IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 9, 2020, accepted January 16, 2021, date of publication January 26, 2021, date of current version February 3, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3054688

Towards Effective Network Intrusion Detection: From Concept to Creation on Azure Cloud

SMITHA RAJAGOPAL, POORNIMA PANDURANGA KUNDAPUR, AND HAREESHA K. S., (Senior Member, IEEE)

Department of Computer Applications, Manipal Institute of Technology, Manipal Academy of Higher Education (MAHE), Manipal 576104, India

Corresponding author: Hareesha K. S. (hareesh.ks@manipal.edu)

ABSTRACT Network Intrusion Detection is one of the most researched topics in the field of computer security. Hacktivists use sophisticated tools to launch numerous attacks that hamper the confidentiality, integrity and availability of computer resources. There is an incessant need to safeguard these resources to avoid further damage. In the proposed study, we have presented a meta-classification approach using decision jungle to perform both binary and multiclass classification. We have established the robustness of our approach by configuring an optimal set of hyper-parameters coupled with relevant feature subsets using a production-ready environment namely Azure machine learning. We have validated the efficiency of the proposed design using three contemporary datasets namely UNSW NB-15, CICIDS 2017, and CICDDOS 2019. We could achieve an accuracy of 99.8% pertaining to UNSW NB-15 whereas the accuracy in the case of CICIDS 2017 and CICDDOS 2019 datasets has been 98% and 97% respectively. A distinctive ability of the proposed model lies in its finesse to detect thirty-three modern attack types considerably well. Unlike conventional stacking ensembles, the proposed solution relies on a train-test ratio of 40:60 to establish the legitimacy of predictions. We also conducted statistical significance tests to compare the performance of classifiers involved in the study. To extend the functionalities further, we have automated the proposed model that can be a reliable candidate for real-time network intrusion detection.

INDEX TERMS Azure, Bayes point machine, Decision jungle, Fisher score, locally deep SVM, meta-classification, mutual information, Spearman correlation coefficient, stacking, significance tests.

I. INTRODUCTION

Technological advancements occurring in the field of cybersecurity emphasize on the application of Artificial Intelligence (AI) techniques to improve the security landscape [1]. Over the years, both adversaries as well as the research community have been relying on AI approaches to offend and defend computer networks. Cyber criminals use well-equipped tools to compromise organizational assets whereas security experts count on modern machine learning algorithms to mitigate the ever increasing cyber threats.

DeepLocker [2], an AI based malware (presented at the Black Hat USA 2018 conference by IBM) conceals its payload and cannot be detected by many antivirus softwares. Furthermore, Generative Adversarial Networks (GAN) are often employed by attackers to gain unauthorized access. GAN [3] is a deep neural network architecture that is endowed with an innate ability to imitate real-world data. Besides, there are many applications built using machine learning techniques that can mimic human voices often used by hacktivists. We have noticed in recent times that vast amount of data is generated on networks. Thus it becomes indispensable to apply machine learning techniques to distinguish between malicious and normal network instances. Most of the recent cybersecurity applications are built using AI that focus on probabilistic, behavioral, mathematical and statistical techniques to address the recurring problem [4]–[7].

It is imperative to analyze the mammoth data generated on the networks using an effective deployment environment. Therefore, harnessing the power of both cloud computing and machine learning helps immensely to speed up the execution process [8]. Some authors have discussed the significance of machine learning algorithms for network intrusion detection by considering the cloud environment due to its scalability and elasticity [9]–[12].

Typically, an intrusion detection system (IDS) can be categorized as host-based or network-based as per their deployment strategies [13], [14]. The former monitors the

The associate editor coordinating the review of this manuscript and approving it for publication was Ahmed A. Zaki Diab.

host systems to detect suspicious activities whereas the latter inspects the network traffic to discover malicious events. Additionally, misuse (signature-based) and anomaly-based are the other two types of IDS based on detection mechanisms.

Signature-based intrusion detection systems operate by comparing the stored signature against an incoming pattern. On the other hand, anomaly-based IDS create normal profiles and diagnose deviations if any. Generally, a normal profile is formulated by closely monitoring the on-going activities of users, network and applications for a specific duration [15], [16].

