International Journal of Engineering Trends and Technology
ISSN: 2231 – 5381 / https://doi.org/10.14445/22315381/IJETT-V71I3P210

Volume 71 Issue 3, 81-90, March 2023
© 2023 Seventh Sense Research Group®

Original Article

A Distributed Attack Detection System for SDN Using Stack of Classifiers

Ravindra Kumar Chouhan[1], Mithilesh Atulkar[2], Naresh Kumar Nagwani[3]

[1]Central Computer Center, NIT Raipur, GE Road, Raipur, CG, India
[2]Department of Computer Applications, NIT Raipur, GE Road, Raipur, CG, India
[3]Department of Computer Science & Engineering, NIT Raipur, GE Road, Raipur, CG, India

[1]Corresponding Author: rchauhan@nitrr.ac.in

Received: 08 August 2022 Revised: 08 January 2023 Accepted: 06 February 2023 Published: 25 March 2023

Abstract *- For the last few years, the Software Defined Network (SDN) architecture has grown in popularity in industries and academia due to its advantages over traditional networks. Because of its emergence, it has attracted many attackers who interfere with the network's normal operation. To defend against such attacks, the SDN controller centrally monitors all network activities and then takes appropriate action. This task consumes the majority of the controller's resources, resulting in controller performance degradation. To address this problem, this paper proposes an architecture in which data plane resources are used for intrusion detection, freeing up the controller for other network-related tasks. In the switch of the data plane, a stack of classifiers composed of Random Forest (RF) and K-Nearest Neighbour (KNN) at level 0 and Logistic Regression (LR) at level 1 is used. Also, to fasten the attack detection process, the appropriate features have been selected using Pearson's Correlation Coefficient and mutual information of the features. The UNSW-NB15 dataset has been used to demonstrate this architecture's performance. The performance has been measured under the metrics Precision, Accuracy, F1 value, Recall, Prediction Time, and Cohen's Kappa Coefficient. In terms of recall, accuracy, CKC, and feature count, the classifier stack surpasses the individual classifiers. Its performance is slightly inferior to that of other classifiers under precision, F1, and prediction time, but the difference is manageable when other parameters are considered. Hence, the stack of the classifier is selected for deployment in the data plane devices.*

Keywords *- Intrusion Detection System (IDS), KNN, Machine Learning, OpenFlow, Random Forest, Software Defined Network (SDN), Stack of Classifiers.*

1. Introduction

SDN is an innovative approach to the design of computer networks that offers advantages over traditional network architectures, such as programmability, centralized control, vendor neutrality, expandability, quickness, and lower capital and operational expenditure [1], [2]. The SDN diagram is depicted in Figure 1. The infrastructure layer comprises all network devices, both physical and virtual, concerned with forwarding data. Examples of such devices include switches and routers. The control layer is the layer that contains the controller, which is the brain of the SDN [3] and thus controls all network-related activities that occur in the network. This functionality gives the controller a comprehensive overview of the network[4]. The SDN controller is the one location for all logic involved in decision-making; thus, all decisions are made by the controller. The third layer, the application layer, sits on top of the SDN architecture and serves as the user interface. Users can interact with the control plane via applications developed by them to provide instructions to the controller, which can then instruct the data plane to carry out those instructions.

Instructions for transferring information between the control and data planes are known as southbound rules. As a southbound rule, the OpenFlow protocol is used. The fields of OpenFlow protocol are used to send and receive various information between the controller and data plane devices, allowing necessary decisions for data coming into or leaving the network. In addition, some APIs known as northbound rules are used to provide communication between the controller and the application layer.

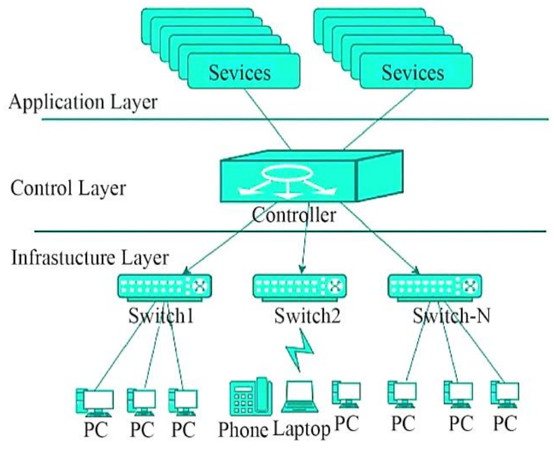

Fig. 1 Diagram of SDN

Because of the popularity of SDN architecture in new environments such as industry networks, data centres, and clouds has attracted many attackers to steal user-related data from the network or clog the network in such a way that it cannot fulfil genuine requests of the users. To choke the network, attackers flood the network with different types of traffic, such as TCP, UDP, or ICMP traffic, causing the network's critical resources to become overburdened in responding to these requests (DDoS attack). To counter such attacks, the controller has an integrated intrusion detection system, which analyses the behaviour of traffic between the controller and data plane devices and prevents attacks by enforcing the rules at the data plane level.

In the preceding scenario, the controller makes all decisions related to intrusion detection, so it consumes a large portion of the controller's resources, causing the controller's performance to deteriorate and, as a result, the network's responsiveness to suffering. At the same time, the resources available in data plane devices remain idle, and while they are doing their work, a large portion of their resources remain idle. If the resources available with these devices could be used for intrusion detection, some of the burdens from the controller would be lifted, allowing the controller to perform its other more important tasks more efficiently. The attacks that were entering the network through data plane devices can only be stopped there, and the spread of these attacks in the network can be stopped.

Apart from the above point, to make the prediction faster, a smaller number of features should be used in the classifier for making a prediction. For this purpose, in this work, feature selection has been performed. Pearson's Correlation Coefficient and Mutual Information of the features have been considered. A smaller number of selected features provide faster prediction in data plane devices of SDN.

There are two types of attack detection techniques used for attack detection: Anomaly-based IDS (AIDS) and Signature-based IDS (SIDS) [5]. SIDS are systems that store the pattern or signature of attacks in a database. An attack occurs when a new request arrives with a signature that matches one previously stored pattern. The advantage of SIDS is that it works more efficiently and quickly but fails when subjected to unforeseen attacks.

