International Journal of Intelligent Information Technologies
Volume 17 • Issue 4

Modelling and Evaluation of Network Intrusion Detection Systems Using Machine Learning Techniques

Richard Nunoo Clottey, University of Ghana, Ghana
Winfred Yaokumah, University of Ghana, Ghana
https://orcid.org/0000-0001-7756-1832
Justice Kwame Appati, University of Ghana, Ghana
https://orcid.org/0000-0003-2798-4524

ABSTRACT

This study aims at modelling and evaluating the performance of machine learning techniques on a recent network intrusion dataset. Five machine learning algorithms, which include k-nearest neighbour (KNN), support vector machines (SVM), voting ensemble, random forest, and XGBoost, have been utilized in the development of the network intrusion detection models. The proposed models are tested using the UNSW_NB15 dataset. Three different K values are used for model with KNN algorithm and two different kernels are utilized in the development of the model with SVM. The best detection accuracy of the model developed with KNN was 84.9% with a K value of 9; the SVM model with the best accuracy is developed with the Gaussian kernel and obtained an accuracy of 83%, and the Voting Ensemble achieved 83.4% accuracy. Random forest model achieved accuracies of 90.2% and 70.8% for binary classification and multiclass classification respectively. Finally, XGBoost model also achieves accuracies of 85% and 51.77% for binary and multiclass classification respectively.

KEYWORDS

Cyber-Attacks, K-Nearest Neighbour, Machine Learning Algorithms, Modelling, Network Intrusion Detection, Random Forest, Support Vector Machines, UNSW-NB15 Dataset, Voting Ensemble, XGBoost

INTRODUCTION

Computer networks are evolving significantly as a result of rapid developments in Internet of Things (IoT), wireless handled devices, networks of moving vehicles, 4G and 5G, and cyber-physical systems (Kasongo & Sun, 2020). These technologies enable the exchange of huge amount of data, thereby making them susceptible to various malicious actions, security threats, and cyber-attacks (Kasongo & Sun, 2020). Cyber-attacks usually exploit vulnerabilities in the current network ecosystem. These attacks target sensitive information and can interrupt the availability of the system. Notable among these attacks are routing attacks, Denial of Service (DoS) attack, flooding attacks, data leakage,

DOI: 10.4018/IJIIT.289971

Distributed Denial of Service (DDoS), spoofing, wormhole attacks, and insecure gateways (Santos et al., 2019; Wazid et al., 2019; Deshmukh-Bhosale & Sonavane, 2019). These attacks and vulnerabilities enable theft of information on computer network systems (Hajisalem & Babaie, 2018), disruption of business operations, and breach of confidentiality, integrity and availability of the information system resources, thereby leading to great financial loss (Faker & Dogdu, 2019). For example, in 2017 the cost on the global economy was $600 billion due to cyber-attacks or systems vulnerabilities and that cost rose to $1 trillion in 2018 (www.technology.Org, 2019).

As network attacks become sophisticated, intelligent network intrusion detection systems are developed to detect and prevent these attacks. Intelligent intrusion detection systems (IDS) can detect unauthorized access (Gendreau & Moorman, 2016), analyze the activities in the network to prevent malicious behavior from disrupting the network (Choudhary & Kesswani, 2018). According to Choudhary and Kesswani (2018), the intelligent devices themselves are vulnerable and exposed to cyber-attacks. Consequently, machine learning (ML) algorithms have proven to be an efficient method for detecting network intrusions. With machine learning algorithms, the essential part is the dataset used. The dataset used should be as definite as possible because little changes to the data in the dataset can cause lots of different outcomes in the detection of the attacks. There are various ML approaches which are used to detect attacks on the networks. Some of the prominent approaches used are deep learning techniques (Al-hawawreh et al., 2019), random neural networks (Saeed et al., 2016), binary logistic regression (Ioannou & Vassiliou, 2018), K-Nearest Neighbor (KNN) (Li et al., 2014), Naïve Bayes classification (Mehmood et al., 2018), and neighbor discovery protocol (Alsadhan et al., 2019).

This study aims at developing machine learning models to improve the detection of network intrusions with a recent network intrusion dataset and evaluating the performance of the developed models. Supervised machine learning techniques are utilized in the implementation of the intrusion detection models. The supervised machine learning techniques employed in this paper are the K-Nearest Neighbor (KNN) Classification algorithm, the Support Vector Machine (SVM) classification algorithm, Voting Ensemble, Random Forest and eXtreme Gradient Boosting (XGBoost) algorithm. Three different values for K are used in the K-Nearest Neighbor model, and two different kernels are used in the Support Vector Machine model. Random Forest and XGBoost are used for binary and multiclass classification. Ten features of the UNSW_NB15 dataset are used in the implementation of the proposed K-Nearest Neighbor (KNN) Algorithm, the Support Vector Machines (SVM), Voting Ensemble, Random Forest Classifier, and XGBoost models.

LITERATURE REVIEW

Supervised Learning Algorithms

Supervised learning is a family of machine learning techniques that are used for searching for the design of computational models which are capable of learning patterns from explained data and to classify new sets of unseen data automatically (Granjal et al., 2018). The performance of supervised learning algorithms depends on the quality of the data and the parameters used in configuring the model. Some important supervised machine learning algorithms, including K-Nearest Neighbor (KNN) Algorithm, the Support Vector Machines (SVM), Voting Ensemble, Random Forest Classifier, and XGBoost Algorithm, are discussed in the following section.

- **K-Nearest Neighbor (KNN) Algorithm:** One of the algorithms used for performing the detection of intrusions is the K-Nearest Neighbor (KNN). K-nearest neighbor (KNN) classification algorithm is a data mining algorithm which is matured theoretically with very low complexity. KNN is a sample-based learning and classification algorithm. The idea of the k-nearest neighbor

classification algorithm is that, when in a sample space, if most of its K nearest neighbor samples belong to a particular category, then the sample belongs to that same category (Li et al., 2014). The nearest neighbor refers to a single or multidimensional feature vector that is used in describing the sample on the closest criteria, and the closest criteria can be the Euclidean distance of the feature vector. In the intrusion detection algorithm, the n-dimensional vector to represent nodes is used. Some examples of these dimensions can be the number of nodes with the same source inter-packet arrival time, the number of nodes that contain the same service, and same address. Therefore, nodes of the same type have the same characteristics.

- **Support Vector Machines (SVM):** Support Vector Machine (SVM) sets sight on finding a hyper-plane which is used for classifying all the training instances into different class (binary classification or multiclass classification) (Suykens & Vandewalle, 1999). SVM algorithm takes observed samples and its associated outputs, that is, binary or N-ary. The SVM algorithm then designs a model that can be used to classify new samples into different classes. Training samples are then mapped as points in coordinate space, thereby, dividing the samples or instances linearly. There can be many hyper-planes that can divide or partition the training samples, but the best choice of the hyper-plane will be that which has the maximum distance from the nearest instance of any class. In a case of two different hyper-planes A and B, hyper-plane A can classify the samples or instances correctly but has less distance from the nearest sample and hyper-plane B has maximum distance but has a small error when classifying. Hyper-plane A will be selected in such a case. SVM is useful for high dimensional spaces.

