separated benign and attack data instances from the training datasets provided by the repositories mentioned above. We used supervised methods that perform better with a balanced dataset. To make a training dataset balanced, we first chose the data type (benign or attack) with the minimum count of data instances within that training dataset. Using this minimum count, we have taken almost the same amount of normal (benign) and anomalous (attack) data instances to prepare a balanced training set for NSL-KDD and UNSW-NB15 datasets. The original amount of testing data instances were considered for preparing the testing set for these datasets. However, the CICIDS2017 dataset doesn't have separate training and testing sets. To prepare a balanced dataset, we followed the same procedure as with the two datasets. Then it was split in a 75 to 25 ratio where 75% are training data, and 25% are testing data. The numbers of benign and attack data in the three datasets are shown in TABLE II.

C. Ensemble Feature Selection (EnFS)

Ensemble feature selection is a process of identifying the best feature subset based on the majority voting technique. The

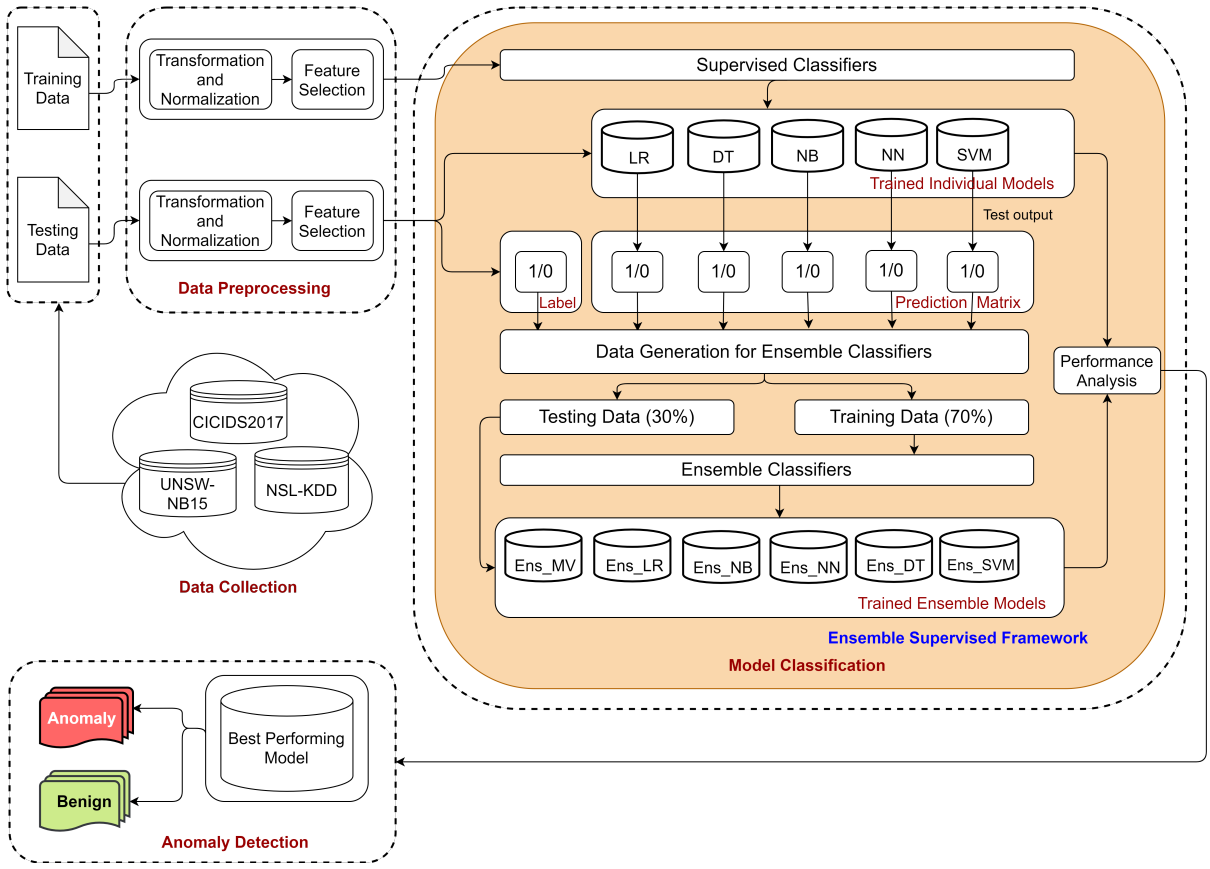


Fig. 1: Process flow of our proposed method

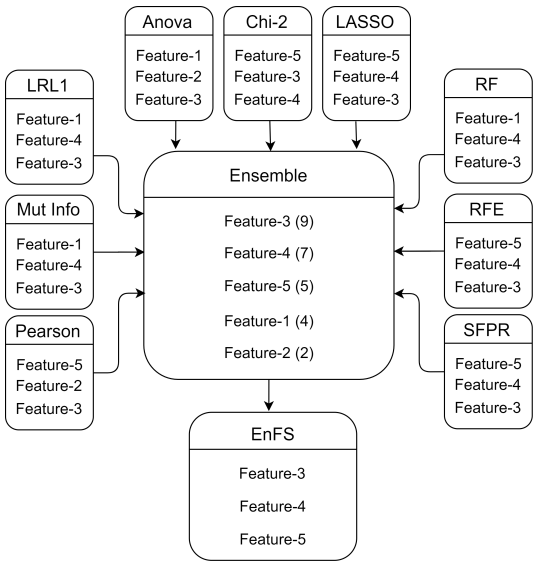


Fig. 2: Ensemble feature selection approach (EnFS) using majority voting.

process is illustrated in Fig. 2, where we use nine individual FS methods. Afterwards, we rank the features according to majority voting where more than half of the FS methods select the features. The major contributions of this research are to use several feature selection algorithms, perform a comparative analysis among them, and ensemble them using majority voting. Based on the performances, we have chosen the top nine FS methods from a list of fifteen well-known methods. They are Anova, Chi-squared, LASSO, Logistic Regression with L1 Penalty, Random Forest, Recursive Feature Elimination, Pearson Correlation, Mutual Information, and SFPR. First, we used these feature selection methods to find the optimal feature sets, and the ensemble supervised framework trained five individual and six ensemble models for performance analysis using these feature sets. We tuned the feature selection methods using several feature counts, like 5, 10, 15, 20, 25, 30, and full set. In most of the cases for all three datasets, feature count 20 generated the optimal feature sets that produced the best results among them. Then we combined all nine feature sets using a majority voting technique and generated a minimal number of features. All features i.e.; nine feature sets from nine FS methods, feature set from EnFS approach, and a full set of features are used to train the models listed in ensemble supervised framework (SupEnML) for analysis purposes.

