

# Analysis of K- Nearest Neighbor for Network Intrusion Detection

Mba Obasi Odim<sup>1,5</sup>; Shalom Oluwapelumi Ojo<sup>2</sup>; Bosede Oyenike Oguntunde<sup>3</sup>; Toluwase Ayobami Olowookere<sup>4</sup>

<sup>1,2,3,4</sup>Department of Computer Science, Redeemer's University, Ede, Nigeria

<sup>5</sup>Department of Computer Science and Mathematics, Mountain Top University, Prayer City, Nigeria

odimm@run.edu.ng<sup>1</sup>, ojo7806@run.edu.ng<sup>2</sup>, oguntunden@run.edu.ng<sup>3</sup>,  
olwookereta@run.edu.ng<sup>4</sup>

DOI: 10.26821/IJSHRE.11.5.2023.110508

## ABSTRACT

*Computer network Intrusion is an unauthorised action or activity on a network. The threat of intrusions on cyber-security has grown significantly recently. Various techniques are used to counteract these dangers with some levels of detection accuracy. This study models and assesses the performance of k- Nearest Neighbor (KNN) for intrusion detection. An online intrusion detection dataset of National State and Local Knowledge Discovery in Database (NSL-KDD) from Kaggle was used for the assessment comprising 43 features and instances of both normal and abnormal data streams. The model is implemented in python and assessed using Precision, Recall, and F1 score. The assessment results show 99.996% accuracy and 1.00 respectively for Precision, Recall and F1 Score for the 2-class classification (normal and abnormal) and 99.988% and also 1.00 respectively for Precision, Recall and F1 Score for each of the abnormal multi-class classification (Denial of service, Remote to user Attack, User to root, and probe), except for the User to root class that records a Precision of 9.99. These results suggest that KNN is an effective algorithm for intrusion detection both for the binary and multi class classification. and, therefore, should be adopted for developing an intrusion detection system.*

**Keywords:** Computer Network, Cyber-attacks, Intrusion Detection, k-Nearest Neighbor

## 1. INTRODUCTION

Internet-connected services are rapidly increasing over the years. In [1] it was estimated that fifty billion devices will have been connected to the internet as of 2020, this number has since been

surpassed. However, information and communication technology systems and devices are continuously being exposed to cyber risks despite the resulting benefits. Malware attacks are getting more complex, challenging to detect, and with substantial economic and social consequences. Yearly, billions of dollars are lost due to breaches of IT services, and this value is expected to grow from one year to another [2]. As a result, cybersecurity has become great issue of concern to modern society. The prevention of such cyber-attacks and malicious activities is the primary function of the intrusion detection system (IDS). An intrusion detection system keeps track of network traffic for suspicious activities and sends an alert when such actions are discovered. Monitoring and analysing network traffic data are crucial to detecting potential attack patterns. Companies and Information Technology (IT) enterprises worldwide have been investing in data science to develop more intelligent Intrusion Detection Systems (IDS) to avert malicious threats and improve cybersecurity. An IDS is a software application that examines a network or a system for dangerous activity or policy violations [3]. Such systems combine a group of methods from the fields of Computer Science, Statistics, and Information Technology referred to as Machine Learning (ML).

In IDS systems, the system decides on what type of abnormalities or attack is about to occur. The various types of Intrusion detection include the host-based (HIDS) and network-based IDS (NIDS). The HIDS resides in a particular computer system (host) and screens the operating system files for volatilities and irregular incidences, while in the NIDS, the network connections are monitored and examined

for malicious events. Furthermore, there are two approaches to detection, these are signature-based and anomaly-based detection. In signature-based IDS, bytes pattern along the network path is examined, whereas, in the anomaly-based IDS, the behaviour of the network is scanned for patterns, which then automatically builds a data-driven model that profiles expected behaviour and detect deviations in the event of any anomalies. Several machine learning algorithm classification techniques, such as Bayesian Network, Naïve Bayes, Random Forest, Decision Tree, and Decision table, have been used to provide intelligent cyber security services, especially for intrusion detection [3]. This study, therefore, modelled and assessed the performance of K-Nearest Neighbour Classification algorithm for both binary and multi-class classification of the network-based intrusion detection.

