Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Network Intrusion Detection using Ensemble Voting Classifier for IoT devices

Osama Javid:230308[1], Muhammad Zorain: 230300
[1]Department of Creative technologies, Faculty of Computing & A.I., Air University, Islamabad, Pakistan

I. PROPOSED METHODOLOGY

To increase the effectiveness of identifying rogue nodes in a network, the researchers is working on a solution. They put out a voting categorization technique that divides nodes into benign and harmful categories using a collection of machine learning methods. The data was collected from three independent datasets, preprocessed, and then input into the model. The model was developed, tested, and the outcomes were assessed using a variety of assessment measures. Figure 1 displays the conceptual diagram for the suggested model.

A. DATA GATHERING

The collection of data for the suggested model's training and testing is the initial phase in this research. the Three datasets are used to evaluate the model: NSL-KDD, UNSW-NB15, and CIC-IDS 2017.

1) NSL-KDD

The KDD- Cup 99 sample, which was use in a challenge for NIDS, has been updated into the NSL-KDD dataset.NSL-KDD dataset comprises 41 unique attributes for each record that represents a peer-to-peer connection. Each record is classified as normal, aberrant, or one of many distinct attack names (as shown in Table 2). The four categories of assaults are DoS, probing, U2R, and R2L.

TABLE 1: NSL-KDD Data Set

Class	No of Occurrence
Normal	77,054
DOS	51,668
Probe	12,762
R2l	6,944
U2R	89

2) UNSW-NB15

A more recent dataset is the UNSW-NB15 was made in 2015 and contains nine contemporary assaults. attack types as opposed to the KDD-99's 14 attack types dataset). It has 49 traits, including both common and unusual ones.attack actions with class names and 25,40,044 attacks overall records. There are 3,21,283 unusual records and 2,18,761 regular ones. assault the dataset's records. The characteristics of UNSW- There are six sections for the NB15 dataset: Basic Features, Features of Flow, Time, Content, and Additional both labelled features and generated features. includes 36-40 features 41–47 are referred to as General Purpose Features. considered to be connecting characteristics. Dataset UNSW-NB15 adds nine more attack types, such as Analysis Fuzzers, Back-backdoor, DoS, Exploits, Generic, Shellcode, Reconnaissance and worms.

TABLE 2: UNSW-NB15 Data Set

Class	No of Occurrence
Normal-data	93,000
Fuzzers-attack	24,246
Analysis-attack	2,677
Backdoors-attack	2,329
Dos-attack	16,335
Exploits-attack	44,525
Generic-attack	58,871
Reconnaissance-attack	13,987
Shellcode-attack	1,511
Worms	174

3) CIC-IDS2017

Since it was introduced, the CICIDS2017 dataset has been widely used by researchers to design new models and algorithms. Dataset, which was collected by the Canadian Institute of Cybersecurity, consists of eight files contains five

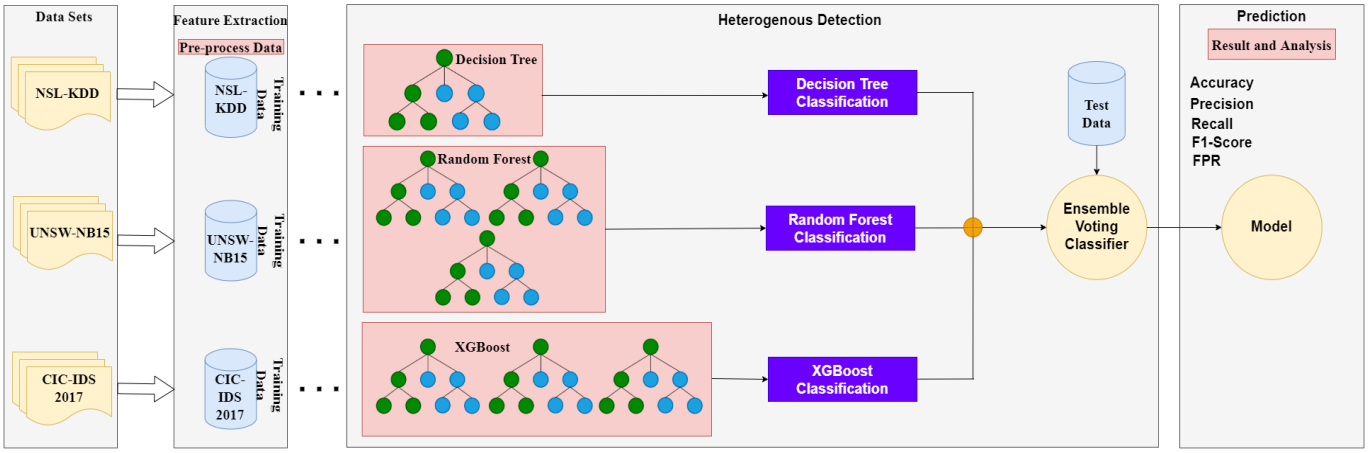

FIGURE 1: Proposed DRX-based network intrusion detection and classification technique

days of normal and attack data There are a total of 2,40,394 distinct instances in the dataset, with 83 features.