We have composed this article in the following manner. **Section II** reviews some state of the art methods in the field of network intrusion detection. **Section III** highlights the importance of automated machine learning. **Section IV** outlines the important aspects of the three datasets used in the study. **Section V** emphasizes upon the application of stacking ensemble in the field of network intrusion detection. **Section VI** describes the experimental strategy followed to formulate the proposed framework. Subsequently, **Section VII** presents the empirical findings along with a brief discussion. Finally, **Section VIII** concludes the article.

To be precise, our contributions are:

- We have proposed a reliable intrusion detection framework that is based on stacking approach to identify thirty-three modern attack types.
- We have relied on some pragmatic base learners to further boost the performance of decision jungle, the meta-classifier.
- We have automated the stacking ensemble on Azure cloud and deployed it as a web service for better accessibility.
- We have validated the potentiality of our approach using standard metrics and experimental results suggest that the model can also handle a larger test set (60%) given a comparatively smaller training set (40%).
- We have compared the performance of both binary and multiclass classifiers using statistical significance tests.

II. RELATED WORK

In the context of the proposed work, we have emphasized on the concept of meta-learning to enable extensibility and adaptability. Although a few research endeavors [17], [18] have aimed at achieving reasonable performance by reducing the incidence of false alarms, execution time was not paid much attention.

We also assert that automation is a primary concern while deploying predictive models. Eventhough unprecedented success has been achieved using deep learning [19], automating the deep learning models for network intrusion detection poses several challenges in terms of Quality of Experience (QOE) and scalability [20]. Therefore, we have relied on shallow learning techniques in this work to achieve the desired performance.

Meta-classification in the proposed work is calibrated by considering some robust algorithms that are quite popular in the purview of machine learning. Combinative approaches, when employed in a classification problem context reduce bias and variance to a substantial extent. Hybrid approaches [21], [22] offer a way of combining algorithms to enhance performance. Thus integrating the functionalities of algorithms are mostly preferred in the field of machine learning.

The application of stacking is noteworthy in the realm of machine learning as it improves validation accuracy and establishes a robust basis for business intelligence [23], [24].

Meta-classification is implicit to stacking that works by acquiring knowledge from different base classifiers and aggregating their outputs to produce the final predictions [25].

Random forest and weighted K means clustering algorithms were used to build a hybrid framework. Random forest served as a classifier whereas k-means clustering was used to develop an unsupervised approach so that anomalous clusters could be determined based on the features [26].

An amalgamation of genetic algorithm and fuzzy logic culminated in a robust approach [27] to detect network intrusions and an accuracy of 96.53% and a false positive rate as low as 0.56% was achieved.

The application of KDD cup 99 dataset was found to be infeasible for this work due to the shortcomings associated with historical traffic and thus network data was obtained from the State University of Londrina. The data found in KDD cup 99 dataset, apart from being decades old also comprises of packet traces rather than flows and thus the anomaly detection approach could become computationally complex whenever the network is large enough [27].

An insightful study [28] that focused on the systematic categorization of anomaly detection was presented. This compendious article emphasized on the application of various analytical methods and usage of benchmark datasets for network intrusion detection.

C4.5 decision Tree and one class SVM were used to formulate misuse and anomaly detection models [29]. The known attack information contributed towards building normal profiles thereby demonstrating the competency of hybrid approaches.

A selective ensemble method called SELECT was proposed to identify anomalies. SELECT worked in two stages for combining multiple results from various detectors and then collectively derive their consensuses too [30]. Ensemble approaches have shown promising results in both supervised and unsupervised learning domains [31]–[34].

A comparative study [35] was conducted to assess the performance of SVM against different classifiers and the results indicated that the stacked implementation of SVM and random forest resulted in an accuracy of 97.5% whereas an accuracy of 91.81% was achieved by SVM individually upon validation using NSL-KDD dataset.

The proficiency of ensemble classification was demonstrated using weighted majority algorithm [36] and results of the classifiers were combined using Particle Swarm Optimization (PSO) generated weights thereby improving accuracy. Recent approaches to anomaly detection rely on distributed architecture that normally consists of a key component namely signature-based module further coalesced by an anomaly detection component to improve the overall attack detection rate [37].