Support Vector Machines (SVMs) are powerful classification methods that use suitable amounts of computational resources for performance. Due to the computational requirements and the time-consuming process of learning, the use of Neural Networks is infeasible; therefore, this is an import requirement on these systems. Support Vector Machines (SVMs) have a faster procedure for classification which aids with a real-time implementation and based on the kernel function used; they have the advantage of being non-linear classifiers (Granjal et al., 2018). Support Vector Machines (SVM) have a feature called kernel function which is used for mapping linear algorithms into non-linear space. Polynomial and radial basis function are kernel functions which are used to construct a hyper-plane by dividing the feature space. At the time of training of the classifiers, the kernel function selects support vectors along the surface of the function. Classification of data by SVM uses the support vectors that are used to outline the hyperplane in the feature space (Mulay et al., 2010). A principled approach to machine learning problems is offered by the support vector machine (SVM) because of its mathematical foundation in statistical learning theory. A subset of the training input is utilized to form its solution by the SVM. SVM has been widely used for classification, novelty detection tasks, feature reduction, and regression.

- **Random Forest:** According to Cutler et al. (2012), Random Forest was developed by Leo Breiman who was influenced decisively by the early work of Amit and Geman on geometric feature selection and was developed to be a challenger for boosting. Random Forest can be utilized for performing classification (a categorical response variable) or regression (a continuous response). Random Forest can handle both classification and regression, are fast to train and predict, can be used for high-dimensional issues or problem, and depend only on one or two tuning parameters. Random Forests are also thought to be one of the most accurate general-purpose learning techniques (Biau, 2012).
- **eXtreme Gradient Boosting (XGBOOST):** XGBoost is an enhanced and scalable variety of the gradient boosting algorithm which is designed for effectiveness, speed of computation, and performance of the model. The XGBoost was developed to increase the accuracy of existing boosting techniques in the shortest amount of time (Malik et al., 2020).

KNN algorithm is chosen for this study because KNN is a non-parametric model as compared to linear regression, Naïve Bayes, and logistic regression which are parametric models. Also, KNN algorithm supports non-linear solution compared to Logistic Regression which supports only linear solutions. When the data used has high signal-to-noise ratio (SNR), KNN performs better than linear regression. Furthermore, Neural networks needs large training data before it can outperform KNN and also neural networks needs a lot of hyper parameter tuning as compared to KNN. SVM is also chosen for this study because SVM supports both linear and non-linear solutions as compared to logistic regression which can handle only linear solutions (Varghese, 2018). SVM also handles outliers better than linear regression and logistic regression.

Random Forest Classifier is selected over Decision Trees because Random Forest overcomes or solves the over fitting problem by using the concept of Bagging. Random Forest Classifier reduces or lowers variance and thereby reduces over fitting by creating various tress and identifying the best split function from a random subset of features for each of the trees. XGBoost was chosen because it is an ensemble techniques whereby new models are added which aids in correcting errors or mistakes made by existing models. Decision Trees and the Boosting technique are the building blocks of the XGBoost.

RELATED WORKS

Deshmukh-Bhosale and Sonavane (2019) designed an Intrusion Detection System (IDS) for Denial of Service (DOS) attack detection using Contiki OS and Cooja simulator. The Intrusion Detection System designed achieved a 93% detection rate and consumed less energy. Jahir Husain and Maluk Mohamed (2019) designed, implemented, and evaluated a novel intrusion detection and prevention mechanism called IMBF, which was used to secure the IoT from Denial of Sleep Attack. The proposed mechanism made use of a lightweight modularized system to detect and prevent the Denial of Sleep Attack by using malicious node alert messages. The proposed system consumed less amount of memory and had a better success rate for detecting the denial of sleep attacks. Venkatraman and Surendiran (2019) proposed an adaptive hybrid Intrusion Detection System (IDS) based on timed automata controller approach. The proposed hybrid IDS was developed to detect intruders in IoT networks. The proposed method achieved a 99.06% accuracy in detecting Denial of Service (DoS) attacks, control hijacking attacks, zero-day attacks, and replay attacks in the IoT environment. Manso et al. (2019) designed and implemented a Software-Defined Intrusion Detection System (IDS) that reactively impairs the attacks at its origin, ensuring the regular operation of the infrastructure of the network. The proposed IDS automatically detect Distributed Denial of Service (DDoS) attacks and notifies a Software-Defined Networking (SDN) controller. The proposed Intrusion Detection System detected the attacks with an average DDoS mitigation time of 0.07s, an average Round Trip Time (RTT) of 0.541 milliseconds, and with no packet loss.

Mehmood et al. (2018) presented a new approach of using Naïve Bayes classification algorithm applied to intrusion detection systems (IDSs) to protect an IoT infrastructure from distributed denial of service attacks generated by intruders. The IDSs are sent in the form of multi-agents throughout the network to sense the misbehaving traffic nodes. The detection of the distributed denial of service attacks was performed very fast. Li et al. (2014) proposed a new intrusion detection system based on the K-nearest neighbor (KNN) classification algorithm in a wireless sensor network. The proposed system was able to separate abnormal nodes from normal nodes by observing their abnormal behaviours. The proposed system achieved high detection accuracy and speed.

Alzahrani et al. (2020) examined the significant and discriminative features to recognize various attacks by using Support Vector Machine (SVMs) methods and Structural Sparse Logistic Regression (SSPLR). The proposed techniques (SVMs and SSPLR) improved the performance compared to other techniques used for Intrusion Detection Systems (IDS). Su et al. (2020) proposed a traffic anomaly detection model called BAT. The proposed model (BAT) combined Bidirectional Long Short-term

memory (BLSTM) and attention mechanism. The proposed model achieved high accuracy rates and performed better compared to other methods. Syarif et al. (2019) proposed reducing the dimensions of the dataset in the preprocessing step by using different state-of-the-art dimension reduction techniques such as Principal Component Analysis (PCA) and Information Gain (IG). The proposed technique achieved a better detection rate than other methods. Devi (2019) presented an Intrusion Detection System using Data mining based Enhanced Framework (DEF). The presented model is aided by the K-means Clustering and Decision Tree (DT) classification techniques in which genetic algorithms (GA) can be used. The proposed model achieved a good accuracy of detection. Anthi et al. (2018) described the initial stages of developing a novel IDS for Internet of Things (IoT) which utilized Machine Learning (ML) techniques called Pulse. The described system (Pulse) was capable of detecting forms of Denial of Service (DoS) attacks. The model was better at detecting probing attacks than flood-type attacks and achieved a detection accuracy of about 97.7%.

Choudhary & Kesswani (2018) designed a detection and prevention algorithms which were Key Match Algorithm (KMA) and Cluster-Based Algorithm (CBA) in MatLab Simulation environment. The designed algorithms were used to detect and prevent routing attacks such as sinkhole and selective forwarding. The results of the two algorithms designed were compared with KMA achieving a detection rate between 50% to 80% whiles the CBA achieved a detection rate between 76% to 96%. Jafier (2018) introduced an innovative approach for Intrusion Detection System (IDS) which is dedicated to building Data Mining (DM) based IoT IDS (IoTIDS). The IoTIDS identified attacks by employing Singular Value Decomposition (SVD) techniques. Chawla & Thamilarasu (2018) proposed a new intrusion detection system that utilizes machine learning algorithms to detect security anomalies in IoT networks. Kasinathan, Costamagna, et al. (2013) presented an Intrusion Detection System (IDS) framework for IoT empowered by IPv6 over low-power personal area network (6LoWPAN) devices. Govindasamy & Punniakodi (2017) proposed an Energy Efficient Intrusion Detection System (EE-IDS) and Energy Efficient Intrusion Detection System with Energy Prediction (EE-IDSEP) for improving the security of ZigBee based wireless sensor networks against wormhole attacks and Distributed Denial of Service (DDoS) attacks. The proposed system obtained better performance as compared to other existing systems in terms of Packet Delivery Ratio (PDR), energy consumption, detection rate, false-positive rate, average end-to-end delay, and average detection time. Fu et al. (2017) analyzed the requirements for intrusion detection of IoT networks and proposed a uniform intrusion detection method for heterogeneous IoT networks based on an automata model. The proposed method detected IoT attacks such as jam-attack, false-attack, and reply-attack.

