Anchor University Journal of Science and Technology (AUJST)
A publication of the Faculty of Natural, Applied and Health Science, Anchor University Lagos

URL: fnas.aul.edu.ng

In AJOL: https://www.ajol.info/index.php/aujst

Vol. 4 No 1, September 2023, Pp. 71 - 80

ISSN: 2736-0059 (Print); 2736-0067 (Online)

SOFTWARE DEFINED-NETWORK INTRUSION DETECTION MODEL USING STACKED ENSEMBLE TECHNIQUES OF MACHINE LEARNING

Yahaya Abbas Yakubu[1], Kabir Ibrahim Musa[2], Umar Muazu[2]

[1]Student, Department of Management and Information Technology, ATBU Bauchi
[2]Lecturer, Department of Management and Information Technology, ATBU Bauchi

***Corresponding author:**
yahayaabbas216@gmail.com

Submitted 29 March, 2023
Accepted 4 July, 2023

Competing Interests:
The authors declare no competing interests.

ABSTRACT

Software defined-network (SDN) brought in so much of flexibility in network management and administrations through its programmability and centralized nature. However, this programmability, exposes SDN to constant evolving network attacks. To addressed this challenge, previous studies have shown that intrusion detection system (IDS) is very effective. So many approaches were adopted to develop IDSs especially machine learning because of its strength in detecting trends in a given data. Unfortunately, this strength depends greatly on the quality of the training dataset which is subject depreciation over time. Couples with the constant evolutions of network attack, the depreciations in quality of IDS training datasets have made it very difficult for machine learning IDSs to detect attacks accurately. In order to address this challenge, this study proposes a software defined-network-based intrusion detection model using stacked ensemble technique of machine learning. The study adopts inSDN dataset as the training dataset because of its of quality in SDN features. From the experimented result, the model performed very well by recording 99.3% of accuracy. Despite the performance of this model, the model has never been evaluated in a real SDN environment.

Keywords: IDS, SDN, ensemble learning, accuracy, confusion matrix, inSDN, Network attacks

1. INTRODUCTION

The easiness of network management and administrations offered by software defined-network (SDN) stands it out among networking frameworks. Leading SDN to the submit of the list of network frameworks with the highest adaption rate in high-tech industries (Jin, *et al.,* 2020). Advantages of SDN over other framework of networks, came as the result of its programable and centralized nature. However, the programmability exposes SDN to security threats which are far more severe compared to that of the conventional network (Fahad, *et al.,* 2019). Intrusion detection system (IDS) is very effective in mitigating this security vulnerability in SDN. It monitors system usage and network traffic in order to detect threats. So many different methods were adopted in developing IDS. Out of these methods, machine learning is very effective because of its skills in finding trends when other approaches failed. Though, machine learning strengths in detecting patterns, always rest on the quality of the training dataset which depreciate over time (Elsayed *et al.*, 2020). Couples with depreciation of IDS dataset quality overtime,

https://dx.doi.org/10.4314/aujst.v4i1.8

the constant evolutions of network attacks, has weaken the predictive power of IDS. To improve the predictive power of machine learning based IDS, for a software defined-network, this study adopts stacked ensemble technique of machine learning to propose intrusion detection model.

The proposed model combines all predictions from the level 0 (base models) to train the level 1 (meta-model) which make the final prediction. The base models comprise of K-nearest neighbor (KNN), Classification and regression tree (CART), Support vector machine (SVM) and Gaussian bayes (BAYES) while, the metal model uses Logistic regression (LGR).

1.1. Software defined-network (SDN)

The major idea of software defined network, is the splitting of the control plane and the data plane. The control plane housed the SDN controller which can be programmed externally. This allow addition of new network services as an application without any change to the hardware or the topology of the network (Hoang, 2015).

The SDN controller generate and send flow tables to switch which handle packets forwarding. Residing between applications and network infrastructures, SDN controller depends on application programmable interfaces (APIs) such as northbound and southbound interfaces to interact with the applications and the infrastructures (Hande & Muddana, 2020).