D. Model Classification

In our proposed ensemble supervised ML framework, we perform two-stage classification using individual and ensemble classifiers sequentially to discern anomalies from the datasets. We train five supervised individual models, such as Logistic regression (LR), Decision tree (DT), Naïve Bayes (NB), Neural network (NN), and Support vector machine (SVM) using the training data. These models are tested using the testing

TABLE II: Data instances for NSL-KDD, UNSW-NB15 and CICIDS2017 datasets used in experimentation

Dataset	Training			Testing			Verification		
	Total	Benign	Attack	Total	Benign	Attack	Total	Benign	Attack
NSL-KDD	125974	67343	58630	22544	12833	9711	1000	200	800
UNSW-NB15	112000	56000	56000	70000	35000	35000	1000	200	800
CICIDS2017	75000	38000	38000	25000	13000	12000	1000	200	800

TABLE III: Hyper-parameter values used for different individual and ensemble classifiers.

	Classifier	**Short Names**	**Hyper-parameter Values**
Supervised Models	Logistic Regression	LR	random_state=0, solver='lbfgs', multi_class='multinomial'
	Decision Tree	DT	default parameters
	Naïve Bayes	NB	alpha=1.0, binarize=0.0, fit_prior=True, class_prior=None
	Neural Network	NN	solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(5, 2), random_state=1
	Support Vector Machine	SVM	C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=True
Ensemble Models	Majority Voting	Ens_MV	none
	Decision Tree	Ens_DT	default parameters
	Naïve Bayes	Ens-NB	alpha=1.0, binarize=0.0, fit_prior=True, class_prior=None
	Logistic Regresion	Ens_LR	random_state=0, solver='lbfgs', multi_class='multinomial'
	Neural Network	Ens_NN	solver='lbfgs', alpha=1e-5, novelty=True hidden_layer_sizes=(5, 2), random_state=1
	Support Vector Machine	Ens_SVM	C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=True

data, and the corresponding test outputs (e.g; 1 for anomalous, 0 for benign) generate a prediction matrix. Appended by the label from testing data, this prediction matrix forms the data for ensemble classifiers. This dataset is divided into training and testing data with a ratio of 70 to 30. Then we train and test six ensemble classifiers, namely Ensemble majority voting (Ens_MV), Ensemble logistic regression (Ens_LR), Ensemble naive bayes (Ens_NB), Ensemble neural network (Ens_NN), Ensemble decision tree (Ens_DT), and Ensemble support vector machine (Ens_SVM). We tuned the hyper-parameters of these classifiers using a grid search algorithm to obtain the best valued hyper-parameters that are listed in our paper earlier [25]. We have performed 10-fold cross-validation during the model training time over a randomly divided dataset to avoid the model overfitting. After a comparative performance analysis of these eleven models using the evaluation metrics, the best performing model is obtained for anomaly detection. The mechanism of combining individual classifiers using the ensemble technique is illustrated in Fig. 1.

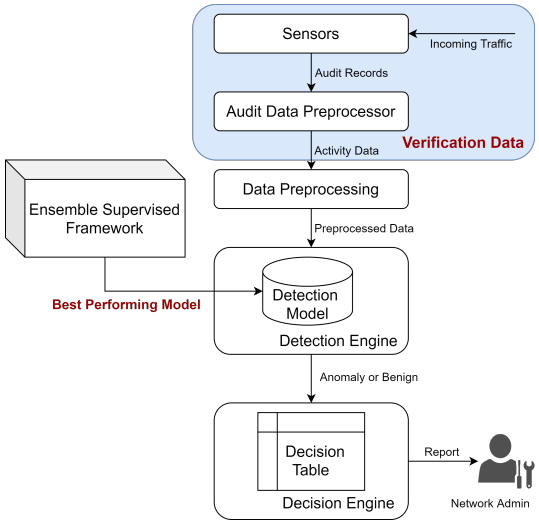


Fig. 3: Ensemble machine learning framework as the detection technique of an IDS

E. Anomaly Detection

In the real world, an IDS as shown in Fig. 3 can be placed at the gateway in a secured network. The IDS consists of sensors and an audit data preprocessor, to convert the incoming traffic into activity data. We use 'Verification data' to mimic the real time data stream. The best performing model obtained from the ensemble supervised framework is used as the detection model of the IDS. The detection model within the detection engine analyzes the verification data and identifies the data as anomaly or benign. Afterwards, based on the rules from the decision table, the decision engine takes necessary actions and reports the network admin about the potential threat.

F. Proposed Algorithm

Algorithm 1: Ensemble Feature Selection and Classification

Input : $train\_X$: train features, $train\_y$: train target, $test\_X$: test features, $test\_y$: test target
Output: m : a best performing model

1 **init**
2 fs = ['ANOVA', 'CHI2', 'LASSO', 'LRL1', '
3 MUTINFO', 'PEARSON', 'RF', 'RFE', 'SFPR']
4 mli = ['LR', 'DT', 'NB', 'NN', 'SVM']
5 mle = ['LR', 'DT', 'NB', 'NN', 'SVM']
6 $result\_df$ = ['LR_Target', 'DT_Target', 'NB_Target',
7 'NN_Target', 'SVM_Target', 'Actual_Target']
8 $best\_feature$ = $[5, 10, 15, 20, 25, 30]$
9 **foreach** $k \in best\_feature$ **do**
10 $feature\_list$ = [[]]
11 **foreach** $f \in fs$ **do**
12 select k best feature using f on $train\_X$ and $train\_y$
13 append k best feature in $feature\_list$
14 **end**
15 $feature\_vote\_map$ = map <feature, vote >
16 **foreach** $fl \in feature\_list$ **do**
17 update value vote in $feature\_vote\_map$ for every key as feature in fl
18 **end**
19 $majority\_vote$ = len(fs)/2
20 $en\_k\_feature$ = a list of feature from $feature\_vote\_map$ with vote $\geqslant$ majority_vote
21 **foreach** $m \in mli$ **do**
22 train m on $train\_X$[en_k_feature] and train_y
23 evaluate performance of m on $text\_X$[en_k_feature] and test_y
24 store performance metric
25 $predicted\_target$ = a list of predicted class on $test\_X$ using m
26 append $predicted\_target$ in $result\_df$
27 **end**
28 append $test\_y$ in $result\_df$
29 $new\_train\_X, new\_train\_y, new\_test\_X, new\_test\_y$ = split $result\_df$ by 70%/30%
30 **foreach** $m \in mle$ **do**
31 train m on new_train_X and new_train_y
32 evaluate performance of m on new_text_X and new_test_y
33 store performance metric
34 **end**
35 analyze the stored performance to determine best performing model m
36 **end**

