Anomaly-based Network Intrusion Detection using Ensemble Machine Learning Approach

Abhijit Das[1]
Research Scholar, Dept. of CSE
PES Institute of Technology & Management
Affiliated to VTU, Shivamogga, India

Pramod[2]
Associate Professor, Dept. of ISE
PES Institute of Technology & Management
Affiliated to VTU, Shivamogga, India

Sunitha B S[3]
Associate Professor, Dept. of CSE
PESITM, Affiliated to VTU
Shivamogga, India

***Abstract*—In this study, an Intrusion Detection System (IDS) is designed based on Machine Learning classifiers, and its performance is evaluated for the set of attacks entailed in the UNSW- NB15 dataset. UNSW- NB15 dataset contains 2,540,226 realistic network data instances and 49 features. Most research uses a representative sample of this dataset with present training and testing subsets, which includes 257,673 records in total. The dataset was submitted to visual data analysis to discover potential reasons or flaws which likely challenge Machine Learning classifiers. Pre-processing strategies are necessary before this data can be used for data-driven prototype development for IDS because of the class representation imbalance with pattern counts and feature overlap. The method used for pre-processing is implemented by min-max scaling in the normalization phase, followed by applying Elastic Net and Sequential Feature Selection (SFS) algorithms. This work employed ensemble methods using three base classifiers, namely Balanced Bagging, XGBoost, and RF-HDDT, augmented to address the imbalance issue. Parameters of Balanced Bagging and XGBoost are tuned for the imbalanced data, and the Hellinger distance metric supplements random Forest to address the limitations of the default distance metric. Two new algorithms are proposed to address the class overlap issue in the dataset and applied during training. These two algorithms are leveraged to help improve the performance on the testing dataset by affecting the final classification decision made by three base classifiers as part of the ensemble classifier, which employs a majority vote combiner. The performance evaluation of the proposed method for binary and multi-category classification was evaluated using standard metrics, including those generated from the confusion matrix, and compared to other studies using the same dataset. The proposed design outperforms those reported in the literature by a significant margin for binary and multi-category classification cases.**

***Keywords*—Machine learning; ensemble method; intrusion detection system; UNSW-NB15 datasets**

I. Introduction

Cybercrime has increased dramatically due to the rapid development of technology and the broad distributed use of internet networks around the world. If 2019 has taught us anything, it's that no firm, no matter how big or little, is immune to a cyberattack. Cyber-attacks have become more sophisticated, difficult to detect, more targeted than ever before. Security must be constantly upgraded as a result. It is vital to network security to have a network intrusion detection system (NIDS) in place, as it alerts the right authorities when an incursion is detected. It is undeniable that we are becoming increasingly dependent on Internet every passing day due to human creativity and innovation. At the same time, the world of cybercrime has been populated with the criminals looking for a perfect crime such as stealing confidential information, data, funds or causing harm to target computing infrastructure. They explore possible avenues through the cyberspace and formulate different strategies called cyberattacks, to gain unauthorized access to the computing systems. These strategies (aka attacks) may be highlighted as follows: Distributed Denial-of-Service (DDoS), Man-In-The-Middle (MITM), Password-Based Attacks like Brute Force, Dictionary, Shoulder Surfing, Phishing, Malware like Virus, Trojan, Worm, Rootkit, Ransomware, Spyware, Botnet, Key Logger, Adware, SQL Injection, Cross-Site Scripting (XSS), Eavesdropping, Social Engineering [1]. The attackers leverage the attacks to get access to system resources and either destroy them or collect valuable information.

To reduce cyber threats or recover from damages caused by cyberattacks, the organizations assess the cyber risks. Any uncertainties related to the data resources and computer devices that threatens the confidentiality, availability and integrity of the information or information systems are identified as cyber risks [2]. Confidentiality represents the system resources protected from unauthorized access. While integrity preserves any piece of information from unauthorized changes, availability guarantees that the authorized users can always gain access to the required data resources [3]. According to study the most commonly reported cyberattack is Ransomware, it is predicted that a Ransomware attack may affect a business every 11 minutes and the concomitant damage reach $20 billion. The total loss caused by cybercrimes are also projected to rise by $6 trillion by 2022 [4, 5].

With the help of Intrusion detection system, malicious network traffic and device activity standard security system might not be able to see can be found and blocked. IDS is extremely successful in detecting, identifying, and monitoring threats, to put it more exactly. It is very important for keeping computer systems safe from threats that could harm their availability, integrity, or secrecy [6]. Traditional methods use a predefined database of known attacks and signature patterns to evaluate network packets. The user couldn't use the system because of the possibility of a new zero-day that isn't represented by any of the signatures detected in the database [7]. When it comes to detecting zero-day attacks, existing IDS systems have been found to be ineffective [8]. Because of this, it may be concluded that no matter how precise an intrusion detection (ID) technology is, malevolent attempts can degrade IDS stability. The main contributions of this work include:

- The development of a novel ensemble-based classification model for detecting intrusions that uses data from the UNSW-NB15 dataset.
- A cross-comparison of several methods for selecting features has been done.
- The suggested IDS has been tested to see how well it performs for binary and multi-class classification

II. Related Work

An ML classifier's accuracy can be improved by picking features that can represent incursion patterns. Multiple classifiers can reduce false positives and produce more accurate classification results than a single classifier, according to the research [9].

Kumar et al. [10] developed the unified intrusion detection system (UIDS) by generating the new training and test subsets out of UNSW-NB15. They utilized the k-means clustering algorithm to increase the attack sensitivity as the k-means clustering algorithm was able to identify the similarities between different attack classes. In each cluster, the number of records in some type of attack classes was more than the rest. The authors randomly picked 65% of the records of the dominating class categories to form a training dataset. The remaining 35% of the instances were used to build the test set. They also used information gain algorithm for the feature selection phase. 13 features out 47 were selected due to the improvement of accuracy scores by C5, Chi-Squared Automatic Inference Detection (CHAID), Classification and Regression Tree (CART) is also known as Decision Tree (DT) and Quick Unbiased Efficient Statistical Tree (QUEST) algorithms. These algorithms were used to form the proposed UIDS model. Their study reported 77.87% and 79.12% for the average sensitivity and F-measure, respectively. It also offered 3.80% false alarm rate for Normal instances and 86.15% attack sensitivity.

The authors in [11] implemented MLP as an anomaly detection system for binary classification. They employed RFE along with the Random Forest classifier for the purpose of dimensionality reduction. This method selected the top four informative features. The MLP-based IDS scored 85% for sensitivity and 89% for accuracy on the test subset of UNSW-NB15: 15% of the attack traces and 2% of the normal records were misclassified.

Bayu et al. [12] applied Gradient Boosted Machine (GBM) on three datasets including UNSW-NB15. No feature selection technique was implemented. All 47 features were kept for training and testing phases. The results showed that GBM outperformed four other algorithms, namely RF, Deep Neural Network (DNN), SVM and CART, with the average accuracy value of 93.64% and missed alarm rate of 0.0206 where GBM performance was evaluated on NSL-KDD, UNSW-NB15 and GPRS datasets. While running GBM on UNSW-NB15 alone provided 95.08% accuracy and 2.97% false alarm rate using 10-fold cross-validation and the accuracy of 91.31% and false alarm rate of 8.60% using the Hold-Out method on the original UNSW-NB15 train and test subsets.