Jiang et al. (2020) presented a network intrusion detection algorithm which combined hybrid sampling with deep hierarchical network. One-side selection (OSS) was used to reduce the noise samples in majority category and increase the minority samples by Synthetic Minority Over-sampling Technique (SMOTE). The proposed network intrusion detection algorithm was used to experiment on NSL-KDD and UNSW-NB15 dataset and obtained a classification accuracy of 83.58% and 77.16% respectively. RBF-based multistage auto-encoders were used in detecting attacks by Kadhim & Mishra (2019). The presented technique involved a two-part multistage auto-encoder which was used to select top and sensitive features from input data and RBF which was used to categorize the input data into attack or no attack label. The proposed technique achieved an accuracy of 98.80% when applied to the UNSW-NB15 dataset. A Support Vector Machine (SVM) with a novel scaling method for binary-classification and multi-classification was presented by Jing & Chen (2019). The presented method obtained an accuracy of 85.99% for binary classification and 75.77% accuracy for multi-classification when the performance was evaluated.

Lin et al. (2019) adopted a time-related deep learning approach to detect network intrusions and a stacked sparse autoencoder (SSAE) was set up to extract the features with the greedy layer-wise strategy. A time-related intrusion detection system based on the variants on Recurrent Neural Network (RNN) was proposed and the performance of the approach was tested on the UNSW-NB15 dataset. The results of the experiment showed that the accuracy of the proposed approach was over

98% and the false alarm rate was as low as 1.8%. Khan & Derhab (2019) proposed a new two-stage deep learning (TSDL) model based on a stacked auto-encoder with a soft-max classifier for the detection of network intrusion. Probability score value was used to classify network traffic as normal or abnormal. Several experiments were conducted on KDD99 and UNSW-NB15 datasets to evaluate the effectiveness of the proposed model. High recognition rates of up to 99.996% and 89.134% were achieved for the KDD99 and UNSW-NB15 datasets respectively. An improved Intrusion Detection System (IDS) based on hybrid feature selection and two-level classifier ensembles was proposed by Tama et al. (2019). The hybrid feature selection was comprised of particle swarm optimization, ant colony algorithm, and genetic algorithm and was used for reducing the feature size of the training datasets which were the NSL-KDD and UNSW-NB15. An accuracy of 85.8%, sensitivity of 86.8%, and detection rate of 88.0% was obtained for the NSL-KDD dataset. Also, the results obtained for UNSW-NB15 datasets was better than other state-of-the-art techniques.

RESEACRH METHODOLOGY

Dataset Selection

The first phase of the detection of network intrusions involves the acquisition of the dataset. This study employs the UNSW_NB15 dataset, which is used for both training and testing. According to Husain et al. (2019), the UNSW_NB15 dataset constitutes modern day network attacks and network traffic as compared to the KddCup99 dataset. Also, the attributes of the Kdd99 dataset are less efficient than the UNSW_NB15 dataset (Moustafa & Slay, 2015a). The datasets which are frequently used for detection of intrusions in the network are UNSW_NB15 (Faker & Dogdu, 2019), NSL-KDD dataset (Mehmood et al., 2018), and KddCup99 dataset (Sharma et al., 2019). The UNSW_NB15 dataset was downloaded from Kaggle (https://www.kaggle.com/mrwellsdavid/unsw-nb15) in a CSV format. The UNSW_NB15 dataset was created by IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) for the generation of a hybrid of real modern normal activities and synthetic contemporary attack behaviors (Moustafa & Slay, 2015b). The UNSW_NB15 dataset contains 175341 different sets of data with 44 different features. The features of the UNSW_NB15 dataset are grouped into flow features, basic features, content features, time features, and additional generated features. This dataset has nine different attacks, namely, Backdoors, Denial of Service (DoS), Fuzzers, Worms, Reconnaissance, Generic, Exploits, Analysis, and Shellcode. Table 1 defines the various attacks.

Data Preprocessing

The second phase of the detection process is the data preprocessing stage. The dataset obtained (UNSW_NB15 dataset) is preprocessed; this is done to clean the data. In order to preprocess the dataset, normalization, label encoding for attack category as well as reduction of the dimension by applying Principal Component Analysis (PCA) are performed. Normalization is a technique where by real valued numeric attributes or values are scaled and shifted into the range of 0 and 1 (Brownlee, 2014). Min-Max Normalization was adopted for preprocessing. A reason for choosing or selecting the Min-Max Normalization is to keep the relationship between the original data in the dataset intact even after the scaling. Label Encoding is an approach which entails converting or encoding categorical values in a column into a number (Yadav, 2019). Label encoding for attack category in the dataset was done to assign a label to each of the categories of attacks. This was done in order to execute or carry out correlation analysis and training of a multiclass classification model. Eighty percent (80%) of the data in the UNSW_NB15 dataset was used for training whiles the remaining 20% was used for testing. The dataset is then uploaded into the intrusion detection models. The Pandas library in Python was imported and used to read the data for uploading into the intrusion detection models.

Table 1. Definition of the various attacks in the UNSW_NB15 dataset (Kumar et al., 2019)

Attack Type	Description
Backdoor	This is the process by which the authentication process is bypassed leading to unauthorized access to any network
Denial of Service (DoS)	This is an attack in which services of a network are disrupted making resources unavailable for the actual user's request
Fuzzers	This is when the attacker floods the network with random data to crash it and detect vulnerabilities in the network
Worms	This attack happens when malicious codes are replicated, and the attacker tries to enter the system using some vulnerabilities
Reconnaissance	This is the process whereby information relating to a network is gathered for the sole purpose of avoiding security controls
Generic	This is an attack performed by the attacker without caring about the crypto-graphical executions of primitives
Exploits	This is an attack in which the attacker makes use of vulnerabilities which are present in the network
Analysis	This is a type of attack used for web applications using spam emails, web scripts and others
Shellcode	This is an attack where a command shell is triggered by injecting codes into an application to take control of the machine.

Also, the Python library - Scikit-learn was imported and used for the learning process in the network intrusion detection models.

Feature Extraction

In the feature extraction stage, the features which will be used as inputs to the machine learning models for the detection of network intrusion are chosen. Devi (2019) based the feature selection on a general feature and a sub-set of features. The roles selected determine the feature identity, whether it was normal, unique, labelled, and others. Kumar et al. (2019) calculated the information gain value of each feature by applying Equations 1 – 4 and selecting only the features whose information gain value were greater than 0. Information gain value of a feature can be defined as the contribution of the feature for classifying the data set:

$$Information\ \ Gain\left(A\right) = Entropy\left(D\right) - Entropy_{A}\left(D\right) \tag{1}$$

$$Entropy\left(D\right) = -\sum_{i=1}^{m} p_{i} * log_{2}\left(p_{i}\right) \tag{2}$$

$$Entropy_{A}\left(D\right) = \sum_{j=1}^{v} \frac{\left|D_{j}\right|}{\left|D\right|} * Entropy\left(D_{j}\right) \tag{3}$$

$$p_i = \frac{|C_i|}{|D|} = \text{Probability than an arbitrary tuple in D belongs to class } C_i \quad (4)$$

Extra Tree Classifier was used to find the important features which will be used for the Random Forest Classifier and XGBoost Algorithm. Extra Tree Classifier which is a Tree based model was selected to obtain the essential features because it generates multiple or various trees and key decisions with decision tree are made with an attribute. Also, the cut point during tree construction is randomized by the Extra Tree Classifier and therefore, it leads to a very good reduction in variation. The features obtained by the Extra Tree Classifier for use by the Random Forest Classifier and XGboost Algorithm are Source Bytes (sbytes), packets count from same destination to the same source port (ct_dst_sport_ltm), mean packet size transmitted from source (smean), number of connections between same source and destination address (ct_dst_src_ltm), and source time to live value (sttl).