TABLE 3: CIC-IDS2017 Data Set

Class	No of Occurrence
Normal	1,17,964
DoS Hulk	46,215
PortScan	31,786
DoS GoldenEye	9,264
DDoS	8,367
FTP-Patator	7,144
SSH-Patator	5,307
DoS slowloris	5,216
DoS Slowhttptest	4,949
Bots	1,966
Web Attack - Brute Force attack	1,507
Web Attack - XSS	652
Infiltration	36
Web Attack - Sql Injection	21

B. DATA PRE-PROCESS

Data pre-processing is an essential step into transforming raw data into meaningful and efficient data that can be used by ML algorithms. The first technique used in this process is data normalization, specifically min-max normalization, which scales all values of the attributes to a range of 0 to 1. The second technique is label encoding, which converts all string values in the data-set to 0s and 1s based on the output class label. The third technique used is Synthetic Minority Oversampling Technique (SMOTE), which oversamples the few classes in the dataset to create a balance distribution and use the correlation to extract the best 30 features. In our preprocessing method, we apply different techniques to generate the best training dataset for the ensemble classifiers. As a result, we have three balanced sample datasets.

C. MACHINE LEARNING ALGORITHMS

In the proposed method, we use the ensemble voting classifier that uses Decision Tree Random Forest and XGBoost machine learning algorithms. These are discussed below:

TABLE 4: Pre-Process NSL-KDD Data Set

Attack Type	Train Data (80%)	Test Data (20%)
Normal	61,604	15,450
DOS	41,330	10,338
Probe	10,300	2,462
R2L	5,506	1,436
U2R	71	18

TABLE 5: Pre-Process UNSW-NB15 Data Set

Attack Type	Train Data (80%)	Test Data (20%)
Normal	74,387	18,613
Generic	47,334	11,537
Exploits	35,445	9,080
Fuzzers	19,415	4,831
Dos	13,061	3,292
Reconnaissance	11,135	2,852
Analysis	21,58	519
BackDoor	1,861	468
Shellcode	1,202	309
Worms	140	30

TABLE 6: Pre-Process CIC-IDS 2017 Data Set

Attack Type	Train Data (80%)	Test Data (20%)
Normal	91607	22902
DoS/DDoS	49386	12346
PortScan	18491	4623
Brute Force	6223	1556
Web Attack	1558	389
Botnet ARES	1694	424
Infiltration	29	7

1) Decision Tree

A feature selection technique and tested using a decision tree and multi-layer perceptron (MLP) for classification. According to their findings, DT performance was good as compared to MLP. DT is the classification algorithm that classifies the objects present in the data set by the value of attributes. The src and des IP addresses, the kind of protocol being used, and the quantity of packets being transferred, for instance, might all be utilized as characteristics by the decision tree

to decide if the traffic is normal or attacked. The method would start by choosing the most pertinent characteristics from the training data and utilizing them as the tree's root node to build the decision tree for a NIDS. Then, depending on each feature's potential values, it would build branches, and each branch would lead to a subnode that represented the following feature to be tested. A leaf node would eventually be reached by the procedure, depending on the values of the characteristics, indicates the tree's ultimate judgment.

According on the values of the characteristics, the tree would then employ a set of rules, similar to those previously discussed, to categorise new network traffic as legitimate or malicious. For instance, the rule in the tree may state, "If the traffic is malicious and the origin IP address is in a malicious network IP range, then the data is malicious." When fresh network traffic is encountered, the tree would categorise it using this rule and the other rules it has learnt from the training data.

Consider, X is equal to the input sample (i.e., the network traffic data), F1, F2, ..., Fn are the the features of the input sample and C is the class label (i.e., normal or malicious). The decision tree would work as follows.

If following equation results in true then C is classified as malicious.

$$(F_1 = a_1) and (F_2 = b_1) or (F_3 = c_1) and (F_4 = d_1) \quad (1)$$

Otherwise, if the following condition is satisfied, the sample is classified as normal.

$$(F_5 = e_1) and (F_6 = f_1) \quad (2)$$

And, if the output of the following condition is true, the C is equal to malicious.

$$(F_7 = g_1) \quad (3)$$

The samples that are not classified by the above conditions are considered as normal.

2) Random Forest

Random forest the classification of network traffic as either normal or malicious is done using a particular kind of machine learning algorithm called RF. A huge number of DT are created using the random forest technique, and each one is train using a different random sub-set of the training data. The final call of the random forest is then reached by combining all of the individual decision trees' decisions, often via a procedure known as majority voting. The method would start by choosing the most pertinent characteristics from the train data and utilising them as the tree's root node to construct a decision tree for an NIDS. Then, depending on each feature's potential values, it would build branches, and each branching would link to a subnode that represented the following feature to be tested. The process would proceed until a leaf node, which symbolizes the tree's ultimate choice according to the values of the traits, was achieved.

