Detecting Anomaly Intrusions in Digital Network Traffic Using Machine Learning Approach

[1]J. Sebastian Nixon
Department of CSE, SoE
Dayananda Sagar University
Bangalore, India.

[2]Admasu Desalegn
Department of Information Technology
Wolaita Sodo University,
Ethiopia.

[3]J. Jeya A Celin
Department of Information Technology
Kalasalingam Academy of Research and Education,
Krishnankovil, India.

[4]S. Gokulakrishnan
Department of CSE, SoE
Dayananda Sagar University
Bangalore, India.

[5]A. Senthil Kumar
Department of CSE, SoE
Dayananda Sagar University
Bangalore, India.

Abstract:- ***The main target of the Organizations is to secure their network from attacks. This requires network administrators to implement different IDS to monitor network traffic for unauthorized and malicious activities. The detection of malicious activities is two types; i. Misuse IDSs versus ii. Anomaly-based IDSs. Misuse IDS is a signature based IDS which can detect known attacks in an efficient way based on hard coded signatures stored in the signature list. The misuse techniques have the advantage of low false positive rate. However, they suffer from high false negative rate due to the sensitivity to any simple variation in the stored signatures. In such case, the variations can be considered as an attack. Misuse IDS fails in detecting unknown and zero-day attacks where they are unavailable in the stored signatures. Because of this, currently, the focus of many researchers is on anomaly detection to overcome the limitations of sign-based IDSs in detecting new attacks. Artificial intelligence, specifically machine learning methods, has been used to develop an effective data-centric intrusion detection system. In most Anomaly-based approaches, the detection rate is low, the training time is high and the false-alarm-rate (far) is high. To solve these problems, we experimented 3 well-known ML algorithms namely Random Forest, K nearest neighbor & Deep Neural Network and used the UNSW NB15 Network traffic dataset. The innovative findings show that the RF Classifier is better than the alternative methods in detecting the data traffic is normal or attack. This with, RF achieved a classification accuracy of 97.57%, detection rate of 97.53%, and 2.35% with a training time of 8.34 sec in binary classification with a accuracy of classification of 80.76% in ten class classification.***

Keywords: ***Attack, Anomaly Detection, Classification, Intrusion Detection System, Machine learning.***

I. INTRODUCTION

In today's world, due to the increasing quantity of Internet users and online transactions hardly demanding a secure channel, ensuring computer network security has become most important. This is because unauthorized users are the biggest threat to companies. An unauthorized person could be a harmful hacker, ex-staff or one of many third-parties [1]. Security is a prime issue for all the industries, individual users, and

institutions at any time. The field of intrusion detection has received increasing attention in recent years. An IDS is a network security technology designed primarily to detect vulnerability in a computer and a NIDS- Network-Intrusion-Detection System helps system administrators to detect network security breaches. Intrusion Detection is an important and critical part of network security systems [2].

The increase in the number of networked machines has resulted to a growth in unauthorized activity, not only from external attackers, but also from internal attackers [3,4]. A substantial research milestone in the information security area is the IDS. It can quickly identify an intrusion, which may be a continuing intrusion or an intrusion which has just occurred [3]. NIDSs are essential tools for the network system administrators to detect various security breaches inside an organization's network. A NIDS observes and analyzes the going in and out of network devices of an organization and alert if an intrusion is observed. NIDS can be classified into two types: Signature-based NIDS (SNIDS) and Anomaly Detection-based NIDS (ADNIDS), depending on the intrusion detection systems used.

A NIDS observes and analyzes incoming and outgoing network traffic of organization and alerting of any detected intrusions. NIDSs are classified into two categories, SNIDS- Signature based NIDS and ADNIDS- Anomaly Detection based NIDS, based on intrusion detection methods. In systems like SNIDS, such as Snort attack signatures are already programmed into the NIDS. Pattern matching is used to detect network intrusions by comparing the traffic against installed signatures. Alternatively, an ADNIDS identifies network traffic as an intrusion if it detects a deviation from the usual traffic pattern. SNIDS is effective in the detection of known attacks and shows high detection accuracy with less false-alarm rates. However, its found difficult in detection of unknown attacks. Although ADNIDS experienced a high level of false positives, its theoretical potential in the discovery of latest attacks has led to widespread acceptance in the research community [4].

Various ML techniques have been implemented to update ADNIDSs, such as ANN-Artificial Neural Networks, SVM-Support Vector Machines, NB-Naive-Bayesian, RF-Random Forests, and SOM-Self-Organized Maps. The NIDSs are created as classifiers to distinguish between regular traffic and abnormal traffic. Several NIDSs perform feature selection to extract a subset of relevant features from the traffic dataset to enhance classification results. Feature selection helps in the elimination of the possibility of incorrect training through the removal of redundant features and noises [4].

The objective of this paper is to propose solutions to improve the quality of detection of Anomaly-based-IDS using ML techniques. Improving the accuracy of detection on known datasets is not enough to achieve this goal, because the results obtained are not transferable to real networks. Indeed, ML-models learn the traffic of a dataset and not the traffic to be monitored. They need to be re-trained on the monitored network, which is hardly possible as it requires labeled datasets containing attacks on a real network. Since the primary target of any IDS is to scan all the data coming to the network's interface, this approach needs, first capturing all packets that pass through the switch/router and then analyzing these packets to detect a deviation in behavior.

II. LITERATURE REVIEW

IDSs are available in the form of hardware and software systems that monitor events occurred on computers and networks in order to analyze security problems. The number and severity of these attacks has been increasing continuously. Consequently, IDSs are now a vital part of organizations' security infrastructure [2].

A. Common Vulnerabilities and Attacks

Vulnerability is a weakness where you are at risk. Risk is the potential for loss or destruction of assets/data. Attack is the potential violation of the security. Present NIDSs rely on human involvement and the presence of administrators in order to function effectively. So we need ML based tools to secure our network [6].