In [13], nominal features were converted to numerical and then the Min-Max normalization method was utilized to scale down the values to the range of 0 to 1. They used 5-fold cross-validation without resampling to generate the test and training subsets. They calculated the average of the sensitivity, false alarm rate and accuracy of 5 folds. The authors utilized SVM by taking advantage of hyper clique property of hypergraph to improve the performance of SVM. This optimization technique implemented the feature selection as well. The SVM algorithm was trained with the entire 47 features and then the results were compared with the case when SVM was trained with the optimal number of feature subsets. The optimal feature subset is not reported. However, the number of optimal features is in the range of 30 to 35. They concluded that the feature selection had significant influence on the proposed model which delivered 98.47% sensitivity and 2.18% false alarm rate. However, 94.11% accuracy and 2.18% false alarm rate suggests a relatively large value for the missed alarm rate, which is not reported.

ML and DL are used to develop various IDS systems. Due to the enormous dimensionality of the data, improving IDS efficiency and classification prediction requires feature selection from the complete dataset. Chaouki et al. [14] employed genetic algorithm and logistic regression methods to choose the optimal collection of attributes for NIDS. They used KDD99 and UNSW-NB15 datasets for analysis. The primary purpose of their work was to locate the subset of features with the highest classification accuracy and the smallest number of features. RF, C4.5 and NB Tree are used in the classification stage to evaluate the performance of the generated features. Their results show an accuracy of 99.81% for the KDD99 and 81.42% for the UNSW-NB15 dataset. The results conclude that the UNSW-NB15 dataset is more complicated than the KDD99 dataset. In order to enhance the classification accuracy for the UNSW-NB15 IDS datasets, we must thus attempt various methodologies.

Tian et al. [15] addressed the issues of overfitting and inadequate classification accuracy. They proposed a model based on DBN by using probabilistic mass function encoding and the Min-Max normalisation technique. The proposed method achieves 96.17% and 86.49% accuracy on the NSL-KDD and UNSW-NB15 public datasets. They did not explore the class overlap problem of the UNSW-NB15 dataset, and the selection of DBN values was challenging to acquire without experimentation. It was challenging to choose the best parameter impacting detection accuracy.

The efficient classification of network traffic has been hampered by repetitive and unnecessary data attributes and this problem can be solved by feature selection method to recognize the most important elements. M. S. Abirami et al. [16] proposed Least Square Support Vector Machine (LSSVM-IDS) feature selection methods for IDS. With a 95% accuracy rate, this LSSVM system has correctly predicted the output 95% of the time. Ensemble learning was applied to the UNSW-NB15 dataset using a stacking classifier approach. They used logistic regression as a meta-classifier to integrate RF, SVM, and NB algorithms, and they reached a 95% accuracy rate. When selecting features, the author should have utilised an embedded process that might have improved accuracy. There is no mention of the problem of overlapping classes following feature selection. The class overlap problem, embedded methodology, and feature selection can further improve the

proposed model's accuracy.

A dynamically scalable ML-based NIDS was proposed by Soulaiman et al. [17] to solve the imbalanced class problem using SMOTE. The author has not mentioned the class overlap problem in the UNSW-NB15 datasets, decreasing the attack prediction accuracy in real-time.

In [18], the authors proposed the ensemble extreme learning machine (ELM) along with one-vs-all method to generate multi-class classification model. The proposed algorithm is a combination of a single hidden layer feedforward neural network and a softmax layer to make a multi-class prediction out of an ensemble of single output which was 0 or 1. This algorithm scored 95.66% average accuracy. Also, the authors implemented ExtraTree classifier in order to reduce the dimensionality of feature space. Accordingly, 21 features were selected. In the final stage, weighted extreme learning machine (WELM) was implemented and the accuracy of each attack type increased but still need more improvment. The training was done on 80% of the original training set and the remaining 20% was used for validation to avoid overfitting. The entire UNSW-NB15 test subset was utilized for the test phase.

D. Papamartzivanos et al. [19] combined the decision tree and genetic algorithm to generate classification rules and called their model Dendron. Wrapper technique was used for feature selection, which resulted in 23 being selected as informative features. The authors reported the sensitivity of 97.39% for Normal records and the average false alarm rate of 2.61%. In this study, 10% of the instances for each of the 9 classes were considered for building the training set and the remaining 90% was kept for the test set. To address the imbalance problem of multi-class classification in UNSW-NB15, 50% of Worms attack class records were included in the training subset and remaining 50% was kept to test the model.

The integrated rule-based model in [20] is trained to detect five class types to avoid overlapping. These rules were generated from four tree-structured classification algorithms, C5, CHAID, CART and QUEST. Training and test sets were built by eliminating some instances from the original training and testing subsets using k-means clustering. These instances belong to Analysis, Backdoor, Fuzzers, Shellcode and Worms attack types. These attack types suffer from overlapping problem and their presence may cause poor results. For the feature selection phase, the genetic algorithm was used and 22 features were picked accordingly. They reported good accuracy and sensitivity values for the Normal records. However, the average accuracy and sensitivity are 93.94% and 65.21%, respectively.

III. Proposed Methodology

A. Data Preprocessing

Data preprocessing transforms raw data into an appropriate framework or format before processing using a learning algorithm. These techniques significantly impact Machine Learning (ML) and Deep Learning (DL). This section explains these strategies and how they help to maximize the proposed ensemble model's performance.

Data Cleaning: It removes duplicate or irrelevant entries or features from the dataset. The UNSW-NB15 has no duplicate characteristics or records. However, all 49 functionalities are not required to be used. IP address and port number all work together to identify a computer's infrastructure. Other characteristics, such as record start time and record last time, may not have a great deal of use either. Because of overfitting, the Machine Learning (ML) model may not generalize correctly if these attributes are retained. The attack_cat feature was used for multiclass classification, and the label was used for binary class classification.

Data Transformation: Nominal features make up three of the 41 features that aren't targeted. Nominal features have transformed into integers to make ML models and scalers easier. The label encoder is used to convert the nominal features to numerical features. All the features have transformed to the same range of values using scalers, which helps most learning algorithms weigh in the entire features equally. The nearest shrunken centroid has been measured for each attack class. The Euclidean and Mahalanobis distance was used to measure the distance between the attack class centroids. The transformation algorithm which maximizes the intra-cluster distances has been identified and used to address the overlapping problem. According to the finding, the min-max scaler increases the distance of the centroids of four attack classes (Backdoor, DoS, Generic, and Reconnaissance) away from the Normal class centroid more than other methods like Normalization, Robust Scaler, Standardization, Quantile and Power transformation. Consequently, min-max scalar has been employed in this study.

Feature Selection: Fourteen different feature selection algorithms were implemented on the dataset to extract the informative as well as representative features. Chi-squared, Information gain (tree-based feature selection), CFS, ReliefF, and mRMR, are employed among filter-based feature selectors; genetic algorithm, Recursive Feature Elimination (RFE), and Sequential Feature Selection (forward selection and backward elimination) are picked as wrapper methods; and Lasso and Elastic Net are chosen among embedded feature selectors. The effect of feature selection algorithms on the classifier performance was assessed and evaluated using ensemble algorithms including Random Forest, Bagging, Balanced Bagging (BB), AdaBoost, XGBoost, Gradient Tree, Extremely Randomized Trees (ERT), Easy Ensemble (EE) and many other algorithms such as naïve Bayes, SVM and MLP Neural Network using Back Propagation (BP) optimization algorithm. Performance evaluation of classifiers on the dataset, which was preprocessed with the set of feature selection algorithms, indicated that the two best feature selection algorithms are Elastic Net and Sequential Forward Selection (SFS) running in conjunction with Random Forest, Bagging and XGBoost classifiers. The Elastic Net feature selection algorithm combined with the Balanced Bagging machine learning classifier and the SFS feature selector with the Random Forest classifier and XGboost performed the best among all possible combinations tested when considering the F1-score as the metric. In conjunction with the Balanced Bagging classifier, the Elastic Net feature selector performed the best for 24 features from the datasets. In conjunction with the Random Forest classifier, the SFS feature selection algorithm performed the best for 8 features that form a proper subset of 24 features.