AIDS is being used to combat this problem. In the case of AIDS, a model is learned by observing normal traffic behaviour. Any traffic that deviates from normal behaviour is considered an attack. In the proposed work, AIDS is used to detect attacks in the SDN data plane using a stack of classifiers.

It is found in this work that using the classifier stack results in better performance. [6] has also demonstrated stack performance in their work. A classifier stack is one in which more than one classifier is used to predict an attack. Fig. 2 depicts the classifier stack diagram. The level 0 classifiers are given the training data, and the predictions made by the level 0 classifiers are utilized as input into the level 1 classifier, also known as the final estimator. The prediction provided by the final estimator is used as the trained model's final prediction.

The paper is divided into seven sections: Section 2 presents Related Work, Section 3 presents Proposed Work, Section 4 presents Experiments and Result Analysis, Section 5 presents Deployment of the Trained Model, Section 6 presents Contribution, and finally, Section 7 presents the Conclusion and Future Work.

2. Related Work

For attack detection and mitigation, much work has been done. Most works are implemented in the control plane of the SDN [7,23]. The paper [8] demonstrates investigating a DDoS attack on a primary server. The attack is probed by flooding the switch that connects to the main server with a huge quantity of packets containing destination IP addresses. The (POX) controller repeatedly installs flow table rules into this switch, causing its flow table space to be exhausted. The article discusses a scenario of a weak attack in which flow entries that the controller directly installs are considered to attack traffic. However, flow entries that are requested by the switch are considered to be legal traffic. The scenario is described in the paper. Mitigation is implemented by manually manipulating the "TIMEOUT" parameter value without an SDN application. [9] proposes a method for detecting DDoS attacks using SVM algorithms. It uses flow data available to controllers to form a 6-tuple, which is then used to learn and classify traffic into normal and spurious traffic on the control plane.