B. Attack Types

- **Confidentiality**: In such kinds of attacks, the attacker obtain entry to the confidential data.
- **Integrity**: In such kinds of attacks, the attacker can modify the system state and alter the data without proper authorization from the owner.
- **Availability**: In this attacks, the system is either shut down by the attacker or made unavailable to general users. Denial of Service attacks fall into this category.
- **Access Control**: In such attacks the attacker gains full control of the system and can alter the access privileges of the system thereby potentially triggering all of the above three attacks.

C. Attacks detected by a NIDS

The current NIDS can detect several types of attacks. A few of them are : Scanning Attack, Denial of Service (DoS) Attacks, Flaw Exploitation DoS, Attacks, Flooding DoS Attacks, and Penetration Attacks[6].

D. Intrusion Detection System Classification

There are several ways to classify IDSs depending on some criteria, such as by location, information source, detection Model or analysis type, this is shown IN Figure1.

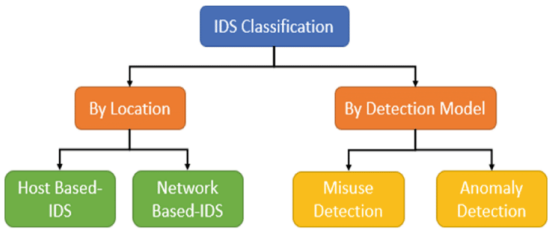

Figure 2.1: IDS_Classification

E. Network-Based IDSs

Most commercial IDSs operate on a network level. These IDSs identify attacks by monitoring and evaluating network traffics. By monitoring a network segment, a single NIDS can analyze the network traffic that impacting various hosts in a network to protect those hosts, by preventing an attack[5]. Network-based IDSs typically include a collection of specialized sensors strategically placed throughout a network.

These devices oversee network data flow, conducting analysis within the local system and reporting attacks to a central management console. As the sensors are limited to running the IDS, they can be more easily secured against attack. Many of these sensors are created to operate "stealth-mode", making it hard for an attacker to identify their location and existence [5]. Figure 2 depicts the design philosophy of a NIDS.

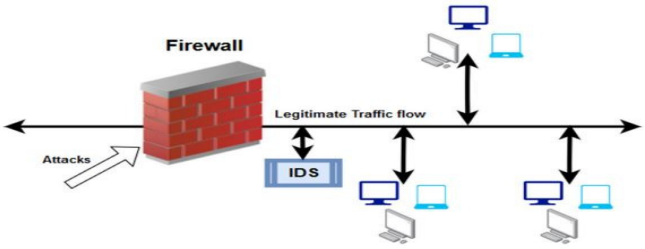

Figure 2.2: The design philosophy of a NIDS

F. Anomaly Detection

Anomaly detectors identify abnormal unusual behavior on a host / network. They operate based on the belief that attacks are distinct from regular

behavior and can be identified by systems that detect these differences. Anomaly detectors create profiles that represent the typical behavior of users, network connections or hosts.

Profiles are built using historical data gathered over a specific period of normal operation. The detectors then collect event data and use a variety of measures to determine when monitored activity deviates from the normal.

The measures and techniques used in anomaly detection include [5]:

- **Threshold detection:** involves expressing specific user and system behavior attributes as counts, with a set level considered acceptable. Characteristics such as the quantity of files a user opens within a specific time frame, the frequency of failed login attempts, the level of CPU consumption by a process, etc..

- **Statistical measures**: can be parametric, assuming a specific distribution for the attributes, or non-parametric, learning the distribution from historical data over time.

- **Rule-based measures**: which are similar to non-parametric statistical measures, behavior patterns are determined by observed data, but unlike them, rules define these patterns, not numeric quantities.

- **Other measures**: which including neural networks, genetic algorithms, and immune system models. The figure 3 shows the Anomaly-based intrusion detection classification.

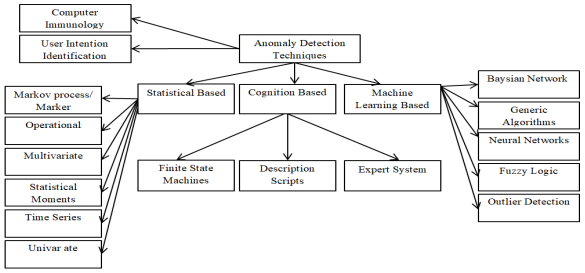

Figure 2.3: Anomaly-based intrusion detection classification.

G. Machine Learning Based Detection Techniques

All computers, keeps track of logs of many activities that users perform on it. For example, there are records of users who log in and how long they are logged in, and there are records of system, application, software updates, etc.

A networked computer keeps records of a lot of additional activities such as records of which user has transferred which files, emails to and from a system and when. Devices that facilitate and control communication activities can keep records of every bit of information that comes from or leave to a machine.

Most activities are normal, but a very small fraction of activities may be outside the realm of what is usual / expected. It is possible that unusual activities are potential intrusions. Such abnormal activities may be found by matching logs and traffic datasets with patterns of known instances of abnormal activities that one has created a priori. However, traffic and log datasets are potentially very large, disparate and ever-growing.

The patterns of intrusion may not be straightforward and easy to find. This is where ideas from machine learning may be able to help in a significant way in fishing out potential intrusion patterns, and provide warnings to network management personnel on time [7].

ML aims to extract valid, useful and meaningful patterns from a dataset, usually large, in a domain by using a nontrivial learning mechanism. A ML algorithm tries to recognize complex patterns in existing datasets to help make intelligent predictions when it encounters new and previously unseen data instances. A fundamental problem that a ML algorithm must address is the problem of generalization.

When learning from data, it is possible that the learner just remembers what it has seen. If this is the case, a learner cannot process data instances it has new. To deal with unseen data, a machine learning program must be cognizant of this possibility and thus, when it learns, must make conscious and diligent efforts to generalize from examples it has seen.
This may necessitate estimating a random distribution for what is acceptable and what is not. It may involve being able to detect data which are sufficiently similar to and dissimilar from seen examples, usually using some mathematical approaches.