Northbound Interface (NBI) is utilized for communication between the application and the SDN controller while the southbound interface is used for communication between the controller and underlying network infrastructure.

1.2 SDN security challenges

The first major SDN security issue comes from its programmability that allow installations of network services as an application runs in the SDN controller (Haas J., et al., 2021). Any application running on SDN controller, have the ability to access network status and also injecting new data forwarding rules to the entire network. This have created a big security challenge to the SDN because once a malicious application gets into the SDN controller, it can access network status and even determine flows of network packets.

Unlike in traditional network, where attacks are only regulated to the portion of the network with same vendors, in SDN a compromised switches or end-users can disrupt the SDN controller, resulting into impairment of the entire network (Abbas *et al*., 2020).

The most perpetrated attack against SDN include; Denial of Service (DoS), Distributed Denial of Service (DDoS), Brutal Force Attack (BFA) etc.

1.3. Intrusion detection system

A network's intrusion detection system (IDS) is used to spot threats by keeping watch on the packets that move across the network or the use of computer resources. IDS can identify suspicious and malicious activity coming from both insiders and outsiders. When it runs on a network host, it is called a host-based IDS and when it monitors a network, it is referred to as network based (Alhadad, et al., 2019).

Based on the techniques used by IDS in detecting threats, IDS can be classified as signature-based detection or anomaly-based

https://dx.doi.org/10.4314/aujst.v4i1.8

IDS is to detect unknown malicious activities (Randy & Wang, 2017).

In developing IDSs, machine learning models have proven to be very effective in detecting network attacks. However, due to the sophistication of networks attacks these days, network attacks are becoming too difficult to be effectively detected by a single machine learning model. Therefore, the ensemble technique of machine learner is now taking over from single model in machine learning based IDS development.

1.4. Ensemble learning

Ensemble learning is a general meta-approach used in machine learning to improve the accuracy of models by combining multiple algorithm as sub models instead of using just one model. Ensemble learning can be classified as committee-based learning or multiple classifier system that combine strength of multiple classifiers to solve a learning problem (Zhou, 2021).

There are almost unlimited numbers of ensembles learning techniques for predictive modeling problems, but three of these techniques dominate the field of ensemble learning. These are bagging, boosting and stacking.

Bagging is a meta-algorithms approach purposefully created for decreasing variance of predictions by creating additional training set (bag) from the original dataset using combinations and repetitions. It fit multiple decision trees on more than one bag and compute average of the predictions from these decision trees to arrive at the final prediction (Brownlee, 2021a). These bags, helps the bagging techniques to archive un-bias sharing of the main dataset.

Boosting techniques uses a sequential learning technique in training models. It trains a model using the entire training set, then subsequent models are built by paying more attention to those observations that were poorly estimated by previous model. It is a chronological process in which successive model rely on the predecessor in order to reduce model's bias (Zhou, 2021). Examples of boosting are the extreme gradient boosting (XGBoosting), gradient boosting machine (GBM) and adaptive boosting (ADABoost) (Paul, 2018).

Stacking combines weaker model in making predictions but unlike bagging and boosting, it employs a different model (level-1) to combine the predictions of the weaker model (the base models). It is a procedure whereby a learner (meta-learner) is trained to make a better prediction by combining individual model (level-0 models) predictions (Brownlee, 2021b). According to Odegua (2019), on an average, boosting will do better than both a single classifier and bagging techniques, but cannot be a match for the stacking techniques because boosting is liable to overfitting most especially in a dataset with lot of noise.