TABLE I. MISSED ALARM RATE VALUES FOR THE SET OF 11 CLASSIFIERS

Classes	SVM	NB	Bagging	NN	RF	ERT	AdaBoost	GT	BB	XGBoost	EE
Analysis	1.00	1.00	0.99	1.00	0.86	1.00	0.87	1.00	0.77	0.87	0.86
Backdoor	1.00	1.00	0.93	0.98	0.76	0.95	0.93	0.95	0.59	0.64	0.60
DoS	0.90	0.99	0.88	0.98	0.87	0.87	0.99	0.93	0.81	0.83	0.99
Exploits	0.90	0.97	0.21	0.82	0.21	0.28	0.70	0.09	0.42	0.04	0.99
Fuzzers	0.94	0.63	0.42	0.89	0.39	0.42	0.93	0.55	0.32	0.29	0.75
Generic	0.51	0.03	0.03	0.03	0.03	0.03	0.42	0.03	0.04	0.02	0.42
Normal	0.04	0.65	0.24	0.24	0.21	0.23	0.86	0.34	0.34	0.39	0.70
Recon	0.65	0.71	0.19	1.00	0.17	0.22	0.17	0.21	0.17	0.18	0.99
Shellcode	0.96	0.98	0.31	1.00	0.30	0.52	0.92	0.60	0.06	0.12	0.81
Worms	1.00	0.95	0.86	1.00	0.04	0.86	1.00	0.56	0.09	0.43	0.88

B. Classifier Development Methodology

This research uses the training and testing subsamples existing on the UNSW website. There are 175,341 records in training and 82,332 records in testing data subsets. Each of which contains the records belonging to 9 different attack classes consisting of Analysis, Backdoor, DoS, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode, Worms, and the Normal class. Exploits, Generic and Normal instances make up a large portion of subsamples at 16%, 22% and 38% of the overall records in the training subset, respectively. In extreme cases, Worms attack instances form a mere 0.06% of training and test sets. Since the number of records of majority classes significantly outnumber the records of minority classes, there exists a severe class imbalance problem.The overlapping problem is present in these datasets between the Normal class and some attack classes. This work develops a design to address these problems and an ensemble classifier to identify the threats as normal or attacks.

The model has been implemented by numerous machine learning classifiers such as SVM, naïve Bayes (NB), multilayer perceptron (MLP) neural network (NN), Bagging, Random Forest (RF), Extremely Randomized Trees (ERT), AdaBoost, Gradient Tree (GT), Balanced Bagging (BB), XGBoost, and Easy Ensemble (EE) on the dataset for the case where the classifier algorithms employed all of the original features without implementing data normalization methods. The results are shown in Table I in terms of missed alarm rate (MAR) which is one of the, if not, most critical performance metrics for intrusion detection context. Table I shows that Balanced Bagging, XGBoost and Random Forest lead in their performances for this dataset with respect to the MAR metric. Consequently, these three classifiers will be employed in the design of an ensemble classifier.

The model also employ the Hellinger distance criterion [21] along with the Random Forest classifier to improve the quality of split. Decision Trees are easy to code, interpretable, fast, and nonlinear. However, they suffer from overfitting, axis-parallel splitting and skewness sensitivity. The overfitting problem is mitigated by tree pruning, while axis-parallel splitting can be addressed by building a forest of orthogonal and oblique decision trees. Skewness sensitivity of decision trees arises due to utilizing some popular splitting criteria including information gain and Gini measure. Hellinger distance measure can address this problem due to its skew insensitivity property. For instance, suppose that the model have two classes and Random Forest is applied on the training subset with 175,341 records. Also, in this scenario, the model have 1% of the entire data in class 'A' and the remaining records are in class 'B'. In the case that Random Forest splits the data on the feature 'rate' using a test or threshold value of 200,000, one splitting scenario for such an imbalanced dataset could be as presented in Fig. 1.

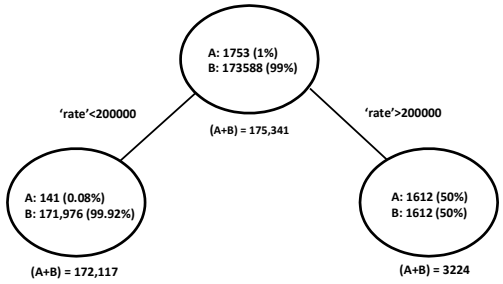

Fig. 1. An Example to Illustrate Hellinger Distance Metric Utility

In fact, regardless of balanced or imbalanced property of a dataset, the best split is made for binary classification when the entire data points in class 'A' are placed in the left node and all data points in class 'B' are placed in the right node. If that is the case, the perfect score for Entropy and Hellinger distance would be 1.0 and $\sqrt{2}$, respectively. In this example, which is a reasonably good but not a perfect split, the scores measured by Entropy and Hellinger are tabulated in table II.

TABLE II. ENTROPY AND HELLINGER DISTANCE SCORE MEASUREMENTS

	Entropy	Hellinger
Perfect Score	1.0	$\sqrt{2} \simeq 1.41$
Evaluated Score	0.015	1.29

According to score, Hellinger distance takes this split into account to form the final prediction. In contrast, given the Entropy formula as

$$E = -\sum_{c=1}^{n} p_c \log_2 p_c \tag{1}$$

and the IG formula as

$$Gain = E(parent) - \frac{W_L}{W_{parent}} E_L - \frac{W_R}{W_{parent}} E_R \tag{2}$$

The split in Fig. 1 does not appear to be "desirable". In equations (1), pi is the probability distribution associated with

class c, and c={1,2,...,n}. Information gain(IG) in equation (2) is the difference of Entropy in parent node (E_{parent}) from the Entropy of its left (E_L) and right (E_R) child where w_L is the number of data points in the left node, w_R is the number of data points in the right node, and wparent is the number of data points in the parent node.

For this split, it looks very probable that a record coming into the right node will be misclassified. This misclassification probability is $\frac{3224}{175341}$(2% of the entire data), while 98% of the data will be most probably classified correctly if they find their way into the left node. Accordingly, it can be considered as a good split. However, the information gained by Entropy measurement indicates that this split is a very bad one since the score is 1.5% of the perfect score shown in Table II. All the while, the Hellinger distance metric values this split by assigning it 91.49% of the perfect or maximum achievable score. Hellinger distance metric thus addressed, for the most part, the problem of skewness sensitivity if utilized by a decision tree classifier and will likely improve the performance of such an algorithm.

C. Training Methodology

The training set is split into two subsets: one subset, aka training subset, is used to train the classifier and the second subset, aka validation subset, is used to calculate its error rate to determine the convergence or stopping point. The training set is split into two subsets using stratified sampling: 90% of the training set is extracted in order to train the proposed model and the remaining 10% is kept to calculate the model's error. Each stratum is formed by ten different classes following a frequency distribution. In other words, the samples are picked randomly from each attack class as well as Normal records.Next, the training subset is processed with the Elastic Net feature selection algorithm. This algorithm selects 24 features out of 40 before the training subset is used to train the Balanced Bagging and XGBoost classifiers. Concurrently, the SFS algorithm selects 8 informative features out of 24 that were already selected among the original 40 by the Elastic Net. The output of SFS, which are 8 features, is used to train the Random Forest classifier. The classifier development schematic diagram has shown in Fig. 2.