Gaussian dissimilarity measure was used to obtain the best subset of features and the application of feature clustering seemed to be a promising solution for anomaly detection [38].

Selecting the best features can contribute towards maximizing the objective function and it can be achieved by choosing the top ranked features. Given the pattern recognition literature, feature selection is performed using filter-based, wrapper based and embedded methods. Quite interestingly, in recent years, nature inspired approaches are being used extensively for feature engineering.

An intelligent dynamic swarm based rough set was employed for feature selection and simplified swarm optimization was used to classify the samples of KDD cup 99 dataset. An accuracy of 93.3% was obtained using this approach by using only six features [39].

In order to eliminate irrelevant features, a wrapper-based approach namely cuttlefish optimization algorithm was employed and ID3 served as a classifier [40]. Cuttlefish operates by using two processes: reflection and visibility. When applied as a global search strategy, the aforesaid processes help to attain an optimal solution. Five features contributed towards achieving a favorable detection rate of 91% [40].

A subset consisting of 16 features contributed towards an average classification rate of 98.4%. The IDS [41] was developed using neurotree as the classification engine.

A random effects logistic regression model was developed to study the uncertainty factors involved in network characteristics and stepwise variable selection was applied to eliminate multicollinearity issues. A classification accuracy of 98.68% was accomplished by the model that considered KDD cup 99 dataset for validation [42].

GA-LR [43], a wrapper based approach that considered 20 features to propose an IDS was put forth using Genetic algorithm and logistic regression. Decision tree was used for classification. False alarm rate was reported as 6.39% with respect to UNSW NB-15 dataset whereas an accuracy of 81.42% was achieved using the aforementioned algorithmic combination.

Information gain was employed to select relevant features from UNSW NB-15 dataset to generate an accuracy of 85.78% although the false alarm rate was definitely higher (15.64%) [44].

III. AUTOMATED MACHINE LEARNING

Over the years, research in the field of network intrusion detection has focused scrupulously on the technical component but somehow the automation has been sidelined [45]. Network intrusion detection study is certainly a step beyond increasing attack detection rate and reducing false alarms. In other words, developing and deploying models that can be made accessible to end-users.

Apart from merely differentiating between normal network patterns and attacks, an attack-specific evaluation is often encouraged whenever we try automating intrusion detection models. Therefore, in the proposed work, we have also emphasized upon multi-class classification task.

The aforementioned line of reasoning compelled us to look into the recent articles that have emphasized on building robust intrusion detection models. We observed that there is a dearth of research initiatives in the field of network intrusion detection using Machine Learning as a Service (MLaaS) paradigm.

The automation of models in a production-ready environment is often overlooked. Therefore, we consider automation in the proposed work so that the model could be made available to the end-users. In an automated machine learning set-up, it is possible to use machine learning models off the shelf thereby advancing business intelligence. Typically, automation is necessary whenever there arises a need to deploy computationally intensive machine learning models. One such feasible option to consider could be MLaaS for automation.

Conceptually, automated machine learning suggests that data scientists need not have to devise a neural network or a logistic regression model from scratch but can use automated algorithmic components available on MLaaS to leverage the benefits. As a proof of concept to automated machine learning, MLaaS helps data scientists to realize the potential of automation. In other words, the task of automation is accomplished using MLaaS paradigm.

Data, being the core of machine learning applications, is often difficult to store and process. MLaaS provides a scalable and efficient interface so that machine learning practitioners can focus only on conducting experiments using their datasets without being concerned about storage, computation and networking hassles.

As and when the complexity of data increases, machine learning poses several challenges as explained in [46] pertaining to processing speed, concept drift, variance and bias issues, noisy data, class imbalance, etc. Thus, automating a machine learning task becomes vitally important.

One of the cornerstones of automated machine learning is certainly Hyper-Parameter Optimization (HPO) [47] often difficult to achieve. Thus, in the proposed work, we employed a production-ready platform namely Azure Machine Learning to simplify the process of developing machine learning pipelines. One of the suggested ways to deploy a predictive model is to provision it on a virtual machine and gather actionable insights about its performance in a real-time environment.