We implement an ensemble feature selection and ensemble classification algorithm to improve the overall performance of our proposed machine learning model. As defined in Algorithm 1, it takes input as features and targets for training and testing purposes of our classification models. We initialize a list of feature selection methods, machine learning models for individual classification, machine learning models for ensemble classification, and the number of best features as shown between lines 1 to 8. Also, the data structure for storing the individual model prediction result has been initialized here. The ensemble feature selection and classification approach execute for all considered best feature sets controlled by the outer loop in line number 9. The k-best features extracted by individual feature selection algorithms have been stored between lines 11 to 14 for ensembling them later. We use the training features and target to extract the best feature set from the dataset. The ensemble feature set has been calculated based on the majority voting technique, i.e.,a feature is essential when it is selected by at least half of the feature selection algorithm. The code block starting from line number 15 to 20 executes to identify the ensemble feature set, and it is the end of the ensemble feature selection part of our algorithm. The ensemble classification part of the algorithm has started from line number 21, where the code block from line number 21 to 27 trains our machine learning models and stores the classification result in the data structure initialized at the beginning of the algorithm. The accumulated result has been used as our new dataset where features are the individual classification results, and the target is the ground truth from the dataset. The new dataset has been trained and classified again by all machine learning models considered for our experiment, and it shows in between line numbers 30 to 34. Finally, we analyze the result at line number 35 to evaluate the performance of the individual classifier to identify the best-performing model.

G. Complexity Analysis of Our Algorithm

Before analyzing the complexity of our algorithm, we need to identify few terms that is, n_{fm} as the number of feature selection methods, n_{mi} as the number of ML models for individual classification, n_{me} as the number of ML models for ensemble classification, n_{fs} as the count of number of best features, n_{fl} as the number of features in feature list, $O(fs)$ as the highest time complexity among all FS methods, $O(mi)$ as the highest time complexity among all individual ML models, and $O(me)$ as the highest time complexity among all ensemble ML models

The complexity of our algorithm can be derived as time and space wise. Therefore the time complexity of our algorithm is as

$$T(n) = n_{fs}\left[n_{fm}\left\{O(fs) + O(n_{fl}) + O(n_{fl}) + n_{mi}O(mi) + n_{me}O(me)\right.\right. \quad (1)$$

As we have used constant number of ML models and FS methods, the simplified version of the time complexity in eq 2 is equal to the time complexity of when we use a single classifier and single FS method.

$$T(n) \approx O(fs) + MAX(O(mi), O(me)) \quad (2)$$

In our algorithm, we have used 5 input variables, 5 static variables, 6 loop iterators, 1 map, and 8 other variables. Most cases, we have used 'list' type variable and one hashmap type.

Since we performed our experiments in 64 bit machine with python script. A 64 bit machine allocates 64 bytes for an empty list and 240 bytes for a hashmap. Thus the minimum space needed for a 64 bit machine is

$$S(n) = 10 * 64 + n_{fs}\left[64 + 64 + n_{fm}(64 + 64) + 240 + n_{f1}\left\{64 + n_{f2}(64 + 64)\right\} + 64 + n_{mi}(64 + 64 + 64) + n_{me}(64 + 64 + 64)\right. \quad (3)$$

V. Experimental Setup and Results

A. Software and Hardware Preliminaries

We have used Python and the machine learning library scikit-learn [35] to conduct the experiments in computers with the configuration of Intel®CORE™i5-6600K CPU@3.50GHz, 8GB memory & 64-bit Ubuntu operating system, and Intel (R) i5-5250U CPU@1.6GHz, 4GB memory & MacOS operating system.

B. Datasets

In this research, three intrusion detection benchmark datasets are used to perform all of the experiments. Details of these datasets are as follows:

NSL-KDD: NSL-KDD [3] dataset is the improved version of the KDD'99 dataset proposed by the Canadian Institute of Cybersecurity and is widely used for detecting network intrusion. It comprises 41 features and a class label to describe 39 common cyberattacks.

UNSW-NB15: The Cyber Range Lab of the Australian Centre for Cyber Security created the UNSW-NB15 dataset [4] using the IXIA PerfectStorm tool. They captured 100 GB of raw traffic that consists of nine types of cyberattacks. The dataset has a total of 49 features with a class label.

CICIDS2017: The Canadian Institute for Cybersecurity generated the CICIDS2017 [5] dataset that represents recent day cyber-attacks by analyzing network traffic using CICFlowMeter. This benchmark dataset exclusively covers 11 necessary criteria that make it reliable to cybersecurity data analysts.

C. Evaluation Metrics

In this research, well established evaluation metrics are required to find the best performing model which can be incorporated within an IDS. Sensitivity, specificity, accuracy, precision, recall, and f-measure are the well-known performance evaluation metrics. These are derived from four basic terms, such as true positive, false positive, true negative, and false negative rates. The evaluation metrics are defined as follows:

1) *Accuracy* $[(TP + TN)/Total]$: the percentage of true DDoS detection over total data instances.
2) *Precision* $[TP/(FP + TP)]$: is the measurement of how often the model correctly identifies a DDoS attack.
3) *False Positive Rate (FPR)* $[FP/(FP + TN)]$: indicates how often the model raise a false alarm by identifying a benign as a DDoS attack.
4) *Recall* $[TP/(FN+TP)]$: is the measurement of how many of the DDoS attacks the model does identify correctly. Recall is also known as true-positive rate, sensitivity, or DDoS detection rate,
5) *F1-Score* $[2TP/(2TN + FP + FN)]$: is the harmonic average of precision and recall.