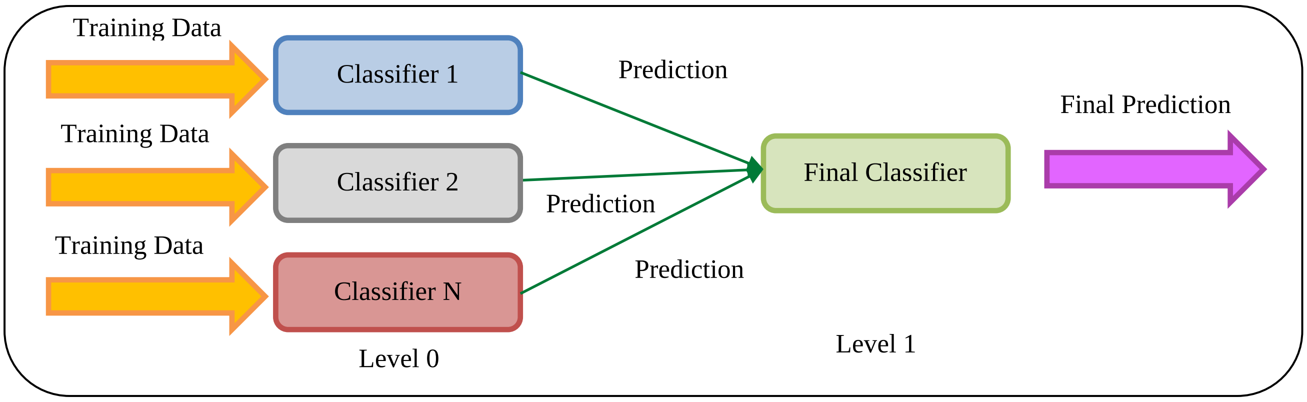

Fig. 2 Stack of the classifiers

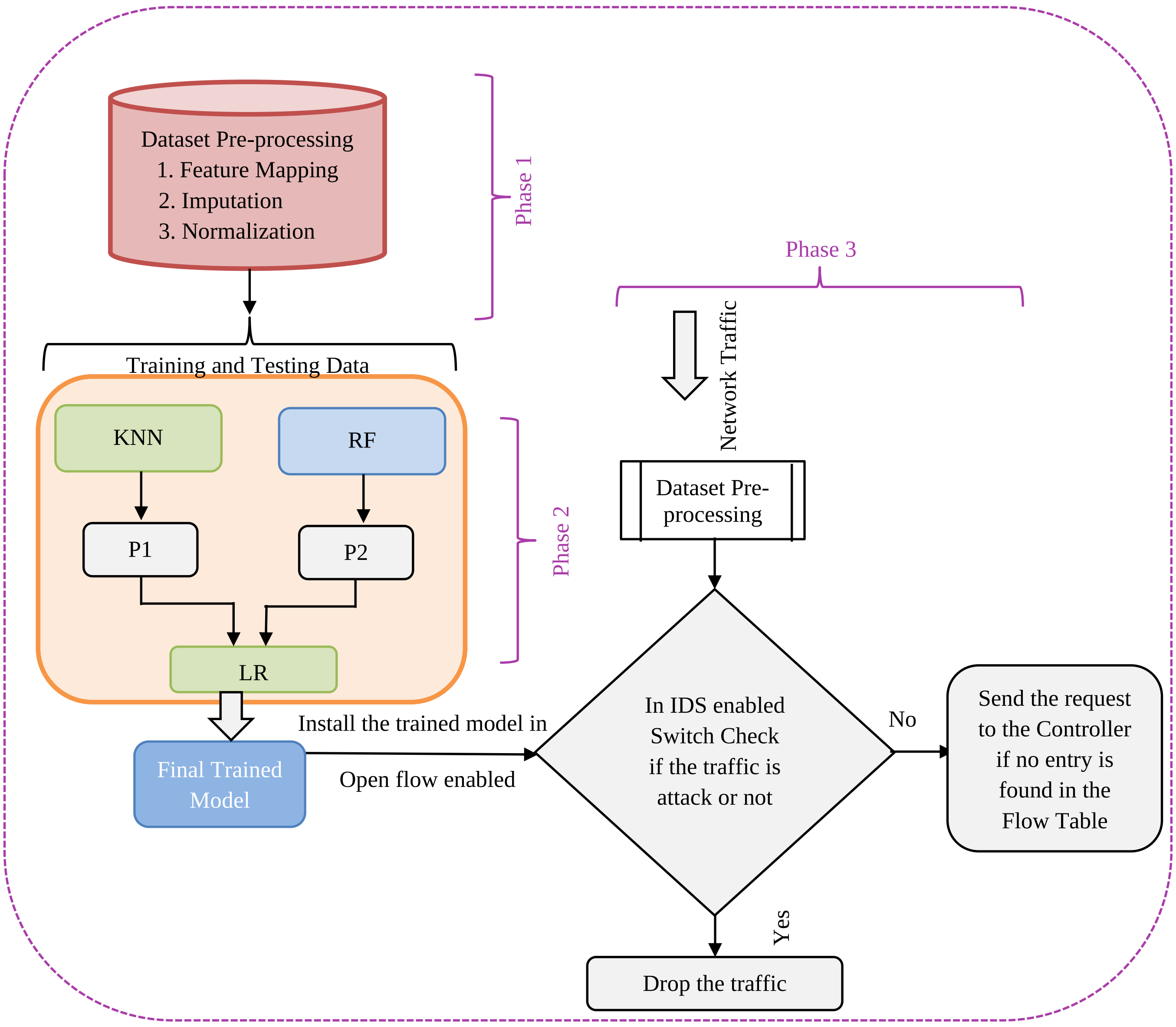

Fig. 3 Flow Diagram of Proposed Method

To prevent attacks, the authors of [10] proposed a distributed design of stateful switches as an alternative to traditional SDN centralized solutions, which may be more susceptible to computational resource limitations. Because these methods require less time to evaluate a significant volume of traffic, entropy-based methods are widely used to identify attacks in SDN networks [11]. [12] has presented a control plane DDoS attack detection solution based on entropy. The Ryu controller and Mininet were used to simulate several attacks for simulation purposes. Using entropy to determine the unpredictability of the flow data [13] has offered a strategy for early detection and mitigation of DDoS attacks. They have used Floodlight and Mininet to develop a variety of attacks and scenarios for their work. [14,15] have employed SVM augmented by kernel principal component analysis (KPCA), utilizing KPCA to reduce the dimension of feature vectors and a genetic approach to optimize SVM parameters. [30] have suggested a technique for attack detection that employs both SDN layers. One controller functions as the master controller, while the other functions as the slave controller. For the demonstration, the Mininet emulator and Ryu Controller were utilized. In the data plane, [14,15] uses attack detection systems based on triggers. In the control plane, KNN and K-means were utilized to determine whether or not the warning provided by the data plane was accurate. In the data plane, the Packet-In message is observed; if its rate is high, the switch notifies the controller by sending an alarm. The controller then retrieves five vector tuples from the respective switch's traffic and utilizes them for prediction. The defence plan is activated if the controller identifies it as an attack. ONOS is employed as the SDN controller to demonstrate their work, and Mininet is used to establish the SDN topology.

3. Proposed Work

In this section, the architecture of the proposed work has been discussed. The work has been completed in 3 phases, as shown in Fig.3. In phase 1; data preprocessing is done.

In phase 2, the stack of the classifiers, an ensemble of classifiers [16], [17], which at level 0 includes KNN and RF and at level 1 includes Logistic regression, was trained. To check the model's performance, the testing data is given to the trained model and the Precision, Accuracy, F1 value, Recall, Prediction Time, and Cohen's Kappa Coefficient are calculated. In Phase 3, the trained and tested model is installed in the data plane, i.e., in the switch of the SDN. Now whenever any new request comes to the switch, the required features are extracted from the live traffic, and then preprocessing is performed. The model (IDS) installed in the switch of the SDN takes the values of the features for which it has been trained and checks whether the coming traffic is normal traffic or attack traffic. If the traffic is predicted as the attack traffic, it is blocked by the switch itself. If the traffic is predicted as normal traffic, then the switch first searches its FlowTable to check whether or not the information related to dealing with this type of traffic exists. Suppose the information is already available in the FlowTable in the form of a flow entry. In that case, the action is taken accordingly. If there is no information available in the switch table, then the request to ask how to deal with these packets is sent to the controller using OpenFlow protocol in the form of a PacketIn request. The controller then sends the response using the same protocol but in the form of PacketOut action, and the sent rule is installed in the FlowTable in the form of flow entry.

3.1. Dataset Preprocessing

Dataset preprocessing is an essential part of any machine learning classifier. Under this, all non-numerical data of the dataset is converted into some numerical form. Also, it might give more weightage to higher numeric values, so all the numeric values are brought in the same scale. Only after performing the task of preprocessing the values of the dataset can it be used for training and testing the classifiers. In the current work, some essential operations mentioned below have been performed on the UNSW-NB15 dataset.

3.1.1. Feature Mapping of Categorical Features

Some features of the dataset contain categorical values, so to deal with such values, Label Encoding (LE) and One Hot Encoding (OHE) have been performed. In the case of Label Encoding, *'n'* different values are generated, and in the case of OHE' *n'*, different columns are generated if there are *'n'* different values in that feature[18].

3.1.2. Imputing the missing values

Some features contain missing values. Before using such features for training and testing, these missing values need to be removed. Many approaches are used to remove those missing values, like taking that feature's maximum, minimum, and average values to fill the missing values. Sometimes those rows/columns are removed altogether. In this work, the number of samples is sufficient, so the records having missing values have been removed.

3.1.3. Normalization

If the difference among the values of features is big, then it can train the model in a biased manner giving more weightage to the features having higher values. To deal with such conditions, standardization has been done to scale the value of this work. In standardization, the value of a feature is scaled in such a manner that its mean comes to zero, and the standard deviation comes to 1. The formula to perform normalization using a standard scaler is shown in Eq. (1)[19].

$$z = \frac{x_i - \mu}{\sigma} \quad (1)$$

where $\mu = \frac{1}{N}\sum_{i=1}^{N}(x_i)$ and $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$

3.2. Proposed Stack of Classifiers

For attack detection, an ensemble approach of classifiers called stacking has been used for the data plane of the SDN in this work. KNN and RF have been used as level 0 classifiers, and Logistic regression has been used as a level 1 classifier. The reason for selecting these classifiers has been mentioned in the below subsections.