The algorithm for random forests would produce several of these decision trees, using separate random selected samples of the train data, . A new input sample would be run through each decision tree in the random forest to categorize it, and the results would then be combined using majority voting. For instance. For example, if 80% of the decision trees classify the sample as normal and 20% classify it as malicious, then the final decision of the Forest algorithm would be normal.

In terms of an equation, the random forest algorithm can be represented as follows:

$$C = majorityvoting(C_1, C_2, \ldots, C_n) \quad (4)$$

where C is the final class label (normal or malicious) and C_1, C_2, ..., C_n are the class labels predicted by the individual decision trees.

3) XGBoost

XGBoost (eXtreme Gradient Boosting) is a machine learning method that is applied to tasks involving classification and regression. It is a method of ensemble learning that integrates the result of many in-efficent models, frequently decision trees, to provide a more precise result. Because it employs gradient descent optimization to reduce the loss function and boost prediction accuracy, XGBoost is extremely successful.

XGBoost can be used to categorize network traffic as legitimate or malicious in the environment of the NIDS. To do this, a group of decision tree models, each trained using a portion of the training data, would be first created via the XGBoost method. Gradient descent optimization would then be used to reduced loss function and raise the model's accuracy. XGBoost model would run a new input sample through each decision tree, using the predictions from the trees to arrive at a final classification. The final forecast would be the weighted sum of the predictions from each decision tree, the weights being established by the optimization procedure.

In terms of an equation, the XGBoost algorithm can be represented as follows:

$$C = \sum_{i=1}^{n} w_i C_i \quad (5)$$

where:

- C is the final class label (normal or malicious)
- C_i is the class label predicted by the ith decision tree
- w_i is the weight assigned to the ith decision tree based on the optimization process.
- n is total number of DT in XGBoost model

To classify a new input sample, the XGBoost model would pass the sample through each of the decision trees, and the final prediction would be the weight sum of the predictions

of the individual DT, with the weights determined by the optimization process.

D. PROPOSED ENSEMBLE VOTING CLASSIFIER

Classifier combines many base model predictions, frequently decision trees, to get a more precise final forecast. Using a process called majority voting, the forecasts of the base models are combined to create the final projection. The same train data is used to train the basic models. The two fundamental subcategories of ensemble polling classifiers are hard voting and soft voting. Class label that the base models most commonly predict is the outcome in hard voting (e.g., normal or malicious). Predictions are generated by the basic models as class labels (e.g., normal or malicious). The underlying models produce probability estimates as predictions for soft voting, with the class label with the greatest average probability serving as the final projection. An ensemble voting classifier may be seen as the following equation: Hard voting:

$$C = majorityvoting(C_1, C_2, \ldots, C_n) \tag{6}$$

Soft voting:

$$C = argmaxc \sum i = 1^n P_{i,c} \tag{7}$$

Where:

- C is the final class label predicted by the ensemble classifier
- $C_1, C_2, \ldots, C_n$ are the class labels predicted by the base models in hard voting
- $P_{i,c}$ is the probability estimate for class label c predicted by the ith base model in soft voting
- n is the total number of base models in the ensemble classifier

Ensemble voting classifiers are typically used when it's important to aggregate the predictions of many models to boost a model's accuracy. The ensemble classifier is useful when the basic models have high accuracy but a variety of mistakes occur since it may correct for these faults by aggregating the predictions of the basis models.

Algorithm 1 Ensemble Voting Classifier

1: **Input:** Training data $\mathcal{TD} = (a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)$, base model classifier type T, voting type V (hard or soft)
2: **Output:** Ensemble voting classifier EC
3: Initialize empty list EC
4: **for** $(i = 1)$ to n **do**
5: Train-base model M_i on $\mathcal{D}$ using classifier type T
6: Append M_i to EC
7: **end for**
8: **if** $V = hard$ **then**
9: **return** EC
10: **else**
11: **return** Soft voting function with arguments EC
12: **end if**

Where:

- $\mathcal{TD}$ is the train data, which consists of pairs of input samples a_i and their class labels b_i
- T is the type of base model classifier, such as a decision tree or random forest
- V is the voting type, which can be either hard voting or soft voting
- EC is the ensemble voting classifier, which is a list of base models in the case of hard voting, or a function that performs soft voting in the case of soft voting
- M_i is the ith base model in the ensemble classifier.

II. EXPERIMENTAL SETUP

The suggested approach uses the DT, RF, and XGBoost algorithms as part of an ensemble classification strategy. A number of measures, such as efficiency, ACC, Recall, F1-score, and FPR, were employed to assess the model's performance.

Acc: It is the proportion of occurrences that were correctly categorized to all of the instances. It is a helpful performance metric and is also known as detection accuracy. It is calculated as follows:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

Precision: It calculates the proportion of right classifications to wrong classifications that are punished. It is calculated as follows:

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

Recall: It is the proportion of correctly predicted positive outcomes to all of the actual positive class cases.