## 2. RELATED WORKS

In [4], an intrusion detection system for identification of attacks on a computer network was developed. By categorizing the attacks into their various classes, the KDDCup99 Test datasets were evaluated utilizing the machine learning methods such as Bayes Net, J48, Random Forest, and Random Tree. The results obtained revealed that Random Forest and Random Tree algorithms proved superiority in classifying Test dataset. The performance of J48, Random Forest, Random Tree, Multilayer Perceptron, Naive Bayes, and Bayes Network classifier for network anomaly detection was investigated in [5]. Random forest proved its superiority with the highest accuracy rate in identifying and categorizing network anomalies into multiclass of DOS, R2L, U2R, and PROBE

The accuracy of three machine learning algorithms for network intrusion on the DARPA KDD CUP 1999 dataset was reviewed and analysed in [6]. WEKA, a large data and machine learning tool, was used in this study to assess the performance of the Naive Bayes, decision tree, and random forest algorithms as they were trained and tested on the DARPA KDD CUP 1999 dataset. The performance of these algorithms was evaluated using precision, sensitivity, and accuracy.

In [7], classification and predictive intrusion detection models using Logistic Regression, Gaussian Naive Bayes, Support Vector Machine, and Random Forest were presented. These techniques were tested with the NSL-KDD data set. Random Forest Classifier recorded the highest accuracy of 99% compared to other techniques in classifying data traffic into normal or attack class. The study suggested that further work could be extended to multi-class classification and

considering only the important attributes for intrusion detection.

The classification performances of Naive Bayes, J48, and Random Forest were compared on a 20% KDD\_NSL dataset in [8], and Random Forest outperformed the Naive Bayes and J48 regarding both accuracy and detection rate. The three techniques attained precision and recall up to 90% each, thus, future works can combine the three techniques for improved performance and compare the result.

In [9], the performance of REPTree and Voting Feature Interval (VFI) algorithms in distinguishing between legitimate and malicious activity on the system for intrusion detection on the KDD Cup dataset was compared using the Weka tool. The receiver operating characteristic curve (ROC) was used for comparing classification algorithms. The findings from the analysis revealed that the REPTree learning algorithm recorded a higher accuracy and a lower error rate than the VFI.

A revised Naive Bayes (RNB) classifiers classifier for Intrusion detection was proposed by [10]. By applying the 10-fold cross-validation, the performance of the study was compared with the standard Naive Bayes, J48, and REPTree on the NSL-KDD data set. The proposed technique provided superior performance in terms of accuracy of detection.

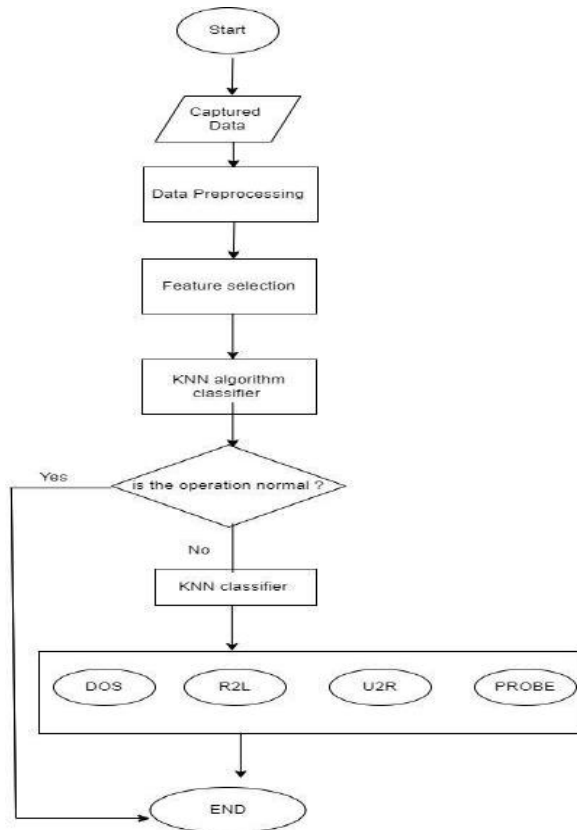
In [11], the performance of the Support Vector Machine (SVM), Naive Bayes, K-Nearest Neighbour (KNN), and the Decision Tree-based C4.5 (J48) and Random Forest algorithms for attack detection was evaluated. The study aimed at investigating the possibility of newer datasets and the most advanced, such as the NSL-KDD and GurKDDcup datasets, offering improved performance compared to the older and overused DARPA KDD Cup 1999 as well as to develop a method that would be useful for reducing the complexity of the first-phase classification models that were developed by narrowing the feature's domain. The results obtained showed that the Random Forest and KNN algorithms worked best with the three datasets with ROC of 1.000.