Each classifier generates a matrix consisting of probability scores. Entries in this matrix are computed by dividing the number of votes for each class by the number of decision trees in each model. For instance, if the model had 30 decision trees in Random Forest and 20 of them vote for normal class on a new sample, the probability of Normal class is 0.67 (20/30). The first seven rows of the matrix produced by the Balanced Bagging is shown in Table III. It consists of 10 columns, representing 10 (9 attack plus Normal) classes, and N rows, where N designates the number of data samples in the validation subset. Since the validation subset has 35,068 records representing 10% of the training set, each model generates a probability matrix consisting of 35,068 rows. Both the probability matrix and the confusion matrix produced by XGBoost and Balanced Bagging classifiers are used as the inputs of Algorithm #1. This algorithm is employed to process the XGBoost and Balanced Bagging outputs to compute the errors caused by class overlap issue associated with the dataset.

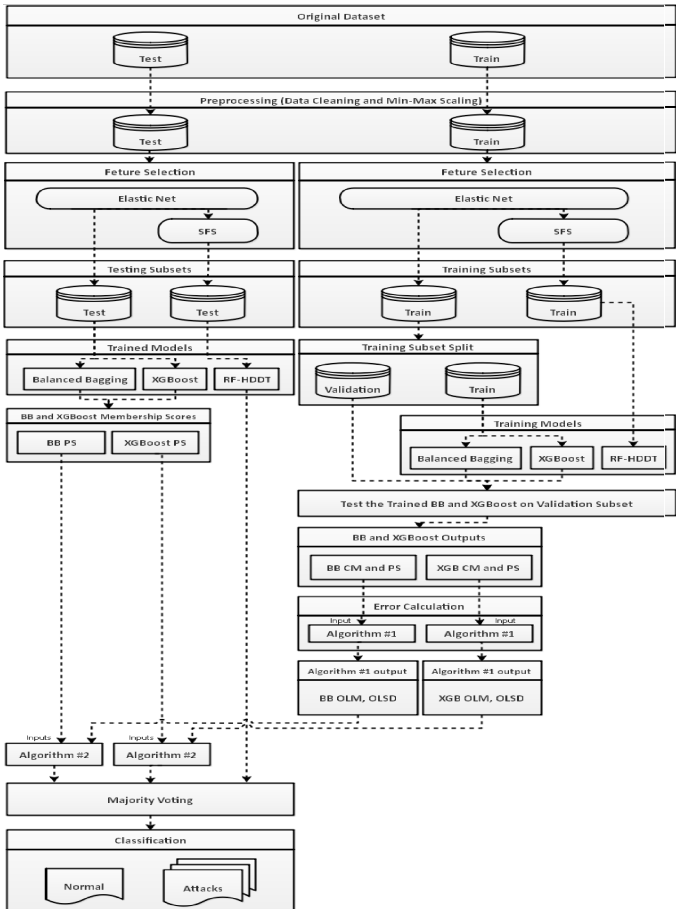

Fig. 2. Classifier Development Schematic Diagram

The output of Algorithm #1 is a nested list in Fig. 3 that consists of 10 items representing all the existing attack classes along with the Normal class which constitute the Level-1. Each item in Level-1 points to 9 sub-items in Level-2 as well. The corresponding sub-items in Level-2 for each item will not hold the item itself in Level-1. For each sub-item in the Level-2 of the nested list, there is a corresponding two-element list in Level-3. To make it more clear, it can consider this sole nested list as two two- dimensional arrays with the same dimensionality as the confusion matrix (10 by 10) storing mean and standard deviation. In other words, one matrix could hold the mean and another would hold the standard deviation values. Each row and column represents nine different attack classes along with the Normal class with the same order, similar to the rows and columns of the confusion matrix. These arrays store zeros along their main diagonal. The reason for that is that the aim of Algorithm #1 along with Algorithm #2 is to find the prediction errors or the errors existing in the membership scores. Since the main diagonal is holding true positives in confusion matrix, there is no error to calculate. That is why in the nested list, these entries are eliminated automatically.

Investigating the confusion matrix using Algorithm #1, if class A is misclassified x different times as class B, this algorithm iterates x times to calculate the difference between

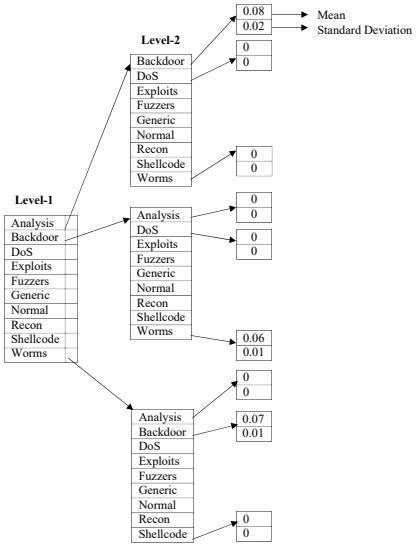

Fig. 3. Illustration of Partial Output by Algorithm #1

TABLE III. FIRST SEVEN ROWS OF A PROBABILITY SCORE MATRIX GENERATED BY BALANCED BAGGING CLASSIFIER

Row Index	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Normal	Recon	Shellcode	Worms
0	0.0083	0.0103	0.0988	0.1506	0.1718	0.0167	0.0750	0.0212	0.4463	0.0003
1	0.0034	0.0044	0.0632	0.1580	0.4279	0.0240	0.2884	0.0118	0.0144	0.0045
2	0.0057	0.0087	0.0641	0.1094	0.5057	0.0141	0.2659	0.0143	0.0111	0.0011
3	0.0062	0.0101	0.0627	0.1301	0.5491	0.0118	0.1968	0.0154	0.0159	0.0017
4	0.0018	0.0036	0.0339	0.0843	0.6023	0.0113	0.2427	0.0079	0.0109	0.0013
5	0.0044	0.0080	0.0827	0.1667	0.5055	0.0165	0.1213	0.0149	0.0774	0.0027
6	0.0034	0.0051	0.0605	0.1625	0.4867	0.0242	0.2271	0.0105	0.0140	0.0061

the probability score of class B and class A as well as class B and the eight remaining classes. If the former difference is smaller than the latter, this difference value (between class B and class A) is stored in a temporary variable (array D) for further calculation. Otherwise, the value is discarded. In the next step, the mean and standard deviation of the stored values are calculated and kept in arrays M and SD, respectively. These two values are placed in the third level of the nested list, which is an output of Algorithm #1, where the class A is the element of the first level of the list and class B is the element of the second level of the list.

To clarify how Algorithm #1 works, its application has been trace step by step next. The first row of Table IV(a) represents a confusion matrix for the Analysis attack and Table IV(b) represents the probability scores generated by the Balanced Bagging classifier in a two-dimensional array or matrix form after it is trained and its performance evaluated on the validation subset. Values of these matrices are held by CM and PS, two-dimensional array variables in the Algorithm #1, respectively. Initially, DL, OLM and OLSD are empty lists and eventually holding values for distances, output for mean values, and output for standard deviation values, respectively. AL is another list that initially contains the Normal and all the attack classes.