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

F1-score: It is a measurement that combine recall and accuracy. It is figured out as the harmonic mean of recall and accuracy. It is calculated as follows:

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{11}$$

False Positive Rate: It is the proportion of inaccurate forecasts to all actual instances of the negative class. It is calculated as follows:

$$FPR = \frac{FP}{TN + FP} \tag{12}$$

False negative rate: It is the ratio of accurate positive class instances to all incorrect negative predictions. It is calculated as follows:

$$FNR = \frac{FN}{TP + FN} \tag{13}$$

III. EXPERIMENTS AND RESULTS

The proposed DRX-base ensemble model was evaluated using the NSL-KDD, UNSW-NB15, and CIC-IDS2017 datasets, three well-known datasets combines datasets are frequently employed to evaluate the efficency of Anomaly detection systems based on machine learning. The datasets were divided into train and test sets, respectively. A number of assessment criteria, such as Acc, Precision, Recall, F1 score, and FPR, were used to assess the suggested technique.

TABLE 7: Pre-Processed NSL-KDD Dataset

Attack Type	Train Data (80%)	Test Data (20%)
Normal	61,604	15,450
DoS	41,330	10,338
Probe	10,300	2,462
R2L	5,506	1,436

A. RESULT ANALYSIS BASED ON NSL-KDD DATASET

After pre-processing, the NSL-KDD dataset was divide into train and test sets. There were 61604 data in the normal class, 41330 records in the DoS class, 10300 records in the probing class, and 5506 records in the R2L class in the trai set. 15450 data in the normal class, 10338 in the DoS category, 2462 in the probe category, and 1436 in the R2L class made up the testing set. Table 7. displays the number of sample in each class.

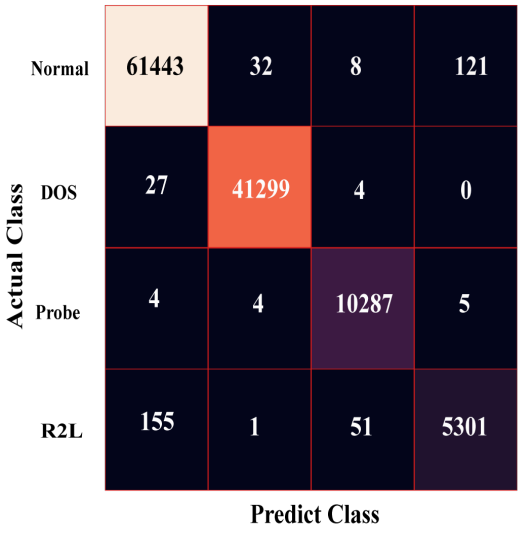

FIGURE 2: Confusion matrices for DRX classifier of NSL-KDD Train Data Set

Figure 2 show confusion matrix (CM) for the DRX classifier on the 80% NSL-KDD training set shows that the classifier correctly classified 61443 samples as normal, 41229 as DoS, 10287 as probe, and 5301 as R2L. In Figure 3 the CM for the DRX classifier on the 20% NSL-KDD testing set shows the classifier correctly classified 15329 samples as normal, 10312 as DoS, 2438 as probe, and 1263 as R2L.

Table 8 shows the result obtained for all class of NSL-KDD dataset after testing the proposed anomaly-based NIDS. The

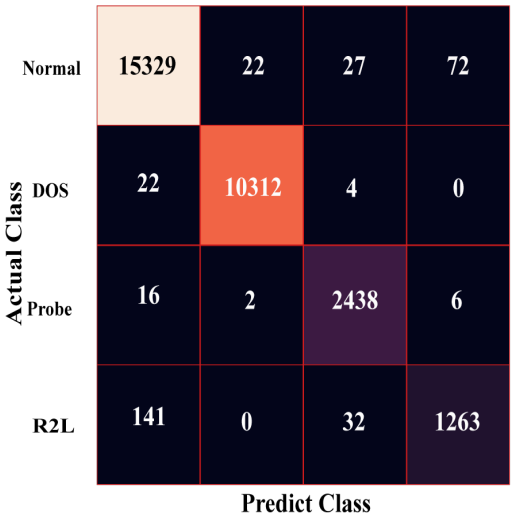

FIGURE 3: Confusion matrices for DRX classifier of NSL-KDD Test Data Set

TABLE 8: Result Analysis for DRX algorithm for 80% Training on NSL-KDD Data Set

Label	ACC	Precision	Recall	F1-Score
Normal	.9969	1	1	1
Dos	.9994	1	1	1
Probe	.9993	.99	1	1
R2L	.9997	.98	.96	.97

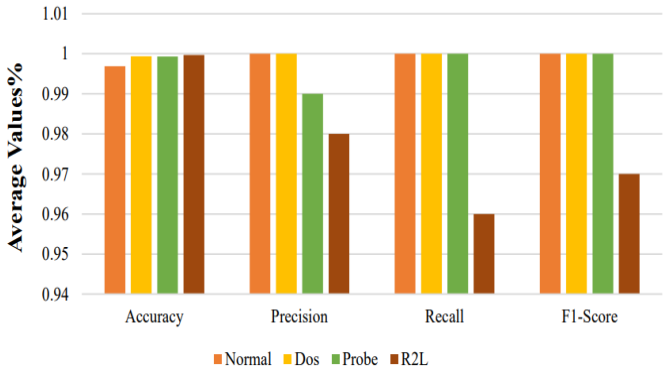

FIGURE 4: Average analysis of DRX classifier using NSL-KDD Train data-set

proposed method achieves 99.68% ACC, 100% Precision 100% Recall, and 100% F1-score for normal samples. For DoS, probe, and R2L samples, the proposed method achieves 99.91%, 99.82%, and 99.10% accuracy, respectively. These results are shown in Figure 4.