A decision tree model for an anomaly-based network intrusion detection system using the categorical characteristics from the Change Control Identifiers (CCIDS) 2017 dataset encoded with a label encoder was presented by [12]. Some best features were selected. Training data was used to form a Decision-Tree-Model where each leaf indicates the possible result. Test data obtained 99% accuracy, with 99.9% true positive and 0.1% false positive rates. The model used new dataset for training and test as against the traditional systems,

which was modelled using the KDD-CUP-99 data set. This work provided a basis for any new algorithm using the CCIDS 2017 dataset.

### 3. METHODOLOGY

This section presents the data description, classification process and algorithm for the study as depicted in Fig. 1.



**Fig. 1 The IDS classification process**

#### 3.1 Data Description

Data was obtained from the National State and Local Knowledge Discovery in Database (NSL-KDD) data set, which is an improved version of the KDD'99 data set, consisting forty-three (43) distinct features of traffic streams. The target classes of the NSL-KDD can be seen in two ways:

- i. The binary class: the target is simply '1 or 0', '0' for normal activities, '1' for malicious activities. Different types of attack are generalised as '1'.
- ii. As Multi-class: has multiple target responses rather than binary values. There were 22 distinct classes of attacks in the dataset, implying that the model would attempt to classify 22 possible target responses (Akshaya, 2016). However, this study selected four classes, namely, Denial of Service (DoS),

Remote to User attacks (R2L), User to Root (U2R) and Probe.

#### 3.2 Data Pre-processing

The data collected were organised converted to a format suitable for the machine learning program to deploy. The data are stored as NumPy arrays, feature extraction was performed to select relevant attributes to the experiment. One of the 43 features contained in the dataset, "difficulty level" was dropped due to inaccuracy of the feature. Data normalization was carried out using the standard score of a sample  $x$  is calculated as:

$$z = \frac{x - u}{s} \quad (1)$$

where  $u$  denotes the mean of the training samples or '0' if with mean = False and  $s$  is the standard deviation of the training samples or '1' if with\_std = False. The data was then split into training and test datasets. Seventy five percent (75%) was used for training, while, twenty five percent (25%) was reserved for testing.

#### 3.3 k-Nearest Neighbors Classification Algorithm

k-Nearest (KNN) is a simple and accurate machine learning algorithm that classifies objects based on the proximity of training object in a feature space. It is well known for regression and classification. KNN is a non-parametric algorithm that makes no assumptions about the structure of data and its distribution. This is an advantage because real-life data scarcely obeys theoretical rules [13]. KNN is directed at estimating the classification of a new unseen object using the classification of the object closest to it. In the  $k$  nearest neighbours, the classification is based on those of the  $k$  nearest neighbours (where  $k$  is a small integer such as 3 or 5), instead of nearest one [14], fixed number ( $k$ ) of training instances closest to the input instance. This implies that for a selected  $k$  value, an input object would be allocated to the same class as the closest number of  $k$  objects nearest to it. An odd number ' $k$ ' should also be chosen to prevent ties in the majority vote [13].

##### Basic k-Nearest Neighbour Classification Algorithm [14]

1. Find the  $k$  training objects that are closest to the unseen objects
2. Take the highest occurring classification for these  $k$  instances.