1.5 RELATED WORK.

So many frameworks have been proposed by researchers for developing intrusion detection system for a software defined-networks by combining machine learning algorithms in so many different ways. To address SDN security flaws, Abbas *et al.,* (2020), adopted voting ensemble techniques of machine to proposed intrusion detection for SDN. The model was pre-trained using NSL-KDD. Five different machine learning algorithms were combined to

https://dx.doi.org/10.4314/aujst.v4i1.8

developed this model. They are: Decision Tree (DT), Random Forest (RF), XGBoost (XGB), Support Vector Machine (SVM), and Deep Neural Network (DNN). This framework recorded a final accuracy of 79.6%

Hareesha, *et al.,* (2020), also proposed intrusion detection system for SDN by using stacked ensemble techniques of machine learning. The proposed model was pre-trained using UNSW NB-15 dataset. The model combines random forest (RF), logistic regression (LR) and K-neighbor nearest (KNN) as base models, and support vector machine (SVM) as the meta-model. Albahar, et al., (2021), combines convolutional neural network (CNN) and ML algorithms such as Support vector machine (SVM), K-nearest neighbor (KNN) and Random forest (RF) to proposed hybrid deep learning-based architectures for attack classification and anomaly detection in SDN.

2. **PROPOSED METHODOLOGY**

This study experimented the proposed model on window 10 pro operating system. The OS runs on a personal computer (PC) of Intel corei3 processor, with a speed of 2.30GHz. The proposed model was built from python Scikit-learn machine learning libraries using python 3. Jupyter notebook 3 was used as the computational environment for this study.

2.1.Experiment Setup

As a two layered stacked ensemble model, the proposed model is made up of two levels of learners; the level 0 models and the level 1 model. The level 0 models comprise of Support Vector Machine (SVR), Decision Tree Regressor (CART), K-nearest Neighbors (KNN) and Gaussian Bayes (GNB) while, the level 1 model, uses Logistic Regression classifier (LR). The level 0 models are the base models while, the level 1 model is the meta-model. The meta model takes the predictions of the base models as input when making the final prediction. This give the meta model the ability to make better predictions.

2.2. Dataset definition and pre-processing

This study adapts the inSDN dataset published by Elsayed et al., (2020). The inSDN dataset contains most recent and dangerous network threats such as denial of service (DoS), Botnet, Distributed denial of service (DDoS), Brutal force attack (BFA), Web attacks, Probe and lot more. The inSDN dataset also contains normal SDN service features such as; FTP, SSH, Email, HTTPS, HTTP DNS etc.

According to Wang et al., (2020), excessive large dataset feature can lead to high computational cost in machine learning. This can result in making machine learning-based IDS not to be sufficient across different applications. Therefore, using python random sample techniques which allow random selections of features without repetition, 3400 features where drawn from the inSDN dataset to formed the dataset used in this study. They are no missing features in the inSDN dataset samples drawn but it contains categorical data which machine learning algorithms can not directly work with. Therefore, label encoder is used to transform all the categorical data to numerical data.

2.3.Algorithm of the proposed stacked model

https://dx.doi.org/10.4314/aujst.v4i1.8

2.3 Algorithm of the proposed stacked model

Input: *training set data* $D = \{x_i, y_i\}^m_{t=1}$ $(x_i \in R^n, y_i \in Y)$
Output: *ensemble model H*
Step 1: adopt python cross_val_score method for training set preparation
n_split D into K of same sizes. where $D = \{D_1, D_2, D_3....D_k\}$

for *k* $\leftarrow$ *1 to K* **do**
Step 2: learn level0 model

for *t* $\leftarrow$ *1 to T* ***do***
Learn h_{kt} *on* $D \backslash D_k$
end for
Step 3: create new data from level0 predictions

for $x_i \in D_k$ ***do***
fetch data $\{X^T_i, y_i,\}$ where $X^T_i = \{h_{ka}(X_i), h_{kb}(X_i), \ldots, h_{kT}(x_i)\}$
end for
end for
Step 4: learned meta-model
Return H
$Dh = \{X^T_i, y_i,\}$, *where* $X^T_i = \{h_1(X_i), h_2(X_3), \ldots, h_T(x_i)\}$
end for
step 4: learn level1 (meta-model)
learn H on $\{X^T_i, y_i,\}$
return H