Filter method was also used to obtain features which were significant to the development of the KNN, SVM, and Voting Ensemble models. The filter method relies on the general characteristics of the training data in order to select feature with independence of any predictor (Sanchez-Marono et al., 2007). The filter method was done on the dataset using Weka. Table 2 shows the top 10 ranked attributes obtained from the filter method through Weka.

Software Tools and Hardware Platform

The study makes use of experimental research approach. The machine learning algorithms utilized for modelling, implementation, and evaluation of the network intrusions detection systems are the K-Nearest Neighbor (KNN) Algorithm, the Support Vector Machines (SVM), Voting Ensemble, Random Forest Classifier, and XGBoost Algorithm. PYTHON programming language is the technology which is used in designing the proposed models. The Python 3.7 programming language, in addition to Keras, Tensorflow, Numpy, Pandas, and Sklearn libraries, are be used to run the algorithm. The accuracy of the proposed intrusion detection models, the precision scores, F1 scores and recall values are used in evaluating the models. The experiment is done with the code written in the python programming language using Intel(R) Core(TM) i7, having a clock speed of 1.80GHz with 12 GB primary memory running on Windows 10 Home.

Table 2. Ranked Attributes obtained from the filter method

Feature	Ranking
Sbytes	1.6403
Smean	1.3322
Sload	1.268
Dbytes	0.9184
Label	0.9037
Dmean	0.7895
Rate	0.7529
Ct_dst_sport_ltm	0.7502
Ct_srv_dst	0.733
dur	0.7265

Development of the Models

The process of implementation of the network intrusion detection system starts with the importation of the necessary libraries. Libraries such as Pandas, Scikit-learn (Sklearn), and Numpy were imported. Other packages such as KNeighborClassifier, preprocessing, classification_report, and StandardScaler were imported from the Scikit-learn library. Then, the dataset that will be used for training and testing of the model, which is the UNSW_NB15 dataset was read with the Pandas library.

Afterwards, the labels of the dataset were encoded into appropriate integer values by using the LabelEncoder() function which can be obtained from the preprocessing package in the Scikit-learn library. The Scikit-learn library provides an efficient tool for the encoding of the levels of certain features into numeric values. The Label Encoder encodes labels with a value between 0 and n_classes – 1 where n is the number of distinct labels. Non-numerical data in the dataset are transformed into numerical values using the fit_transform() function. The fit_transform() function is used to fit and transform the data. Also, all the data under each label or column was converted into a list which returns a NumPy array. A NumPy array is a grid of values, all of which are of the same type, and indexed by a tuple of a non-negative integer. Ten (10) features or labels out of the forty-four (44) features which were dur, sbytes, smean, dmean, sload, dbytes, label, rate, ct_dst_sport_ltm, ct_srv_dst were used.

Next, the ten selected features of the UNSW_NB15 dataset were put into a list using the zip() functionality. The zip() function returns a zip object, which is an iterator of tuples where the first item in each passed iterator is paired together, and then the second item in each passed iterator is paired together, and so on. The returned list of the features of the dataset is given as the x values and the feature to be predicted, the network intrusion category, is given as the y values. The network intrusion category or y values are also converted into a list. The x values are then standardized using the StandardScaler() function package from the Scikit-learn library. Standardization of datasets is a common specification for many machine learning methods implemented in Scikit-learn. The machine learning methods might not work appropriately if the individual features do not look more or less like standard normally distributed data. The StandardScaler() function standardizes features by removing the mean and scaling to unit variance, that is, it transforms the data so that its distribution will have a mean value of 0 and a standard deviation of 1.

Next, the UNSW_NB15 dataset is split into the training dataset (x_train, y_train) and testing dataset (x_test, y_test). Eighty per cent (80%) of the dataset will be used as training dataset whiles the remaining twenty per cent (20%) of the dataset will be used as testing dataset. The training dataset is used to train the network intrusion detection model whiles the testing dataset is used to test the model to verify if the model can detect the network intrusions. The division of the UNSW_NB15 dataset into the respective training and testing datasets is achieved using the train_test_split() functionality from the Sci-kit learn library. The train_test_split() function splits arrays or matrices into a random train and test subsets. The train_test_split() takes arguments or parameters such as the x values, y values, train_size, test_size, and random_state amongst others. Since 80% of the data is used as training data and 20% as testing data, the parameter test_size was used to address the 20% test data.

The next step is the creation of the classifier. The first classifier used was the KNeighborsClassifier() which accepts only one argument, which is the number of neighbors. The number of neighbors chosen is generally odd. Three values for the number of neighbors were used, which were $k = 5$, 7, and 9. After, the KNeighborsClassifier() function is used to implement the fit() function. The fit() function is used to fit the model using the training data (x_train) and the target values (y_train). The class label of the model is then predicted using the predict() function. The predict() function takes the test data (x_test) as its parameter. Using the values obtained from the predict() function in addition to the test data (y_test) as the input parameters, the accuracy of the model was predicted using the accuracy_score() function. To measure the quality of the prediction from a given algorithm, the classification report is used. The classification report for the model was then obtained using the classification_report package.

The Support Vector Classification (SVC) from Support Vector Machine (svm.SVC()) is the second classifier used. The svm.SVC() accepts only one parameter which is the type of kernel to be used. The type of kernel to be used in the algorithm must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. Two types of kernels were used in the implementation of the model. They were 'poly' and 'rbf'. 'Poly' represents a polynomial kernel, and 'rbf' represents a Gaussian kernel. The fit() method of the SVC is called and used to train the algorithm with the training data (x_train, y_train) as the parameters. The predict() function or method of the SVC class is used to check for the accuracy of prediction. The classification_report package from Scikit-learn library is then used to print out the classification report of the model. The classification report is used for measuring the quality of the predictions from a given classification algorithm. The metrics of a classification report can be predicted using true positives, false positives, true negatives, and false negatives.

Evaluation Metrics

Accuracy is the ratio of all instance classified correctly (TP, TN) to all the instances (TP, TN, FP, and FN). Equation 5 represents how accuracy is calculated. Precision is the ratio of true positive (TP) to the total of true positive (TP) and false positive (FP), that is, precision is also the ability of a classifier not to label an instance positive that is negative. Precision is represented by equation 6. A classifier's ability to find all positive instances is known as recall, that is, the ratio of true positive (TP) to the sum of true positive (TP) and false negative (FN). The recall is represented with equation 7. The recall and weighted harmonic mean of precision such that the best score is 1.0, and the worst being 0.0 is called the F1 score. F1 score is represented by equation 8:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

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

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

$$F1\ Score = \frac{2 * \left(Recall * Precision\right)}{\left(Recall + Precision\right)} \tag{8}$$

ANALYSIS

The Network Intrusion Detection System Models aim to help with the detection of intrusion in a network using the UNSW_NB15 dataset as the working dataset. Five algorithms are used in the development of the models, namely, K-Nearest Neighbor Classifier (KNN), the Support Vector Machine (SVM), Voting Classifier, Random Forest and XGBoost algorithms. Three different values of the k-neighbor were used in the implementation of the network intrusion detection model using the

KNN. Also, two different kernels were used in the design of the model using the SVM algorithm. The network intrusion detection models were designed with the different. machine learning techniques to be able to compare how well each technique performed in the detection of the intrusions. The accuracy scores of the algorithms showed the model which performed better. The first step towards the modeling of the network intrusion detection system models is the selection of a suitable programming language to be used for this purpose. The network intrusion detection system models were designed using the Python Programming Language. The Python programming language is stable, flexible, and has already available tools.