Fundamentally, it is imperative to understand that there is a substantial difference between executing algorithms on a stand-alone device and cloud enabled platforms. Although

TABLE 1. Summary of recent seminal works on network intrusion detection.

Ref	Year	Algorithm or technique	Usage of MLaaS
[42]	2010	Random Effects logistic regression	NO
[41]	2012	Decision Tree	NO
[39]	2012	Simplified swarm optimization	NO
[40]	2015	Cuttlefish	NO
[36]	2016	SVM-kNN-PSO ensemble	NO
[53]	2016	C5.0 and ripper	NO
[43]	2017	Genetic algorithm, decision tree and logistic regression	NO
[54]	2017	Stacking dilated convolutional autoencoder	NO
[27]	2018	Genetic algorithm & Fuzzy logic	NO
[55]	2019	Convolutional neural network	NO
This study	**2021**	**Stacking [Decision Jungle as meta-learner]**	**Yes, Azure Machine Learning. Automated as a web service to detect thirty-three modern attack types. A relatively larger testing set is also employed to validate the approach and it consumes less training and execution time.**

we have used well-established algorithms in our work, it is important to note that all these algorithms are executed on an MLaaS paradigm that offers ample scope for automatic tuning of hyper-parameters thereby reducing human effort.

Enabled with Machine Learning Operations (MLOps), Azure Machine Learning helps in improving the performance of workflows.

In a real-time scenario that involves enormous networking data, the credibility of traditional machine learning frameworks is highly questionable. On the other hand, if emphasis is laid on the application of MLaaS environments like BigML, Algorithmia, DataRobot [48] and Azure Studio (used in the proposed work), it can help immensely to achieve auto-scaling or load balancing [49], [50].

Table 1 presents an overview of some recent approaches and also highlights the significance of our proposed approach.

IV. AN OVERVIEW OF THE DATASETS

An intrusion detection dataset is a representation of specific attack types targeting a network and it becomes imperative to choose appropriate datasets that reflect present-day attack scenarios and possibly assist further towards real-time implementations. In view of the above mentioned considerations, we have used three publicly available intrusion detection datasets namely UNSW NB-15, CICIDS 2017, and CICDDOS 2019 for experimentation.

UNSW NB-15 dataset has 42 features and samples found in the dataset belong to nine attack categories namely analysis, backdoor, denial of service, exploits, fuzzers, generic, reconnaissance, shellcode and worms. $\text{UNSW}_{\text{train}}$ and $\text{UNSW}_{\text{test}}$ are the pre-determined splits that consist of 1,75,341 and 82,332 samples respectively [51], [52]. The distribution of the various samples pertaining to each class is given in **Section VI**.

CICIDS 2017 dataset [56] is comparatively a newer offering that was created using benign-profile system for recording the abstract behavior of users. It encompasses benign and attack traces that were derived based on user characteristics of various protocols like HTTP/S, FTP, SSH and others. Seventy-eight features are found in this dataset and for the current study, we have considered 112,148 samples out of which 32,372 belong to benign class and 79,776 samples belong to various attack categories. Network instances pertaining to fourteen attack types are found in this dataset namely bots, infiltration, brute force, distributed denial of service, DOS golden eye, DOS HULK, DOS Slowhttptest, DOS Slowloris, FTP-patator, SSH-patator, heartbleed, portscan, sql injections and cross-site scripting.

CICDDOS 2019 [57] is another new dataset that has only DDOS attack instances in its topology with 87 features. Some attack scenarios captured in this dataset include LDAP, portmap, DNS, UDP-lag, UDP, NetBIOS, SSDP, MSSQL, NTP and Syn. In order to derive a reliable estimate of the proposed stacking ensemble, we selected 129,88 random samples for experimentation. For additional information on the taxonomy and testbed of DDOS attacks, [57] can be consulted.

V. STACKING ENSEMBLE FOR INTRUSION DETECTION

In the field of network intrusion detection, some inherent issues exist like low attack detection rate, high false alarm rate and lack of sufficient training examples. In order to address these challenges, relying on one learning algorithm may not be ideal.

