COMPARISON ANALYSIS OF CLASSIFICATION ALGORITHMS FOR INTRUSION DETECTION

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Binita Saha

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

July 2021

Fargo, North Dakota

North Dakota State University
Graduate School

Title

Comparison Analysis of Classification Algorithms for Intrusion Detection

By

Binita Saha

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Simone Ludwig
Chair

Dr. Anne Denton

Dr. Mohiuddin Quadir

Approved:

July 19,2021
Date

Dr. Simone Ludwig
Department Chair

ABSTRACT

As the Internet of Things becomes more prevalent in our lives, we are confronted with more security concerns. Network attacks have become more common in the cyber world these days. Denial of service, Prove, Remote to Local attacks, and other types of attacks are on the rise. In our research, we used five machine learning classifiers and conducted a comparison analysis to see which one performed better in predicting network anomalies. Since the dataset we used is unbalanced, we experimented with oversampling and under sampling techniques for the minority and majority groups to improve the model's prediction. Then, in order to test and compare our models, we measured accuracy, F1 ranking, and confusion matrix.

ACKNOWLEDGMENTS

I would like to express my deep and sincere gratitude to my advisor, Dr. Simone Ludwig, for her constant support and providing valuable guidance throughout this research. Beside my advisor, I would like to thank my committee members Dr. Anne Denton and Dr. Mohiuddin Quadir to show interest on my research.

Lastly, I am extremely grateful to my husband, parents, my sister Puja and in-laws for their love, prayers and sacrifices.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

The internet has become an integral component of our daily lives. from waking up with smart watches to communicating on social media or watching Netflix to go bed, we are dependent on the internet. Behind every contact with smart devices or any IoT services, there are thousands of Network nodes, hubs and routers are involved which we take for granted. When a group of devices use the same communication protocol, this indicates that they are connected to the same network. These networks make a variety of services available to us, but they are also subject to a variety of threats. There are numerous policies and processes in place to keep these networks secure. There are 4.72 Billion Internet users all over the world [1]. With these engaged end consumers, there is a lot of room for vulnerability. To support these networks, a security system must be in place. Intrusion Detection is implemented as a result of this motive. Over the last few years, the number of network-based attacks has increased [2]. There are several reasons behind this. One reason is that wireless network is more in use these days. To protect potential attacks, the network security should be put on higher priority. Intrusion Detection System also known as IDS monitors network traffic searching for suspicious activity that known as threats. When an intruder gains access of a network and attempts to change data or render information in a system, the system becomes unreliable. The IDS's responsibility is to inform appropriate stakeholder that a network intrusion may be ongoing. This can be in the form of an alert which should include information about the source address of the intrusion, target/victim address and type of attack that is suspected. An IDS is a combination of hardware and software components that detects malicious or dangerous network activity. IDS can keep track of every network activity and, as a result it detects signals of intrusion. The fundamental goal of IDS is to notify the system administrator if any suspicious behavior occurs. Anomaly detection and misuse detection [3] are two kinds of techniques of intrusion detection. Anomaly detection produces very large number of false positive whereas misuse based system has a very low false positive rate [4].

Data cleaning and preprocessing, model preparation, and model validation or testing are the three phases of Machine Learning or Data Mining. Excluding instances of missing values, correcting any corrupted data, and deleting unnecessary data or columns from the data set are all examples of data cleaning. The process of transforming data into an understandable format is known as data preprocessing. Real-world data always has a broad range between the minimum and maximum value of each element. There are several scaling methods that can be used to reduce the variation between the values. When the feature set is broad, dimensionality reduction techniques are often used to make the learning process simpler and faster. In the machine learning method, the entire data cleaning and preprocessing technique can be considered a single step known as data handling. Another important aspect of data handling in the preprocessing phase is splitting the data into two sections, the training set and the testing set. In most cases, the training and testing data are in a 7:3 ratio. Our model learns about the trend in the data as well as which class an instance belongs to using the training data. Each instance's class is attempted to be predicted using the testing data. We use a different model evaluation methodology to see how well the model performs on unseen data until the expected class values are visible. Confusion matrix, and F1-score are some of the techniques we use to test our model in addition to accuracy. In our work we tried to detect the intrusion of networks using the NSL KDD dataset using five machine learning classifier algorithms. NSL stands for Network Security Laboratory. This dataset is provided by NSL. This is a refined version of the KDD’99 dataset. We have used the original dataset as well as the balanced dataset. Then, we rank them based on their performance on certain metrics such as accuracy, precision, F1 score etc.

The following is a breakdown of the paper's structure. In Section 2, the related work is described in the area of intrusion detection using machine learning classifiers. Section 3 describes briefly the dataset we have used. Section 4 contains our proposed approach. Section 5 describes the experiment conducted as well as result we got. The last section describes the conclusion and future work.

2. RELATED WORK

There are numerous publications that discuss and apply various aspects of IDS. In this section, we will go through some of the important literature work based on feature selection and classification.

The paper in [5] compares two of the most widely used intrusion detection datasets, KDDCUP99 and NSL-KDD. The classifiers trained on the KDDCup99 dataset demonstrated a bias towards the redundancies within it, allowing them to attain better accuracies, which conclude that NSL-KDD is of greater quality than KDDCUP99.

The NSL-KDD dataset is used to analyze the effectiveness using various machine learning classification algorithms (SVM, Nave-Bayes, J48) [6] to detect irregularity in network traffic pattern. Furthermore, the NSL-KDD dataset is used to deduce the protocols' connections from the commonly used network protocol stack from an intruders' attack that generates network traffic irregularities in network flow patterns.

The paper in [7] describes about the training of a neural network which is used to distinguish the various types of new attacks using the Principal Component Analysis Neural Network Algorithm (PCANNA). A refined version of the NSL-KDD dataset has been used over the KDD-CUP99 dataset for checking and comparing the datasets. The resulted data features can be reduced by 80.4 percent using this analysis and the time reduction varies from 70% (for time testing) to 40% (for time training and other purpose).

In [8], five machine learning classifiers (J48 decision tree, SVM, Decision table, Bayesian network, Back propagation neural network) have been used to compare and three feature selection have been used which helps to decrease the dimensionality of NSL-KDD dataset to get stable and better results. Among them, information gain feature selection gives the best efficient result with the accuracy of 80.3% for the training dataset and 93.9% for the testing dataset.