ML algorithms can be primarily categorized into two types: **Supervised** and **Unsupervised**. A supervised algorithm requires a training dataset where data instances are labeled. This labeling is usually performed by humans and is expensive because it requires experts in relevant domains to spend precious time doing the labeling. An example of a label may be the class or sub-group a data instance belongs to. An unsupervised algorithm usually attempts to group data instances into categories or sub-groups based on inherent properties contained in them, using a measure or metric to compute similarity between an arbitrary pair of data instances. Alternatively, one can compute dissimilarity or distance between each pair of occurrences as well [7].

III RESEARCH METHODOLOGY

3.1 Proposed Architecture

ML solutions are useful in solving a wide range of problems, Data generation and collection, training, and evaluation are must-haves, but we may need domain-specific components too. Based upon these, understanding of data, for data class imbalance problem we use Synthetic-Minority-Oversampling-Technique [SMOTE]. This basis for method resides in the idea of oversampling the minority class by generating synthetic instances from its elements and keeping the majority number as it is.

In ML, there is different algorithm that is used for training data, these categorized into 1. Supervised, 2.Unsupervised & 3.Reinforcement Learning. For conducting our research we used Supervised Learning, Classification algorithms, to detect computer network intrusions by identifying the behavior of the Normal and Malicious user profile. In the proposed model, to detect intrusions, there are two mains phases which were done: Training and Testing.

The observations in the training set create the experience that the algorithm used to learn. The testing set is a set of observations that are not used or included in the training set, which is used to asses and validates the effectiveness of the model using the evaluation metrics. This proposed model can improve the detection performance of the IDS. Figure 3.1 shows the various components of the proposed model.

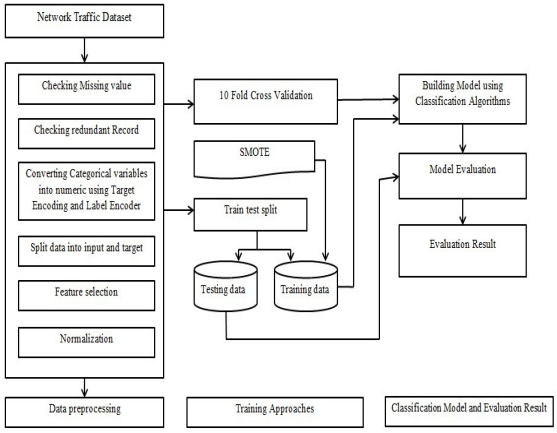

Figure 3.1: System Architecture

3.2 Data Collection

The Australian-Center-for-Cyber Security (ACCS) in collaboration with a number of researchers worldwide took the initiative and creates the UNSW-NB-15-dataset. The IXIA Perfect Storm tool was used to create a diverse mix of both regular and unusual network traffic [9]. Table 3.1 shows the Description of The Unsw-Nb-15-Data Set.

Table 3. 1 A Part of the UNSW-NB15-Data Set Distribution

Category	Training set	Testing set
Normal	56,000	37,000
Analysis	2,000	677
Backdoor	1,746	583
Dos	12,264	4,089
Exploits	33,393	11,132
Fuzzers	18,184	6,062
Generic	40,000	18,871
Reconnaissance	10,491	3,496
Shellcode	1,133	378
Worms	130	44
Total Records	**175,341**	**82,332**

Table 3. 2 Description of attack categories of UNSW-NB-15-dataset

Traffic Type	Description
Normal	Traffic that contains no threat
Analysis	Generic type for describing port scanning, spam and html file penetration
Backdoor	Malware type that negates normal authentication to grant remote access to resources such as databases and file servers
Dos	Deprives legitimate users from using web services through flooding the network/server with invalid authentication attempts forcing it to crash or stall
Exploits	Code that take advantage of a software vulnerability or security flaw. Often incorporated into malware, allowing a fairly easy and fast propagation.
Fuzzers	Automated process of finding 'hack able' software bugs by randomly feeding different permutations of data into a target program until one of those permutations reveals a vulnerability
Generic	Collision attack on the secret keys of ciphers. Works against all block ciphers.
Reconnaissance	Collection of simple techniques that gather information about the target network/server, such as nmap.
Shellcode	Set of instructions/statements which are injected and executed by a flawed program. Directly manipulates registers and the functions of a program
Worms	Malicious self-replicating code. Consumes too much system memory

Features are categorized into six groups:

1. **Flow features**: Includes the protocol used between hosts.

2. **Basic features**: involves the attributes that represent protocols connections (i.e. session characteristics).

3. **Content features:** encapsulates the attributes of TCP protocol, also they contain some attributes of http services.

4. **Time features**: contains the attributes time, for example, arrival time between packets(jitter), start/end packet time, and round trip time of TCP protocol

5. **Additional features**: this category can be further divided into two groups: general purpose features (i.e., 3640), whereby each feature has its own purpose, according to protect the service of protocols, and (2) connection features (i.e., 4147) that are constructed using 100 data connections following the sequential order of the latest timestamp.

6. **Labeled features**: used for class identification (i.e. label) the data set. Two attributes were provided: attack-category represents the nine categories of the attack and the normal, and label is 0 for normal and 1 otherwise.