Algorithm 1 Algorithm to train the Stack of Classifiers

```
Require: Training_Dataset $D_{TRN}$ (F, T), Testing_Dataset
    $D_{TST}$ (F, T), k for K-Fold cross-validation, list of
    classifiers C= {$C_1$, $C_2$, ....., $C_M$ }
Ensure: Stack of Trained Models
1:  Split the Training Dataset $D_{T RN}$ into k subsets
2:  $D_{T RN}$ = {$D_1$, $D_2$, . . . . . . , $D_k$}
3:  Train the base classifiers
4:  for m ← 1 To M do
5:    Train classifier with each subset of the dataset
6:    for d ← 1 To k do
7:        $C_m(F_d, T)$ ► T is the target or label of the
          dataset
8:    end for
9:  end for
10: Generate dataset for next level classifier (Final
    estimator C ' )
11: for m ← 1 To M do
        F ' = $C_m(D_{T ST}$ (F, T))
12: end for
13: Get the dataset for next level classifier
        D' = (F ', T)
14: Train second-level classifier C.'
        M' = C ' (D')
15: Save the Model
                dump( M' , "Model.pkl" )
```

3.2.1. K-Nearest Neighbor (KNN)

KNN is a supervised learning algorithm used for solving regression and classification problems. It is called a lazy learner [20], [21] as for any new point, it calculates the distance between the new point and other points. KNN handles the noise, giving the best accuracy in attack detection[22]. The default distance measuring method in KNN is Minkowski, but Manhattan has been found to perform best in [31], so it has been used in this work.

3.2.2. Random Forest (RF)

Random Forest (RF) is an ensemble supervised classifier that works on the bagging principle, whose diagram is shown in Fig.4. In bagging, the original dataset is divided into subsets. Let this be {S1, S2,…… ,and Sn}. These subsets are used to train base classifiers {M1, M2,………., Mn} respectively. Let the prediction given by these classifiers be {P1, P2,……., Pn}. The final decision is taken based on a majority voting basis. The reason for selecting RF is that RF not only selects the samples randomly but also selects features randomly. The second reason is that it reduces variance[24] and hence has been used in many similar works [25].

3.2.3. Logistic Regression

Logistic regression is a technique that is both easier to use and more effective for solving issues involving binary and linear classification. It is a classification model that can be implemented easily and works very well when applied to classes that can be separated linearly. It is a categorization method that sees extensive application in the business world. A statistical method for binary classification that may be extended to multiclass classification is known as the logistic regression model[26].

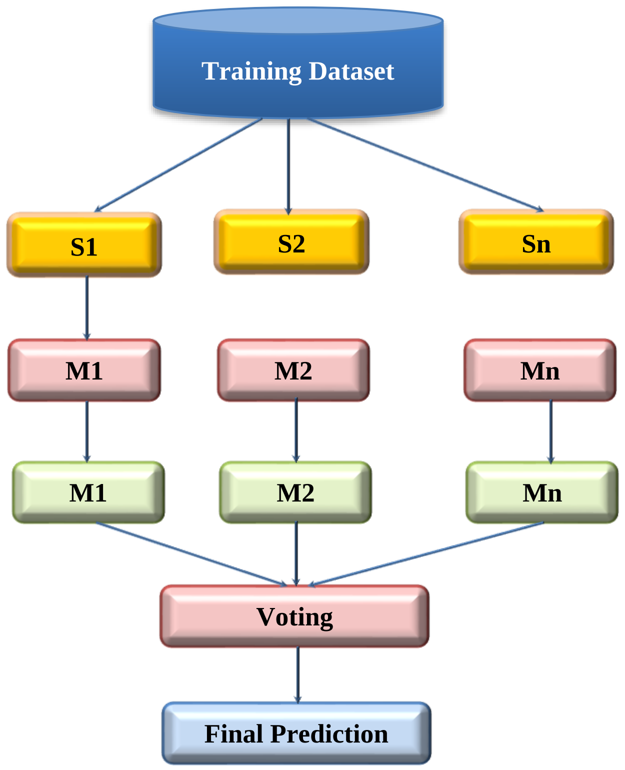

Fig. 4 Diagram of Bagging Method

4. Experiments and Result Analysis

This section describes the work done for the proposed SDN attack detection solution in the data plane. Python has been used as a programming language, SKlearn as the machine learning library, and Pandas, NumPy, Seaborn, and other libraries as supporting libraries. The UNSW-NB15 dataset has been used to evaluate the proposed work's performance. The operating system is Ubuntu 18.04, and the system has 8GB of RAM and an AMD Pro processor with a frequency of 1867 MHz.

4.1. Details about the UNSW-NB15 dataset

The suggested model has been evaluated using the UNSW-NB15 dataset [27]. UNSW Canberra's Cyber Range Lab used the IXIA PerfectStorm system to compile this dataset. They generated normal and synthetic contemporary attack traffic with the Argus and Bro-IDS tools and extracted 49 features and 9 attack classes. The dataset provided as training and testing data in CSV format has been used in this work. The train and test datasets have a total of 175,341 and 82,332 records, respectively. Fig.5 illustrates the sample counts for the training dataset based on various classes. Similarly, Fig.6 illustrates the sample counts for the testing dataset based on various classes.