### 3.3.1 Distance Measurement

There are a number of ways to measure the distance, **dist.**  $(X, Y)$  between two points  $X$  and  $Y$ , in  $n$ -dimensional space., with three conditions required as follows:

1. The distance of any point  $X$  from itself is zero, i.e. **dist.**  $(X, X) = 0$ .
2. the *symmetry condition*: The distance from  $X$  to  $Y$  is the same as the distance from  $Y$  to  $X$ , i.e. **dist.**  $(X, Y) = \mathbf{dist.} (Y, X)$ ,
3. The *triangle inequality*, i.e. ‘the shortest distance between any two points is a straight line’. The condition says that for any points  $X, Y$  and  $Z$ : **dist.**  $(X, Y) \leq \mathbf{dist.} (X, Z) + \mathbf{dist.} (Z, Y)$ .

This study adopted the Euclidean Distance, which is the most popular distance measures among others that exist. The Euclidean distance between points  $(x_1, x_2, \dots, x_n)$  and  $(y_1, y_2, \dots, y_n)$  in  $n$ -dimensional space is defined as [14]. Euclidean distance,

$$\begin{aligned} d(x, y) &= d(y, x) \\ &= \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \dots + (x_n - y_n)^2} \\ &= \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \end{aligned} \quad (2)$$

where  $x, y$  denote, respectively, two points in Euclidean  $n$ -space;  $x_i, y_i$  are Euclidean vectors, starting from the origin of space; and  $n$  denotes the  $n$ -dimensional space.

### 3.4 Classification Evaluation Metrics

The following evaluation measures: classification accuracy, precision, F1 score and recall, computed from Confusion matrix were used to assess the performance of the classification algorithm. There are four essential parameters of the confusion matrix are

- i. True Positives (TP): computes the instances of the dataset in which the model predicted positive, whose actual outputs are positive
- ii. True Negatives (TN): computes the instances of the dataset in which the model predicted negative, whose actual outputs are negative
- iii. False Positives (FP): computes the instances of the dataset in which the model predicted positive, whose actual outputs are negative
- iv. False Negatives (FN): computes the instances of the dataset in which the model predicted negative, whose actual outputs are positive

The performance measures were, thus, calculated as follows:

(i) **Accuracy:** Accuracy is the average of the true parameters and the total number of instances of the data set, defines as

$$\begin{aligned} \text{Accuracy} &= \frac{TP + TN}{TP + TN + FP + FN} \end{aligned} \quad (3)$$

(ii) **Recall:** (also called sensitivity, hit rate, or true positive rate) is the number of correct positive predictions divided by the all-positive instance of the dataset, given

$$\text{recall} = \frac{TP}{TP + FN} \quad (4)$$

(iii) **Precision:** the number of correct positive predictions divided by the number of predicted positives, given as

$$\text{precision} = \frac{TP}{TP + FP} \quad (5)$$

(iv) **F1 score/measure:** F1 Score/Measure is the Harmonic Mean between precision and recall. It denotes how many instances it classifies correctly, as well as how well it does not miss a significant number of instances, defines as

$$\begin{aligned} \text{F1} &= \frac{1}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} \end{aligned} \quad (6)$$

## 4. RESULTS AND DISCUSSION

This section presents the implementation environment, results and discussion of the findings of the study.

### 4.1 Implementation Environment

The following python tools were employed for the implementation, among others, on Goggle collab environment.

1. Numpy: offers a wide range of mathematical functions, random number generators, and linear algebra routines, etc.
2. Pandas: created for data manipulation and analysis.
3. Sklearn: for classification and regression
4. Matplot: for plotting

### 4.2 Classification Results

The classifications results are presented in this section.