Algorithm 1 – Compute Means and Standard Deviations

Require: Mean-Standard-Deviation(CM, PS)
in:
two-dimensional array CM10×10 holding the confusion matrix and two-dimensional array PSn×10 holding the membership scores, n = the number of samples of validation subset
out:
two-dimensional array OLM10×10 initialized with zero
two-dimensional array OLSD10×10 initialized with zero
local:
empty array DL representing the minimum value of the membership scores difference and empty variable SD representing the computed Standard Deviation value and empty variable μ representing the computed Mean value and empty variable D representing the value obtained by subtracting the membership
scores constant:
array AL = {Analysis, Backdoor, DoS, Exploits, Fuzzers, Generic, Normal, Recon, Shellcode, Worms}

```
1: for all x ∈ AL do ▷ AL is an ordered list
2:   for all y ∈ ( AL - x) do
3:     if CM_{x,y} > 0 then
4:       j ← 0
5:       for i ← 1 ... CM_{x,y} do
6:         D ← PS_{i,y} - PS_{i,y} ▷ where x is misclassified as y
7:         for all z ∈ (AL - (x, y)) do
8:           DT ← PS_{i,y} - PS_{i,z}
9:           if DT < D then
10:                count ← 1
11:            end if
12:          end for
13:          if count = 0 then
14:            DL_j ← D
15             j ← j + 1
16:          end if
17:        end for
18:        sum ← 0
19:        for h ← 1 ... length(DL) do
20:            sum ← sum + DL_h
21:        end for
22:        μ ← SUM/N
23:        SD ← sqrt((1/N) Σ_{i=1}^{N} (DL_i − μ)^2)
24:        OLM_{x,y} ← μ
25:        OLSD_{x,y} ← SD
26:      end if
27:    end for
28:  end for
29: return OLM, OLSD
```

TABLE IV. (A) CONFUSION MATRIX AND (B) FIRST TWO ROWS OF PROBABILITY SCORES MATRIX.

Class No.	Analysis (Class 0)	Backdoor (Class 1)	DoS (Class 2)	Exploits (Class 3)	Fuzzers (Class 4)	Generic (Class 5)	Normal (Class 6)	Recon (Class 7)	Shellcode (Class 8)	Worms (Class 9)
0	501	124	0	0	46	0	3	3	0	0
1	0	302	0	0	0	12	0	0	0	1
2	3	0	270	4	0	0	0	0	5	0
3	0	0	0	254	8	0	50	0	0	0
4	3	0	0	32	345	0	23	0	0	0
5	1	0	0	0	9	138	0	0	0	0
6	0	0	0	54	78	0	876	0	0	0
7	0	0	0	0	0	0	8	132	0	0
8	0	0	0	17	0	0	0	1	187	0
9	1	0	0	1	0	0	0	0	2	74

(A)

Index	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Normal	Recon	Shellcode	Worms
1	0.78	0.81	0.02	0.01	0.24	0.08	0.11	0.19	0.08	0.22
2	0.09	0.54	0.01	0.03	0.41	0.04	0.01	0.06	0.12	0.14

(B)

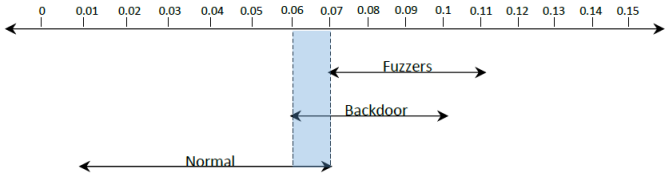

Fig. 4. An Example to Show how Algorithm 1 Calculates the Prediction Error Range

TABLE V. (A) THE CONFUSION MATRIX, (B) MEAN AND STANDARD DEVIATION ARRAYS THROUGH ALGORITHM #1 AND (C) MEAN AND STANDARD DEVIATION ARRAYS THROUGH ALGORITHM #2

	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Normal	Recon	Shellcode	Worms
Analysis	501	124	0	0	46	0	3	3	0	0

(a)

Analysis	Mean	0.08	0	0	0.09	0	0.04	1.3	0	0
	SD	0.02	0	0	0.02	0	0.03	0.1	0	0

(b)

Analysis	Mean	0.08	0	0	0	0	0	1.3	0	0
	SD	0.02	0	0	0	0	0	0.1	0	0

(c)

Fig. 4 shows that the Fuzzers, Backdoor, and Normal overlap. In this figure, Fuzzers represents a range of value between 0.09 -0.02= 0.07 and 0.09+0.02=0.11, Backdoor has a range of values between 0.08 -0.02= 0.06 and 0.08 +0.02= 0.1, and Normal is associated with a range of values between 0.04 - 0.03= 0.01 and 0.04+ 0.03= 0.07 where Analysis is incorrectly predicted as Fuzzers, Backdoor, and Normal, respectively. These values are taken from Table V(b) and after finding the overlaps, Table V(c) would be the final Mean and SD values generated for all the classes in preparation for the test phase. In Table V(c) the Mean and SD values for Recon remains unchanged since its error range is from 1.3-0.1= 1.2 to 1.3+0.1=1.4 and does not have any overlap with other ranges. Since Fuzzers, Backdoor, and Normal ranges overlap as well as the greater number of Analysis records are misclassified as Backdoor, Mean and SD values calculated for Backdoor remain unaltered and the Mean and SD values associated with Fuzzers and Normal are changed to zero. The process of finding the range overlaps and addressing them takes place for all of 10 classesThis procedure gives us a list of Means and SDs that is shown in Fig. 3. The list is eventually utilized in the test phase to minimize the prediction errors caused by data overlap.

D. Testing Methodology

In this phase, Mean and SD values computed by Algorithm #1 and revised after finding the overlaps are used to reduce the errors made by XGBoost and Balanced Bagging classifiers in identifying unseen samples. Due to data overlap issue associated with UNSW-NB15, classifiers may tend to predict class membership for certain new samples incorrectly. A new sample mimicking the behavior of data points belonging to another attack class is most likely to be misclassified. The probability scores matrix generated by the classifiers for a new sample in class 'A' contains error if this sample is incorrectly classified as class 'B'. In this case, the probability matrix score for class 'A' is lower than that of class 'B', while it must be just the opposite. The model consider the difference between the probability scores for classes 'A' and 'B' as error.

Algorithm 2 Membership Score Modification

Require: *Membership-Score-Modification(PS, OLM, OLSD)*
in: two-dimensional array PSn×10 holding the membership scores, n = the number of samples of test subset two-dimensional array OLM10×10 two-dimensional array OLSD10×10
out: two-dimensional array PSn×10 holding the (modified) membership scores, n = the number of samples of test subset
1: **for** i ← 1 … n **do**
2: max ← 0 ▷ holding zero in max to find the maximum membership score from line 3 to 8
3: **for** j ← 1 … 10 **do**
4: **if** $PS_{i,j} >$ max **then**
5: max ← $PS_{i,j}$
6: index_max ← j
7: **end if**
8: **end for**
9: min ← 10^{10} ▷ holding a very big constant value in min to find the minimum values obtaining from line 10 to 15
10: **for** j ← 1 … 10 **do**
11: **if** ((max-$PS_{i,j}$) < min) and (index_max ≠ j) **then**
12: min ← max - $PS_{i,j}$
13: index_min ← j
14: **end if**
15: **end for**
16. **if**(min≥($OLM_{index\_min,index\_max}$ - $OLSD_{index\_min,index\_max}$)) and (min≤($OLM_{index\_min,index\_max}$+$OLSD_{index\_min,index\_max}$)) **then**
17: $PS_{i,index\_min}$ ← ($PS_{i,index\_min}$ + $OLM_{index\_min,index\_max}$)
18: **end if**
19: **end for**
20: **return** PS

To reduce this error, Algorithm #2 has been applied on the probability score matrices generated through XGBoost and Balanced Bagging classifiers.