Using the NSL-KDD dataset, [9] focuses on feature selection for intrusion detection. To get a better outcome, a decision tree classifier was combined with a new feature selection method that employed the feature average of total and individual classes. Three important features selection methods (CFS, IG, GR) were taken into account. CFS's accuracy was 99.781 percent with 25 features, IG was 99.781 percent with 23 features, and GR was 99.794 percent with 19 features.

[10] gives an overview of current Intrusion Detection Systems and their fundamental concepts. It also discusses how data mining can aid in the development of IDS-based data mining with its core feature (knowledge discovery). When compared to traditional IDS, the resulting data mining can show more consistent behavior and achieve a higher level of accuracy when it comes to different types of instruction.

The NSL-KDD dataset has been used to design the IDS [11] with a neural network assembled method to classify different kinds of attacks. It detects the false alarm rate and measure the detection rate of neural network assemble method along with confusion matrix, classification accuracy and area under curve.

To detect IDS in the NSL KDD dataset, [12] has proposed coupling SVM with other classifiers such as BayesNet, AdaBoost, Logistic, IBK, J48, RandomForest, JRip, OneR, and SimpleCart. SVM and Random Forest outperform best with an accuracy of roughly 97.50 percent, which is higher than SVM's 91.81 percent.

[13] [14] uses five machine learning classifier algorithms to analyze chosen features of the NSL-KDD dataset: J48, Nave Bayes, CART, Random Forest, and SVM. SVM and Random Forest are the most effective among them.

To create a reduced feature subset of the NSL-KDD dataset, the SMOTE function [15] is applied to the training dataset and use the Information Gain feature selection approach. According to research, using a random forest classifier with the SMOTE function and information gain improves the performance of IDS design.

Decision tree algorithm is developed [16] using the feature selection method (Information Gain) and split value to perform IDS on the NSL-KDD dataset. The proposed Decision Tree Split (DTS) can be used for signature-based Intrusion detection.

A new feature selection method (average of total and each classes) [9] has been introduced using the decision tree classifier for evaluating the performance of applying the feature selection method. The NSL-KDD dataset was used for the experiment. The authors showed a comparison between their new method and the existing methods.

On the NSL-KDD dataset, [17] assesses ten of the most prominent machine learning classifiers, ranking them based on numerous parameters such as specificity, sensitivity, and accuracy. The experiment reveals that Random Forest (with or without features selection) produces the greatest results in terms of accuracy in a very short period of time.

3. DATASET

We used the NSL-KDD dataset [13] to test the proposed process, which is the refined version of the KDD cup'99 dataset [18] obtained from the University of New Brunswick Canadian Institute for Cybersecurity. KDD cup'99 was widely used by researchers before, but it has some drawbacks. It has duplicate data entries and hand injected data, which does not provide accurate classifier result. On the other hand, NSL-KDD has the advantage over KDD cup'99 in terms of duplicate data entries and better detection rate. For researchers, there is a range of downloadable files available which is given in Table 1 with a brief description.

Table 1: List of NSL-KDD Dataset with Description

Index	File Name	Description
1	KDDTrain+.ARFF	The complete NSL-KDD train dataset in ARFF format with binary labels
2	KDDTrain+.TXT	The complete NSL-KDD train dataset in text format with binary labels
3	KDDTrain+_20Percent.ARFF	KDDTrain+_20Percent.ARFF is a 20% subset of the KDDTrain+.arff file
4	KDDTrain+_20Percent.TXT	KDDTrain+_20Percent.TXT is a 20% subset of the KDDTrain+.TXT
5	KDDTest+.ARFF	The complete NSL-KDD test dataset in ARFF format with binary labels
6	KDDTest+.TXT	The complete NSL-KDD test dataset in text format with binary labels
7	KDDTest-21.ARFF	KDDTest-21.ARFF is a 20% subset of the KDDTest+.ARFF file
8	KDDTest-21.TXT	KDDTest-21.TXT is a 20% subset of the KDDTest+.TXT file

KDDTrain+.TXT and KDDTest+.TXT are used in this paper since these two datasets covered all of the details. Each record in the data set has 43 attributes, among them 41 are relating to the traffic input and the last two being labels for normal and malicious traffic, as well as a score indicating the severity of the traffic input. Among the 41 functions, there are 7 discrete and 34 continuous variables and 39 different forms of cyber-attacks, which can be categorized into four categories such as Denial of Service, Probe, Root to Local, User to Root. All normal kinds of attack named as standard in our experiment.

Standard: This is a normal type of attack named as Standard.

Denial of Service (DoS): DoS is a form of attack in which the victim's resources are depleted, rendering it unable to manage valid requests.

Probe: The aim of surveillance and other probing attacks such as port scanning, is to obtain information about the remote victim. "Connection duration" and "source bytes" are two features that are important in Probing.

Root to Local (R2L): The intruder intrudes into a remote machine and achieves all local access to the victim machine by unauthorized access.

User to Root (U2R): U2R is a form of attack in which an attacker logs into a victim local system with a regular account and attempts to guess root/administrator privileges by exploiting a flaw in the victim.

Table 2: Attack Classes with Attack Type and Values

Attack Class	Attack Type	Value
Standard	Normal	0
DoS	Back, Land, Neptune, Pod, Smurf, Teardrop, Apache2, Udpstorm, Processtable, Worm	1
Probe	Satan, Ipsweep, Nmap, Portsweep, Mscan, Saint	2
R2L	Guess_Password, Ftp_write, Imap, Phf, Multihop, Warezmaster, Warezclient, Spy, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendmail, Named	3
U2R	Buffer_overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, Ps	4

Table 2 describes the attack type classes with a particular value, which helps classifiers to detect various kinds of attacks. During a specific attack, an attack creates a link between a source IP address and a target IP address and sends data to attack the target.

4. APPROACH

This section describes the used Machine learning classifier algorithms along with feature selection method in details.

4.1. Classification Algorithms

In our experiment, we used five different binary classifiers such as Logistics Regression, Decision Tree, K-nearest Neighbor, Random Forest and Gaussian Naïve Bayes.