### 4.2.1 Binary Classification

A copy of the DataFrame was created for Binary Classification, two categories of attack, 'normal' and 'abnormal' were defined under attack label ('label' attribute). LabelEncoder() was used to encode the label and saved in 'intrusion', 'label' is one-hot-encoded as shown in Fig. 2.

```
In [4]: # dataset with binary labels and label encoded column
bin_data.head()

Out[4]:
duration  protocol_type  service  flag  src_bytes  dst_bytes  ...  wrong_fragment  urgent  hot  num_failed_logins  logged_in  num_compromised
0  0.110243  tcp  http  SF  -0.007758  -0.004916  ...  -0.004946  -0.007758  -0.009075  -0.007023  -0.004362  -0.010564
1  1.011043  udp  other  SF  -0.007757  -0.004916  ...  -0.004946  -0.007758  -0.009075  -0.007023  -0.004362  -0.010564
2  2.011043  tcp  private  SI  -0.007757  -0.004916  ...  -0.004946  -0.007758  -0.009075  -0.007023  -0.004362  -0.010564
3  3.011043  tcp  http  SF  -0.007757  -0.004916  ...  -0.004946  -0.007758  -0.009075  -0.007023  -0.004362  -0.010564
4  4.011043  tcp  http  SF  -0.007758  -0.004916  ...  -0.004946  -0.007758  -0.009075  -0.007023  -0.004362  -0.010564
```

Fig. 2 Dataset with binary labels and label encoded column

Fig. 3 shows the attack classification labels categorized into normal and abnormal and Fig. 4 presents the distribution.

```
In [33]: # changing attack labels into two categories 'normal' and 'abnormal'
bin_label = pd.DataFrame(data.label.map(lambda x: 'normal' if x == 'normal' else 'abnormal'))

In [34]: # creating a dataframe with binary labels (normal,abnormal)
bin_data = data.copy()
bin_data['label'] = bin_label

In [35]: # label encoding (0,1) binary labels (abnormal,normal)
lel = preprocessing.LabelEncoder()
enc_label = bin_label.apply(lel.fit_transform)
bin_data['intrusion'] = enc_label

In [36]: lel.classes_

Out[36]: array(['abnormal', 'normal'], dtype=object)

In [41]: np.save("lel_classes.npy",lel.classes_,allow_pickle=True)
```

Fig. 3 Attack classification into normal and abnormal

The classification result is depicted in Fig. 4. 53.46% for normal and 46.54% for abnormal traffic streams.

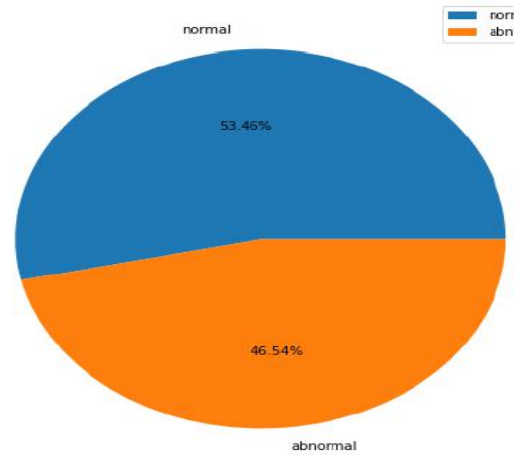


Fig. 4 Classification distribution into normal and abnormal.

### 4.2.2 Multi-class classification

A copy of DataFrame was created for Multi-class classification, attack label ('label' attribute) was classified into five groups 'normal', 'U2R', 'R2L', 'Probe', 'Dos'. The 'label' was encoded using LabelEncoder() and saved in 'intrusion.' The 'label' is one-hot-encoded as shown in Fig. 5.

```
# creating a dataframe with multi class labels (Dos,Probe,R2L,U2R,normal)
multi_data = data.copy()
multi_label = pd.DataFrame(multi_data.label)

# label encoding (0,1,2,3,4) multi class labels (Dos,normal,Probe,R2L,U2R)
lel = preprocessing.LabelEncoder()
enc_label = multi_label.apply(lel.fit_transform)
multi_data['intrusion'] = enc_label

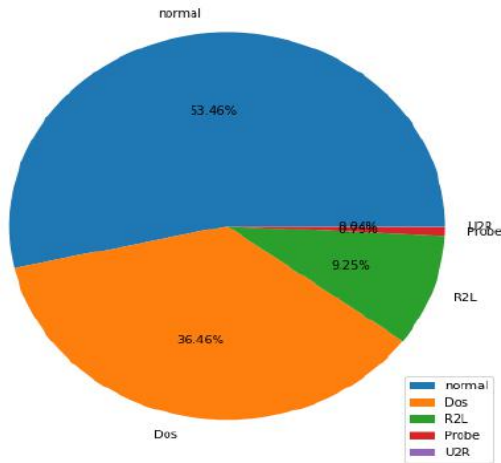
np.save("lel_multi_classes.npy",lel.classes_,allow_pickle=True)

# one-hot encoding attack label
multi_data = pd.get_dummies(multi_data,columns=['label'],prefix="","prefix_sep="")
multi_data['label'] = multi_label
multi_data
```