The average analysis of the proposed anomaly-based technique DRX classifier using NSL-KDD train dataset shows that proposed scheme receives 99.88% accuracy, 99.25% precision, 99.0% recall, and 99.25% F1-score.

TABLE 9: Result Analysis for DRX algorithm for 20% Testing on NSL-KDD Data Set

Label	ACC	Precision	Recall	F1-Score
Normal	.9993	1	1	1
Dos	.9983	1	1	1
Probe	.9971	.98	.99	.98
R2L	.9915	.94	.88	.91

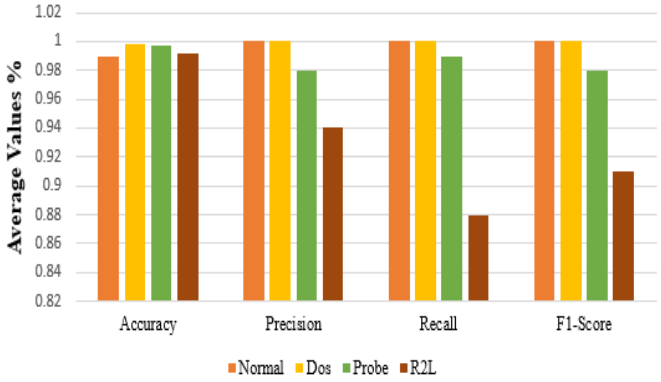

FIGURE 5: Average analysis of DRX classifier using NSL-KDD Test dataset

Table 9 provides the detail about the results obtained for each class of NSL-KDD dataset after testing of the proposed DRX ensemble classifier. The proposed technique attains 98.93% ACC, 100% Precision, 100% Recall and 100% F1-score for normal samples. While it achieves 99.83%, 99.71%, 99.15% accuracy for DoS, probe and R2L samples respectively. These are depicted in Figure 5. The average scores achieve by the proposed DRX classifier are 99.40% accuracy, 98% precision, 96.50% recall, and 98.50% F1-score against 20% test dataset of NSL-KDD.

B. RESULT ANALYSIS BASED ON UNSW-NB15 DATASET

The UNSW dataset is split 80% training and 20% testing datasets after pre-processing. Table 10. displays the number of records in each class after pre-processing. The number of records collected for training purposes against the labels of common is 74387, 47334, 35445, 19415, 13061, 11135, 1861, and 140. While 18613, 11537, 9080, 4831, 3292, 2852, 468, 309, and 30 samples were collected for testing, correspondingly.

The suggested DRX ensemble classifier of UNSW-confusion NB15's matrix versus the 80% training data-set is shown in Figure 6 It show the suggested method, which correctly classifies all records, offers the maximum accuracy against common, generic, exploits, fuzzers, denial-of-service, reconnaissance, and analysis samples. While correctly classifying 1812 and 1200 samples for shellcode and backdoors out of 1861 and 1202 samples, respectively.

TABLE 10: Pre-Process UNSW-NB15 Data Set

Attack Type	Training Data Set (80%)	Test Data Set (20%)
Normal	74,387	18,613
Generic	47,334	11,537
Exploits	35,445	9,080
Fuzzers	19,415	4,831
Dos	13,061	3,292
Reconnaissance	11,135	2,852
Analysis	21,58	519
BackDoor	1,861	468
Shellcode	1,202	309
Worms	140	30

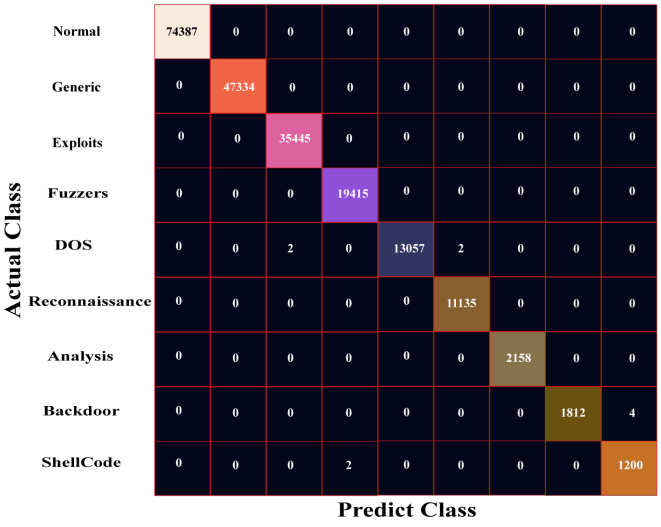

FIGURE 6: Confusion matrices for DRX classifier of UNSW-NB15 Train Data Set

TABLE 11: Result Analysis for DRX algorithm for 80% Training on UNSW-NB15 Data Set