4.1.1. Logistic Regression

Logistic Regression is a binary classifier which predicts a value between 0 and 1. It is used to determine the probability of observation to be part of a certain class or not which is expressed by a binary selection (0 or 1). To characterize data and illustrate the relationship between one dependent binary variable and nominal, ordinal, interval, or ratio-level independent variables, logistic regression is used. The central feature of logistic regression is called the logistic function. This function is also known as the sigmoid function, which was created by statisticians to explain the properties of population growth in ecology, such as growing rapidly and eventually reaching the environment's carrying capacity. Random state is a random number generator that is one of the parameters used for this algorithm. None is the default random condition. Random state has been set to 111 in our case.

4.1.2. Decision Tree Classifier

Another systematic approach to building classification models is the decision tree, which is a very simple and widely used technique. The decision tree's key method of operation is to learn decision rules from training data and use them to predict groups. Knowledge benefit and Gini index are two types of attribute selection measures that we used to build two types of decision trees for both the initial dataset and the oversampled dataset. As decision tree works with categorical data, so it is important to convert all continuous data to discrete or continuous data. We have used two attribute selection techniques which criterion is entropy and splitter is none in our decision tree classifier.

4.1.3. K-nearest Neighbor

K-nearest Neighbor algorithm is one of the simplest classifier which is also known as lazy algorithm. It can be used to solve both classification and regression problems but are mostly used for classification problems. This algorithm is very simple to interpret and faster in terms of calculating time. It finds the distance between two points of data with the closest unique number then votes for the most frequent mark. In this algorithm, n-neighbor is a very important parameter which has a default value of 5. There is no specific formula to determine the best value for it. However, one way to determine this value is to pick a range and see how accurate it is. After that, one can compare and chose the n-neighbor with a higher accuracy value.

4.1.4. Random Forest

Random Forest is one of the various other models we have used in our experiment which can solve both classification and regression problems. From a randomly chosen subset of the training set, the random forest classifier generates a set of decision trees. It then combines the votes from various decision trees to determine the test object's final class. There are basic parameters such as how many trees are generated, minimum split of the trees, split criteria, which are import to Random Forest classifier. One of the most important parameters are n-estimator, which has a default value of 100.

4.1.5. Gaussian Naïve Bayes

Gaussian Naive Bayes is an extended version of Naïve Bayes that uses a Gaussian normal distribution which can handle continuous data. The Bayes' theorem is the basis for a group of supervised machine learning classification algorithms known as Naive Bayes. It is a simple classification technique and the main difference between Naïve bayes and Gaussian Naïve bayes is that standard Naïve bayes only supports categorical data whereas Gaussian Naïve Bayes supports continuous data. One of the important parameters is 'priors' which set to none by default as well as in our experiment.

4.2. Undersampling and Oversampling

Undersampling and oversampling methods are used to achieve a balanced dataset when the dataset is unbalanced. These are data analysis techniques that alter the distribution of a dataset. In majority classes, the undersampling approach deletes or merges data, whereas in minority classes, the oversampling method duplicates or creates additional synthesis examples. These methods are typically applied to a training dataset that will be used to fit into a machine learning model.

We have five different types of attacks in our NSL-KDD dataset: normal, DoS, Probe, R2L, and U2R, with 77054, 53094, 14077, 4173, and 119 attacks, respectively. We choose 2500 data points from each class to make it balanced. We utilize an undersampling strategy to merge or eliminate data in the majority classes because the first four attacks have over 2500 data points. We employed the Smote function for U2R, which is a synthetic minority oversampling technique that is widely used for undersampling. The main concept behind this strategy is to use the closest neighbors of these cases to artificially generate new examples of the minority class. In addition, the majority of class examples are under-represented, resulting in a more balanced collection.

4.3. Performance Metrics

In order to evaluate the algorithm, we compare our mechanism with some performance metrics such as Accuracy, Recall, Precision, Accuracy and F1 score.

Confusion matrix: In our experiment, we have used confusion matrix to get an overall idea of the overall performance of the model. It shows the performance of a classifier in a tabular form. It describes the performance of a matrix visually. The rows of the table belong to the actual class while the column indicates the predicted class. There are four important terms used in a confusion matrix:

***True Positive*:** A true positive is when the model predicts the positive class correctly.

***True Negative*:** A true negative is when the model predicts the negative class correctly.

***False Positive*:** A false positive is when the model predicts the positive class incorrectly.

***False Negative*:** A false negative is when the model predicts the negative class incorrectly.

Accuracy: One of the most basic metrics for describing the quality of an algorithm is its accuracy. It is easy to calculate by dividing the number of valid predictions by the total number of predictions.

$$\text{Accuracy} = (TP+TN)/(TP+TN+FP+FN)$$

Recall: The ratio of correctly predicted positive observations to all observation of the actual class is known as Recall.

$$\text{Recall} = TP/(TP+FN)$$

Precision: The ratio of correctly predicted positive observations to total predicted positive observations is known as precision.

$$\text{Precision} = TP/(TP+FP)$$

F1 Score: This is another model evaluation method. F1 score is the weighted average of precision and recall. The F1-Score typically reaches a value of 1.

$$\text{F1 Score} = 2*(\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$$

We tested the five different classifiers in two different datasets to see which one provides the best results: one is using the imbalanced dataset often known as original dataset and the another is the balanced dataset. For the original dataset, we have 77054,53094,14077,4173,119 data points for Standard, DoS, Prove, R2L, and U2R, respectively, and we used 2500 data from each form of attack for the balanced dataset. For the first three kinds of attack, we use under sampling. Random under sampling with replacement is used where a sample is selected from the dataset in which every data point has equal chances of being selected. The probability of each sample being selected from the population is 1/N where N is the size of the population. As R2L and U2R attack has less data, we do oversampling using smooth function which generates synthetic data for minority classes. Generated synthetic data lies between the range of minority class. To produce new instances, this approach needs three parameters: T, N, and k, where T is the number of minority class samples to be over-sampled, N is the percentage explaining the minority class to be over-sampled, and k is the number of nearest neighbors to be considered for the oversampling technique.

5. EXPERIMENTS AND RESULTS

In this section, result of different binary classifier will be observed for both datasets.

5.1. Logistic Regression Classifier

In our first experiment, Logistic regression is applied to both the original and the balanced datasets. Table 3 shows the measured value for training and testing accuracy. The models performance is determined by the accuracy of testing. The original and balanced datasets have reached an accuracy of 80% and 49%, respectively. Though, the original dataset is giving better accuracy, it provides almost 0% accuracy for two attack classes as shown in Table 4.