Since only the test set is utilized to evaluate the performance of our model, it is definitely unknown to the model. So, the model does not know the real target variable and it cannot calculate the errors using ground truth. This algorithm is designed to go through the membership scores generated by XGBoost and Balanced Bagging classifiers for the test subset in order to calculate the errors regardless of real target variables. The following steps discuss the functionality of Algorithm #2:

Using the revised mean and standard deviation values obtained by implementing Algorithm #1, Algorithm #2 is able to realize and correct the errors in the probability matrices where the errors arise from data overlap. Although, errors are not zeroed out using this algorithm, they may be reduced appreciably. The modified membership scores were used along with the membership scores obtained by the augmented Random Forest in the testing phase to make the final prediction using the majority vote. Using this method, the most voted prediction wins and taken as a final prediction. In other words, if more than two classifiers cast a vote for a particular class, the class will gain the final vote.

IV. Experiment and Result Analysis

An Intel Core i7 processor with 16 GB of RAM and Python with TensorFlow was used to create the proposed system. Before getting into the results, it is necessary to provide some background information on the UNSW-NB15 dataset. This dataset has been selected because of its advantages over older standard datasets. The lack of current cyberattacks types in the KDD98, KDDCUP99, and NSLKDD datasets and insufficient normal traffic and an unbalanced distribution of classes in the training and testing sets cause the datasets to suffer. The UNSW-NB15 benchmark dataset, explicitly designed for IDS design, has been presented to address these issues.

A. Experimental Results Evaluation

The confusion matrix is used to calculate performance measures for classifiers. The confusion matrix is used to calculate True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). TP, TN, FP, and FN denote records that are correctly identified as positives, negatives, or incorrectly identified as positives. The most standard measures include sensitivity, specificity, false positive, false negative, precision, and accuracy [22]. Abnormalities in a dataset must be taken into account while evaluating it. Due to the imbalanced datasets, accuracy is not appropriate. Insufficient datasets can use F-measure. This study aims to evaluate accuracy using the same parameter as previous studies has shown in equation (3). Accuracy is identified as the ratio of the correct classifications to the total number of samples and defined by the following formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

Sensitivity or Detection Rate (DR), shown in equation (4), corresponds to the proportion of true positives to all positives. It measures the probability of a sample being actually positive from all positive data points. Specificity indicates the proportion of false positives to all negatives. It measures the probability of a sample being actually positive from all data points that are predicted to be positive. They are used when the only positive or negative matter.

$$Sensitivity = \frac{TP}{TP + FN} \tag{4}$$

False Positive Rate (FPR) or False Alarm Rate (FAR) represents the ratio of incorrect positive predictions to the overall number of negatives. At the same time, the Precision measures the probability of samples classified as positives for actually being positive. These two metrics are defined as shown in equation (5) and (6).

$$FPR = \frac{FP}{FP + TN} \tag{5}$$

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

False Negative Rate (FNR) or Missed Alarm Rate, as shown in equation (7), indicates the ratio of incorrect negative predictions to the total number of positives. The evaluation metrics used in this study are based on the parts of the confusion matrix.

$$FNR = \frac{FN}{FN + TP} \tag{7}$$

F-measure is the harmonic mean of Sensitivity and Precision and is given by equation (8).

$$F - Measure = \frac{2 * Precision * Sensitivity}{Precision + Sensitivity} \tag{8}$$

B. Binary Classification

Table VI shows that 790 attack records are misclassified among the entire 45,332 attack records. It means that less than 2% of the overall attacks are misclassified as non-attack or Normal which leads to 0.017 missed alarm rate. In other words, the sensitivity for the attack class is 98.26%. On the other hand, 1022 of 37,000 Normal records are incorrectly classified as attacks which indicates less than 3% false alarm rate. Several evaluation metric values are shown in Table VII in order to comprehensively assess the performance of the proposed classifier design.

TABLE VI. Confusion Matrix on Test Data Subset for Binary Classification

PredictedActual	Normal	Attack
Normal	35978	1022
Attack	790	44542

TABLE VII. Performance Evaluation of the Proposed Model for Binary Classification

Metrics Class Type	Sen (%)	Spe (%)	FPR	FNR	Precision (%)	F-measure (%)
Normal	97.24	98.26	0.017	0.028	97.85	97.75
Attack	98.26	97.24	0.028	0.017	97.76	98.01

The main objective of any intrusion detection system (IDS) is to identify the pattern of the network traffic that may imply a suspicious activity. Accordingly, the performance of proposed IDS on UNSW-NB15 data is competitive in comparison with the other studies reported in the literature as shown in Table VIII. Two columns are dedicated to present the performance of the proposed classifier design. The first column indicates the average of calculated metrics in Table VII for both the attack and Normal records. On the other hand, the second column shows the values of performance metrics associated with the attack class in Table VII, which suggests that the performance of the proposed classifier design.

TABLE VIII. COMPARISON OF THE PROPOSED CLASSIFIER DESIGN WITH FOUR OTHERS CITED (NR INDICATES NOT REPORTED)

Metrics	Proposed Design (Avg)	Proposed Design	[10] (Avg)	[11]	[12]	[13]	[14]	[15]	[16]
Sensitivity	97.75	98.26	79.12	85	NR	98.47	NR	NR	NR
FNR (%)	2.25	1.74	NR	15	NR	NR	NR	NR	NR
FPR (%)	2.25	2.76	NR	2	8.6	2.18	NR	5.56	NR
Precision (%)	97.81	97.76	NR	99	NR	NR	NR	NR	96
F-measure (%)	97.88	98.01	77.87	91	NR	NR	NR	NR	95
Accuracy (%)	97.8	97.8	NR	89	91.31	94.11	81.42	86.49	95

C. Multiclass Classification

In this research, the performance of different estimators were evaluated separately by implementing them on the refined dataset. The results are shown in Table I. This study combined the first three estimators produced satisfactory performances to form an ensemble method along with Elastic Net and Sequential Forward Selection while Min-Max scaler had already implemented on the refined dataset. The results of the model evaluation is shown in Table IX in terms of confusion matrix. Although the outcome of the ensemble method has effectively improved the performance of each classifier contributed in the method, the model still suffered from generating large number of false negatives. In order to cover this issue, the study proposed two algorithms utilizing the basic statistic methods such as mean and standard deviation. Comparing Table IX with the performance of the proposed model depicted in detail for the multi-class case in Table X, represents the significant improves in most classes, such as Normal class to shrink the number of false negatives. This improvement verifies the effectiveness of Algorithm #1 and Algorithm #2. Although the study observe the increasing number of false negatives in some elements in confusion matrix, such as Fuzzers incorrectly classified as Backdoor, when the proposed algorithms implemented on the final decision, they are mostly seen between two attack class rather than an attack class and Normal records. On the other words, false negatives in one attack class may be increase due to the attack class to attack class misclassification. In Table X, the share of attack records which are incorrectly categorized as Normal traces is 4.14% for the 28% overall missed alarm rate. The remaining 23.86% missed alarm rate is associated with misclassification among attack classes which is not equally as problematic for network transactions. Although, 4.14% missed alarm rate for attack records alone could be detrimental for an intrusion detection system, our design achieves better results in comparison with the classifiers in other studies as shown in Table XI. Normal traces are also misclassified as Shellcode, Fuzzers and Analysis which suggests approximately 3% false alarm rate and 97.24% sensitivity.