Label	ACC	Precision	Recall	F1-Score
Normal	1.0	1.0	1.0	1.0
Generic	1.0	1.0	1.0	1.0
Exploits	1.0	1.0	1.0	1.0
Fuzzers	1.0	1.0	1.0	1.0
Dos	1.0	1.0	1.0	1.0
Reconnaissance	1.0	1.0	1.0	1.0
Analysis	1.0	1.0	1.0	1.0
Backdoor	0.9999	0.9998	0.9999	0.9997
Shellcode	0.9999	0.9999	0.9999	0.9999

Table 11 shows the results acquired after training of the proposed anomaly-based NIDS for each class of 80% UNSW-NB15 dataset. The proposed scheme achieves 100% accuracy for normal, generic, exploits, fuzzers, DoS, reconnaissance and analysis samples. While it achieves 99.99% ACC for backdoor and shellcode samples. These are depicted in Figure 7. The average score achieve by the proposed methodology is 99.94% ACC, 99.92% Precision, 99.92% Recall and 99.93% F1-score.

The confusion matrix on 20% UNSW-NB15 testing data-set is represented in Figure 8. It shows, the proposed DRX clas-

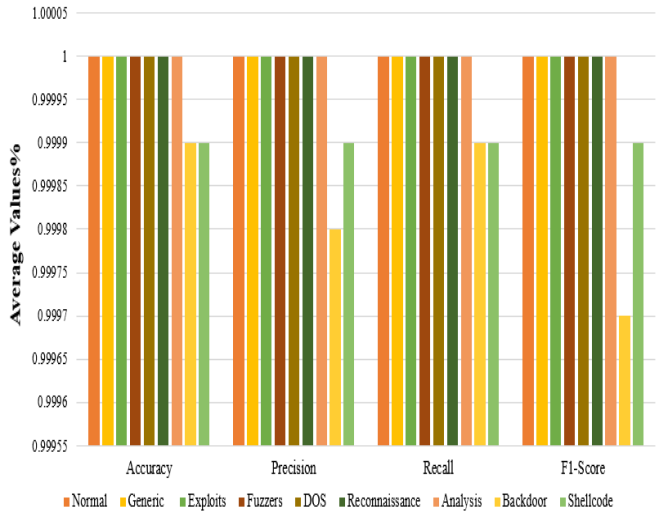

FIGURE 7: Average analysis of DRX classifier using UNSW-NB15 Train dataset

sifier effectively classifies the samples for normal, generic, exploits, fuzzers, DoS and reconnaissance with highest accuracy. While, it correctly classify 518, 465 and 308 out of 519, 468 and 309 respectively for analysis, backdoor and shellcode samples.

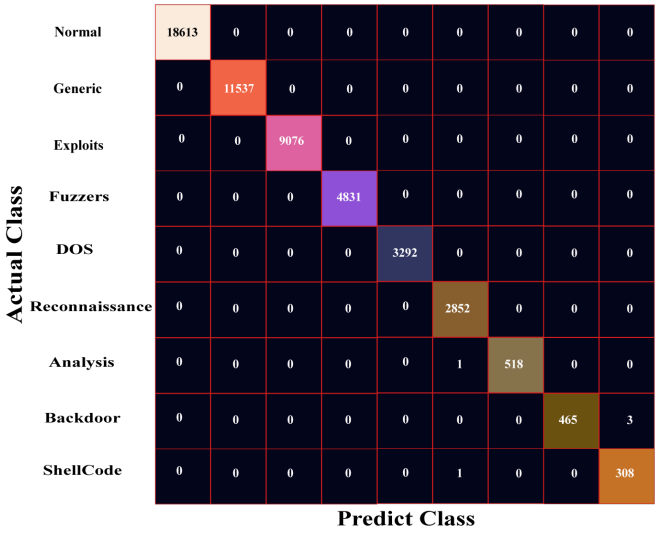

FIGURE 8: Confusion matrices for DRX classifier of UNSW-NB15 Test Data Set

The results receive from apply the proposed DRX ensemble classifier on UNSW-NB15 20% test dataset are shown in 12. It shows that the proposed ML-based technique achieves 100% accuracy for normal, generic, exploits, fuzzers, DoS and reconnaissance samples. While, it achieves 99.99% accuracy for analysis and shellcode samples and 99.98% accuracy for backdoor samples. These are further depicted in Figure 9.