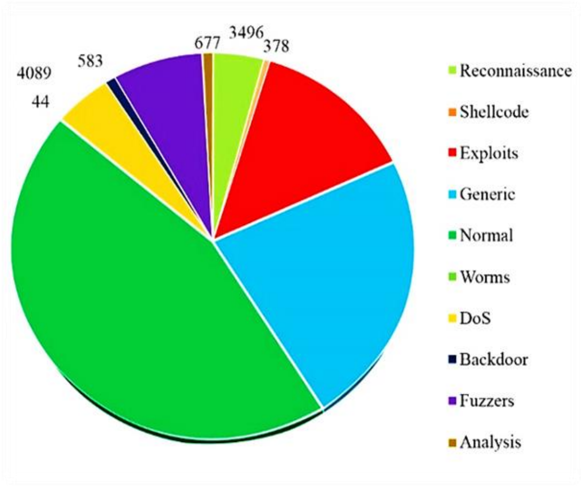

Fig. 5 Distribution of attacks in the UNSW-NB15 Training Dataset

4.2. Performance Measuring Metrics

Precision, Accuracy, F1 value, Recall, Prediction Time, and Cohen's Kappa Coefficient were used to assess the suggested model's performance. These have been explained in [28], [29]. Let the confusion matrix as shown in Table 1. The performance evaluation metrics are described as:

4.2.1. Accuracy

This is the ratio of all true predictions to total predictions, as given in Eq.2. A higher model accuracy rating indicates better performance.

Table 1. Confusion Matrix

		Predicted Output	
		Attack	Normal
Actual Output	Attack	TP	FN
	Normal	FP	TN

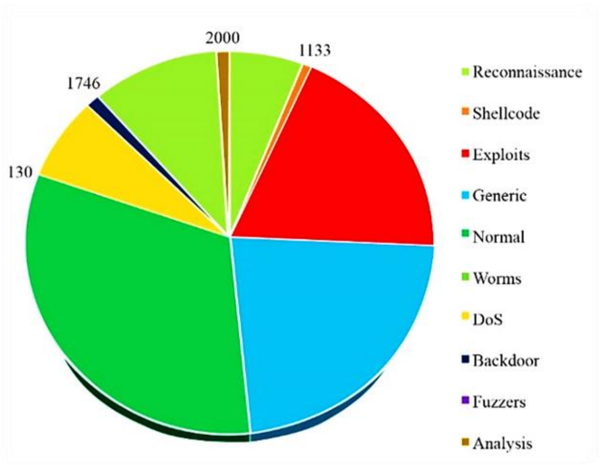

Fig. 6 Distribution of attacks in UNSW-NB15 Testing Dataset

4.2.2. Precision

It is the ratio of TP to TP & FP. The formula to calculate precision is shown in Eq.3. A model with a higher precision value has superior performance.

4.2.3. Recall

It is the ratio of TP to TP & FN. The formula to calculate recall is shown in Eq.4. A model with a higher recall value has superior performance.

4.2.4. F1-Score

It is the weighted average of Precision and Recall. The formula to calculate F1-Score is shown in Eq.5. A model's F1-Score with a higher value indicates superior performance.

4.2.5. Prediction Time

The prediction time shows the time taken by a classifier to predict the outcome by taking features as input.

4.2.6. Cohen's Kappa Coefficient

This metric comes in handy when dealing with multiclass and unbalanced datasets. In certain instances, accuracy, precision, recall, and other metrics cannot tell the whole story, but Kappa can. The Kappa coefficient is calculated using Eq. 6. p_o & p_e are observed agreement and projected agreement, respectively. Cohen's Kappa can be less or equal to one. The classifier is meaningless if the value is less than zero, whereas values near 1 (i.e., near 100 percent) indicate better performance.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

$$F1 - Score = \frac{2 * Recall * Precision}{Recall + Precision} \tag{5}$$

$$\kappa = \frac{p_o - p_e}{1 - p_e} = 1 - \frac{1 - p_o}{1 - p_e} \tag{6}$$

4.3. Feature Selection and Model Evaluation

Feature selection has been performed to get the best performance of the model. Two filter-based methods, namely Pearson's Correlation Coefficient (PCC) and Mutual Information, have been used sequentially to get the best number of features for the best performance. The value of PCC has been taken from 0.9 to 0.3. For one PCC value, features with a higher value than that PCC are removed for that pass. The features (K) having a lower value than the PCC of that pass are taken, and all the classifiers, namely Stack, RF, KNN, and LR, are evaluated for the said metrics.

Further, 10 percent of features with low mutual information are removed, the top 90 percent of features based on mutual information (new K) are taken, and the model is again evaluated. Reduction by 10 percent based on this approach continues for each cycle till 30 percent of the number of features for that pass remains. This complete approach has been shown in algorithm 2. Finally, a list of features giving the best performance is prepared, which are recommended for attack detection in the data plane. The different values of K for each PCC are shown in Table 2.

Table 2. Combination of PCC and K for each pass

Pass	**PCC**	**K**							
1	**0.9**	29	26	23	20	17	14	11	8
2	**0.8**	26	23	20	17	14	11	8	-
3	**0.7**	22	19	16	13	10	7	-	-
4	**0.6**	19	17	15	13	11	9	7	5
5	**0.5**	17	15	13	11	9	7	5	-
6	**0.4**	12	10	8	6	-	-	-	-
7	**0.3**	9	8	7	6	5	4	3	-

4.4. Performance analysis with the UNSW-NB15 dataset

The proposed method has been evaluated for all the mentioned metrics one by one, namely Precision, Accuracy, F1 value, Recall, Prediction Time, and Cohen's Kappa Coefficient. Stack has been found to perform best for the metrics Accuracy, Recall, F1, and CKC, while RF has been found to perform best for Precision, Prediction Time. This has been shown in Table 3.

Table 3. Name of the classifiers performing best for the mentioned metrics

SN	**Name of metrics**	**Best classifier**	**PCC**	**K**
1	Accuracy	Stack	0.3	5
2	Recall	Stack	0.3	5
3	F1	Stack	0.9	14
4	CKC	Stack	0.3	5
5	Precision	RF	0.9	26
6	Prediction Time	RF	0.7	10

4.4.1. Evaluation of the models under Accuracy, Recall & CKC

After running the process as shown in algorithm 2, it is found that the stack of the classifier for PCC=0.3 and K=5 gives the best accuracy. It is also found that the overall best values of Recall and CKC are also given by the stack for the same combination of PCC and K. These have been shown in Table 4.

Black-coloured values are the values given by the classifiers; blue values show the best value for the current combination of PCC and K, and green-coloured values show the overall best values for all the combinations of PCC and K. The greater the number of green colours, the better the model's performance.