Besides, ensembles contribute towards improving classification accuracy and facilitate a reliable mechanism to deal with minority classes quite effectively [58]. Another significant advantage of using ensemble approaches is that they assist in adapting to the ever increasing demands of dynamic network traffic. As elaborated in [58], the primary techniques used to combine classifiers are bagging, boosting and stacking.

The primary advantage of stacking is meta-classification that helps in generalizing to unseen/unfamiliar samples adeptly. As highlighted in [59], stacking also helps in reducing false positive rate to a great extent. Ensemble-based approaches have an advantage of combining the power of multiple classifiers thereby reducing misclassifications. Hence it is advisable to employ ensemble approaches in the field of network intrusion detection [59]. A value-added

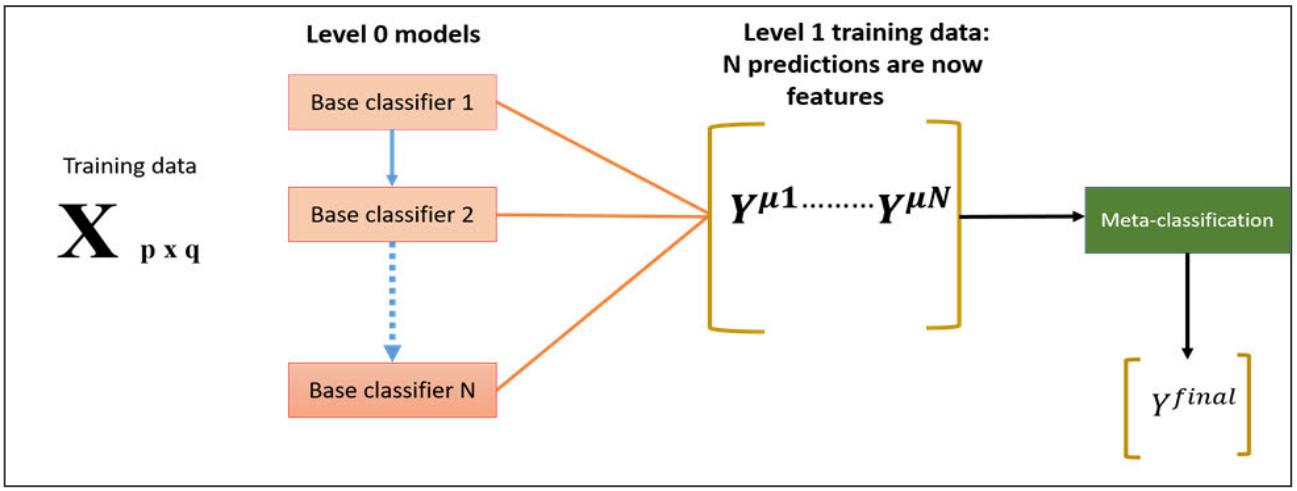

FIGURE 1. Conceptual procedure of stacking ensemble.

contribution to the existing literature could be the development of ensembles using MLaaS paradigm.

The predictive capability of a stacked ensemble can be tested vigorously using MLaaS paradigm since it offers a production-ready environment. Besides simplifying and customizing machine learning pipelines, another significant advantage of using MLaaS paradigm pertains to efficient resource allocation.

Typically, it is quite tedious to train, test and deploy ensembles but with newer and scalable paradigms like MLaaS, the entire process can be simplified to achieve superior performance. The current literature lacks discussion related to the application of ensemble based approaches to network intrusion detection using MLaaS paradigm.

Therefore, we have made an attempt to use an integrated development environment like Azure Machine learning to train, test and deploy the predictive model thereby accentuate its benefits.

We used Azure Machine Learning Studio to build the stacking ensemble. The reason to choose the aforementioned predictive analytics interface can be attributed to its prowess to merge data science and cloud resources thereby offering a convenient API to both data scientists and cloud enthusiasts. The conceptual workflow of stacking ensemble is depicted in **FIGURE 1**.

The current literature lacks sufficient evidence regarding the statistical significance needed to determine the performance difference among algorithms. We have performed statistical significance tests in our work as discussed in **Section VI**.

When we started our preliminary investigation of choosing the base classifiers and meta-learner, there were various options to choose from in terms of some well-established algorithms. Initially, we executed the algorithms independently and the overall performance of decision jungle was found to be the best for both binary and multiclass classification tasks.