TABLE IX. CONFUSION MATRIX SHOWING THE PERFORMANCE OF THE ENSEMBLE METHOD

Class No.	Analysis (Class 0)	Backdoor (Class 1)	DoS (Class 2)	Exploits (Class 3)	Fuzzers (Class 4)	Generic (Class 5)	Normal (Class 6)	Recon (Class 7)	Shellcode (Class 8)	Worms (Class 9)
0	327	193	4	1	18	0	130	0	4	0
1	298	154	8	8	23	0	76	1	15	0
2	1477	737	775	435	221	0	228	45	164	7
3	1322	1583	51	6197	257	0	667	490	420	145
4	651	386	8	14	1837	0	2604	7	540	15
5	15	30	37	390	99	18145	63	7	72	13
6	842	1	3	28	1427	0	34545	0	154	0
7	97	210	0	35	14	0	31	2967	128	14
8	11	0	0	0	7	0	9	3	347	1
9	0	0	0	1	0	0	1	0	3	39

TABLE X. CONFUSION MATRIX SHOWING THE PERFORMANCE OF THE PROPOSED DESIGN

Class No.	Analysis (Class 0)	Backdoor (Class 1)	DoS (Class 2)	Exploits (Class 3)	Fuzzers (Class 4)	Generic (Class 5)	Normal (Class 6)	Recon (Class 7)	Shellcode (Class 8)	Worms (Class 9)
0	327	193	4	1	18	0	130	0	4	0
1	126	405	0	0	13	0	39	0	0	0
2	1604	625	1110	228	92	0	178	59	164	29
3	779	830	39	8511	57	0	338	221	213	144
4	482	491	4	34	4380	0	0	10	646	15
5	15	30	37	390	99	18145	63	7	72	13
6	200	0	0	0	668	0	35978	0	154	0
7	103	207	0	35	14	0	31	2967	126	13
8	0	0	0	0	7	0	9	3	358	1
9	0	0	0	1	0	0	2	0	5	36

This is because these attack types mimic the behavior of Normal records [20, 23, 24]. This is the main reason that some attacks are also incorrectly predicted as Normal activity. In the associated confusion matrix, it can see that 19.20% of Analysis, 6.69% of Backdoor, 4.35% of DoS, 3.04% of Exploits, 0.33% of Generic, 0.88% of Reconnaissance, 2.38% of Shellcode, and 4.55% of Worms attack records are confused with Normal records.

Performance comparison of the proposed model with those studies reported in the literature is presented in Table XII and Table XIII. Many of the relevant performance metrics including the missed alarm rate, which is one of the most critical ones, are not reported in these studies by others. Consequently, performance comparison is done only for accuracy and sensitivity as these are the only metrics commonly reported in the cited studies.

Fig. 5 depicts the performance of the model in comparison with two models, Integrated and Dendron [18,19] that are proposed recently in terms of the F-measure. The proposed model in this study outperforms the other two given the F-measure values. The main reason is that the imbalance and overlapping problems in our model are addressed. This

TABLE XI. PERFORMANCE OF THE PROPOSED DESIGN FOR MULTI-CLASS CASE

Metric / Class	Accuracy	Sensitivity	Specifity	FPR	FNR	Precision	F-measure
Class0	95.56	48.30	95.95	0.041	0.51	0.15	64.25
Class1	96.89	69.47	97.09	0.029	0.31	0.84	80.99
Class2	96.27	27.15	99.89	0.001	0.73	0.42	42.70
Class3	95.98	76.46	99.03	0.009	0.24	0.84	86.29
Class4	96.78	72.25	98.73	0.012	0.28	0.77	83.44
Class5	99.12	96.15	100.0	0.000	0.28	1.00	98.04
Class6	97.81	97.24	98.26	0.017	0.02	0.97	97.75
Class7	95.39	84.87	95.86	0.041	0.15	0.61	90.03
Class8	99.31	94.71	98.33	0.017	0.05	0.34	96.49
Class9	99.73	81.82	99.74	0.003	0.18	0.24	89.90

CLASS0: ANALYSIS CLASS1: BACKDOOR CLASS2: DOS CLASS3: EXPLOITS CLASS4: FUZZERS CLASS5: GENERIC CLASS6: NORMAL CLASS7: RECONNAISSANCE CLASS8: SHELLCODE CLASS9: WORMS

TABLE XII. THE ACCURACY OF PROPOSED MODEL VS. THE ACCURACY OF DIFFERENT MODELS (NR: NOT REPORTED)

Attack Type	Accuracy	Accuracy [17]	Accuracy [18]	Accuracy [19]	Accuracy [20]	Difference
Analysis	95.56	**99.44**	99.26	99.3	NR	-3.88
Backdoor	96.89	99.06	**99.11**	97.93	NR	-2.22
Dos	**96.27**	96.14	94.9	95.71	94.52	0.11
Exploits	**95.98**	93.91	90.12	93.58	89.72	2.07
Fuzzers	**96.78**	96.52	91.47	95.04	NR	1.74
Generic	**99.12**	98.34	98.23	98.7	87.7	0.26
Normal	97.81	98.16	93.54	94.59	**98.64**	-0.3
Reconnaissance	95.39	98.74	95.33	96.18	**99.1**	-3.71
Shellcode	99.31	99.22	**99.4**	98.33	NR	-0.09
Worms	99.73	97.28	**99.92**	99.78	NR	-0.14

TABLE XIII. THE SENSITIVITY OF PROPOSED MODEL VS. THE SENSITIVITY OF DIFFERENT MODELS (NR: NOT REPORTED)

Attack Type	Sensitivity	Sensitivity [19]	Sensitivity [20]	Difference
Analysis	48.30	20.45	NR	+27.85
Backdoor	69.47	67.32	NR	+02.15
Dos	27.15	14.29	5.0	+12.86
Exploits	76.46	76.22	54.64	+00.24
Fuzzers	72.25	64.42	NR	+07.83
Generic	96.15	81.37	96.72	-00.57
Normal	97.24	97.39	98.00	-00.76
Reconnaissance	84.87	46.04	71.70	+13.17
Shellcode	94.71	36.39	NR	+58.32
Worms	81.82	18.37	NR	+63.45

work have a combination of ensemble methods to handle the imbalance and if-then-else rules to mitigate the adverse effects of overlapping issue and using Hellinger distance criterion to choose the best split considering the imbalance problem.

V. CONCLUSION

This study presents design and performance evaluation of an intrusion detection (and identification) system using machine learning for the UNSW-NB15 dataset. The study evaluated the performance of classifier design which employs three ensemble classifiers and two proposed algorithms where the later is developed for minimizing the errors due to one of the two issues inherent to the UNSW-NB15 dataset, namely the class overlap and class imbalance. To deal with the imbalanced data, this work utilized the Balanced Bagging and the XGBoost ensemble classifiers which offer a set of hyper-parameters that,