Additionally, ROC curve is used to portrait the relationship between the True Positive and False Positive rates.

D. Discussion of Results

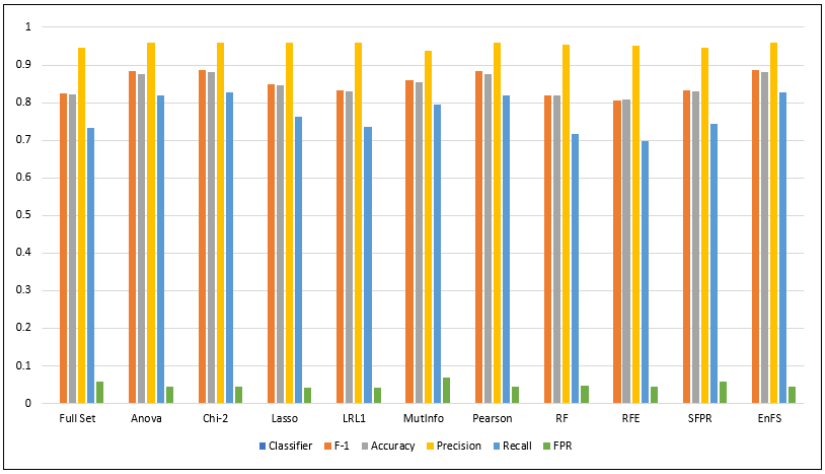
In this section, we describe the results obtained from our experiments. As mentioned earlier, we have performed similar experiments with three datasets, NSL-KDD, UNSW-NB15 and CICIDS2017, in two phases: feature selection and data modeling. The best performing results are shown in this section, where the detailed results are listed in this URL, https://github.com/CSMLGroup/NIDS-SupEnML-EnFS/.

TABLE I, II and III listed in the above mentioned URL show the features versus feature selection methods for NSL-KDD, UNSW-NB15 and CICIDS2017, respectively, where 1 represents a feature (in a row) is selected by a corresponding FS method (in a column) and 0 represents not selected. The last column counts the total selection by the FS methods for a single feature. The features are listed in a descending order of total count value. As we have used nine FS methods, majority wins when the count is greater than 4.5 (i.e; 5). The count values for the features below 5 are discarded from the feature list and are not shown on these tables. This majority voting technique is used to select a total of 20 out of 43, 19 out of 42, and 22 out of 80 optimized features from NSL-KDD, UNSW-NB15, and CICIDS2017 datasets, respectively. Our proposed EnFS technique significantly reduces the feature set for classification model by 53.5%, 54.8%, and 72.5 % respectively for NSL-KDD, UNSW-NB15, and CICIDS2017 dataset.

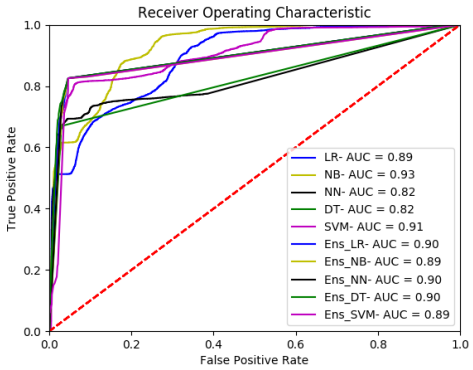
From each of these three datasets, eleven feature sets are extracted: nine of them are from nine individual FS methods, a feature set from ensemble feature selection framework, and the full feature set (i.e., no feature selection was applied). Using these eleven feature sets, we trained and tested our classifiers which are listed in ensemble supervised framework. For each feature set, the ensemble supervised framework is used to train five individual models and six ensemble models. Therefore, a total of 121 models are generated for eleven feature sets. After analyzing the performances of the eleven trained models for each feature set in terms of F-1 score, FPR and training time (Elp_time as elapsed time in tables), the best performing models are listed in TABLE IV, V and VI for NSL-KDD, UNSW-NB15 and CICIDS2017 dataset, respectively. Our experimental results shows an out-performance of proposed ensemble classification model in comparison with the individual classifier. In case of NSL-KDD and CICIDS2017 dataset, our proposed SupEnFS framework shows 100% better classification accuracy for all individual and ensemble feature set while 7 out of 11 (63.3%) best performing classifier were our proposed ensemble machine learning model for UNSW-NB15 dataset. Overall, our proposed ensemble classification model outperform than the individual classification model for all datasets in the experiments.

TABLE IV: Best performing results using nine feature sets, full feature sets and ensemble feature sets for NSL-KDD dataset

Feature Sets	Classifier	F-1	Accuracy	Precision	Recall	FPR	ROC_auc	Elp_time
Full Set	Ens_DT	0.825	0.823	0.945	0.732	0.057	0.889	0.209
Anova	Ens_LR	0.884	0.877	0.96	0.819	0.045	0.895	0.296
Chi-2	Ens_LR	0.887	0.881	0.959	0.826	0.046	0.898	0.278
Lasso	Ens_DT	0.85	0.846	0.959	0.763	0.043	0.871	0.238
LRL1	Ens_DT	0.832	0.831	0.959	0.735	0.041	0.857	0.263
MutInfo	Ens_NB	0.86	0.853	0.938	0.795	0.07	0.841	0.211
Pearson	Ens_LR	0.884	0.877	0.96	0.818	0.045	0.895	0.3
RF	Ens_NN	0.818	0.819	0.953	0.717	0.047	0.843	1.455
RFE	Ens_DT	0.805	0.807	0.952	0.697	0.046	0.835	0.232
SFPR	Ens_DT	0.832	0.829	0.945	0.743	0.057	0.858	0.509
EnFS	Ens_NB	0.887	0.881	0.959	0.826	0.046	0.892	0.234



(a) Performance Analysis



(b) ROC Curves

Fig. 4: a) Comparative performance analysis using nine feature sets, full feature sets and ensemble feature sets for NSL-KDD dataset, b) ROC curves for ensemble feature selection using NSL-KDD datasets

From TABLE IV, a better performance measure in terms of F-1 (0.825), Accuracy (0.881), Precision (0.959), Recall (0.826), False Positive Rate (0.046), etc. is achieved using our EnFS method compared to any individual FS method except Chi-square. However, our EnFS method requires comparatively smaller training time (0.234 Sec.) than that for Chi-square (0.278 Sec.). The graphical representation of this table is illustrated in Fig. 4 (a). From this bar chart, it is clear that the bunch of bars at the rightmost side which is for EnFS is found better compared to any other bunches. Fig. 4 (b) represents the ROC curves for all 11 models using EnFS feature set for NSL-KDD dataset.