KNN Model

The purpose of this section is to discuss the findings obtained using KNN algorithm. The neighbor values for the algorithm are changed various times to detect the accuracy of the model using the detection accuracy score and F1 scores of the various attack categories. The number of features used from the dataset to train and test the model was 10. The results obtained after training and testing the model with KNN and neighbor value of 5 was an accuracy value of 84.6%. Observations after training and testing the model with KNN and seven (7) as the neighbor value showed that the accuracy of detection was 84.8%. Training and testing the model on the UNSW_NB15 dataset with KNN and a neighbor value of 9 produced an accuracy of detection score of 84.9%. Table 3 shows the full results obtained from the model after training and testing.

SVM Model

This section discusses the results using SVM. Two different kernel types were used for this algorithm to obtain the detection accuracy and F1 score of the different categories of attack. Ten (10) features were used in training and testing the model. After training and testing the model with SVM algorithm with a kernel of polynomial of degree 2 on the UNSW_NB15 dataset, the model achieved a detection accuracy of 81%. A detection accuracy of 83% was achieved when the model with SVM using Gaussian kernel was trained and tested on the USNW_NB15 dataset. The results obtained results obtained are displayed in table 4.

Voting Ensemble

The next classifier used was the Voting Classifier. The Voting Classifier trains on an ensemble of diverse models and the output is predicted based on the probability of the selected class. The VotingClassifier accepts parameters such as the estimators and voting. The estimators used were KNN and SVM. The Voting Classifier accepts two types of voting which are hard and soft. The hard voting uses majority rule voting for predicted class labels whiles the prediction is based on average of probability given to that class in soft voting. The accuracy and f1 scores are predicted using the accuracy_score and f1_score respectively. The results obtained from the Voting ensemble which combines both the KNN and SVM algorithms are examined in this section. Ten (10) features out of the 44 features included in the dataset were used in training and testing the model. Training and testing of the voting ensemble model were done on the UNSW_NB15 dataset using K-Nearest Neighbor (KNN) Classifier and Support Vector Machines (SVM). The detection accuracy achieved by the model was 83.6% and the weighted F1 score of the model was 83.6%. Also, the Voting Ensemble model obtained precision and recall scores of 84.5% and 83.6% respectively.

Random Forest

This section discusses the findings obtained from the model through binary classification and multiclass classification. The accuracy of detection, precision, recall scores and the Area under the Receiver Operating Characteristics (AUROC) are the performance metrics used to assess the model:

Table 3. Performance of KNN with neighbor value of 5, 7, 9

Neighbors	Network Intrusion Category	Evaluation Metrics			
		Precision	Recall	F1 score	Accuracy
Value = 5	Analysis	0.53	0.23	0.32	0.846
	Backdoor	0.25	0.06	0.10	
	DoS	0.34	0.43	0.38	
	Exploits	0.67	0.72	0.69	
	Fuzzers	0.89	0.85	0.87	
	Generic	0.99	0.98	0.98	
	Normal	1.00	1.00	1.00	
	Reconnaissance	0.84	0.72	0.77	
	Shellcode	0.63	0.44	0.52	
	Worms	0.60	0.12	0.20	
Value = 7	Analysis	0.69	0.23	0.34	0.848
	Backdoor	0.27	0.05	0.09	
	DoS	0.33	0.35	0.34	
	Exploits	0.65	0.77	0.70	
	Fuzzers	0.90	0.85	0.87	
	Generic	1.00	0.98	0.99	
	Normal	1.00	1.00	1.00	
	Reconnaissance	0.86	0.72	0.78	
	Shellcode	0.63	0.44	0.52	
	Worms	0.33	0.08	0.13	
Value = 9	Analysis	0.50	0.26	0.34	0.849
	Backdoor	0.53	0.05	0.09	
	DoS	0.32	0.28	0.30	
	Exploits	0.64	0.80	0.71	
	Fuzzers	0.90	0.84	0.87	
	Generic	1.00	0.98	0.99	
	Normal	1.00	1.00	1.00	
	Reconnaissance	0.85	0.71	0.78	
	Shellcode	0.61	0.45	0.52	
	Worms	0.00	0.00	0.00	

- **Binary Classification:** Training and testing the Random Forest model using the UNSW_NB15 dataset with the binary classification achieved a detection accuracy of 90.15%, precision score of 91.59% and a recall score of 90.15%. The model had a high count for True Positive and True Negative values. Table 5 shows the values obtained from the model for accuracy, precision, and recall.

Table 4. Performance of SVM model with polynomial kernel of degree 2

Kernel	Network Intrusion Category	Evaluation Metrics			
		Precision	Recall	F1 score	Accuracy
Polynomial with degree of 2 kernel	Analysis	1.00	0.03	0.06	0.81
	Backdoor	0.00	0.00	0.00	
	DoS	0.09	0.00	0.00	
	Exploits	0.55	0.89	0.68	
	Fuzzers	0.83	0.61	0.70	
	Generic	0.98	0.96	0.97	
	Normal	1.00	1.00	1.00	
	Reconnaissance	0.54	0.66	0.59	
	Shellcode	0.00	0.00	0.00	
	Worms	0.00	0.00	0.00	
Gaussian kernel	Analysis	0.89	0.14	0.25	0.83
	Backdoor	0.00	0.00	0.00	
	DoS	0.55	0.01	0.01	
	Exploits	0.57	0.92	0.71	
	Fuzzers	0.81	0.72	0.76	
	Generic	1.00	0.97	0.98	
	Normal	1.00	1.00	1.00	
	Reconnaissance	0.65	0.65	0.65	
	Shellcode	0.00	0.00	0.00	
	Worms	0.00	0.00	0.00	

Table 5. Performance of Random Forest ()

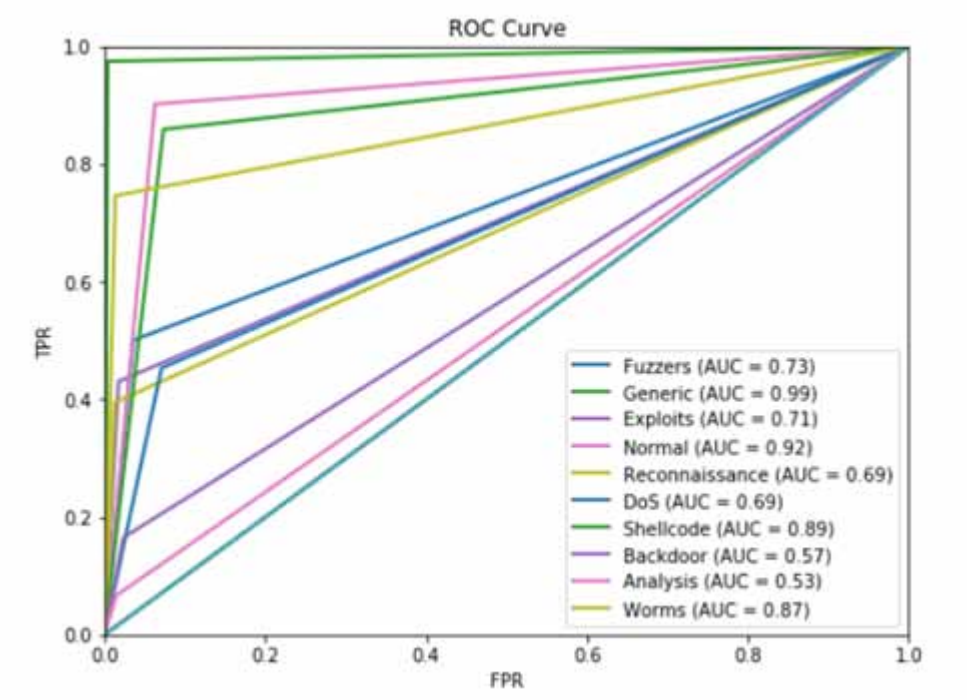
	Accuracy	Precision	Recall
Binary Classification	90.15%	91.59%	90.15%
Multiclass Classification	70.79%	79.46%	70.79%