Here stack gives the overall best performance for Accuracy, Recall, and CKC for PCC=0.3 and K=5. For the remaining metrics, i.e., precision, F1, and execution time, evaluation is shown one by one below sections.

Algorithm 2 Algorithm to select the best features.

```
Require: Training and Testing Datasets: train_x, test_x,
train_y, test_y.
    List of classifiers $C = \{C_1, C2, ....., Cm\}$
Ensure: List of best features.
1: Loop to select the features having Pearson's Coefficient
value from 0.9 to 0.2
2:  for x← 0.9 To 0.2, x = x - 0.1 do
3:  Select the features based on Person's Correlation
          Coefficient (PCC)
              corr_features = correlation(train_x,x) ► Get
          the Correlated Features
                  no_corr_features = count(corr_features) ►
              Count the selected features
4:  Select the features based on mutual information
(Between features and target)
               Mutual_info_of_ftr =
              mutual_info_classif((Train_x_no_corr_features),
              train_y)
5:  Select K best features from 100% to 20% of selected
features
       start  = count(mutual_info_of_f tr)
       end = math.ceil(count(mutual_info)/10) *2
     dec_val = math.ceil (count(mutual_in fo)/10)
6:   for y← start To end, y= y – dec_val do
            sel_k_ftr = SelectKBest(mutual_info_of_ftr, k= y)
              ► Select K best features
7:   Train & Test all the classifiers
8:      for m ← 1To M do
            $C_m$.fit(train_x, train_y)
            start_time = current_time()
            predict_y = $C_m$.predict(test_x)
            total_time = (current_time() - start_time)
                a_scr = accuracy_score(test_y, predict_y)
                p_scr = precision_score(test_y, predict_y)
               r_scr = recall_score(test_y, predict_y)
                f1_scr = f1_score(test_y, predict_y)
               ckc_scr = cohen_kappa_score(test_y,predict_y)
  9:      end for
10:   end for
11:  Compare the metrics given by all the classifiers and
select the list of features giving the best performance
12: end for
```

4.4.2. Evaluation of the Models Under Precision

The overall best precision value is given by RF for PCC=0.9 and K=26. It is shown in Table 5. RF is outperforming other classifiers under precision for this combination of PCC and K, but its overall performance under other metrics is lower as it secures only one green value.

4.4.3. Evaluation of the models under F1

The overall best value of F1 is given by stack for PCC=0.9 and K=14. It is shown in Table 6. Stack is outperforming other classifiers under F1 for this combination of PCC and K; its overall performance is lower under other metrics as it secures only one green value.

4.4.4. Evaluation of the models under Prediction Time

The overall best value of prediction time is given by RF for PCC=0.7 and K=10. It has been shown in Table 7. RF is outperforming other classifiers under prediction time for this combination of PCC and K, but its overall performance is lower under other metrics as it is securing only one green value.

Summarily, it is stated that the stack is giving the best performance under four parameters, namely, number of features(K), accuracy, recall, and CKC for the PCC=0.3 and K=5. For precision and prediction time, the value of RF is better than the stack, where RF gives values of 84.12 and 3.31, respectively, and the stack gives values of 81.95 and 4.65, respectively. Here the performance of RF is a bit better, but it requires a greater number of features, and it is not giving overall better values for other metrics. So, the stack can be used instead of RF for attack detection.

Table 4. Performance of different models under different metrics while considering the best accuracy

PCC	K	Classifier	Accuracy	Precision	Recall	F1	CKC	Time
0.3	**5**	Stack	**80.93**	81.95	**80.93**	**79.85**	**0.74**	**4.65**
0.3	5	RF	78.60	**82.02**	78.60	78.68	0.71	8.50
0.3	5	KNN	72.20	79.27	72.20	74.49	0.63	5.77
0.3	5	LR	57.48	50.16	61.34	51.41	0.38	39.20

Table 5. Performance of different models under different metrics while considering the best precision

PCC	K	Classifier	Accuracy	Precision	Recall	F1	CKC	Time
0.9	26	Stack	**79.04**	81.36	**79.04**	**79.83**	**0.71**	291.04
0.9	26	RF	75.43	**84.12**	75.43	77.75	0.67	**4.59**
0.9	26	KNN	73.81	79.40	73.81	75.88	0.65	852.85
0.9	26	LR	64.33	72.36	65.67	66.45	0.55	77.95

Table 6. Performance of different models under different metrics while considering the best F1-Value

PCC	K	Classifier	Accuracy	Precision	Recall	F1	CKC	Time
0.9	14	Stack	**80.06**	82.35	**80.72**	**80.83**	**0.73**	9.66
0.9	14	RF	76.87	**82.95**	76.87	78.00	0.69	**4.95**
0.9	14	KNN	73.75	79.53	73.75	75.79	0.65	6.70
0.9	14	LR	55.70	65.16	56.86	58.19	0.43	53.45

Table 7. Performance of different models under different metrics while considering the least prediction time

PCC	K	Classifier	Accuracy	Precision	Recall	F1	CKC	Time
0.7	10	Stack	**79.29**	79.98	**79.29**	**78.76**	**0.71**	5.52
0.7	10	RF	77.17	**81.98**	77.17	77.94	0.69	**3.31**
0.7	10	KNN	72.34	77.85	72.34	74.47	0.63	5.45
0.7	10	LR	60.61	62.88	61.87	60.90	0.47	43.88

5. Deployment of the Trained Model

In the case of the best F1 value, the stack gives better values with PCC=0.9 and K=14. In this case, the stack gives the value of F1 80.83. This is a bit better than 79.85, which is given by a stack with PCC=0.3 and K=5, but a greater number of features is the cost.

Also, the stack with PCC=0.9 and K=14 does not perform better for other metrics. So, it is recommended to use the stack for attack detection in the data plane with the top 5 features based on mutual information having the value of PCC less than 0.3. The names of such features are *sbytes*, *dbytes*, *sload*, *smean*, and *dmean*.

Once the model has been trained, it is distributed across all OpenFlow-enabled data plane devices from which attacks are conceivable.