TABLE V shows a better performance measure in terms of Accuracy (0.881), Precision (0.808), False Positive Rate (0.22), etc. except the similar F-1 score (0.867) with the Chi-squared and Recall (0.935) deviation of 0.064 from the best score (0.999 for SFPR and RFE) using our EnFS method compared to any individual FS method. The table is graphically represented in Fig. 5 (a) in order to visualize the performances of all FS methods. Fig. 5 (b) represents the ROC curves for all 11 models using EnFS feature set for UNSW-NB15 dataset, where Ens_NN model shows the best auc value.

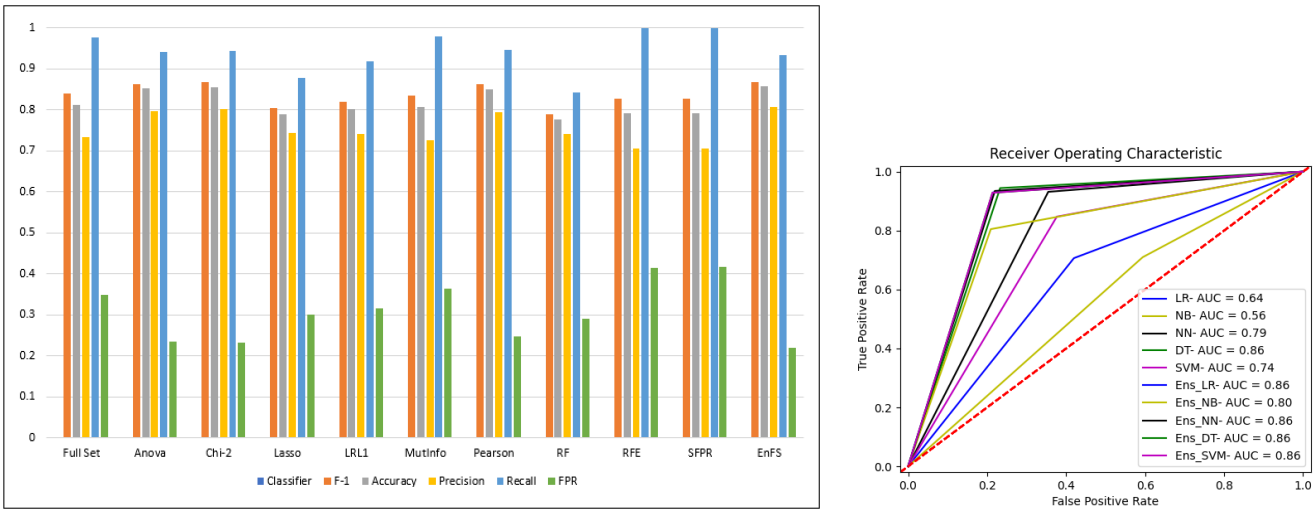
In TABLE VI, we see the better performance measure in terms of F-1 (0.995), Accuracy (0.995), Precision (0.995), Recall (0.996), False Positive Rate (0.006), etc. using our EnFS method compared to any individual FS method except the LASSO method, which has lesser FPR (0.005) than that for the EnFS method (0.006). Fig. 6 (a) shows the visual representation of the table. In Fig. 6 (b), we see the ROC curves for all 11 models using EnFS feature set for CICIDS2017 dataset.

From TABLE IV, V, and VI, it is clear that Chi-squared and LASSO methods generate equal or better performances for some metrics like, FPR in CICIDS and Recall in UNSW-NB15 dataset but not for all performance metrics when compared with our EnFS method. On the other hand, our EnFS method always yields better performances for most of the performance metrics and especially in case of classification accuracy and F1-score which is very crucial for any machine learning based IDS. The details of 121 models' performance measures and rest of the ROC curves for the other ten feature sets are listed in this URL, https://github.com/CSMLGroup/NIDS-SupEnML-EnFS/.

Comparing TABLE IV, V and VI, and Fig. 4, 5, and 6, we can summarize that the feature set obtained from the EnFS framework provides the best performances in terms of F-1 score and Accuracy for all experimental setups. Also the FPR (False Positive Rate) was better for most of the dataset (2 out of 3) where for CICIDS2017 dataset the FPR was a

TABLE V: Best performing results using nine feature sets, full feature sets and ensemble feature sets for UNSW-NB15 dataset

Feature Set	Classifier	F-1	Accuracy	Precision	Recall	FPR	ROC_auc	Elp_time
Full Set	Ens_DT	0.839	0.813	0.735	0.977	0.348	0.815	0.241
Anova	Ens_DT	0.864	0.853	0.798	0.942	0.235	0.853	0.242
Chi-2	DT	0.867	0.856	0.802	0.944	0.233	0.856	1.196
Lasso	Ens_SVM	0.805	0.789	0.743	0.879	0.3	0.79	301.68
LRL1	Ens_DT	0.821	0.801	0.742	0.919	0.316	0.802	0.245
MutInfo	Ens_DT	0.835	0.808	0.727	0.98	0.363	0.809	0.242
Pearson	DT	0.863	0.85	0.794	0.946	0.246	0.85	1.193
RF	Ens_SVM	0.789	0.776	0.741	0.844	0.291	0.776	315.367
RFE	NN	0.828	0.792	0.706	0.999	0.415	0.792	14.363
SFPR	SVM	0.828	0.792	0.706	0.999	0.416	0.792	6681.006
EnFS	Ens_NN	0.867	0.857	0.808	0.935	0.22	0.858	6.019