- **Multiclass Classification:** While performing multiclass classification of the categories, the Random Forest model obtained an accuracy of detection value of 70.79%, precision score of 79.46%, and a recall score of 70.79%. The low scores of accuracy, precision, and recall of the Random Forest for multiclass classification is due to the higher class imbalance of the attacks in the dataset. Table 5 represent the scores obtained from the model. Figure 1 represents the ROC curve for the model.

XGBoost

This section discusses the results which were obtained from the model through binary classification and multiclass classification. The performance metrics used in assessing the model include detection accuracy, precision, recall scores and the Area under the Receiver Operating Characteristics (AUROC).

Figure 1. ROC Curve for Random Forest (Multiclass Classification)



- **Binary Classification:** UNSW_NB15 dataset was used in training the model and the model achieved 85% detection accuracy, 88.1% precision score and 90.14% recall score. The XGBoost model also had a high count for True Positive and True Negative values. Table 6 shows the values obtained from the model for accuracy, detection, and recall.
- **Multiclass Classification:** Multiclass classification was done by the XGBoost model on the UNSW_NB15 dataset and a detection accuracy of 51.77% was achieved. Also, precision and recall scores of 51.77% respectively were obtained by the XGBoost model. The high class imbalance in the data accounts for the low scores obtained by the model. Table 6 represents the scores obtained and Figure 2 shows the ROC curve.

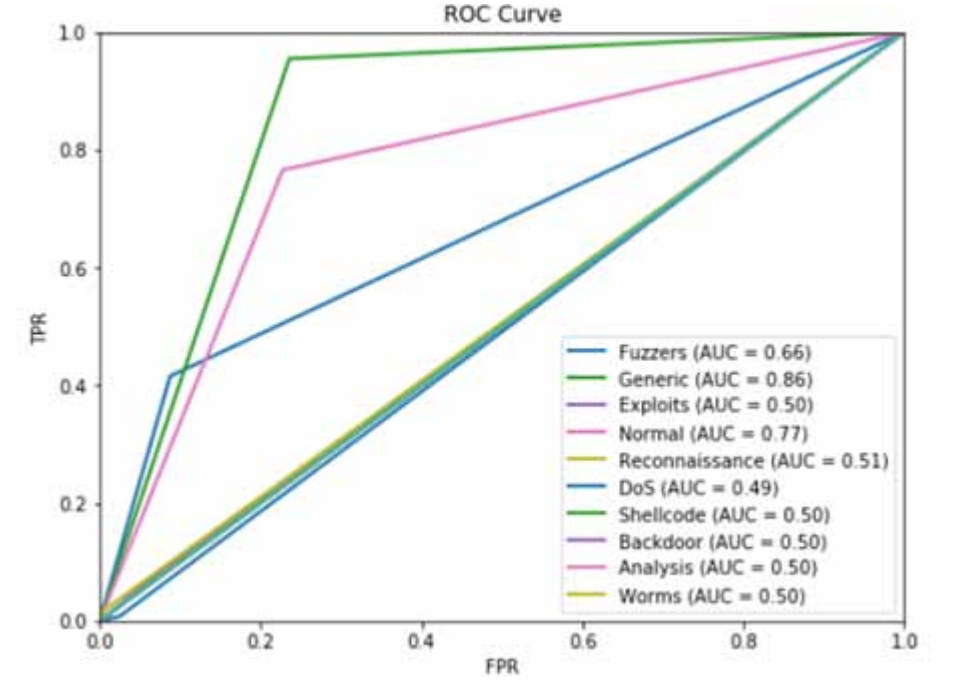
DISCUSSION

The objectives of this research were to develop machine learning models for network intrusion detection, evaluate the performance of the developed models, and compare the performance of the developed models to other network intrusion detection models. The network intrusion detection models were developed by utilizing machine learning techniques such as K-Nearest Neighbor (KNN), Support Vector Machines (SVM), Random Forest, Voting Ensemble and XGBoost Classifier.

Table 6. Performance of XGboost

	Accuracy	Precision	Recall
Binary Classification	85.0%	88.1%	90.14%
Multiclass Classification	51.77%	51.77%	51.77%

Figure 2. ROC curve for XGBoost (Multiclass Classification)



Analysis of the performance of the network intrusion detection models shows that the models implemented with the KNN algorithm obtained better accuracy than the models implemented with SVM. Three values of K, which were 5, 7, and 9, were used for the model implemented with the KNN classifier. The model with the K value of 5 (5NN) obtained the least accuracy amongst the models designed with the KNN algorithm whiles the model with K value of 9 (9NN) obtained the highest accuracy. The accuracy obtained by the 5NN model was 84.6%, the accuracy of 7NN model was 84.8%, and the 9NN model achieved an accuracy of 84.9%. The SVM models were implemented using two kernels, the polynomial kernel with a degree of two and a Gaussian kernel. The Gaussian kernel performed better than the polynomial kernel in terms of accuracy. The accuracy of the Gaussian kernel was 83%, and that of the polynomial kernel was 81%. The best performing SVM kernel (Gaussian kernel) was outperformed by the worst-performing KNN model (5NN). Random Forest model obtained an accuracy of 90.15% and 70.79% for binary and multiclass classification respectively which outperformed that of XGBoost which obtained accuracy values of 85% and 51.77% for binary and multiclass classification respectively. The SVM model using the polynomial kernel was unable to detect Backdoor and Shellcode intrusions whiles the SVM model using Gaussian kernel was unable to detect Backdoor intrusions. Also, the KNN model with K value of 9 was unable to detect worms.

The proposed models performed well compared to the existing systems. The SVM and KNN models performed better in terms of accuracy than the intrusion detection algorithm combined sampling with deep hierarchical network which obtained a detection accuracy 77.16% when tested on UNSW_NB15 dataset and network intrusion detection model based on the multivariate correlation analysis-long short-term memory network (MCA-LSTM) which obtained 77.74% accuracy when using UNSW_NB15 dataset. Also, the support Vector Machine (SVM) with a novel scaling method for binary-classification and multi-classification presented by Jing & Chen (2019) obtained accuracies of 85.99% and 75.77% for binary and multiclass classification which was outperformed by the Random Forest algorithm.