Table 3. 3 Features of UNSW-NB15 Dataset

No.	Name	Features	No.	Name	Features
1	Dur	Basic	23	Dwin	Content
2	Proto	Flow	24	Tcprtt	Time
3	Service	Basic	25	Synack	Time
4	State	Basic	26	Ackdat	Time
5	Spkts	Basic	27	Smean	Content
6	Dpkts	Basic	28	Dmean	Content
7	Sbytes	Basic	29	trans-depth	Content
8	Dbytes	Basic	30	response_body_len	Content
9	Rate	Basic	31	ct_srv_src	Connection
10	Sttl	Basic	32	ct_state_ttl	General
11	Dttl	Basic	33	ct_dst_ltm	Connection
12	Sload	Basic	34	ct_src_dport_ltm	Connection
13	Dload	Basic	35	ct_dst_sport_ltm	Connection
14	Sloss	Basic	36	ct_dst_src_ltm	Connection
15	Dloss	Basic	37	is_ftp_login	General
16	Sinpkt	Time	38	ct_ftp_cmd	General
17	Dinpkt	Time	39	ct_flw_http_mthd	General
18	Sjit	Time	40	ct_src_ltm	Connection
19	Djit	Time	41	ct_srv_dst	Connection
20	Swin	Content	42	is_sm_ips_ports	General
21	Stcpb	Content	43	label	Labeled
22	Dtcpb	Content	44	attack_cat	Labeled

3.3 Split Input and Target Variable

In machine learning, the concept of dependent & independent variables is important to understand. After we convert categorical data into a numeric using Target encoding and label encoder method, in order to learning algorithms to learn, it is important to split input data and target (output variable) data in supervised machine learning. Without differentiating a target (class label) in supervised ML algorithms, would be unable to map available data to outcomes.

This because, supervised ML uses historical-data, to learn patterns, and uncover-relationships between other features of the input and the target data. In order to work with these data forms, all input features need to be merged into one vector. This, (X) as an input (independent variable) and (y) as dependent or target. This task includes making predictions in the future (y) based on new input variables (X) for learning cases.

3.4 Resampling Technique Used

In this study, SMOTE is used to select records for the experimental analysis. Dealing with uneven categories in a dataset can be easily done using this method. Due to the challenge of creating a uniform balanced dataset, preprocessing the current dataset needs to start by either decreasing the majority, increasing the minority, or both [11].

The basis of the SMOTE method resides in the idea of over collecting the minority class by generating synthetic instances from its elements and keeping the majority number as is. The newly acquired samples are not just exact copies of the minority samples, but are actually made by combining characteristics from the minority cases and their closest neighbors in the feature space [11].

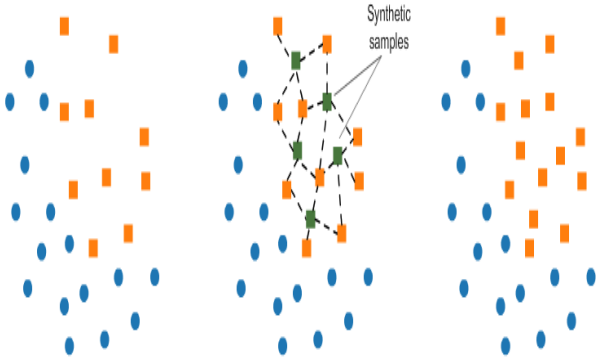

Figure 3.2 This How SMOTE works

3.5 Train-Test Split

The observations in the training set create the experience that the algorithm used to learn. In order to estimate the performance on new data, we need to have a set of unseen data.

This can be done by splitting the data into training &testing. The training data is used to predict correctly or if not correct itself to predict correctly next time [8]. The accuracy will be decided by the testing. In this experiment we used 80% for training & remaining 20% for testing as shown in the table 3.4.

Table 3. 4 Input dataset split

Input Features(X)	Target Class(Y)	% Split
X-training	Y-training	80%
X-test	Y-test	20%

3.6Training Data with Classifier Algorithms

In classification phase utilize the data received from the previous phase for detecting whether the normal packet or attack packet. Depending on feature values, our work is to design a NIDS with the different supervised machine learning classifiers namely Random Forest [RF], K-Nearest Neighbor, Deep Neural Network in intrusion detection.

The main reason for the selection of those three algorithms is:

- ✓ Being able to work very efficiently on large datasets that include many features.
- ✓ The ability for both binary & multi-class classification problem.
- ✓ According to Literature review those algorithms are preferred using this complex pattern dataset.

- ✓ The ease of understanding the results of the model on the nature of data.

The algorithms are selected for reason explained here. Firstly, RF is a classifier made up of multiple models working together, contains numerous decision trees and outputs the class. That means, the mode of the class's output by individual trees, Accuracy and variable importance generated automatically, over fitting is not a problem, not very sensitive to outliers in training data and easy to set parameters. Secondly, the KNN classifier algorithm is chosen.

It is among the simple forms of ML algorithms used for classification. It categorizes data points by considering the classification of neighboring data points and then assigns new data points based on their resemblance to existing data points. Finally, we used DNN Algorithm techniques because of many researchers used this algorithm in the areas computer network intrusion detection for both binary classification and multiclass classification. One of the basic targets of the data analytics is to compare different models and to select the better classification accuracy accordingly. Those classifiers are discussed below:

A. *Model Performance Evaluation*

In this section, the performance evaluation of the proposed machine learning algorithms towards detecting network intrusion have been discussed. Two validation techniques have also been used. These are, the tenfold cross-validation technique and by applying a separate test data on the generated classification model in order to estimate the effectiveness of the algorithms when they face a new dataset. This task as the main objective of the research was proceeded by performing the comparison of algorithms mentioned.

To evaluate and compare the performance, the well-known machine learning tool; Python, implementation is be done in Anaconda environment using jupyter notebook was used and five performance evaluation metrics have been considered for experimental comparison. These are classification accuracy, recall, precision, F-measure, confusion matrix.