In the proposed work, decision jungle plays the role of meta-learner to produce the optimal predictions. It is noteworthy that the application of decision jungle in the field of network intrusion detection has not been explored much. As compared to traditional algorithms, decision jungle uses multiple learning models and is equipped with the power of ensemble learning thereby surpassing other conventional classifiers.

- **Decision Jungle**

A recent extension to decision forest is the decision jungle algorithm that was selected as the meta-learner in the proposed work. Directed Acyclic Graphs (DAGs) form the core of this algorithm that is also an established method to conserve memory space [60].

As compared to a typical decision tree model, decision jungle contributes towards improving generalization. Apart from exhibiting strong discriminative power, it also limits the exponential growth of decision trees [60].

Decision jungle has an innate ability to overlook noisy features. We also contemplated on using decision forest for meta-classification but the execution time of decision forest was considerably longer during the trials.

For imbalanced data, quite common in machine learning, decision jungle seems to be appropriate. We adopted a weighted mechanism to ensure that the minority classes are not dominated by the majority classes. Generally, classifiers are affected by skewed distribution and become biased towards majority classes.

In order to overcome bias issues, minority classes are assigned larger weights. The weight assigned to a specific class (W_{class}) can be calculated using equation (1). n_{class} signifies the number of samples found in a class whereas n_i indicates the total number of samples found in the dataset. 'p' represents the number of class labels.

$$W_{class} = \frac{\frac{1}{\log(n_{class}+1)}}{\sum_{i=1}^{p} \frac{1}{\log(n_i+1)}} \quad (1)$$

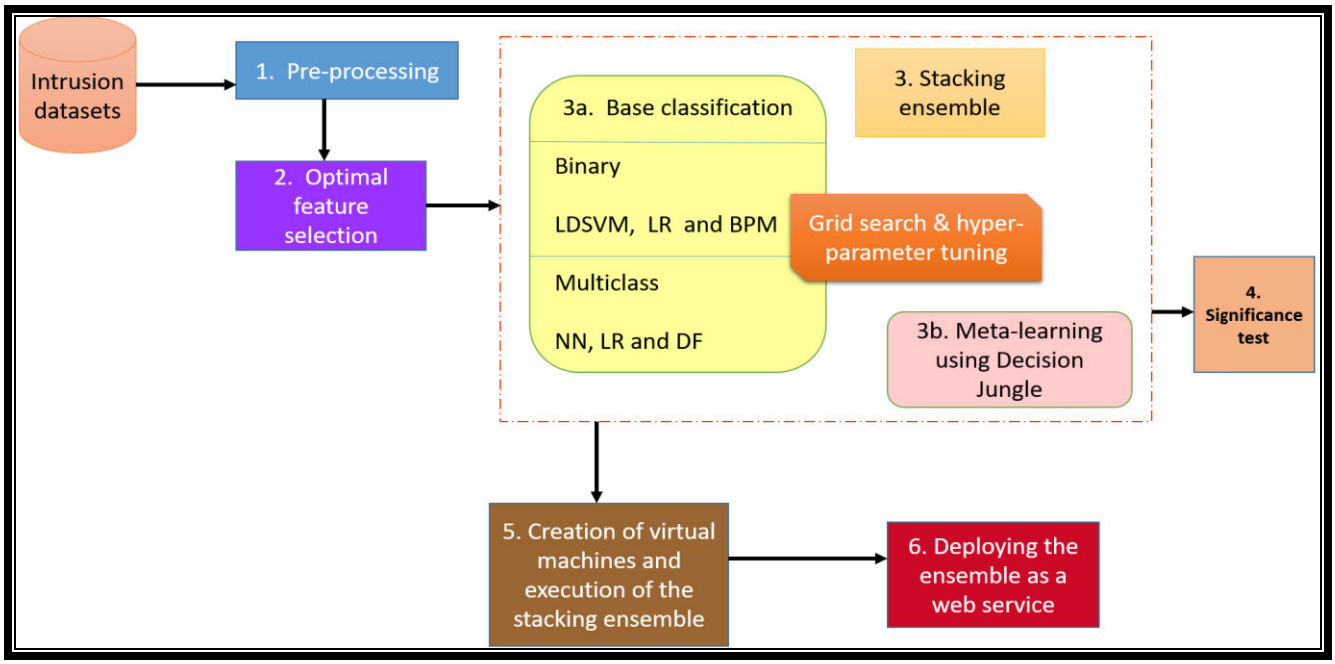

FIGURE 2. Overall Methodology of the proposed framework.