(a) Performance Analysis (b) ROC Curves

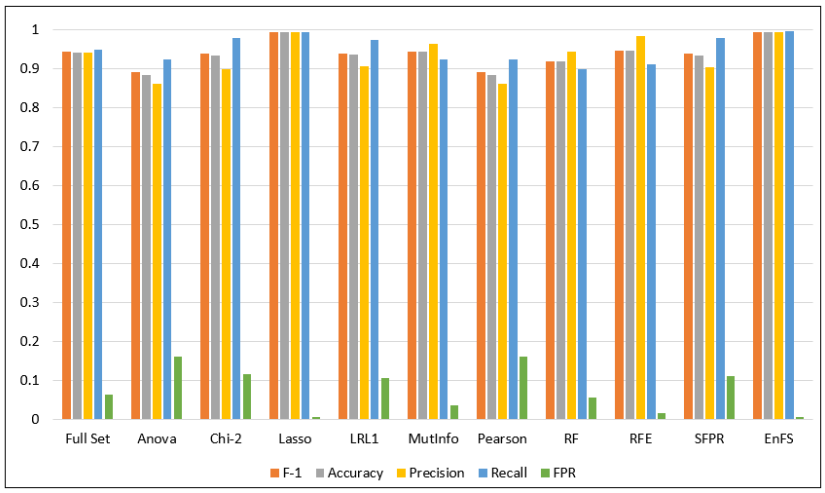
Fig. 5: a) Comparative performance analysis using nine feature sets, full feature sets and ensemble feature sets for UNSW-NB15 dataset, b) ROC curves for ensemble feature selection using UNSW-NB15 datasets

TABLE VI: Best performing results using nine feature sets, full feature sets and ensemble feature sets for CICIDS2017 dataset

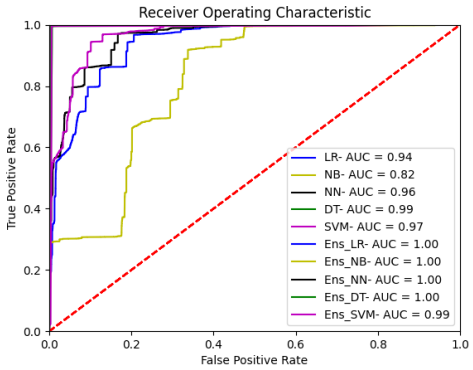
Feature Set	Classifier	F-1	Accuracy	Precision	Recall	FPR	ROC_auc	Elp_time
Full Set	Ens_DT	0.945	0.942	0.941	0.948	0.063	0.978	0.105
Anova	Ens_DT	0.892	0.884	0.861	0.925	0.16	0.955	0.102
Chi-2	Ens_DT	0.938	0.933	0.9	0.978	0.116	0.961	0.103
Lasso	Ens_DT	0.995	0.995	0.995	0.995	0.005	0.998	0.1
LRL1	Ens_DT	0.94	0.936	0.907	0.975	0.106	0.967	0.103
MutInfo	Ens_DT	0.945	0.944	0.965	0.925	0.036	0.97	0.101
Pearson	Ens_DT	0.892	0.884	0.861	0.925	0.16	0.955	0.101
RF	Ens_DT	0.92	0.92	0.944	0.898	0.057	0.969	0.102
RFE	Ens_DT	0.946	0.946	0.985	0.911	0.015	0.982	0.101
SFPR	Ens_SVM	0.94	0.935	0.904	0.978	0.111	0.953	4.037
EnFS	Ens_NN	0.995	0.995	0.995	0.996	0.006	0.998	0.286

slightly lower (0.001) than the best FPR (0.006). From these tables, it is also clear that our proposed ensemble classification models (i.e., the Classifier in the tables that starts with Ens_) outperform single models for more than 85% (29 classification model out of total 33 model) of our experimental models. So, it is very clear from extensive experimental result that our proposed ensemble feature selection (EnFS) based SupEnML, an anomaly classification framework, generally perform better than the individual feature selection and classification approach. As per our knowledge, we have come across a very few existing related works on either ensemble classification or feature selection research, and results are not produced for similar datasets that we have used. Therefore, we were unable to provide a detailed comparison between our methods and results with the existing related works. However, we have added a comparison of the results from any of the three datasets irrespective of existing works' ensemble techniques which is summarized in TABLE VII.

Furthermore, we verified the best performing models for each datasets obtained from our proposed methods using the verification data consisting of 1000 instances. Among these instances, 800 of them are attacks and 200 are benign.



(a) Performance Analysis



(b) ROC Curves

Fig. 6: a) Comparative performance analysis using nine feature sets, full feature sets and ensemble feature sets for CICIDS2017 dataset, b) ROC curves for ensemble feature selection using CICIDS2017 datasets

TABLE VII: A performance comparison of our method compared to existing methods

Methodolgy	Ensemble feature Selection	Ensemble machine learning	Verification Dataset	Best performance (Accuracy)
Ensemble, [6]	Not implemented	Adaptive voting	NSL-KDD	85.2%
Ensemble, [7]	Not implemented	Bagging (Base classifier - J48)	NSL-KDD	84.25%
Ensemble, [8]	Not implemented	DAREnsemble	NSL-KDD	78.8%
SVM, [9]	Not Implemented	Not Implemented	NSL-KDD	82.68%
Deep Learning, [14], [15], [16]	Not Implemented	Not Implemented	NSL-KDD, KDDCup 99, UNSW-NB15, WSN-DS, CICIDS 2017	NSL-KDD (DNN): 78.9%, UNSWNB-15 (DNN): 78.4%, CICIDS 2017 (DNN): 96.3%, NSL-KDD (ML): 93.4%, UNSWNB-15 (ML): 90.3%, CICIDS 2017 (ML): 94.1%
Our work	EnFS	SupEnML	NSL-KDD, UNSWNB-15 CICIDS	NSL-KDD: 88.1%, UNSWNB-15: 85.7%, CICIDS 2017: 99.5%

TABLE VIII: Verified the best performing models' performances using verification set for three datasets

Dataset	Best Model	Instances		Predicted Results				F-1 Score	Accuracy	Precision	Recall	FPR
		Attack	Benign	Attack		Benign						
				TP	FP	TN	FN					
NSL-KDD	Ens_NB	800	200	640	9	191	160	0.883	0.831	0.986	0.8	0.04
UNSW-NB15	Ens_NN	800	200	618	21	179	182	0.859	0.797	0.967	0.773	0.105
CICIDS2017	Ens_NN	800	200	790	1	199	10	0.993	0.989	0.999	0.988	0.005

Table VIII shows the performances of three best models using the verification data. From the table, we observe that the performance measures are almost the same with the results of EnFS from TABLE IV, V, and VI.