2.5 Samples of the model implementation in python

```
#create a list to hold all the base models
baseModels = list()
   baseModels.append((‘CART’, DecisionTreeClassifier()))
   baseModels.append((‘KNN’, KNeighborsClassifier ()))
   baseModels.append((‘SVM’, SVC()))
   baseModels.append((‘BAYES’, GaussianNB()))
   #create the meta model
   metaModel = LogisticRegression()
   #establish the stacked ensemble
   myStack = StackingClassifier(estimators = baseModels, final_estimator = metaModel, cv = )
   return myStack
```

```
# RepeatedStratifiedKFold cross validation
 def myCrossVal
     cv = RepeatedStratifiedKFold(n_repeats= 2, n_split = 10 random_state = 3)
     Valuator = cross_val_score(MyStack, X, y, scoring = ‘accuracy’ , n_jobs=-1, cv=cv)
           Return Valuator
```


https://dx.doi.org/10.4314/aujst.v4i1.8

2.6 Framework of the proposed model

From figure 1, framework flows from left to right. Starting from the dataset to the final prediction. Using repeated stratification, the dataset is subdivided into strata using the n-fold parameter. The base models are fitted on all the strata one after the other, and repeatedly base on the value of n-repeats. The predictions generated from the base models is used for training the meta-model. The meta-model is evaluated in order to generate the final prediction. Parameters that internal are internal to all the classification algorithm used in the model framework, are left at the default value. Only the hyperparameters are tuned to the values shown in the table 1.

2.7. Performance evaluation metric

To quantify the performance of the proposed model, this study adopts accuracy to ascertain the value of correct predictions made by the model in relation to the total numbers of the model's predictions. Normally, accuracy of a machine learning model is generated at machine level. In order to have a clearer view of the model's performance, confusion matrix can used to depicts the model's performances. From the confusion matrix, we can see the number of rightly and wrongly predicted features in a form of True positive (TP), False positive (FP), True negative (TN) and False negative (FN). Generally, accuracy can be described as follow:

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

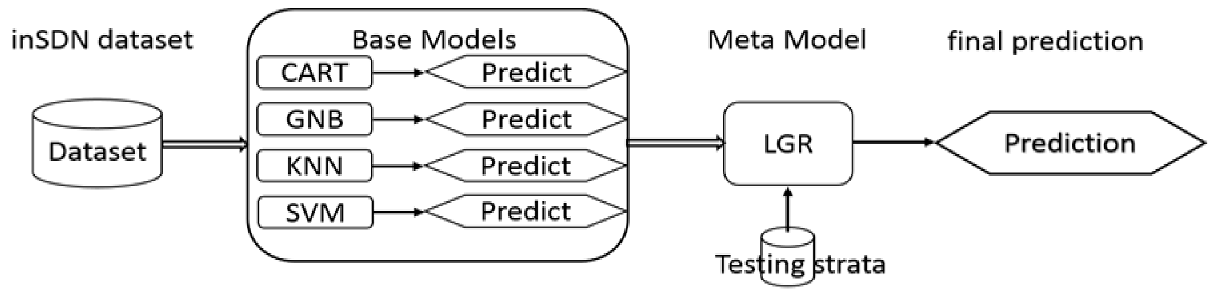

Figure 1: Framework of the proposed model

Table 1: Hyperparameters of the model and their values

Parameter	Value
CV	RepeatedStratifiedKFold
n-jobs	-1
n-splits	10
n-repeat	2
Solver	Newton-cg
Scoring	Accuracy

https://dx.doi.org/10.4314/aujst.v4i1.8

However, as a multi class model, the proposed model's accuracy can be described as follow:

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

Where: $\sum TP$ = summation of TP across all classes, $\sum TN$ = summation of TN across all classes,

$\sum FP$ = summation of FP across all classes, $\sum FN$ = summation of FN across all classes.