In the proposed work, decision jungle (meta-classifier) is equipped with the aforementioned weighted mechanism to address the issue of class imbalance in order to generate superior predictions.

The theoretical aspects of the remaining base-classifiers used in the proposed work and their technical significance can be consulted from [61]–[64].

The overall methodology of the proposed work was devised in 6 stages as shown in **FIGURE 2**. As a value added contribution to the existing literature, we have extended the functionality of the proposed predictive model to operate on three virtual machines using Azure portal. The idea behind using this strategy is to facilitate a web-service to serve the cloud tenants who subscribe for it as per their requirements.

We capitalized on the Azure cloud infrastructure to develop the stacking ensemble owing to its evolutionary maturity over the years [65]. We created machine learning pipelines on Azure cloud by importing the machine learning libraries from its rich collection of modules aimed at pre-processing, feature selection and classification.

VI. EXPERIMENTAL APPROACH

We conducted experiments using three train-test ratios such as 60:40, 50:50 and 40:60. It is noteworthy that the most dependable estimate of any implementation can be affirmed using a train-test split that considers three characteristics like size, randomness and uniqueness of the samples present in the test set. It is worthwhile to mention that we have based our inference about the performance of the stacking ensemble by considering the largest test set from the aforementioned train-test distributions i.e., 40:60.

Our comparative study also revealed that Dendron [17] is a recent research endeavor that considered a larger testing set for experimentation while training the algorithms using a relatively smaller training set. Such breakthroughs further establish the fact that machine learning practitioners are relying on reliable estimates to build robust intrusion detection models.

We have enumerated the distribution of the three datasets in **TABLES 2 – 4**. It must be noted that independent test sets have been considered for experimentation.

Step 1: Pre-processing

With reference to intrusion detection, the datasets used for experimentation encompass different kinds of features like discrete, continuous and symbolic with contrasting ranges that may lead to slow convergence. Machine learning algorithms cannot deal with such formats and thus it becomes necessary to do pre-processing. Boolean features do not require any pre-processing whereas all nominal features require a mapping to integer values. We applied logarithmic scaling (base 10) to decrease their range. Min-max normalization was applied to determine the minimum and maximum value of i^{th} feature so that each feature value could be transformed to [0,1] by using equation (2).

$$v' = \frac{vi - \min i}{\max i - \min i} \tag{2}$$

min_i *and* max_i denote the minimum and maximum values of the feature respectively whereas v_i is the value of feature at time i.

The idea behind doing this is to ensure that the training, validation and test sets include the same proportion of instances of each class as contained in the original dataset.

TABLE 2. Distribution of samples in UNSW NB-15 train and test sets.

Class	*Training set (40%)*	*Testing set (60%)*
Normal	37000	56000
Analysis	677	2000
Backdoor	583	1746
Denial of Service	4089	12264
Exploits	11132	33393
Fuzzers	6062	18184
Generic	18871	40000
Reconnaissance	3496	10491
Shellcode	378	1133
Worms	44	130
Total	**82332**	**175341**

TABLE 3. Distribution of samples in CICIDS 2017 train and test sets.

Class	*Training set (40%)*	*Testing set (60%)*
Benign	5584	8171
LDAP	3641	5506
UDP	4671	6976
UDP-lag	4733	7143
Portmap	4874	7279
DrDOS_MSSQL	3070	4500
DrDOS_NTP	4172	6478
DrDOS_NetBIOS	3261	4808
DrDOS_DNS	7580	11563
DrDOS_SSDP	8298	12310
SYN	2068	3195
Total	**51952**	**77929**

TABLE 4. Distribution of samples in CICDDOS 2019 train and test sets.