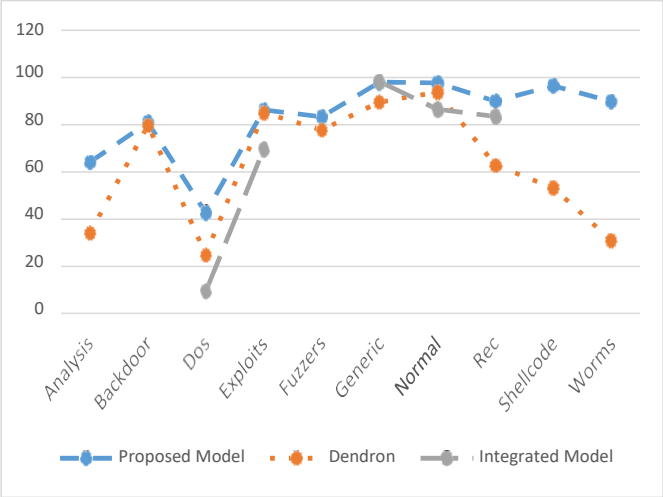

Fig. 5. F-measure Comparison

through judicious adjustments of the same, help contribute to improved performance in the presence of the imbalanced data. To address the class overlap issue, the study proposed two algorithms and utilized them to process and modify the classification outputs from the Balanced Bagging and the XGBoost ensembles. Outputs of three ensemble classifiers, namely Random Forest, Balanced Bagging and XGBoost, were provided as inputs to a combiner that implemented majority voting to determine the final class membership of an input data record under test. The performances of the classifiers are assessed by employing six different normalization methods on the modified UNSW-NB15. The results showed that min-max scaler enhanced the performance of the classifiers in terms of accuracy. Min-max scaler as a normalization method helped increase the distances between the data points of two different attack classes reducing the degree of class overlap. Application of the combination of preprocessing, feature selectors, tree-based ensemble classifiers, and the proposed algorithms for this design resulted in superior performance when compared to seven other classifiers, reported in the recent literature, implemented on the UNSW-NB15 dataset for both multi-class and binary classification cases. Performance of the proposed model was compared with both the binary classifiers and multi-class classifiers cited in the literature. In the binary class classification case, this model could classify more than 98% of the attack classes correctly. The model also performed highly for the classification of Normal records with more than 97%.

In comparison with other studies reported in the current literature on the UNSW-NB15 dataset, this model achieved impressive results. It addresses two major issues that a dataset may suffer from, overlap and imbalance. The study employed Balanced Bagging and XGBoost offering a range of hyperparameters in order to address the dataset imbalance. Also, the study utilized the Hellinger distance for the Random Forest for the same reason. The study further proposed two new post-processing algorithms for the outputs of training models to minimize the errors caused by the large number of impure nodes generted during the training phase due to the data overlap issue. In future this work plan to use Hellinger distance

as split criterion for both Balanced Bagging and XGBoost to enhance the performance of this model. The future works are also aimed at utilizing the proposed algorithms, known as Algorithm #1 and Algorithm #2, along with Random Forest.

REFERENCES

[1] A. K. Pandey, A. K. Tripathi, G. Kapil, V. Singh, M. W. Khan, A. Agrawal, et al., "Trends in Malware Attacks: Identification and Mitigation Strategies," in Critical Concepts, Standards, and Techniques in Cyber Forensics, ed: IGI Global, 2020, pp. 47-60.

[2] J. L. Cebula and L. R. Young, "A taxonomy of operational cyber security risks," Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst2010.

[3] E. C. Thompson, Cybersecurity Incident Response: How to Contain, Eradicate, and Recover from Incidents: Apress, 2018.

[4] S. Morgan, "Official annual cybercrime report," Sausalito: Cybersecurity Ventures,2019.

[5] J. Armin, B. Thompson, D. Ariu, G. Giacinto, F. Roli, and P. Kijewski, "2020 cybercrime economic costs: No measure no solution," in 2015 10th International Conference on Availability, Reliability and Security, 2015, pp. 701-710.

[6] Uikey, R.; Gyanchandani, M.: Survey on classification techniques applied to intrusion detection system and its comparative analysis. Int. Conf. Commun. Electron. Syst. 2019, 1451–1456 (2019)

[7] Soe, Y.N.; Feng, Y.; Santosa, P.I.; Hartanto, R.; Sakurai, K.: Machine learning-based iot-botnet attack detection with sequential architecture. Sensors 20(16), 4372 (2020)

[8] Ahmad, Z.; ShahidKhan, A.;Wai Shiang, C.;Abdullah, J.;Ahmad, F.: Network intrusion detection system: a systematic study of machine learning and deep learning approaches. Trans. Emerg. Telecommun. Technol. 32(1), e4150 (2021)

[9] Chebrolu, S.; Abraham, A.; Thomas, J.P. Feature deduction and ensemble design of intrusion detection systems. Comput. Secur. 2005, 24, 295–307. [CrossRef]

[10] V. Kumar, A. K. Das, and D. Sinha, "UIDS: a unified intrusion detection system for IoT environment," Evolutionary Intelligence, pp. 1-13, 2019.

[11] V. Kumar, D. Sinha, A. K. Das, S. C. Pandey, and R. T. Goswami, "An integrated rule based intrusion detection system: analysis on UNSW-NB15 data set and the real time online dataset," Cluster Computing, pp. 1-22, 2019.

[12] B. A. Tama and K.-H. Rhee, "An in-depth experimental study of anomaly detection using gradient boosted machine," Neural Computing and Applications, vol. 31, pp. 955-965, 2019.

[13] M. G. Raman, N. Somu, S. Jagarapu, T. Manghnani, T. Selvam, K. Krithivasan, et al., "An efficient intrusion detection technique based on support vector machine and improved binary gravitational search algorithm," Artificial Intelligence Review, pp. 1-32, 2019.

[14] Khammassi, Chaouki and Saoussen Krichen. "A GA-LR wrapper approach for feature selection in network intrusion detection." Comput. Secur. 70 (2017): 255-277.

[15] Tian, Q., Han, D., Li, KC. et al. An intrusion detection approach based on improved deep belief network. Appl Intell 50, 3162–3178 (2020). https://doi.org/10.1007/s10489-020-01694-4

[16] Abirami, M. & Yash, Umaretiya & Singh, Sonal. (2020). Building an Ensemble Learning Based Algorithm for Improving Intrusion Detection System. 10.1007/978-981-15-0199-9_55.

[17] Soulaiman Moualla, Khaldoun Khorzom, Assef Jafar, "Improving the Performance of Machine Learning-Based Network Intrusion Detection Systems on the UNSW-NB15 Dataset", Computational Intelligence and Neuroscience, vol. 2021, Article ID 5557577, 13 pages, 2021. https://doi.org/10.1155/2021/5557577

[18] J. Sharma, C. Giri, O.-C. Granmo, and M. Goodwin, "Multi-layer intrusion detection system with ExtraTrees feature selection, extreme learning machine ensemble, and softmax aggregation," EURASIP Journal on Information Security, vol. 2019, p. 15, 2019.

[19] D. Papamartzivanos, F. G. Mármol, and G. Kambourakis, "Dendron: Genetic trees driven rule induction for network intrusion detection systems," Future Generation Computer Systems, vol. 79, pp. 558-574, 2018.

[20] V. Kumar, D. Sinha, A. K. Das, S. C. Pandey, and R. T. Goswami, "An integrated rule based intrusion detection system: analysis on UNSW-NB15 data set and the real time online dataset," Cluster Computing, pp. 1-22, 2019.

[21] D. A. Cieslak, T. R. Hoens, N. V. Chawla, and W. P. Kegelmeyer, "Hellinger distance decision trees are robust and skew-insensitive," Data Mining and Knowledge Discovery, vol. 24, pp. 136-158, 2012.

[22] Abhijit Das, S G Balakrishnan and Pramod, "Network Intrusion Detection System based on Generative Adversarial Network for Attack Detection" International Journal of Advanced Computer Science and Applications(IJACSA), 12(11), 2021. http://dx.doi.org/10.14569/IJACSA.2021.0121186

[23] N. Moustafa and J. Slay, "The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," Information Security Journal: A Global Perspective, vol. 25, pp. 18-31, 2016.

[24] Y. Yang, K. Zheng, C. Wu, X. Niu, and Y. Yang, "Building an effective intrusion detection system using the modified density peak clustering algorithm and deep belief networks," Applied Sciences, vol. 9, p. 238, 2019.