Even though this study proposes network intrusion detection models using machine learning techniques such as K-Nearest Neighbor (KNN) Classification algorithm, and Support Vector Machine (SVM) classifier; there were some limitations. The network intrusion detection model using SVM was unable to detect some intrusions in the UNSW_NB15 dataset. The SVM model using the polynomial kernel was unable to detect Backdoor and Shellcode intrusions whiles the SVM model using Gaussian kernel was unable to detect Backdoor intrusions. Also, only two kernels types were used for the SVM model. Other kernel types for the SVM model were not tested to check their ability in detecting intrusions. This made the decision to conclude that models using KNN outperformed models using SVM bias. Furthermore, although the UNSW_NB15 dataset is a good fit for training and testing intrusion detection models, it would not be able to detect newer attacks such as Distributed Denial of Service (DDoS), Man in the middle (MitM), SQL injection, and other different types of attacks. Lastly, imbalance in terms of the attack types in the dataset could affect the performance of the model while performing multiclass classification.

CONCLUSION

In this paper, network intrusion detection system models are presented and discussed. Supervised machine learning techniques are utilized in the implementation of the intrusion detection models. The supervised machine learning techniques employed in this paper are the K-Nearest Neighbor (KNN) Classification algorithm, the Support Vector Machine (SVM) classification algorithm, Voting Ensemble, Random Forest and XGBoost Classifier. Different values for K were used in the K-Nearest Neighbor model, and two different kernels were used in the Support Vector Machine model. Random Forest and XGBoost classifiers were used for binary and multiclass classifications of the attack categories. Ten (10) features of the dataset were used in the implementation of the proposed models. The network intrusions in the UNSW_NB15 dataset were grouped into nine separate categories, that is, Analysis, Backdoor, DoS, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode, and Worms. Python programming language was used in designing and developing the proposed network intrusion detection models. Python programming language was chosen because of its vast collection of libraries and frameworks, its platform independence, and its consistency. Some of the python libraries include Keras, Scikit-learn, NumPy, and Pandas.

The network intrusion detection model using SVM was unable to detect some intrusions in the UNSW_NB15 dataset. The SVM model using the polynomial kernel was unable to detect Backdoor and Shellcode intrusions whiles the SVM model using Gaussian kernel was unable to detect Backdoor intrusions. Although only KNN, SVM, Voting Ensemble, Random Forest were used as classifier algorithms, the proposed network intrusion detection system models can be extended to include other algorithms, such as Neural Networks or other ensemble techniques. This will aid in coming up with a solution to battle the class imbalance in the dataset. Therefore, in future works regarding the network intrusion detection system models, the implementation and evaluation of the alternative algorithms in the context of the proposed framework will be done. By including other algorithms, there will be an opportunity to evaluate the scalability of the proposed systems using machine learning techniques that demand more in terms of computational resources. Also, a different dataset will be used in training and testing the models to verify the ability of the models to detect intrusions.

REFERENCES

Al-hawawreh, M., Sitnikova, E., & Hartog, F. Den. (2019). *An Efficient Intrusion Detection Model for Edge System in Brownfield Industrial Internet of Things*. Academic Press.

Alsadhan, A., Hussain, A., Liatsis, P., Alani, M., Tawfik, H., Kendrick, P., & Francis, H. (2019). Locally weighted classifiers for detection of neighbor discovery protocol distributed denial-of-service and replayed attacks. *Transactions on Emerging Telecommunications Technologies*, (March), 1–15. doi:10.1002/ett.3700

Alzahrani, A. S., Ali, R., Qian, Y., & Ali, M. (2020). A novel method for feature learning and network intrusion classification. *Alexandria Engineering Journal*, *59*(3), 1159–1169. Advance online publication. doi:10.1016/j.aej.2020.01.021

Anthi, E., Williams, L., & Burnap, P. (2018). Pulse: An adaptive intrusion detection for the internet of things. *IET Conference Publications, 2018*(CP740), 1–4. doi:10.1049/cp.2018.0035

Biau, G. (2012). Analysis of a Random Forests Model. *Journal of Machine Learning Research*, *13*, 1063–1095. doi:10.5603/AIT.a2017.0074

Chawla, S., & Thamilarasu, G. (2018). Security as a service: Real-time intrusion detection in internet of things. *ACM International Conference Proceeding Series*. doi:10.1145/3212687.3212872

Choudhary, S., & Kesswani, N. (2018). Detection and Prevention of Routing Attacks in Internet of Things. *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 1537–1540. doi:10.1109/TrustCom/BigDataSE.2018.00219

Cutler, A., Cutler, D. R., & Stevens, J. R. (2012). Random Forests. *Ensemble Machine Learning, February 2014*. 10.1007/978-1-4419-9326-7

Deshmukh-Bhosale, S., & Sonavane, S. S. (2019). Design of intrusion detection system for dos attack in 6lowpan and RPL based iot network. *International Journal of Innovative Technology and Exploring Engineering*, *8*(11), 3840–3844. doi:10.35940/ijitee.K2288.0981119

Devi, S. S. (2019). *Intrusion Detection System using Datamining Based Enhanced Framework.* 10.35940/ijitee.A4472.119119

Faker, O., & Dogdu, E. (2019). Intrusion Detection Using Big Data and Deep Learning Techniques. *ACM Southeast Conference (ACMSE 2019)*, 86–93. doi:10.1145/3299815.3314439

Fu, Y., Yan, Z., Cao, J., Koné, O., & Cao, X. (2017). An Automata Based Intrusion Detection Method for Internet of Things. *Mobile Information Systems*, *6–10*, 1–13. Advance online publication. doi:10.1155/2017/1750637

Gendreau, A. A., & Moorman, M. (2016). Survey of intrusion detection systems towards an end to end secure internet of things. *Proceedings - 2016 IEEE 4th International Conference on Future Internet of Things and Cloud, FiCloud 2016*, 84–90. doi:10.1109/FiCloud.2016.20

Govindasamy, J., & Punniakodi, S. (2017). Energy efficient intrusion detection system for ZigBee based wireless sensor networks. *International Journal of Intelligent Engineering and Systems*, *10*(3), 155–165. doi:10.22266/ijies2017.0630.17

Granjal, J., Silva, J. M., & Lourenço, N. (2018). Intrusion detection and prevention in CoAP wireless sensor networks using anomaly detection. *Sensors (Switzerland)*, *18*(8), 2445. Advance online publication. doi:10.3390/s18082445 PMID:30060498

Hajisalem, V., & Babaie, S. (2018). A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection. *Computer Networks*, *136*, 37–50. doi:10.1016/j.comnet.2018.02.028

Husain, A., Salem, A., Jim, C., Dimitoglou, G., & College, H. (2019). *Development of an Efficient Network Intrusion Detection Model Using Extreme Gradient Boosting (XGBoost) on the UNSW-NB15 Dataset*. IEEE. doi:10.1109/ISSPIT47144.2019.9001867

Ioannou, C., & Vassiliou, V. (2018). *An Intrusion Detection System for Constrained WSN and IoT Nodes Based on Binary Logistic Regression*. Advance online publication. doi:10.1145/3242102.3242145

Jafier, S. H. (2018). Utilizing feature selection techniques in intrusion detection system for Internet of Things: Extended abstract. *ACM International Conference Proceeding Series*. doi:10.1145/3231053.3234323

Jahir Husain, A., & Maluk Mohamed, M. A. (2019). IMBF counteracting denial-of-sleep attacks in 6LoWPAN based internet of things. *Journal of Information Science and Engineering*, *35*(2), 361–374. doi:10.6688/JISE.201903_35(2).0007

Jiang, K., Wang, W., Wang, A., & Wu, H. (2020). Network Intrusion Detection Combined Hybrid Sampling With Deep Hierarchical Network. *IEEE Access: Practical Innovations, Open Solutions*, *8*(3), 32464–32476. doi:10.1109/ACCESS.2020.2973730

Jing, D., & Chen, H. (2019). SVM Based Network Intrusion Detection for the UNSW-NB15 Dataset. IEEE.