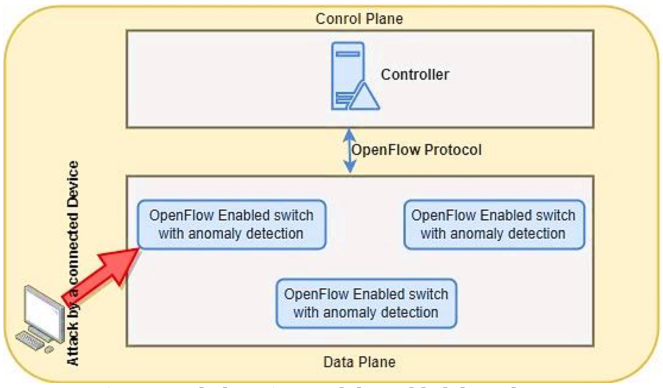
Conrol Plane
Controller
OpenFlow Protocol
OpenFlow Enabled switch with anomaly detection
OpenFlow Enabled switch with anomaly detection
OpenFlow Enabled switch with anomaly detection
Attack by a connected Device
Data Plane

Fig. 7 Attack detection module enabled data plane

This is seen in Fig. 7 Once an attack is launched from any data plane device, it can be detected by a module put in the data plane itself. The attack detection procedure is depicted in algorithm 3. If attack traffic is identified, it is

Algorithm 3 Algorithm for live attack detection.

```
Require: Network traffic coming to the Openflow-enabled devices.
Ensure: Prediction by the model M as Attack or Normal traffic
1: Capture the required features
      $F = \pi_{(f1,f2,f3...fn)}$ (Traffic)
2: Check if the traffic is an attack or a normal traffic
3: if M.predict(F) = attack then
4:    Drop the packets and exit.
5: else
6:        Search flow entries of all the FlowTables of all switches.
7:    for n ← 1 To count(Switch[flowtable]) do
8:        for i ← 1 To len($Switch_n$[flowtable]) do
9:            Check if any such entry already exists
10:           if Packet_Heater_Info == $[flow\_entry\_Match]_i$, then
```

```
11:                Follow the pre-installed rule.
12:            else
13:                Send Packet_In request to the controller.
14:            end if
15:        end for
16:      end for
17: end if
```

blocked in the data plane. If the traffic is identified as normal traffic, the SDN will function normally.

6. Contribution

This study examines a decentralized attack detection strategy for Software-Defined Networks. In this technique, instead of forwarding all traffic to the controller, the OpenFlow-enabled devices of the SDN perform preliminary threat detection work. This offers two distinct benefits. First, the data plane devices can identify the attack, reducing the controller's workload and allowing it to conduct other crucial tasks. Second, idle data plane resources can be exploited for some meaningful activities. Feature selection employing Pearson's Correlation Coefficient and Mutual information has been performed to improve attack detection lighter and more efficiently.

To achieve this goal, the data plane devices of the SDN employ a stack of classifiers that outperforms the individual classifiers used to build the stack. The algorithms for training the stack and classifiers, selecting the best characteristics, and detecting an attack in live traffic are described in detail.

7. Conclusion and Future Work

This work highlights the disadvantage of centralized IDS implemented in the controller. On the one hand, while the controller becomes tremendously busy after the attack, the data plane's resources are idle. To counteract these shortcomings, a structure with dual benefits is designed. First, a suggested classifier, stack, outperforming individual classifiers in attack detection is selected. Second, a decentralized IDS is installed in the data plane's OpenFlow-enabled devices as opposed to the controller, such that network attacks are only detected and dropped in the data plane.

The performance of the model is estimated utilizing the UNSB-NB15 datasets. The classifier stack surpasses the individual classifiers regarding the accuracy, recall, CKC, and feature count. Its performance is slightly inferior to that of other classifiers in terms of precision, F1, and prediction time, but when other parameters are considered, the difference is manageable; therefore, the classifier stack is proposed to be installed in the network's data plane to identify the attacks in the live traffic.

Future improvements can be made for multi-controller SDN, and the auxiliary controller can be utilized to cross-validate whether the attack detected by the switch is accurate. Additionally, deep learning can be utilized to increase the model's performance.

References

[1] Huseyin Polat, Onur Polat, and Aydin Cetin, "Detecting DDoS Attacks in Software-Defined Networks Through Feature Selection Methods and Machine Learning Models," *Sustainability*, vol. 12, no. 3, 2020, [CrossRef] [Google Scholar] [Publisher link]

[2] Jian Su et al., "Redundant Rule Detection for Software-Defined Networking," *KSII Transactions on Internet and Information Systems*, vol. 14, no. 6, pp. 2735–2751, 2020, [CrossRef] [Google Scholar] [Publisher link]

[3] Safaa Mahrach, and Abdelkrim Haqiq, "DDoS Flooding Attack Mitigation in Software Defined Networks," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 1, 2020, [CrossRef] [Google Scholar] [Publisher link]

[4] Lusani Mamushiane, Albert Lysko, and Sabelo Dlamini, "A Comparative Evaluation of the Performance of Popular SDN Controllers," *Wireless Days (WD)*, pp. 54–59, 2018, [CrossRef] [Google Scholar] [Publisher link]

[5] Ansam Khraisat et al., "Survey of Intrusion Detection Systems: Techniques, Datasets and Challenges," *Cybersecurity*, vol. 2, no. 1, 2019, [CrossRef] [Google Scholar] [Publisher link]

[6] Smitha Rajagopal, Poornima Panduranga Kundapur, and Katiganere Siddaramappa Hareesha, "A Stacking Ensemble for Network Intrusion Detection Using Heterogeneous Datasets," *Security and Communication Networks*, vol. 2020, 2020, [CrossRef] [Google Scholar] [Publisher link]

[7] Shanshan Yu et al., "A Cooperative DDoS Attack Detection Scheme Based on Entropy and Ensemble Learning in SDN," *EURASIP Journal on Wireless Communications and Networking*, vol. 2021, 2021. [CrossRef] [Google Scholar] [Publisher link]

[8] R. Sanjeetha et al., "Mitigation of Controller Induced DDoS Attack on Primary Server in High Traffic Scenarios of Software Defined Networks," *International Symposium on Advanced Networks and Telecommunication Systems (ANTS)*, pp. 1-6, 2018. [CrossRef] [Google Scholar] [Publisher link]