B. *Normalization*

Normalization is a technique often applied as part of data preparation for machine learning. To make the network learn easily, the data fed to the algorithm should take small values (in the range 0-1), all features should take values in the same range. In this research, Min-max Scaler technique is used to normalize the features. The issue is which one to choose to normalize our data. The best practice is to pick it empirically (i.e. the technique that gives better result it is the suitable to the nature of our data). Since that Min-Max technique preserves the relationships among the original data values [10], it has been utilized to normalize USNW-NB15 dataset features. This method performs a linear conversion on the original data. The normalized feature is given by:

$$z = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{3.1}$$

C. *Classification Accuracy*

The classification accuracy is a most commonly used measures for the classification performance, and it is defined as a ratio between the correctly

classified instances to the overall number of occurrences frequently presented as a ratio with 100% as the maximum performance an algorithm can reach [9]. Mathematically, it can be put as in equation.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (3.2)$$

Where;

True Positive (TP): denotes the Number of instances, where the exact label is positive (attack) and whose class is correctly predicted to be positive (attack).

True Negative (TN): denotes the Number of instances, where the exact label is negative (normal) and whose class is correctly predicted to be negative (normal).

False Negative (FN): specify the Number of instances, where the exact label is positive (attack) and whose class is incorrectly predicted to be negative (normal).

False Positive (FP): points the number of instances, where the true label is negative (normal) and whose class is incorrectly predicted to be positive (attack).

D. Confusion Matrix

In binary classification, the result is often displayed as a two dimensional confusion matrix with a row and column for each class. Each matrix element represents the quantity of test examples for which the actual class is the row and the predicted class is the column.

Table 3. 5 Confusion Matrix for IDS System

	Predicted Class		
Actual Class		Normal	Attack
	Normal	TN	FP(False alarm)
	Attack	FN	TP

The diagonal elements represent the two types of correct classification (TP and TN) and the diagonal elements represent the two types of error (FP and FN).

E. F-measure or F-score

The F-measure, which is also known as F-score is the harmonic mean of the precision & recall, it measures the effectiveness of prediction when just as much importance is given to recall as to precision. That is it tries to get balance between the two and it is calculated using equation.

F1 - score of a model is calculated as:

$$\text{Fl score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3.3)$$

Where, "Precision" is the ratio of accurate positive results to positive results predicted by the classifier, while "Recall" is the ratio of accurate positive results to all relevant samples and all samples that should have been recognized as positive as shown in below.

$$Precision = \frac{TP}{TP + FP} \quad (3.4)$$

$$Recall = \frac{TP}{TP + FN} \quad (3.5)$$

IV. EXPERIMENTATION AND RESULTS

In this chapter the comparative performance analysis of three different ML algorithms used for intrusion detection is discussed. The dataset size and for class imbalance problem the resampling technique used described in this Section. For algorithm, with the two test techniques percentage split and cross-validation are used in binary classification. For multi class classification we used percentage split method only. The results of the two validation techniques on algorithms were recorded with respect to their processing time and performance measurement parameters. Beside neural network classifier, each algorithm model trained with ML tools using the default configuration parameters. Conclusively, the best model for intrusion detection proposed.

A. *Experimental Setup*

To implement the proposed system the following software tools are used. Classification models are implemented using python, jupyter note book.

The modules that used for our implementation are the following:

- ✓ **Pandas, NumPy, Scikit-learn, Matplotlib , Imbalanced-learn, Tensor flow**
- ✓ Dataset Used : UNSW-NB15 intrusion detection network dataset

B. *Implementation Details*

Primarily, we import necessary libraries into our python anaconda jupyter note book working environment and data acquisition followed next. For data acquisition, we only use 61738 records for experimentation from the existing UNSW-NB15 observation training dataset. This because, based on understanding on data, we are going to address the issue of class imbalance by using SMOTE on UNSW-NB15 network data. In this, using SMOTE on total UNSW-NB15 training dataset is highly computationally expensive because when we apply SMOTE on the training data, it expands the data generation area by creating synthetic samples that generated for the minority class. The following shows our sample codes and the corresponding outputs.

Read the Data and Analyzing

```
# Loading our data
Data = pd.read_csv('Network_Traffic_Data.csv')
#Print the first 5 rows of the data
Data.head(5)
```

	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	...	ct_dst_sport_ltm	ct_dst_src_ltm	is_ftp_login	ct_ftp_cmd	ct_flw_http_mth
0	1	0.000011	udp	-	INT	2	0	496	0	90909.0902	...	1	2	0	0	
1	2	0.000008	udp	-	INT	2	0	1762	0	125000.0003	...	1	2	0	0	
2	3	0.000005	udp	-	INT	2	0	1068	0	200000.0051	...	1	3	0	0	
3	4	0.000006	udp	-	INT	2	0	900	0	166666.6608	...	1	3	0	0	
4	5	0.000010	udp	-	INT	2	0	2126	0	100000.0025	...	1	3	0	0	

5 rows × 45 columns

Figure 4.1 loaded UNSW-NB15 network traffic data

```
Data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 61738 entries, 0 to 61737
Data columns (total 45 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 61738 non-null  int64
 1   dur                61738 non-null  float64
 2   proto              61738 non-null  object
 3   service            61738 non-null  object
 4   state              61738 non-null  object
 5   spkts              61738 non-null  int64
 6   dpkts              61738 non-null  int64
 7   sbytes             61738 non-null  int64
 8   dbytes             61738 non-null  int64
 9   rate               61738 non-null  float64
 10  sttl               61738 non-null  int64
 11  dttl               61738 non-null  int64
 12  sload              61738 non-null  float64
 13  dload              61738 non-null  float64
 14  sloss              61738 non-null  int64
 15  dloss              61738 non-null  int64
 16  sinpkt             61738 non-null  float64
 17  dinpkt             61738 non-null  float64
 18  sjit               61738 non-null  float64
 19  djit               61738 non-null  float64
 20  swin               61738 non-null  int64
 21  stcpb              61738 non-null  int64
 22  dtcpb              61738 non-null  int64
 23  dwin               61738 non-null  int64
 24  tcprtt             61738 non-null  float64
 25  synack             61738 non-null  float64
 26  ackdat             61738 non-null  float64
 27  smean              61738 non-null  int64
 28  dmean              61738 non-null  int64
 29  trans_depth        61738 non-null  int64
 30  response_body_len  61738 non-null  int64
 31  ct_srv_src         61738 non-null  int64
 32  ct_state_ttl       61738 non-null  int64
 33  ct_dst_ltm         61738 non-null  int64
 34  ct_src_dport_ltm   61738 non-null  int64
 35  ct_dst_sport_ltm   61738 non-null  int64
 36  ct_dst_src_ltm     61738 non-null  int64
 37  is_ftp_login       61738 non-null  int64
 38  ct_ftp_cmd         61738 non-null  int64
 39  ct_flw_http_mthd   61738 non-null  int64
 40  ct_src_ltm         61738 non-null  int64
 41  ct_srv_dst         61738 non-null  int64
 42  is_sm_ips_ports    61738 non-null  int64
 43  attack_cat         61738 non-null  object
 44  label              61738 non-null  int64
dtypes: float64(11), int64(30), object(4)
memory usage: 21.2+ MB
```