wrong_fragment	urgent	hot	...	dst_host_srv_error_rate	dst_host_error_rate	dst_host_srv_error_rate	intrusion	Dos	Probe	R2L	U2R	normal	label
0.00496	0.00726	0.00975	...	0.02477	0.02462	0.03637	4	0	0	0	0	1	normal
0.00496	0.00726	0.00975	...	0.02477	0.02765	0.03637	4	0	0	0	0	1	normal
0.00496	0.00726	0.00975	...	1.01855	0.00705	0.03637	0	1	0	0	0	0	Dos
0.00496	0.00726	0.00975	...	0.02492	0.02765	0.04004	4	0	0	0	0	1	normal
0.00496	0.00726	0.00975	...	0.02477	0.02765	0.03637	4	0	0	0	0	1	normal

Fig. 5: Creating multi-class labels

The percentage distribution of the multiclass classification is shown in Fig. 6.



**Fig 6: Classification percentage distribution of the multiclass labels**

The classification shows that was classified as normal, 36.46% as DOS, 9.25% as R2L, 0.64% as U2R and 0.19% as Probe.

### 4.3 Classification Evaluation

The classification performance of the algorithm is described in this section.

#### 4.3.1 Binary Classification

The screenshot of the code segment of the binary classification and the output of the evaluation metrics is shown in Fig. 7.

```

y_pred=knn.predict(X_test) # predicting target attribute on testing dataset
ac=accuracy_score(y_test, y_pred)*100 # calculating accuracy of predicted data
print("KNN-Classifier Binary Set-Accuracy is ", ac)

KNN-Classifier Binary Set-Accuracy is 99.9968095852352

[ ] # classification report
print(classification_report(y_test, y_pred, target_names=le1.classes_))

precision  recall  f1-score  support
abnormal   1.00    1.00    1.00    11833
normal     1.00    1.00    1.00    13362
    
```

**Fig 7: The binary classification performance evaluation**

The classification shows an accuracy of 99.996% with 1.00 for precision, Recall, F1 Score and 11833 supports for the normal class cation and 1.00 for precision, Recall, F1 Score and support of 13362 for abnormal classification.

#### 4.3.2 Multi-class Classification

The performance of the multiclass classification is depicted in the screenshot code segment as Fig. 8, with classification accuracy of 99.988. The precision, Recall, F1 Score and supports for each of the abnormal class category is shown in table 3.

```

[ ] y_pred=knn.predict(X_test) # predicting target attribute on testing dataset
ac=accuracy_score(y_test, y_pred)*100 # calculating accuracy of predicted data
print("KNN-Classifier Multi-class Set-Accuracy is ", ac)

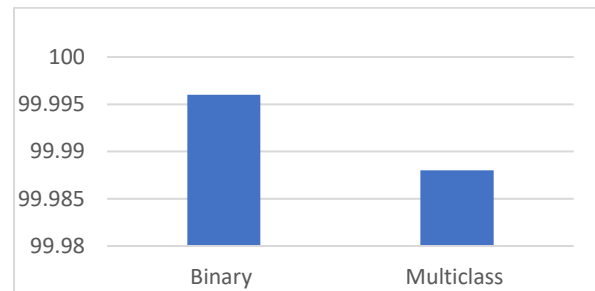
KNN-Classifier Multi-class Set-Accuracy is 99.9880926757054
    
```

**Fig 8: Classification accuracy of the multiclass**

**Table 2. The other performance measures of the categories of the abnormal class.**

Abnormal class	Precision	Recall	F1 Score	Support
DOS	1.00	1.00	1.00	9339
Probe	1.00	1.00	1.00	13362
R2L	1.00	1.00	1.00	2306
U2R	0.99	1.00	1.00	178

Fig. 9 shows the accuracy distribution of the binary and multi class classification.