Table 3: Measured Value for Logistic Regression

Measure Name	Original Dataset	Balanced Dataset
Training accuracy	0.80	0.51
Testing accuracy	0.80	0.49

The diagonal value of the confusion matrix (Figure 1) indicates the true positive values. From a total of 53837 objects, 20883 are correctly categorized as Standard attack. Out of 37204, 9914, 2923, and 83 items, 14182, 501, 2, 0 are correctly classified as DoS, Prove, R2L and U2R, respectively.

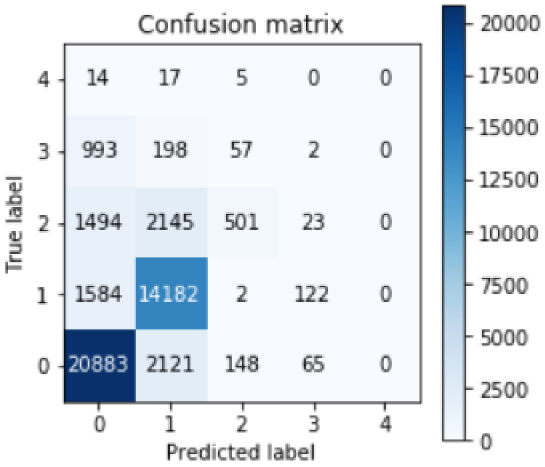

Figure 1: Confusion Matrix for Logistic Regression (Original Dataset)

The confusion matrix for the balanced dataset (Figure 2) shows 569, 665, 56, 443, 101 correctly identified as Standard, DoS, Prove, R2L and U2R, respectively, out of 1747,1778, 1728, 1743, 1754 samples.

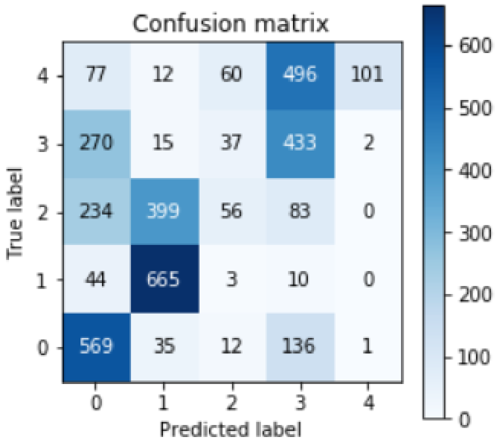

Figure 2: Confusion Matrix for Logistic Regression (Balanced dataset)

The original dataset gives almost a null value for class 3 and 4, while the balanced dataset gives more balanced value for all types of attacks. It is better to identify different kinds of attacks in the moderate level rather than one or two kinds of attacks with the highest rate.

Table 4: F1 Score for Logistic Regression (Original Dataset)

Label	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.83	0.90	0.87	53837
1	0.77	0.89	0.82	37204
2	0.72	0.13	0.22	9914
3	0.03	0.01	0.01	2923
4	0.00	0.00	0.00	83
Macro avg	0.47	0.39	0.38	103961
Weighted avg	0.78	0.80	0.77	103961

Table 4 and 5 show the macro and weighted average along with the F1 score for the original and the balanced dataset, respectively. For the original dataset, the first two kinds of attacks are identifying more than 80% whereas the last three are identifying 22%, 1% and 0%, respectively with macro average of 38% and micro average of 77%.

Table 5: F1 Score for Logistic Regression (Balanced Dataset)

Label	Precision	Recall	F1-Score	Support
0	0.50	0.77	0.61	753
1	0.58	0.94	0.72	722
2	0.38	0.07	0.11	772
3	0.37	0.59	0.45	757
4	0.99	0.13	0.22	746
Macro avg	0.56	0.50	0.42	3750
Weighted avg	0.56	0.49	0.42	3750

The balanced dataset of logistic regression (Table 5) has an overall accuracy of almost 50% but it identifies each class with almost 50% accuracy, which is better than the original dataset. As in the original dataset the last two kinds of attacks are not identified correctly at all.

5.2. Decision Tree Classifier

The second model of our experiment is the decision tree classifier. Table 6 shows the measured value for the training and testing accuracy. The original and balanced datasets have reached the accuracy of 99% and 98%, respectively.

Table 6: Measured Value for Decision Tree

Measure Name	Original Dataset	Balanced Dataset
Training accuracy	1.00	1.00
Testing accuracy	0.99	0.98

The diagonal value of the confusion matrix indicates the true positive values. Figure 3 describes the confusion matrix for the decision tree. From a total of 23217 objects, 23068 are correctly categorized as Standard attack. Out of 15890, 4162, 1250 and 36 items 15867,4125,1149,14 are correctly classified as DoS, Prove, R2L and U2R, respectively

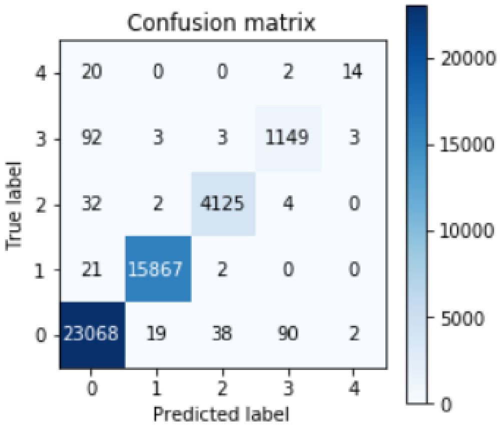

Figure 3: Confusion Matrix for Decision Tree (Original Data)

The confusion matrix for the balanced dataset (Figure 4) shows 721,712,749,735,739 correctly identified as Standard, DoS, Prove, R2L and U2R, respectively, out of 1747,1778, 1728, 1743, 1754 items.

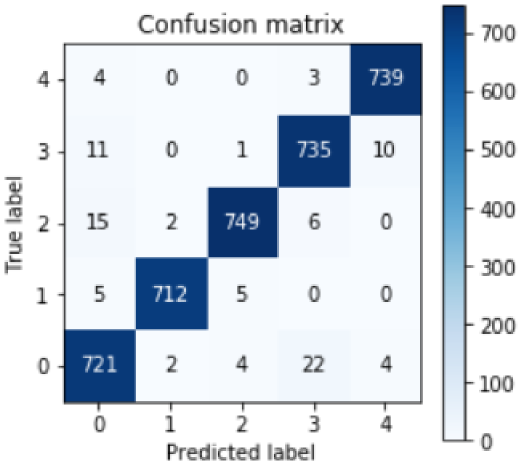

Figure 4: Confusion Matrix for Decision Tree (Balanced Dataset)

Table 7 and 8 show the macro and weighted average along with the F1 score for the original and the balanced dataset respectively. For the original dataset, it is giving almost 99%

accuracy for the first four kinds of attacks but for U2R it is giving only 51% accuracy with a macro average of 88% and weighted average of 99%.