Figure 4.2 Data type of UNSW-NB15 dataset column

As we can see here, on this Dataset, four columns, proto, service, state and attack_cat, are is the categorical feature as it is represented by the **object data type** and the rest of them are numerical features as they are represented by *int64 and float64*. So, in order to convert those categorical columns into numeric, we used Target encoding technique for 'proto','state','service' those which at the side of independent variable and we convert attack_cat column into numeric form using Label encoder as it is dependent variable for multi class classification purpose. In this Label encoding technique, each label is assigned a unique integer based on alphabetical ordering.

C. *Split Independent and Dependent Variable*

We have to differentiate dependent and independent variables why because without differentiating a target (class label) in supervised ML algorithms would be unable to map available data to outcomes.

All input features need to be merged into a unique vector (X) and (Y) in order to work with data forms, This, (X) as an input and (y) as dependent or target. This is a general understanding that we want to make predictions about future occurrences (Y) based on new instances of input factors (X). In this paper, X refers individual feature of inputs in classification, and Y defines attack class Normal data and Malicious attacks. This is can be done, by dropping class label from total features of the dataset. Finally, the final output of the Vector Assembler (X) and (Y) is given as a training data to a ML algorithm.

In this, we have two target or dependent variables, in our UNSW-NB15 Dataset. Those are '**attack_cat'** and '**label**' columns. The 'label' contains 'attack' and 'normal' which represented by **0** and **1** respectively, while the '**attack_cat'** column is for multi class classification, this contain Ten class, Normal, Analysis, Reconnaissance, Generic, Exploits, Fuzzers, DoS, Backdoor, Shell code, Worms. We use those class labels for two different experiments.

D. *Feature Selection*

The SFS approach starts with an empty set of features. The SFS-Random Forest Feature Selection method is a method for searching all the features and has the advantage of accurately determining the importance of the features. This makes it possible to eliminate unnecessary and duplicate features. The features contributing to the classification accuracy are expected to be capable of excluding unnecessary features.

Step Forward Feature Selection (SFS)

```
from mlxtend.feature_selection import SequentialFeatureSelector as SFS

sfs = SFS(RandomForestClassifier(n_estimators=100, random_state=0, n_jobs =-1),
        k_features = (1,42),
        forward = True,
         floating = False,
         verbose = 2,
        scoring = 'accuracy',
        cv = 4,
        n_jobs =-1
       ).fit(X_train, y_train)
```

Figure 4. 3 Feature selection Method used

After these wrapper based step FFSD is performed on the 42 features, only 21 most significant features are selected with accuracy score 97.13 %.

```
# this about the best score feature names
sfs.k_feature_names_
('proto',
 'service',
 'spkts',
 'sbytes',
 'dbytes',
 'rate',
 'sttl',
 'dttl',
 'swin',
 'smean',
 'dmean',
 'trans_depth',
 'response_body_len',
 'ct_srv_src',
 'ct_dst_ltm',
 'ct_src_dport_ltm',
 'ct_dst_sport_ltm',
 'ct_dst_src_ltm',
 'ct_flw_http_mthd',
 'ct_src_ltm',
 'ct_srv_dst')

#this about the best score
sfs.k_score_*100
97.13504864163210
```

Figure 4.4 Best Score Feature

E. Data Normalization

In this paper we used Min-max Scaler technique is used to normalize the features. The issue is which one to choose to normalize our data.

The best practice is to pick it empirically (i.e. the technique that gives better result it is the suitable to the nature of our data). Since that Min-Max technique preserves the relationships among the original data values, it has been used to normalize USNW-NB15 dataset features.

Feature Scaling (Using MinMaxScaler)

```
from sklearn.preprocessing import MinMaxScaler

min_max= MinMaxScaler()

X= pd.DataFrame(min_max.fit_transform(X),columns=X.columns)

X.head()
```

	proto	service	spkts	sbytes	dbytes	rate	sttl	dttl	swin	smean	...	trans_depth	response_body_len	ct_srv_src	ct_dst_ltm	ct_src_d
0	0.81728	0.536117	0.000094	0.000033	0.0	0.090909	0.996078	0.0	0.0	0.151351	...	0.0	0.0	0.016129	0.000000	
1	0.81728	0.536117	0.000094	0.000121	0.0	0.125000	0.996078	0.0	0.0	0.579054	...	0.0	0.0	0.016129	0.000000	
2	0.81728	0.536117	0.000094	0.000073	0.0	0.200000	0.996078	0.0	0.0	0.344595	...	0.0	0.0	0.032258	0.000000	
3	0.81728	0.536117	0.000094	0.000061	0.0	0.166667	0.996078	0.0	0.0	0.287838	...	0.0	0.0	0.032258	0.017241	
4	0.81728	0.536117	0.000094	0.000146	0.0	0.100000	0.996078	0.0	0.0	0.702027	...	0.0	0.0	0.032258	0.017241	