VI. Conclusion

Typically, the network anomalies can be detected using an intrusion detection system (IDS). In this research, we have combined ensemble feature selection and ensemble machine learning approaches as the detection mechanism within an IDS to detect the network anomalies. For the ensemble feature selection framework, we first have experimented with the feature sets obtained from nine feature selection methods and then have combined these feature sets to get the minimal number of features using majority voting. Our experiments demonstrate that the feature set obtained from the EnFS approach has better outcomes compared to any individual FS method. In addition, we have used this feature set in our ensemble supervised ML framework to find the best performing model which can be incorporated into any IDS. A total of eleven feature sets (nine from nine different FS methods, a feature set from EnFS, and a full set of features) have been used for the experimentation. We have also performed a comparative analysis among these feature sets where the EnFS set outperforms individual feature sets in most cases. Moreover, for each feature set, we have used our ensemble supervised ML framework to train eleven models (five individual and six ensemble models). Almost 80% of ensemble models outperform single models. For this extensive experimentation, we have used NSL-KDD, UNSW-NB15, and CICSIDS2017 datasets.

Using a full set of data instances for all three datasets can exact a toll in terms of longer experimentation runtime. Therefore, sometimes we have used a reduced amount of data for training purposes to boost the simulation time, and thus compromising the probability of a near-perfect performance. In future, we plan to incorporate the full amount of data as well as new datasets (including other domains) to verify the efficiency of our method. Furthermore, we plan to consider unsupervised learning with our ensemble classification which will solidify our method. In addition, adversarial machine learning (AML) can be added as an extension of our research in the future to remove the malicious adversaries by enabling safe adoption of the ML techniques in adversarial settings, and thus to retain the whole system security [36].

REFERENCES

[1] T. G. Dietterich, "Ensemble methods in machine learning," in *International workshop on multiple classifier systems*, pp. 1–15, Springer, 2000.

[2] S. Das, A. M. Mahfouz, D. Venugopal, and S. Shiva, "Ddos intrusion detection through machine learning ensemble," in *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pp. 471–477, IEEE, 2019.

[3] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*, pp. 1–6, IEEE, 2009.

[4] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 military communications and information systems conference (MilCIS)*, pp. 1–6, IEEE, 2015.

[5] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization.," in *ICISSP*, pp. 108–116, 2018.

[6] X. Gao, C. Shan, C. Hu, Z. Niu, and Z. Liu, "An adaptive ensemble machine learning model for intrusion detection," *IEEE Access*, vol. 7, pp. 82512–82521, 2019.

[7] N. T. Pham, E. Foo, S. Suriadi, H. Jeffrey, and H. F. M. Lahza, "Improving performance of intrusion detection system using ensemble methods and feature selection," in *Proceedings of the Australasian Computer Science Week Multiconference*, pp. 1–6, 2018.

[8] D. Gaikwad and R. Thool, "Darensemble: Decision tree and rule learner based ensemble for network intrusion detection system," in *Proceedings of First International Conference on Information and Communication Technology for Intelligent Systems: Volume 1*, pp. 185–193, Springer, 2016.

[9] M. S. Pervez and D. M. Farid, "Feature selection and intrusion classification in nsl-kdd cup 99 dataset employing svms," in *The 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014)*, pp. 1–6, IEEE, 2014.

[10] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, "Intrusion detection by machine learning: A review," *expert systems with applications*, vol. 36, no. 10, pp. 11994–12000, 2009.

[11] S. Mukkamala, A. H. Sung, and A. Abraham, "Intrusion detection using ensemble of soft computing paradigms," in *Intelligent systems design and applications*, pp. 239–248, Springer, 2003.

[12] S. Chebrolu, A. Abraham, and J. P. Thomas, "Feature deduction and ensemble design of intrusion detection systems," *Computers & security*, vol. 24, no. 4, pp. 295–307, 2005.

[13] F. Amiri, M. R. Yousefi, C. Lucas, A. Shakery, and N. Yazdani, "Mutual information-based feature selection for intrusion detection systems," *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1184–1199, 2011.

[14] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.

[15] R. Vinayakumar, K. Soman, and P. Poornachandran, "Evaluating effectiveness of shallow and deep networks to intrusion detection system," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1282–1289, IEEE, 2017.

[16] R. Vinayakumar, M. Alazab, S. Srinivasan, Q.-V. Pham, S. K. Padannayil, and K. Simran, "A visualized botnet detection system based deep learning for the internet of things networks of smart cities," *IEEE Transactions on Industry Applications*, vol. 56, no. 4, pp. 4436–4456, 2020.

[17] H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet, "A survey on ensemble learning for data stream classification," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–36, 2017.

[18] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018.

[19] J. Kittler, M. Hatef, R. P. Duin, and J. Matas, "On combining classifiers," *IEEE transactions on pattern analysis and machine intelligence*, vol. 20, no. 3, pp. 226–239, 1998.

[20] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014.

[21] R. Sheikhpour, M. A. Sarram, S. Gharaghani, and M. A. Z. Chahooki, "A survey on semi-supervised feature selection methods," *Pattern Recognition*, vol. 64, pp. 141–158, 2017.

[22] S.-W. Lin, K.-C. Ying, C.-Y. Lee, and Z.-J. Lee, "An intelligent algorithm with feature selection and decision rules applied to anomaly intrusion detection," *Applied Soft Computing*, vol. 12, no. 10, pp. 3285–3290, 2012.

[23] O. Osanaiye, H. Cai, K.-K. R. Choo, A. Dehghantanha, Z. Xu, and M. Dlodlo, "Ensemble-based multi-filter feature selection method for ddos detection in cloud computing," *EURASIP Journal on Wireless Communications and Networking*, vol. 2016, no. 1, p. 130, 2016.

[24] S. Das, D. Venugopal, S. Shiva, and F. T. Sheldon, "Empirical evaluation of the ensemble framework for feature selection in ddos attack," in *2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, pp. 56–61, IEEE, 2020.