Table 7: F1 Score for Decision Tree (Original Dataset)

Label	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.99	0.99	0.99	23217
1	1.00	1.00	1.00	15890
2	0.99	0.99	0.99	4162
3	0.92	0.92	0.92	1250
4	0.74	0.39	0.51	36
Macro avg	0.93	0.86	0.88	44556
Weighted avg	0.99	0.99	0.99	44556

According to the balanced dataset of the Decision Tree classifier (Table 8), F1 score is 96%, 99%, 98%, 97% and 99% respectively for the different kinds of attacks. Macro and weighted average is 98%. Original dataset can only detect 51% for U2R whereas the balanced dataset can detect 99% attacks of that class. Other classes are pretty close for both datasets. So, for this case also the balanced dataset works best to detect all five kinds of attacks.

Table 8: F1 Score for Decision Tree (Balanced Dataset)

Label	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.97	0.95	0.96	753
1	0.99	1.00	0.99	722
2	0.98	0.99	0.98	772
3	0.96	0.98	0.97	757
4	0.99	0.99	0.99	746
Macro avg	0.98	0.98	0.98	3750
Weighted avg	0.98	0.98	0.98	3750

5.3. K-Nearest Neighbour Classifier

The third model of our experiment is the K-Nearest Neighbor Classifier. Table 9 shows the measured value for the training and testing accuracy for the K-Nearest Neighbor classifier. The original and balanced datasets have reached an accuracy of 99% and 94%, respectively.

Table 9: Measured Value for K-Nearest Neighbor Classifier

Measure Name	Original Dataset	Balanced Dataset
Training accuracy	1.00	1.00
Testing accuracy	0.99	0.94

The diagonal value of the confusion matrix indicates the true positive values. From a total of 23217 objects, 22985 are correctly categorized as Standard attack (Figure-5). Out of 15890, 4162, 1250 and 36 items 15809,3959,1139,9 are correctly classified as DoS, Prove, R2L and U2R, respectively.

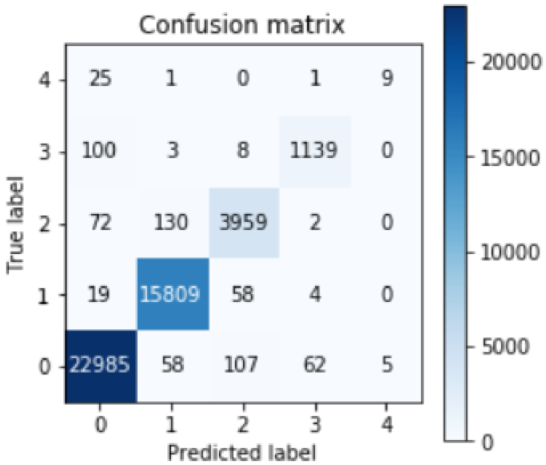

Figure 5: Confusion Matrix for Original Dataset (K-Neighbor Classifier)

The confusion matrix for the balanced dataset (Figure 6) shows 659,680,731,728,713 correctly identified as Standard, DoS, Prove, R2L and U2R, respectively, out of 753,722,772,757,746 items.

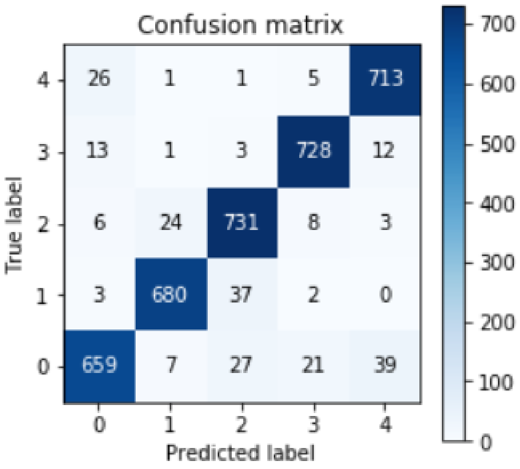

Figure 6: Confusion Matrix for Balanced Dataset (K-Neighbor Classifier)

Table 10 and 11 shows the macro and weighted average along with the F1 score for the original and balanced dataset, respectively. For the original dataset (Table 10), it is giving more than 93% accuracy for the first four kinds of attacks but for U2R it is giving only 36% accuracy with the macro average of 84% and the weighted average of 99%.

Table 10: F1 Score for Original Dataset (K-Neighbor Classifier)

Label	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.99	0.99	0.99	23217
1	0.99	0.99	0.99	15890
2	0.96	0.95	0.95	4162
3	0.94	0.91	0.93	1250
4	0.64	0.25	0.36	36
Macro avg	0.90	0.82	0.84	44556
Weighted avg	0.99	0.99	0.99	44556

According to the balanced dataset of the K-Nearest Neighbor classifier (Table 11), the F1 score is 90%, 95%, 93%, 96% and 94%, respectively, for the different kinds of attacks. The macro and weighted average is 94%.

Table 11: F1 Score for Balanced Dataset (K-Neighbor Classifier)

Label	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.93	0.88	0.90	753
1	0.95	0.94	0.95	722
2	0.91	0.95	0.93	772
3	0.95	0.96	0.96	757
4	0.93	0.96	0.94	746
Macro avg	0.94	0.94	0.94	3750
Weighted avg	0.94	0.94	0.94	3750

The original dataset can only detect 36% for U2R whereas the balanced dataset can detect 94% attacks of that class. The other classes are pretty comparable for both datasets. So, for this case the balanced dataset works best to detect all five kinds of attacks.

5.4. Random Forest Classifier

The fourth model of our experiment is the Random Forest classifier. Table 12 shows the measured value for the training and testing accuracy. The original and balanced datasets have reached an accuracy of 99%.