Kadhim, Y., & Mishra, A. (2019). Radial Basis Function (RBF) Based on Multistage Autoencoders for Intrusion Detection system (IDS). IEEE.

Kasinathan, P., Costamagna, G., Khaleel, H., Pastrone, C., & Spirito, M. A. (2013). Demo: An IDS framework for internet of things empowered by 6LoWPAN. *Proceedings of the ACM Conference on Computer and Communications Security*, 1337–1339. doi:10.1145/2508859.2512494

Kasongo, S. M., & Sun, Y. (2020). A deep learning method with wrapper based feature extraction for wireless intrusion detection system. *Computers & Security*, *92*, 101752. Advance online publication. doi:10.1016/j.cose.2020.101752

Khan, F., & Derhab, A. (2019). A Novel Two-Stage Deep Learning Model for Efficient Network Intrusion Detection. *IEEE Access: Practical Innovations, Open Solutions*, 7.

Kumar, V., Das, A. K., & Sinha, D. (2019). UIDS: A unified intrusion detection system for IoT environment. *Evolutionary Intelligence*. Advance online publication. doi:10.1007/s12065-019-00291-w

Li, W., Yi, P., Wu, Y., Pan, L., & Li, J. (2014). A new intrusion detection system based on KNN classification algorithm in wireless sensor network. *Journal of Electrical and Computer Engineering, 2014*(1). 10.1155/2014/240217

Lin, Y., Wang, J., Tu, Y., Chen, L., & Dou, Z. (2019). *Time-related Network Intrusion Detection Model : A Deep Learning Method*. IEEE.

Malik, S., Harode, R., & Kunwar, A. S. (2020). *XGBoost: A Deep Dive into Boosting (Introduction Documentation).* 10.13140/RG.2.2.15243.64803

Manso, P., Moura, J., & Serrão, C. (2019). SDN-based intrusion detection system for early detection and mitigation of DDoS attacks. *Information (Switzerland)*, *10*(3), 1–17. doi:10.3390/info10030106

Mehmood, A., Mukherjee, M., Ahmed, S. H., Song, H., & Malik, K. M. (2018). NBC-MAIDS: Naïve Bayesian classification technique in multi-agent system-enriched IDS for securing IoT against DDoS attacks. *The Journal of Supercomputing*, *74*(10), 5156–5170. doi:10.1007/s11227-018-2413-7

Moustafa, N., & Slay, J. (2015a). *The significant features of the UNSW-NB15 and the KDD99 data sets for Network Intrusion Detection Systems*. 10.1109/BADGERS.2015.14

Moustafa, N., & Slay, J. (2015b). UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). *2015 Military Communications and Information Systems Conference, MilCIS 2015 - Proceedings*. doi:10.1109/MilCIS.2015.7348942

Mulay, S. A., Devale, P. R., & Garje, G. V. (2010). Intrusion Detection System Using Support Vector Machine and Decision Tree. *International Journal of Computers and Applications*, *3*(3), 40–43. doi:10.5120/758-993

Saeed, A., Ahmadinia, A. L. I., Javed, A., & Larijani, H. (2016). *Intelligent Intrusion Detection in Low-Power IoTs r r.* Academic Press.

Sanchez-Marono, N., Alonso-Betanzos, A., & Tombilla-Sanroman, M. (2007). Filter Methods for Feature Selection – A Comparative Study. *International Conference on Intelligent Data Engineering and Automated Learning*, 178–187. doi:10.1007/978-3-540-77226-2_19

Santos, A. L., Cervantes, C. A. V., Nogueira, M., & Kantarci, B. (2019). Clustering and reliability-driven mitigation of routing attacks in massive IoT systems. *Journal of Internet Services and Applications*, *10*(1), 18. Advance online publication. doi:10.1186/s13174-019-0117-8

Sharma, J., Giri, C., Granmo, O.-C., & Goodwin, M. (2019). Multi-layer intrusion detection system with ExtraTrees feature selection, extreme learning machine ensemble, and softmax aggregation. *EURASIP Journal on Information Security, 15*.

Su, T., Sun, H., Zhu, J., Wang, S., & Li, Y. (2020). BAT. *Deep Learning Methods on Network Intrusion Detection Using NSL-KDD Dataset.*, *8*, 29575–29585.

Suykens, J., & Vandewalle, J. (1999). Least Squares Support Vector Machine Classifiers. *Neural Processing Letters*, 1–11. doi:10.1023/A

Syarif, A. R., Gata, W., Wahyudi, M., & Humaira, S. (2019). Classifier Algorithm with Attribute Selection and Optimization for Intrusion Detection System. *IOP Conf. Series: Materials Science and Engineering*. doi:10.1088/1757-899X/662/2/022066

Tama, B. A., Comuzzi, M., & Rhee, K. (2019). TSE-IDS : A Two-Stage Classifier Ensemble for Intelligent Anomaly-Based Intrusion Detection System. *IEEE Access: Practical Innovations, Open Solutions*, *7*, 94497–94507. doi:10.1109/ACCESS.2019.2928048

Venkatraman, S., & Surendiran, B. (2019). Adaptive hybrid intrusion detection system for crowd sourced multimedia internet of things systems. *Multimedia Tools and Applications*. Advance online publication. doi:10.1007/s11042-019-7495-6

Richard Nunoo Clottey received the M.Sc. degree in Computer Science from the University of Ghana, in 2020. He currently works as an Assistant Information Systems Officer at Cocoa Health and Extension Division (COCOBOD). His current research interests include the modeling and evaluation of Intrusion Detection Systems using machine learning algorithms.

Winfred Yaokumah is a researcher, cyber security expert and senior faculty at the Department of Computer Science of the University of Ghana. His work appears in several reputable journals including Information and Computer Security, International Journal of Distributed Artificial Intelligence, Journal of Information Technology Research, Information Resources Management Journal, IEEE Xplore, International Journal of e-Business Research, and International Journal of Enterprise Information Systems. He is an editor of the Modern Theories and Practices for Cyber Ethics and Security Compliance. His research interest includes Cyber Security, Machine Learning, Network Security, and Information Systems Security. He also serves on an International Review Board for the International Journal of Technology Diffusion.

Justice Kwame Appati is a lecturer in the School of Physical and Mathematical Science (SPMS) and the Department of Computer Science. He began his teaching career at Kwame Nkrumah University of Science and Technology in Kumasi as a graduate assistant and then later moved to University of Ghana in 2017 as a lecturer. Dr Appati earned a PhD, in Applied Mathematics from Kwame Nkrumah University Science and Technology in 2016. He also graduated in 2010 and 2013 with a BSc. Mathematics and MPhil Applied Mathematics from the same institution. His current research includes the automatic detection and classification of human intestinal worm and his most recent publication is "Multi-criteria ranking of voice transmission carriers of a telecommunication company using PROMETHEE, 2018". Academically, Dr. Appati is a self-disciplined and motivated person who is passionate with his job and continually challenges his students and set high expectation for them. He has also singly and jointly supervised undergraduate and postgraduate students from Kwame Nkrumah University of Science and Technology (KNUST), National Institute of Mathematical Sciences (NIMS), African Institute of Mathematical Sciences (AIMS) and University of Ghana Currently, Dr. Appati handle course like Design and Analysis of Algorithm, Artificial Intelligence, Formal Methods and Computer Vision. He looks forward to working with everyone interested in his field of study more especially, Intelligence and Data Science.