The average score that is achieved by the proposed technique

TABLE 12: Result Analysis for DRX algorithm for 20% Test on UNSW Data Set

Label	ACC	Precision	Recall	F1-Score
Normal	1	1	1	1
Generic	1	1	1	1
Exploits	1	1	1	1
Fuzzers	1	1	1	1
Dos	1	1	1	1
Reconnaissance	1	1	1	1
Analysis-attack	0.9999	0.9998	0.9999	0.9997
Backdoor-attack	0.9998	0.9998	0.9999	0.9998
Shellcode-attack	0.9999	0.9997	0.9998	0.9999

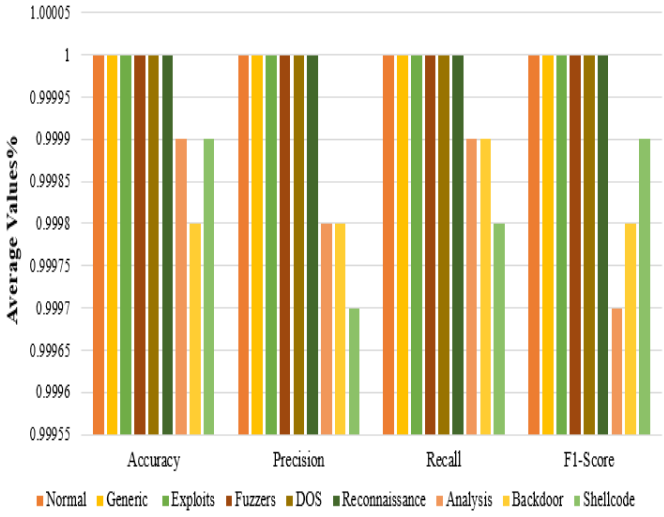

FIGURE 9: Average analysis of DRX classifier using UNSW-NB15 Train dataset

against ACC, Precision, Recall and F1-score are 99.93%, 99.92%, 99.92% and 99.91% respectively.

C. RESULT ANALYSIS BASED ON CIC-IDS2017 DATASET

After pre-process, the CIC-IDS 2017 dataset is divided into 80% train and 20% test data-sets. Number of records after pre-processing in each class is shown in Table 13. The number of records acquired against normal, DoS/DDoS, portscan, brute force, web attack, botnet and infiltration labels are 91607, 49386, 18491, 6223, 1558, 1694, and 29 respectively for training. While, the number of samples obtained for testing are 22902, 12346, 4623, 1556, 389, 424, and 7 respectively.

Figure 10 shows the CM for the CIC-IDS2017 dataset acquired after applying the proposed DRX ensemble for 80% training dataset. It portrays that the proposed ensemble classifier technique provides highest accuracy against normal, web attack, brute force, infiltration, and portscan samples while it classifies 49384 and 1691 samples correctly for DoS/DDoS and botnet ARES out of 49386 and 1694 respectively.

TABLE 13: Pre-Process CIC-IDS 2017 Data Set

Attack Nature	Train Data	Test Data
Normal.	91607	22902
DoS/DDoS	49386	12346
PortScan	18491	4623
Brute Force	6223	1556
Web Attack	1558	389
Botnet ARES	1694	424
Infiltration	29	7

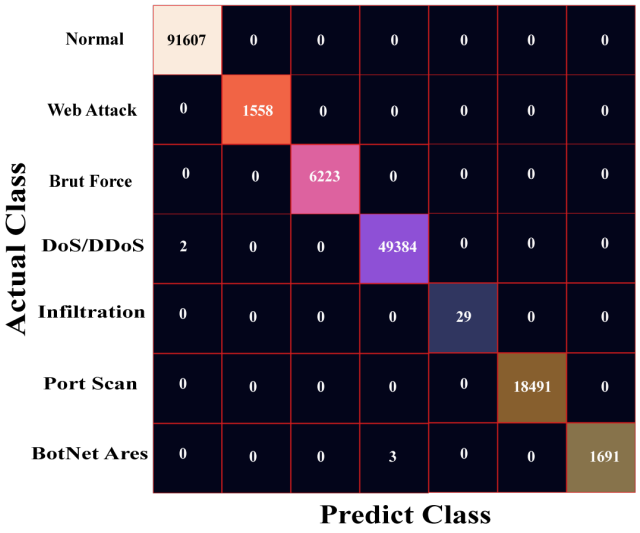

FIGURE 10: Confusion matrices for DRX classifier of CIC-IDS2017 Data Set

Table 14 depicts the results that are acquired for for each class of 80% CIC-IDS2017 training dataset after applying the propose ensemble classifier. Results show that propos scheme achieves 100% accuracy for normal, web attack, brute force, infiltration, and portscan samples. While it accomplishes 99.99% accuracy for the DoS/DDoS and botnet ARES samples. These results are shown in Figure 11. The average score that is achieved by the proposed methodology for accuracy, precision, recall and f1-score is 99.98%, 100%, 100% and 100% respectively.