Table 12: Measured Value for Random Forest

Measure Name	**Original Dataset**	**Balanced Dataset**
Training accuracy	1.00	1.00
Testing accuracy	0.99	0.99

The diagonal value of confusion matrix indicates true positive values. From a total of 23217 samples, 23130 are correctly categorized as Standard attack (Figure 7). Out of 15890, 4162, 1250 and 36 items 15871,4138,1160,14 are correctly classified as DoS, Prove, R2L and U2R, respectively.

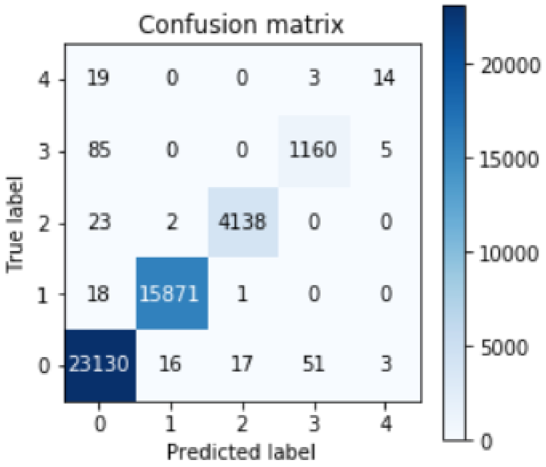

Figure 7: Confusion Matrix for Original Dataset (Random Forest)

The confusion matrix for the balanced dataset (Figure 8) shows 731,718,763,734,743 correctly identified as Standard, DoS, Prove, R2L and U2R, respectively, out of 753,722,772,757,746 items.

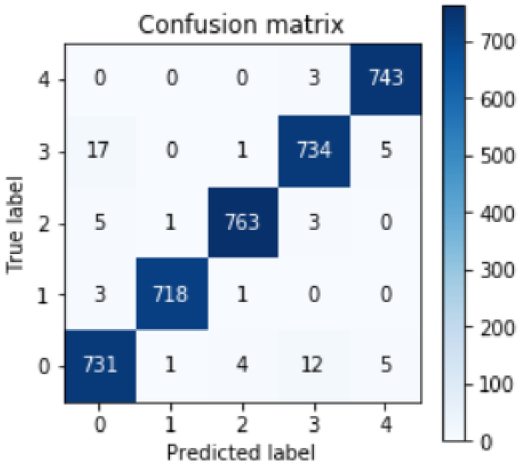

Figure 8: Confusion Matrix for Balanced Dataset (Random Forest)

Table 13 and 14 shows the macro and weighted average along with the F1 score for original and balanced dataset, respectively. For the original dataset (Table 13), it is giving more than 94% accuracy for the first four kinds of attacks but for U2R it is giving only 51% accuracy with the macro average of 89% and the weighted average of 99%.

Table 13: F1 Score for Original Dataset (Random Forest)

Label	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.99	1.00	0.99	23217
1	1.00	1.00	1.00	15890
2	0.99	0.99	0.99	4162
3	0.96	0.92	0.94	1250
4	0.65	0.42	0.51	36
Macro avg	0.92	0.87	0.89	44556
Weighted avg	0.99	0.99	0.99	44556

According to the balanced dataset of the Random Forest classifier (Table 14), the F1 score is 99%, 100%, 99%, 98% and 99%, respectively for the different kinds of attacks. The macro and weighted average is 99%. The original dataset can only detect 51% for U2R whereas the balanced dataset can detect 99% attacks of that class. Other classes are pretty close for both datasets. So, the balanced dataset works best to detect all five kinds of attacks.

Table 14: F1 Score for Balanced Dataset (Random Forest)

Label	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.97	0.98	0.99	753
1	1.00	0.99	1.00	722
2	0.99	0.99	0.99	772
3	0.98	0.98	0.98	757
4	0.99	1.00	0.99	746
Macro avg	0.99	0.99	0.99	3750
Weighted avg	0.99	0.99	0.99	3750

5.5. GaussianNB Classifier

The last model of our experiment is the GaussianNB Classifier. Table 15 shows the measured value for the training and testing accuracy. The original and balanced datasets have reached an accuracy of 42% and 28%, respectively.

Table 15: Measured Value of GaussianNB

Measure Name	Original Dataset	Balanced Dataset
Training accuracy	1.00	1.00
Testing accuracy	0.42	0.28

The diagonal value of confusion matrix indicates true positive values. From a total of 23217 samples, only 3306 are correctly categorized as Standard attack (Figure 9). Out of 15890, 4162 items 15169,268 are correctly classified as DoS, Prove. The last two attacks (R2L and U2R) do not identified at all.

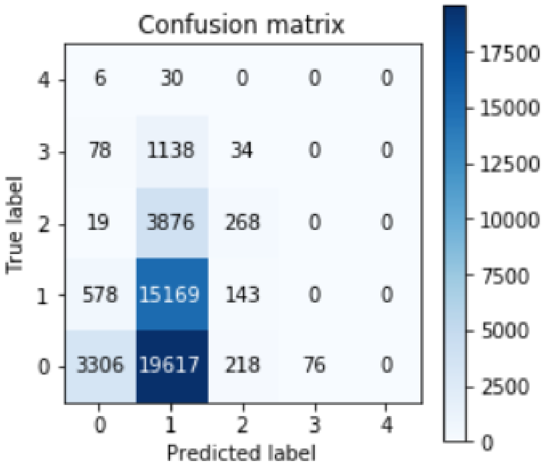

Figure 9: Confusion Matrix of Original Dataset (GaussianNB)

The confusion matrix for the balanced dataset (Figure10) shows 4,690,53,3,313 correctly identified as Standard, DoS, Prove, R2L and U2R, respectively, out of 753,722,772,757,746 items.

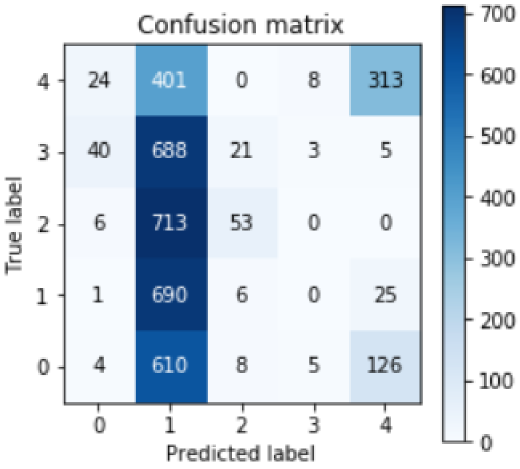

Figure 10: Confusion Matrix of Balanced Dataset (GaussianNB)

Table 16 and 17 show the macro and weighted average along with the F1 score for the original and the balanced dataset, respectively. For the original dataset (Table 16), it can be identified 24%, 54%, 11%, 0%, 0% for five kinds of attack, respectively, with the macro average of 18% and the weighted average of 33%.