[25] S. Das, D. Venugopal, and S. Shiva, "A holistic approach for detecting ddos attacks by using ensemble unsupervised machine learning," in *Future of Information and Communication Conference*, pp. 721–738, Springer, 2020.

[26] S. Khalid, T. Khalil, and S. Nasreen, "A survey of feature selection and feature extraction techniques in machine learning," in *2014 Science and Information Conference*, pp. 372–378, IEEE, 2014.

[27] L. C. Molina, L. Belanche, and À. Nebot, "Feature selection algorithms: A survey and experimental evaluation," in *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pp. 306–313, IEEE, 2002.

[28] S. Adams and P. A. Beling, "A survey of feature selection methods for gaussian mixture models and hidden markov models," *Artificial Intelligence Review*, vol. 52, no. 3, pp. 1739–1779, 2019.

[29] M. Howard and S. Lipner, *The security development lifecycle*, vol. 8. Microsoft Press Redmond, 2006.

[30] A. V. Uzunov and E. B. Fernandez, "An extensible pattern-based library and taxonomy of security threats for distributed systems," *Computer Standards & Interfaces*, vol. 36, no. 4, pp. 734–747, 2014.

[31] W. Xiong and R. Lagerström, "Threat modeling–a systematic literature review," *Computers & security*, vol. 84, pp. 53–69, 2019.

[32] N. Shevchenko, T. A. Chick, P. O'Riordan, T. P. Scanlon, and C. Woody, "Threat modeling: a summary of available methods," tech. rep., Carnegie Mellon University Software Engineering Institute Pittsburgh United . . . , 2018.

[33] Praerit Garg and Loren Kohnfelder, "The stride threat model." accessed on August 15, 2021.

[34] M. Deng, K. Wuyts, R. Scandariato, B. Preneel, and W. Joosen, "A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements," *Requirements Engineering*, vol. 16, no. 1, pp. 3–32, 2011.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[36] Wikipedia, "Adversarial machine learning." accessed on August 15, 2021.

Saikat Das is an Assistant Professor in the Dept. of Computer Science at Utah Valley University. He also served as an Assistant Professor at Midwestern State University in Computer Science Dept. Dr. Das received his Ph.D. and M.Sc. in the Computer Science, both from the University of Memphis. Before finishing Ph.D., he served as a Cyber Defense Instructor at Southwest TN Community College and in industry, he worked more than five years as a Senior Software Engineer in various tech companies including Samsung Electronics. Dr. Das's research interests include intrusion detection system with machine learning for cyber-attacks, explanation-based machine learning in cybersecurity, machine learning software lifecycle, stealth migration protocol in cloud computing, runtime verification in software systems, etc.

Sajal Saha is a faculty member in the Dept. of Computer Science at the Patuakhali Science and Technology University (PSTU), Bangladesh. Before joining PSTU, he was a Software Engineer at Samsung RD Institute, Bangladesh. He received the B.S. degree in Computer Science and Engineering from Patuakhali Science and Technology University, Bangladesh, in 2012, and the master's in Information Technology from Jahangirnagar University in 2014. He also completed another MSc degree in computer science from Brock University, Canada, in 2020. He is currently pursuing a Ph.D. degree with the WING Lab, Western University, Canada. His current research interests include intelligent network, cyber-security, and machine learning, focusing on network QoS and network reliability.

Annita Tahsin Priyoti is a faculty member in the Dept. of Computer Science at the Daffodil International University (DIU), Bangladesh. She received her B.Sc. degree in Computer Science and Engineering from the Military Institute of Science and Technology (MIST), Bangladesh, in 2017. She is currently pursuing an M.Sc. degree with the WING Lab, Western University, Canada. Her current research interests include smart city/home, smart grid systems, cyber-security, and machine learning, focusing on smart services and applications.

Etee Kawna Roy started her PhD in Electrical Engineering'21 at The University of Utah with research interests in fabrication and electrical measurement of microstructural properties in CdTe/Perovskite solar cells. She completed her Master's in Engineering from Arkansas State University where she worked as a graduate research assistant in dynamics modeling of electrostatic self-assembly of charged nanoparticles in inverted system. She also completed her Master's in Electrical Engineering from the University of Dhaka, Bangladesh and her thesis was on Artificial Neural Network and Fuzzy Inference System and their application in Acute Myeloid Leukemia Prediction. She also served as a faculty member at the University of Dhaka and Ranada Prasad Shaha University, Bangladesh.

Frederick T. Sheldon has 35+ years of experience from academia, industry and government in various roles working on a diverse set of computer science problems within the scope of software engineering, formal methods, information assurance and security in domains such as embedded real-time avionics/vehicular, energy delivery systems, supply chain, and cryptographic key management. He received the Sigma Xi research and UT-Battelle key contributor and significant event awards and is senior member of IEEE and ACM. He has degrees from the University of Minnesota and University of Texas at Arlington. Currently he is a professor of computer science at the University of Idaho.

Anwar Haque is an Assistant Professor in the Dept. of Computer Science at the University of Western Ontario, Canada. Before joining Western, he was an Associate Director at Bell Canada. He is a leading international expert on next-generation communication network resources and performance management, cyber security, and smart city applications. Dr. Haque has authored/co-authored over 70 peer-reviewed research publications in leading journals and conferences, authored many industry technical papers, and held a number of patent/licenses. He has been awarded several national/provincial-level research grants, including NSERC, MITACS, OCE, and SOSCIP. Dr. Haque's collaborative research grants are valued at more than $15 million. Dr. Haque is serving in the inaugural advisory committee for the newly established Bell-Western 5G research centre, and he established an industry consortium to promote and support smart systems and digital services research at Western. Dr. Haque is the director of the Western Information & Networking Group (WING) Lab at Western

Sajjan Shiva is a Professor of computer science and served as the Director and founding chairman of the Department from 2002 to 2015. He is the Director of Game Theory and Cyber Security laboratory. He has served on the computer science faculty at the University of Alabama in Huntsville and Alabama A&M University and has been a consultant to industry and Government. His current research spans cyber security, cloud security, secure software development and machine learning applications. He is a Life Fellow of IEEE and and has authored four books (10 editions) on computer architecture used in more than 120 universities around the world.