**Fig 9: Accuracy of the classification types compared.**

The Binary classification accuracy was marginally higher than the multiclass by 0.008%. Overall, the K NN model has shown an impressive performance for intrusion detection, and therefore could be effectively deployed for in an intrusion detection system.

## 5. CONCLUSION

This study modelled and assessed the performance of KNN for intrusion detection, implemented using python. on the NSL-KDD, an online dataset obtained from Kaggle. A binary (normal and abnormal) and multi-class classification of the abnormal activities (*Denial of service, Remote to user Attack, User to root, and probe*) of network intrusion were examined. The results showed the accuracy of the binary classification as 99.99%, and that of the multi-class classification as 99.98%. The findings indicate that the KNN could be used as a powerful tool for developing an Intrusion Detection System.

## 6. ACKNOWLEDGMENTS

The authors wish to appreciate Redeemer's University, Ede, Nigeria, for providing laboratory, power and internet connection facilities during the research and preparation of the manuscript.

## 7. REFERENCES

- [1] I. Lee and K. Lee, "The Internet of Things (IoT): Applications, investments, and challenges for enterprises," *Business Horizons*, vol. 58, no. 4, pp. 431-440, 2015.
- [2] S. Goel, K. Williams and E. Dincelli, "Got phished? Internet security and human vulnerability," *Journal of the Association for Information Systems*, vol. 18, no. 1, pp. 22-44, 2017.
- [3] I. H. Sarker, Y. B. Abushark, F. Alsolami and A. I. Khan, "IntruDTree: A Machine Learning Based Cyber Security Intrusion Detection Model," *Symmetry*, 2020.
- [4] C. J. Ugochukwu and E. O. Bennett, "An Intrusion Detection System Using Machine Learning Algorithm," *International Journal of Computer Science and Mathematical Theory*, vol. 4, no. 1, pp. 39-47, 2018.
- [5] I. Obeidat, N. Hamadneh, M. Alkasassbeh, M. Almseidin and M. I. AlZubi, "Intensive Pre-Processing of KDD Cup 99 for Network Intrusion Classification Using Machine Learning Techniques," *International Journal of Interactive Mobile Technologies*, vol. 13, no. 1, 2019.
- [6] N. A. Azeez, A. O. J. S. Misra, A. Adewumi, R. Ahuja and R. Maskeliunas, "Comparative evaluation of machine learning algorithms for network intrusion detection using Weka.," in *Towards Extensible and Adaptable Methods in Computing.*, 2018, pp. 195-208.
- [7] M. C. Belavagi and B. Muniyal, "Performance evaluation of supervised machine learning algorithms for intrusion detection," *Procedia Computer Science*, vol. 89, pp. 117-123, 2016.
- [8] N. Ashraf, W. Ahmad and R. Ashraf, "A comparative study of data mining algorithms for high detection rate in intrusion detection system," *Annals of Emerging Technologies in Computing (AETiC)*, vol. 2, no. 1, pp. 49-57, 2018.
- [9] N. Maharaj and P. Khanna, "A comparative analysis of different classification techniques for intrusion detection system," *International Journal of Computer Applications*, vol. 95, no. 17, 2014.
- [10] K. Kumar and J. S. Batth, "Network intrusion detection with feature selection techniques using machine-learning algorithms," *International Journal of Computer Applications*, vol. 150, no. 12, 2016.
- [11] N. Sainis, D. Srivastava and R. Singh, "Feature classification and outlier detection to increased accuracy in intrusion detection system," *Journal of Applied Engineering Research*, vol. 13, no. 10, pp. 7249-7255., 2018.
- [12] R. A. Jamadar, "Network intrusion detection system using machine learning," *Indian Journal of Science and Technology*, vol. 7, no. 48, pp. 1-6, 2018.
- [13] K. Chumachenko, "Machine learning methods for malware detection and classification.," XAMK, 2017.
- [14] M. Bramer, *Principles of Data Mining*, London: Springer-Verlag London Ltd., 2016.