Table 16: F1 Score of Original Dataset (GaussianNB)

Label	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.83	0.14	0.24	23217
1	0.38	0.95	0.54	15890
2	0.40	0.06	0.11	4162
3	0.00	0.00	0.00	1250
4	0.00	0.00	0.00	36
Macro avg	0.32	0.23	0.18	44556
Weighted avg	0.61	0.42	0.33	44556

According to the balanced dataset of the GaussianNB classifier (Table 17), the F1 score is 1%, 36%, 12%, 1% and 52%, respectively, for different kinds of attacks. The macro and weighted average is 20%. The original dataset cannot identify a single attack for R2L and U2R, the balanced dataset can detect a few. But, the overall performance of this dataset is not satisfied and thus, this is not a good classifier to detect anomalies of network.

Table 17: F1 Score of Balanced Dataset (GaussianNB)

Label	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.05	0.01	0.01	753
1	0.22	0.96	0.36	722
2	0.60	0.07	0.12	772
3	0.19	0.00	0.01	757
4	0.67	0.42	0.52	746
Macro avg	0.35	0.29	0.20	3750
Weighted avg	0.35	0.28	0.20	3750

Table 18 lists the classification report and accuracy of all supervised machine learning algorithms we have used for the original dataset. Decision tree, Random forest, K-Nearest Neighbor gives the best accuracy which is 99%, followed by Logistic regression which gives 80% accuracy. In this dataset, GaussianNB has the weakest performance, with a score of 49%. Though all three classifiers provide 99% accuracy, the classification report shows that Random forest provides the best results for all five types of attacks, with K-Nearest Neighbor and Decision tree classifiers coming in second and third.

Table 19 lists the classification report and accuracy of all supervised machine learning algorithm we have used for the balanced dataset. From our experiments we can see, Random Forest performs best with an accuracy of 99%, decision tree shows 98% accuracy which is second best performance. K-Nearest neighbor gives an accuracy of 94%. Logistic Regression and GaussianNB shows the accuracy of 49% and 28%, respectively.

Table 18: Classification Report & Accuracy of Supervised Machine Learning Algorithm (Original Dataset)

Algorithm Name	**Classification Report**					**Accuracy**
Logistic Regression	Class	Precision	Recall	F1 Score	Support	
	0	0.83	0.9	0.87	53837	0.8
	1	0.77	0.89	0.82	37204	
	2	0.72	0.13	0.22	9914	
	3	0.03	0.01	0.01	2923	
	4	0	0	0	83	
Decision Tree	0	0.99	0.99	0.99	23217	0.99
	1	1	1	1	15890	
	2	0.99	0.99	0.99	4162	
	3	0.92	0.92	0.92	1250	
	4	0.74	0.39	0.51	36	
K-Nearest Neighbor	0	0.99	0.99	0.99	23217	0.99
	1	0.99	0.99	0.99	15890	
	2	0.96	0.95	0.95	4160	
	3	0.94	0.91	0.93	1250	
	4	0.64	0.25	0.36	36	
Random Forest	0	0.99	1	0.99	23217	0.99
	1	1	1	1	15890	
	2	0.99	0.99	0.99	4162	
	3	0.96	0.92	0.94	1250	
	4	0.65	0.42	0.51	36	
GausianNB	0	0.83	0.14	0.24	23217	0.42
	1	0.38	0.95	0.54	15890	
	2	0.4	0.06	0.11	4162	
	3	0	0	0	1250	
	4	0	0	0	36	

Table 19: Classification Report & Accuracy of Supervised Machine Learning Algorithm (Balanced Dataset)

Algorithm Name	Classification Report					Accuracy
Logistic Regression	Class	Precision	Recall	F1 Score	Support	
	0	0.48	0.79	0.6	1747	0.49
	1	0.61	0.92	0.73	1778	
	2	0.33	0.07	0.12	1728	
	3	0.37	0.54	0.44	1743	
	4	0.98	0.12	0.21	1754	
Decision Tree	0	0.97	0.95	0.96	753	0.98
	1	0.99	1	0.99	722	
	2	0.98	0.99	0.98	772	
	3	0.96	0.98	0.97	757	
	4	0.99	0.99	0.99	746	
K-Nearest Neighbor	0	0.93	0.88	0.9	753	0.94
	1	0.95	0.94	0.95	722	
	2	0.91	0.95	0.93	772	
	3	0.95	0.96	0.96	757	
	4	0.93	0.96	0.94	746	
Random Forest	0	0.97	0.98	0.99	753	0.99
	1	1	0.99	1	722	
	2	0.99	0.99	0.99	772	
	3	0.98	0.98	0.94	757	
	4	0.99	1	0.99	746	
GaussianNB	0	0.05	0.01	0.01	753	0.28
	1	0.22	0.96	0.36	722	
	2	0.6	0.07	0.12	772	
	3	0.19	0	0.01	757	
	4	0.67	0.42	0.52	746	

The logistic regression and GaussianNB classifier give higher accuracy in the original dataset than the balanced dataset because it gives false positive values for some of the classes.