TABLE 14: Result Analysis for DRX algorithm for 80% Train on CIC-IDS2017 Data Set

Label	Accuracy	Precision	Recall	F1-Score
Normal	1	1	1	1
Web Attack	1	1	1	1
Brute Force	1	1	1	1
DoS/DDoS	99.99	99.99	99.99	99.99
Infiltration	1	1	1	1
PortScan	1	1	1	1
Botnet ARES	99.99	99.99	99.99	99.99

The confusion matrix for the DRX classifier of CIC-IDS2017

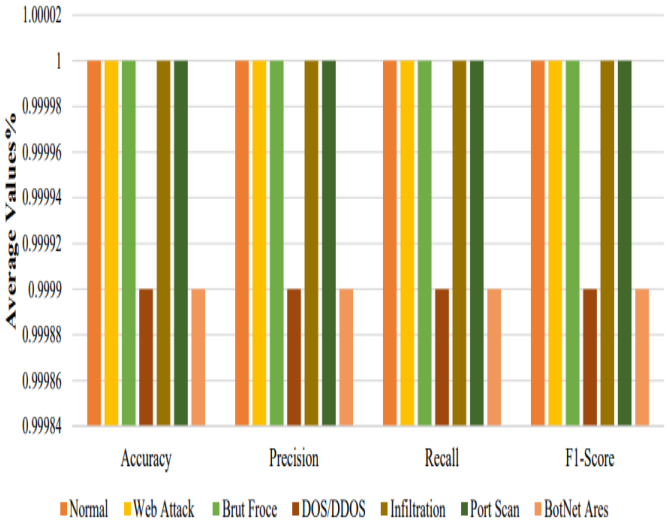

FIGURE 11: Average analysis of DRX classifier using CIC-IDS2017 Train dataset

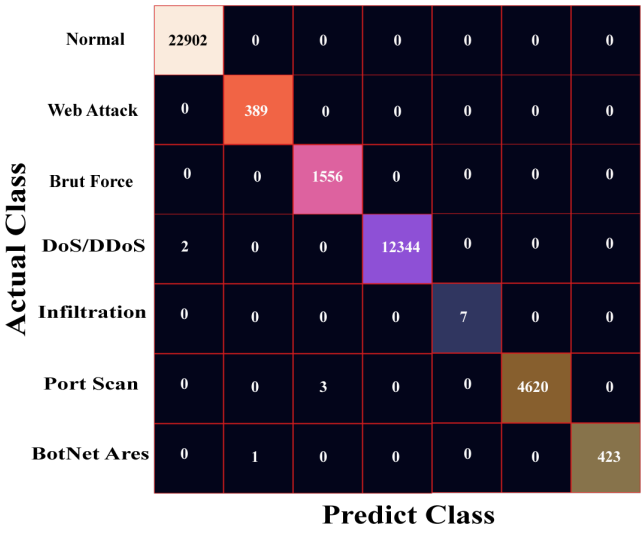

FIGURE 12: Confusion matrices for DRX classifier of CIC-IDS2017 Test Data Set

20% test data-set is depicted in Fig 12 shows that the propose ensemble classifier technique classifies 22902, 389 1556, 12344, 7, 4620 and 423 for normal, web attack, brute force, DoD/DDoS, infiltration, port scan, and botnet ares samples correctly.

Table 15 provides the detail about the results obtain for each class of the CIC-IDS2017 dataset after testing of the propose DRX ensemble classifier. The proposed technique attains 100% accuracy., precision., recall and f1-score for normal, web-attack, brute force, infiltration, and port scan samples. While it achieves 99.99% accuracy, precision, recall and f1-score for DoD/DDoS and botnet ares samples. These results shown in Fig 13.

TABLE 15: Result Analysis for DRX algorithm for 20% Test on CIC-IDS2017 Data Set

Label	Accuracy	Precision	Recall	F1-Score
Normal	1.0	1.0	1.0	1.0
Web Attack	1.0	1.0	1.0	1.0
Brute Force	1.0	1.0	1.0	1.0
DoS/DDoS	99.99	99.99	99.99	99.99
Infiltration	1.0	1.0	1.0	1.0
PortScan	99.99	99.99	99.99	99.99
Botnet ARES	99.99	99.99	99.99	99.99

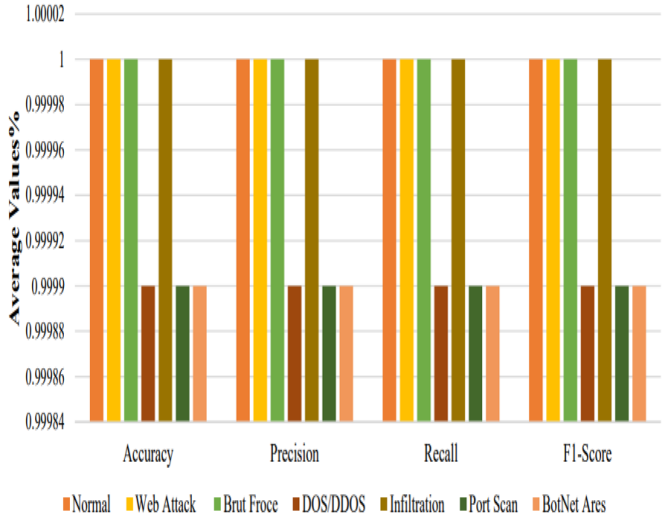

FIGURE 13: Average analysis of DRX classifier using CIC-IDS2017 Test dataset

The average scores achieve by the proposed DRX classifier are 99.97% accuracy, 100% precision, 100% recall, and 100% F1-score against 20% test dataset of CIC-IDS2017.