True positive refers to total numbers of features that are positive and were rightly classified positive by the model. True negative refers to numbers of negative features that were rightly classified by the model. False positive rate refers to numbers of negative features that were wrongly classified as positive. False negative refers numbers of positive features that were wrongly classified as negative. Therefore, accuracy of the proposed model can be recomputed as follow:

$$\text{Accuracy} = \frac{Total\ numbers\ of\ correct\ prediction}{Total\ numbers\ of\ predictions\ made}$$

Experimental Results

This study proposed and experimented a software defined-network based intrusion detection model. With the aim of improving the predictive power machine learning-based IDS, the model was built using stack ensemble techniques and evaluated with inSDN dataset. As a stacked ensemble model, the proposed model comprises of two levels of learners; the base models and the meta-model. Predictions from the base models, serve as the training inputs for the meta-model. The meta-model is responsible for making the final prediction of the proposed model. Table 2 presents the predictions of the base models.

Table 2, indicated that KNN and CART are very effective in handling multiclass problems while, SVM and BAYES are not very effective in dataset that are not geometrically separable. These base models' outputs now become the training inputs for the level 1 model (meta-model) whose prediction, is used as the final prediction of the proposed model.

Table 3 shows the performance accuracy of the meta-model which recorded 99.3% generated at the machine level. Even though this is a very high accuracy, sometimes accuracy of a multiclass model can be deceptive. Therefore, the model accuracy is revalidated via confusion matrix as shown in Table 4. This will give a true picture of where the model got it right or wrong.

From Table 4, it can be seen that, the model classified a total of 3400 features. 3376 features were successfully classified to their rightful classes while, 24 features were wrongly classified. therefore, dividing the total number of the rightly classified features by the total number of classified features will verify the accuracy of the model.

$$Accuracy = \frac{Numbers\ of\ rghtly\ classified\ features}{Total\ numbers\ of\ features}$$

$$= \frac{3376}{3400} = 0.9929 \approx 0.993$$

Table 5 compares the proposed methodology to an existing one. From the table, an existing approach adopted voting ensemble technique

https://dx.doi.org/10.4314/aujst.v4i1.8

Table 2: Predictions of the base models

Proposed Model's Base Classifiers	Accuracy	Percentage%
K-nearest Neighbors (KNN)	0.945	94.5
Classification and Regres-	0.951	95.1
Support Vector Machine	0.440	44.0
Gaussian Bayes (BAYES)	0.382	38.2

Table 3: Performance of meta-model

Evaluation metrics	Performance
Accuracy	99.3%

Table 4: The proposed model's confusion matrix

LABELS	Normal	BFA	Botnet	DDoS	DoS	Probe	Web-attack
Normal	168	0	0	0	0	0	0
BFA	0	150	0	0	2	0	0
Botnet	0	1	849	0	0	0	4
DDoS	0	0	0	920	0	5	0
DoS	10	0	0	0	490	0	0
Probe	0	0	0	0	0	648	0
Web-attack	0	0	0	2	0	0	151

Table 5: Comparison of the proposed methodology to existing methodology

Previous	methodology	Dataset	Metric	Result
Abbas, et al., (2020)	Voting ensemble technique of machine learning	NSL-KDD	Accuracy	79.6%
Proposed model	Stacked ensemble technique of ma-	inSDN	Accuracy	99.3%

https://dx.doi.org/10.4314/aujst.v4i1.8

5. Conclusion

This study improves the predictive power of machine learning-based IDS by proposing a software defined-network intrusion detection model using two layered stacked ensemble technique. The proposed model recorded a very good prediction accuracy of 99.3%. This performance can be attributed two major factors. Firstly, the choice of the training dataset and secondly the ensemble techniques used. Unlike Abbas *et al.,* (2020) that adopted NSL-KDD as training dataset, the proposed model uses inSDN dataset because quality of training dataset in machine learning-based IDS depreciate overtime as a result of evolution of newer attacks features that were not captured previously. NSL-KDD is over 15years older than inSDN which was generated in 2020. Another quality of inSDN dataset is, it was generated specifically from SDN platform. The second factor that led to the high accuracy of the proposed model is, unlike the voting ensemble technique adopted by Abbas *et al.,* (2020), that lacked which of the base models to trust while making the final prediction, the proposed model uses the stacked ensemble techniques to combine all the base models predictions to train the meta-model which is responsible for making the final prediction of the proposed model. This gives the meta-model all the individual strength of the base models in order to boost its own predictive power. This model performance was judged base on the experimental result. Therefore, future studies should investigate the model performance in a real SDN environment.