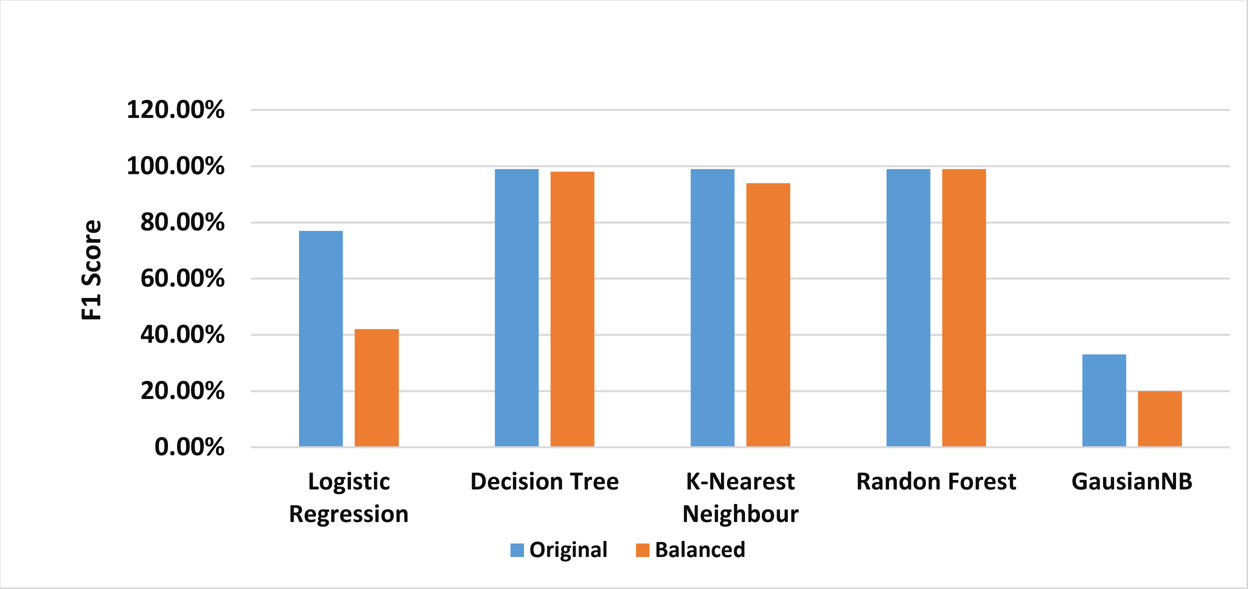

Figure 11: F1 Score of Five Supervised Algorithms

Figure 11 shows the graphical representation of the F1 score for the five classifiers in the original and balanced dataset. F1 score is one of the performance matrix to evaluate the performance of the model.

6. CONCLUSION AND FUTURE WORK

In the above experiment it is seen that the model has better accuracy weighted average but not for the different kinds of attacks separately. On the other hand, in a balanced dataset, all types of attacks perform better. Random forest outperforms the other five binary machine learning classifiers we tested, with 99 percent accuracy for both macro and weighted average. The Decision Tree classifier then provides a 98 percent accuracy for both cases. The K-Neighbor binary classifier is the third best binary classifier, with a 94 percent accuracy for the macro average and weighted average.

This time we did not check how much time each classifier takes to perform the task. We can check the accuracy in terms of time complexity in future. Also, we can merge multiple classifiers to get a better accuracy. This work can be extended by merging multiple classifiers to improve the performance of the model and thus reduce the false negative values.

REFERENCES

[1] 2021. [Online]. Available: https://datareportal.com/global-digital-overview#:~:text=The%20number%20of%20internet%20users,900%2C000%20new%20users%20each%20day..

[2] S. A. Ludwig, "Applying a Neural Network Ensemble to Intrusion Detection," *Journal of Artificial Intelligence and Soft Computing Research,* vol. 9, pp. 177-188, 2019.

[3] R. R. Devi and M. Abualkibash, "Intrusion Detection System Classification Using Different Machine Learning Algorithms on KDD-99 and NSL-KDD Datasets," in *International Journal of Computer Science & Information Technology*, 2019.

[4] I. Aljarah and S. A. Ludwig, "MapReduce Intrusion Detection System based on a Particle Swarm Optimization Clustering Algorithm," *IEEE Congress on Evolutionary Computation,* pp. 956-962, 2013.

[5] P. A. K. I. Suchet Sapre, "A Robust Comparison of the KDDCup99 and NSL-KDD IoT Network Intrusion Detection Datasets Through Various Machine Learning Algorithms".

[6] L. Dhanabal and S. P. Shantharajah, "A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms," *International Journal of Advanced Research in Computer and Communication Engineering,,* vol. 4, no. 6, 2015.

[7] S. Lakhina, S. Joseph and B. Verma, "Feature Reduction using PCA for effective Anomaly-Based Intrusion detection on NSL-KDD," *International Journal of Engineering Science and Technology,* vol. 2, no. 6, pp. 1790-1799, 2010.

[8] J. H. Assi1 and A. T. Sadiq2, "NSL-KDD dataset Classification Using Five Classification Methods and Three Feature Selection Strategies," *Journal of Advanced Computer Science and Technology Research,* vol. 7, pp. 15-18, 2017.

[9] H.-s. Chae, B.-o. Jo and T.-k. P. Sang-Hyun Choi, "Feature Selection for Intrusion Detection using NSL-KDD," in *Recent Advances in Computer Science*.

[10] L. M. Zibusiso Dewa, "Data Mining and Intrusion Detection Systems," *International Journal of Advanced Computer Science and Applications,* vol. 1, pp. 61-71, 2016.

[11] S. A. Ludwig, "Intrusion Detection of Multiple Attack Classes using a Deep Neural Net Ensemble," *IEEE Symposium Series on Computational Intelligence (SSCI),* 2017.

[12] N. Chand, P. Mishra, C. R. Krishna, E. S. Pilli and M. C. Govil, "A Comparative Analysis of SVM and its Stacking with other Classification Algorithm for Intrusion Detection," 2016.

[13] S. Revathi and A. Malathi, "A Detailed Analysis on NSL-KDD Dataset Using Various Machine Learning Techniques for Intrusion Detection," *Int. J. Eng. Res. Technol.,* 2013.

[14] Y. K. Ever, B. Sekeroglu and K. Dimililer, "Classification Analysis of Intrusion Detection on NSL-KDD Using Machine Learning Algorithms," 26 July, 2019.

[15] A. Tesfahun and D. L. Bhaskari, "Intrusion Detection Using Random Forests Classifier with SMOTE and Feature Reduction," in *International Conference on Cloud & Ubiquitous Computing & Emerging Technologies*, Pune, 2013.

[16] K. Rai, A. Guleria and M. S. Devi, "Decision Tree Based Algorithm for Intrusion Detection," in *Int. J. Advanced Networking and Applications*, 2016.

[17] H. M. Prachi and P. Sharma, "Intrusion Detection using Machine Learning and Feature Selection," in *I. J. Computer Network and Information Security*, 2019.

[18] S. S. Kaushik and D. Prof.P.R.Deshmukh, "Detection of Attacks in an intrusion Detection System," *International Journal of Computer Science and Information Technologies,* vol. 2, no. 3, pp. 982-986, 2011.