5 rows × 21 columns

Figure 4.5 Feature Scaling method used

SMOTE Resampling techique used

```
from imblearn.over_sampling import SMOTE

# SMOTE for percentage split method
sm = SMOTE(random_state=42)
X_res, y_res = sm.fit_resample(X_train.astype('float'), y_train)

from collections import Counter
print("Before SMOTE:", Counter(y_train))
print("After SMOTE: ",Counter(y_res))

Before SMOTE: Counter({1: 34454, 0: 14936})
After SMOTE:  Counter({0: 34454, 1: 34454})
```

Figure 4.6 SMOTE Technique used for binary classification

F. Model Performance Evaluation on Random Forest Classifier

As discussed in section 3.5, Random Forest Classifier is, it is a supervised ensemble classifier that contains numerous decision trees and outputs a class. That is, the way of the class's output by individual trees, Accuracy and variable importance generated automatically, over fitting is not a problem, not very sensitive to outliers in training data and also easy to set parameters. It makes use of standard divide and conquers approach to improve the performance. The experimentation of this model has been done by employing two train and test approaches, which are percentage split and 10-fold cross-validation approach. Table 4.1 presents the result of Random Forest experimentation.

Table 4.1: Performance metrics of RF classifier

Train and test Approach	Result					
	Time to build model	Time to test model	Recall	Precision	F-measure	Accuracy
Percentage split (80:20)	8.34 sec	0.27 sec	97.53 %	98.96%	98.23%	97.57%
10 –fold cross validation	49.63 sec	50.42 sec	96.06%	96.2%	96.12%	94.64%

The highest classification accuracy that achieved in this experiment is 97.57%, from which developed applying Percentage split method. This method also achieves better precision recall, and F-Measure results. The time taken to execute the model is 8.34 seconds and it is 41.29 second less than the model built by Cross Validation which is 49.63 seconds. In general, we can say that the model built by percentage split approach is measured a better model than Cross Validation.

Table 4.2 Confusion matrix for Random forest classifier

Train and test Approach	Correctly Classified		Incorrectly Classified	
	TP	TN	FP	FN
Percentage split(80:20)	8399	3649	88	212
10-Fold Cross Validation	41370	17042	1631	1695

The confusion matrix for Random Forest classifiers in table 4.2 shows that percentage split method achieves relatively very small incorrectly classified instances which are 88 False Positive and 212 False Negative result from the total 12,348 instances.

G. Model Performance Evaluation on K-Nearest Neighbor Classifier

These algorithms use the relative distance and relative density to find top N neighbors. The key assumption with nearest neighbor algorithms is that normal points have close neighbors while anomalies are located far from other points. K-Nearest Neighbor models also were developed using the two train test approaches. The models are presented in table below 4.3.

Table 4.3 performance metrics of k-Nearest Neighbor Algorithm

Train and test Approach	Result					
	Time to build model	Time to test model	Recall	Precision	F-measure	Accuracy
Percentage split (80:20)	0.011 sec	32.33 sec	93.99%	98.22%	96.05%	94.63%
10 –fold cross validation	112.96 sec	114.59 sec	94.14%	94.63%	94.38%	92.19%

The highest classification accuracy that achieved in this experiment is 94.63%, from which developed applying. The time taken is 0.011 seconds and it is 112.94 second less than the model built by Cross Validation which is 112.96 seconds.

Also, in this experiment, the model built by percentage split approach is measured a better model than Cross Validation.

Table 4.4 Confusion matrix for K-Nearest Neighbor Algorithm

Train and test Approach	Correctly Classified		Incorrectly Classified	
	TP	TN	FP	FN
Percentage split(80:20)	8094	3591	146	517
10-Fold Cross Validation	40543	16374	2299	2522

H. Model Performance Evaluation on Deep Neural Network Classifier

Keras is a deep learning API written in Python, running on top of the machine learning

platform Tensor Flow. The complete data structures of Keras are **layers** and **models**.

The basic type of model is the Sequential model, a linear stack of layers. After doing some experiments in order to identify the best network's setting such as the hyper-parameters, we end up with hyper-parameter set as shown in the tables 4.5 for binary classification using percentage split method.

Table 4.5 Network configuration in binary classification

Hyper Parameters	Setting
Hidden layer	4
Neurons	100
Activation function in input and hidden layer	Relu
Activation function in the output layer	Sigmoid
Optimizer	Adam
Batch size	100
Epochs	300
Loss Function	Binary_crossentropy
Kernel- initializer	random_uniform

Table 4.6 Binary classification DNN results

ANN	Metrics				
	Accuracy	Precision	Recall	FPR	FNR
	95.01%	97.34 %	95.45%	5.99%	4.54%

Table 4.7 Confusion matrix for DNN Classifier Algorithm

Train and test Approach	Correctly Classified		Incorrectly Classified	
	TP	TN	FP	FN
Percentage split (80:20)	8220	3513	224	391

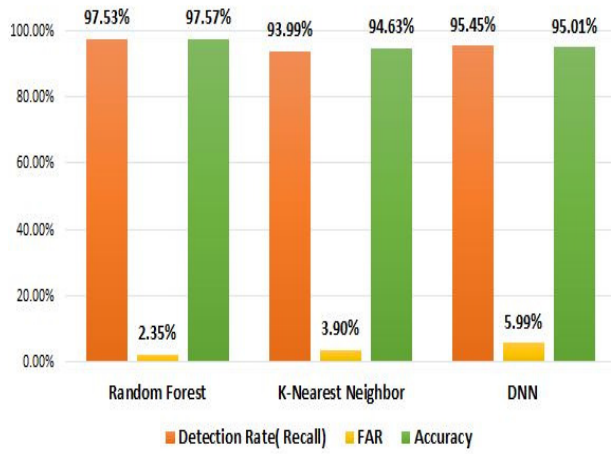

Figure 4.7 Graph comparing Detection rate (Recall), FAR and Accuracy of three Classifiers using percentage split method

I. Ten Class Classification

In Ten class classification, the classification performance result shown in the classification accuracy for the different algorithms. In this, the accuracy of the RF is comparatively higher than the other two classifiers. The overall accuracy of 80.76 %, 73.97% and 76.05% is gained by RF, KNN and DNN respectively.