REFERENCES

Abbas, O., Assora, M., & Khorzom, K. (2020). Machine learning based intrusion detection system for software defined networks. *International Journal of Engineering Research and Technology*, *9* (09), 0181-0185.

Albahar, J., Elsayed, M., & KhacMarwan, A. (2021). A novel hybrid model for intrusion detection systems in SDNs based on CNN and a new regularization technique. *Journal of network and computer applications, 191.* https://doi.org/10.1016/j.jnca.2021.103160

Alhadad, R., Chilamkurti, N., Sultana, N., & Peng, W. (2019). Survey on SDN based network intrusion detection system using machine learning approaches. *Peer-to-Peer Networking and application, 12* (2), 493-501.https://doi.org/10.1007/s12083-017-0630-0

Brownlee, J. (2021a). *Make better predictions with bagging, boosting and stacking.* https://machinelearningmastery/ ensemble-learning-algorithms-with-python/

Brownlee, J. (2021b). *A gentle introduction to ensemble learning algorithm.* Machinelearningmastery.com/tour-of-ensemble-learning-algorithm/

Elsayed, S., Le-Khac, N., & Jurcut, D. (2020). InSDN: A Novel SDN Intrusion Dataset. *IEEEA ccess, 8,* 165263-165284. https://doi- 0.1109/ ACCESS.2020.3022633

Fahad, A., Fatima, M., Farwa, I., Maham, I., & Muhammad, R. (2019). Security issues in software defined networking(SDN): risk, challenges and potential solutions. *International Journal of Advanced Computer Science and Application.*

Haas, J. Z., Culver, L. T., & Sarac, K. (2021). Vulnerability challenges of software defined networking. *IEEE Communications Magazine*

Hande, Y., & Muddana, A. (2020). A survey on intrusion detection system for software defined networks (SDN). *International Journal of Business Data Communications and Networking*, *16* (1) 1-17. https://doi:10.4018/978-1-7998-7705-9.ch03

Hareesha, K., Kundapur, P., & Rajagopal, S. (2020). A stacking ensemble for network intrusion detection using a heterogenous datasets. *Hindawi security and communication, (2020).* https://doi.org/10.1155/2020/4586875

Hoang, D. (2015). Software defined networking shaping up for the next disruptive step. *Australian journal of telecommunication and digital economy, 3*. https://telsoc.org/ajde-v3-n4/a28

Jin, L., Keeping, Y., Yang, X., & Wenduam, L. (2020). Challenge-based Collaborative Intrusion Detection in Software Defined Networking. *Digital Communications and Networks*, *7*(2), 257-263. https://doi.org/10.1016/j.dcan.2020.09.003

Odegua, R. (2019). An empirical study of ensemble techniques (bagging, boosting and stacking). *Deep learning indabax Nigeria*. https://doi.org/10.13140/RG.2.2.35180.10882

Paul, S. (2018, September 6). *Ensemble learning in python.* www.datacamp.com/tutorials/ensemble-learning-python

Randy, J. & Wang, L. (2017). Big data analytics for network intrusion detection: A Survey. *International Journal of Networks and communications, 7*(1), 24-31. https://doi.org/10.5923/j.ijnc.20170701.03

Wang, M., Fu, W., He, X., Hao, S., & Wu, X. (2020) A survey on large-scale machine learning. *IEEE Transactions on Knowledge and Data Engineering*

Zhou, Z. (2021). Ensemble learning in machine learning. *Springer Singapore*. https://doi.org/10.1007/978-981-15-1967-3_8