[9] Jin Ye et al., "A DDoS Attack Detection Method Based on SVM in Software Defined Network," *Security and Communication Networks*, vol. 2018, 2018, [CrossRef] [Google Scholar] [Publisher link]

[10] Yehuda Afek, Anat Bremler-Barr, and Lior Shafir, "Network Anti-Spoofing with SDN Data Plane," *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017. [CrossRef] [Google Scholar] [Publisher link]

[11] Liang Tan et al., "A New Framework for DDoS Attack Detection and Defense in SDN Environment," *IEEE Access*, vol. 8, pp. 161908–161919, 2020, [CrossRef] [Google Scholar] [Publisher link]

[12] Kubra Kalkan et al., "JESS: Joint Entropy-Based DDoS Defense Scheme in SDN," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2358–2372, 2018. [CrossRef] [Google Scholar] [Publisher link]

[13] Prashant Kumar et al., "SAFETY: Early Detection and Mitigation of TCP SYN Flood Utilizing Entropy in SDN," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1545–1559, 2018. [CrossRef] [Google Scholar] [Publisher link]

[14] Kshira Sagar Sahoo et al., "An Evolutionary SVM Model for DDOS Attack Detection in Software Defined Networks," *IEEE Access*, vol. 8, pp. 132502–132513, 2020. [CrossRef] [Google Scholar] [Publisher link]

[15] Nidhi Dandotiya, Abhinandan Singh Dandotiya, and Shashikant Gupta, "Impact of Software Defined Networking for Wireless Sensor Networks," *SSRG International Journal of Computer Science and Engineering*, vol. 6, no. 4, pp. 6-10, 2019. [CrossRef] [Publisher link]

[16] Sapna Singh Kshatri et al., "An Empirical Analysis of Machine Learning Algorithms for Crime Prediction Using Stacked Generalization: An Ensemble Approach," *IEEE Access*, vol. 9, pp. 67488–67500, 2021, [CrossRef] [Google Scholar] [Publisher link]

[17] Mohammed Al-Sarem et al., "An Optimized Stacking Ensemble Model for Phishing Websites Detection," *Electronics*, vol. 10, no. 11, 2021, [CrossRef] [Google Scholar] [Publisher link]

[18] Mwamba Kasongo Dahouda, and Inwhee Joe, "A Deep-Learned Embedding Technique for Categorical Features Encoding," *IEEE Access*, vol. 9, pp. 114381–114391, 2021, [CrossRef] [Google Scholar] [Publisher link]

[19] Gautam Srivastava et al., "An Ensemble Model for Intrusion Detection in the Internet of Softwarized Things," *ACM International Conference Proceeding Series*, pp. 25–30, 2021. [CrossRef] [Google Scholar] [Publisher link]

[20] Shi Dong, and Mudar Sarem, "DDoS Attack Detection Method Based on Improved KNN with the Degree of DDoS Attack in Software-Defined Networks," *IEEE Access*, vol. 8, pp. 5039–5048, 2020, [CrossRef] [Google Scholar] [Publisher link]

[21] Mamta Punjabi, and Gend Lal Prajapati, "Lazy Learner and PCA: An Evolutionary Approach," *2017 Computing Conference*, pp. 312–316, 2017. [CrossRef] [Google Scholar] [Publisher link]

[22] Anupama Mishra et al., "Classification Based Machine Learning for Detection of DDoS Attack in Cloud Computing," *IEEE International Conference on Consumer Electronics*, 2021, [CrossRef] [Google Scholar] [Publisher link]

[23] Sangeetha M.V, and Bhavithra J, "Applying Packet Score Technique in SDN for DDoS Attack Detection," *SSRG International Journal of Computer Science and Engineering*, vol. 5, no. 6, pp. 20-24, 2018. [CrossRef] [Publisher link]

[24] S. R. Khonde, and V. Ulagamuthalvi, "Ensemble and Feature Selection-Based Intrusion Detection System for Multi-Attack Environment," *2020 5th International Conference on Computing, Communication and Security (ICCCS)*, 2020. [CrossRef] [Google Scholar] [Publisher link]

[25] Rifkie Primartha, and Bayu Adhi Tama, "Anomaly Detection Using Random Forest: A Performance Revisited," *Proceedings of 2017 International Conference on Data and Software Engineering*, ICoDSE, pp. 1–6, 2017. [CrossRef] [Google Scholar] [Publisher link]

[26] Abdulhamit Subasi, *Practical Machine Learning for Data Analysis Using Python*, 2020. [Google Scholar] [Publisher link]

[27] Nour Moustafa, and Jill Slay, "UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set)," *2015 Military Communications and Information Systems Conference (MilCIS)* pp. 1-6, 2015. [CrossRef] [Google Scholar] [Publisher link]

[28] Salma Elhag et al., "A Multi-Objective Evolutionary Fuzzy System to Obtain A Broad and Accurate Set of Solutions in Intrusion Detection Systems," *Soft Computing*, vol. 23, pp. 1321–1336, 2019, [CrossRef] [Google Scholar] [Publisher link]

[29] Samson Ho et al., "A Novel Intrusion Detection Model for Detecting Known and Innovative Cyberattacks Using Convolutional Neural Network," *IEEE Open Journal of the Computer Society*, vol. 2, pp. 14–25, 2021. [CrossRef] [Google Scholar] [Publisher link]

[30] Yang Wang et al., "SGS: Safe-Guard Scheme for Protecting Control Plane against DDoS Attacks in Software-Defined Networking," *IEEE Access*, vol. 7, pp. 34699–34710, 2019, [CrossRef] [Google Scholar] [Publisher link]

[31] Diyana Tehrany Dehkordy, and Abbas Rasoolzadegan, "DroidTKM: Detection of Trojan Families Using the KNN Classifier Based on Manhattan Distance Metric," *2020 10th International Conference on Computer and Knowledge Engineering, ICCKE*, pp. 136–141, 2020. [CrossRef] [Google Scholar] [Publisher link]