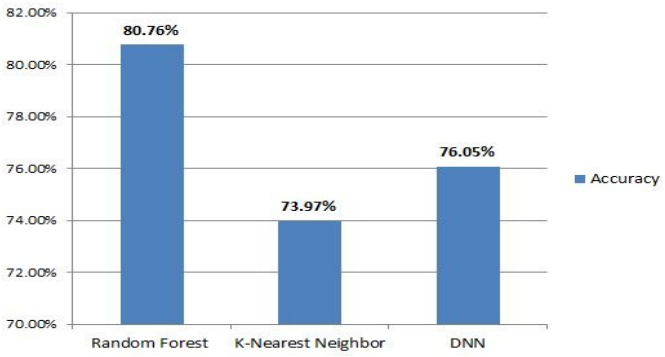

Figure 4.8 Graph Shows the percentage accuracy of three classifiers on 10-class

For multi (Ten) class classification, For Neural Network Classifier, we use hyper parameter set as shown in below table 4.8.

Table 4.8 Network configuration in Ten classification

Hyper Parameters	Setting
Hidden layer	4
Neurons	100
Activation function in the hidden layer	Relu
Activation function in the output layer	Softmax
Optimizer	Adam
Batch size	100
Epochs	300
Loss Function	categorical_crossentropy
Kernel- initializer	random_uniform

The detection performance of the three classifiers on the Ten-class is presented in Table 4.9. This table shows the detection rate and FAR results of the four models for the nine attack classes: Analysis, Exploits, Backdoor, Fuzzers, Dos, Reconnaissance, Generic, Shellcode and Worms. The findings show that the classification models performed very well in the detection of Generic attacks.

Table 4.9 The Detection and false positive rate result of the three algorithms

Classifier	Metrics	Analysis	Backdoor	Dos	Exploits	Fuzzy	Generic	Reconn aissance	Shell code	Worms
RF	DR	0.37	0.34	0.66	0.60	0.74	0.97	0.84	0.85	0.81
	FP	1.69%	2.3%	6.58%	3.88%	3.42%	0.08%	0.58%	0.94%	0.16%
KNN	DR	0.37	0.28	0.57	0.55	0.70	0.95	0.75	0.59	0.80
	FP	3.37%	2.79%	7.21%	3.96%	0.94%	0.22%	1.66%	1.98%	0.69%
DNN	DR	0.66	0.20	0.53	0.49	0.59	0.95	0.82	0.92	0.76
	FP	6.9%	1.98%	7.0%	1.64%	1.6%	0.14%	0.51%	0.74%	0.69%

J. Comparison of Results

The collected dataset has been preprocessed and selected relevant features using wrapper based step forward feature selection method. In this we used from total 42 features only 21 most significant features were selected. We suggested an oversampling method in that the minority class is over-sampled using SMOTE technique.

V. CONCLUSION

In this research, we implement the following ML approaches using the reduced feature space: RF, KNN, DNN. In our test, we look into both the multiclass and binary classification configurations. The results shows that the RF worked efficiently compared to the other methods in predicting the class of unseen packets especially in terms of accuracy, recall and false alarm rate. RF scored classification accuracy of 97.67%, detection rate of 97.53%, & 2.35% false alarm rate. KNN scored classification accuracy of 94.63%, detection rate of 93.99% and with 3.9% false alarm rate and DNN scored classification accuracy of 95.01%, detection rate of 95.45%, and with 5.99% false-alarm-rate (far) in binary classification. For multi class classification, the accuracy score of RF, KNN, and DNN is 80.76 %, 73.97%, 76.05% respectively. The evaluation result in binary classification shows the proposed approach acts effectively to detect network intrusions.

REFERENCES

[1]. M. Assefa , et.al., "A Combined Reasoning System for Knowledge Based Network Intrusion Detection," p. 10.

[2]. M. A. Aydın, et.al., "A hybrid intrusion detection system design for computer network security," Computers & Electrical Engineering, vol. 35, no. 3, pp. 517–526, 2009.

[3]. L. Zhiqiang, et.al., "Modeling NIDS Using Feed-Forward Neural Network Using UNSW-NB15 Dataset," in 2019 IEEE 7th International Conference on Smart Energy Grid Engineering (SEGE), Oshawa, ON, Canada, Aug. 2019, pp. 299–303. doi: 10.1109/SEGE.2019.8859773.

[4]. A. Javaid, et.al., "A Deep Learning Approach for NIDS", presented at the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS), New York City, United States, 2016. doi: 10.4108/eai.3-12-2015.2262516.

[5]. R. Bace , et.al., "NIST-a special publication on IDS", Booz-allen and Hamilton Inc MCLEAN VA, 2001.

[6]. "INTERNATIONAL JOURNAL FOR RESEARCH & DEVELOPMENT IN TECHNOLOGY," p. 7.

[7]. D. K. Bhattacharyya, et.al., " Network anomaly detection-A ML perspective. Crc Press, 2013.

[8]. T. Hidru, "College Of Natural Sciences", p. 89.

[9]. N. Moustafa, et.al., "UNSW-NB-15: A comprehensive data set for NIDS (UNSW-NB15 network data set)," in 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, Nov. 2015, pp. 1–6.

[10]. G. Manikandan, "Achieving Privacy in Data Mining Using Normalization," IJST, vol. 6, no. 4, pp. 1–5, Apr. 2013.

[11]. N. V. Chawla, et.al., "SMOTE: synthetic minority over-sampling technique", Journal of artificial intelligence research, vol. 16, pp. 321–357, 2002.