Class	*Training set (40%)*	*Testing set (60%)*
Benign	12955	19417
FTP-patator	3233	4705
SSH-patator	2347	3550
DOS Golden Eye	2296	3548
DOS HULK	8413	12475
DOS Slowloris	1131	1710
DOS Slowhttptest	2045	3187
Bot	772	1194
Portscan	6039	9055
Heartbleed	5	6
SQL injections	6	15
Cross-site scripting	256	400
Brute Force	614	894
DDOS	4738	7111
Infiltration	4	27
Total	**44854**	**67294**

It is important to note that only few samples are found pertaining to heartbleed (11) SQL injection (21) and infiltration (31) in the original dataset also.

Step 2: Optimal feature selection

We applied three filter-based feature selection methods namely Fisher score, mutual information and spearman correlation to formulate the best subset of features. **Tables 5-7** depict the VIF values of features pertaining to three datasets. It is worthwhile to mention that 9 features were selected based on mutual information from the set of 42 features found in UNSW NB-15 dataset. In order to select the 12 relevant features from CICIDS 2017 dataset that contains 78 attributes, we used Fisher score. A non-parametric measure called

TABLE 5. VIF of pertinent features related to UNSW NB-15 dataset.

Feature index	*VIF*
4	1.65
6	1.6
8	2.2
9	2.36
11	2.7
13	2.9
14	3.12
17	3.4
19	3.58

TABLE 6. VIF of pertinent features related to CICIDS 2017 dataset.

Feature index	*VIF*
2	1.93
16	1.98
17	2.44
18	2.68
19	2.82
20	2.94
21	3.14
22	3.16
23	3.29
24	3.30
25	4.15
35	4.27

TABLE 7. VIF of pertinent features related to CICDDOS 2019 dataset.

Feature index	*VIF*
8	1.85
10	1.9
12	2.36
14	2.47
16	2.6
18	2.8
20	3.06
21	3.33
24	3.69
29	3.77
30	3.89
32	4.17
36	4.4

spearman correlation was applied on CICDDOS 2019 dataset to select 13 features from the available set of 87 features.

- **Fisher score**

Fisher score is relatively a simple method to obtain the scores of attributes in the dataset. As per equation (3), Fisher Scores (FS) were calculated and 12 pertinent features were included for classification from CICIDS 2017 dataset. Based upon threshold that is computed by considering the average fisher score value, the attribute space is decided. It can be noted that a and b are the classes in consideration. n_a refers to the samples found in the dataset. μ_i indicates the mean score of the features and $\mu_{i,a}$ signifies the mean score of the features in a^{th} class. $\sigma_{i,a}$ indicates the variance score of the features found in a^{th} class.

$$FS = \frac{\sum_{a=1}^{b} na(\mu_{i,a} - \mu_i)^2}{\sum_{a=1}^{b} na\sigma_{i,a}^2} \quad (3)$$

- **Mutual Information (MI)**

MI is often applied in information theory to determine the dependency between two random variables X and Y. MI is a measure of information on Y furnished by X [66]. If MI between X and Y is zero, then are supposedly independent. Formally, MI can be calculated using equation (4).

$$I(X:Y) = \sum_{x \in X} \sum_{y \in Y} P(x,y) \log \frac{P(x,y)}{P(x)P(y)} \tag{4}$$

P(X) and P(Y) are the marginal distributions of X and Y.

- **Spearman correlation coefficient**

Spearman correlation coefficient is calculated using the ranks of the variables but not by considering actual values. In accordance with Pearson correlation coefficient, Spearman coefficient also ranges from -1 to +1 to quantify monotonic relationships between two variables [67].

Using equation (5), Spearman rank correlation coefficient (ρ) is calculated. R_i is the rank score of the ith x- value whereas S_i is the rank of the ith y-value. R indicated the mean score obtained by considering all x-values. Similarly, S indicates the mean score derived by taking into account all y-values. Thirteen features were chosen for classification using Spearman correlation from CICDDOS 2019 dataset that originally has 88 features.

$$\rho = \frac{\sum (R_i - \bar{R})(S_i - \bar{S})}{\sqrt{\sum (R_i - \bar{R})(S_i - \bar{S})}} \tag{5}$$

Depending solely on the top ranked features introduces multicollinearity. Therefore, each predictor variable was investigated further to refine the feature subset using both feature ranking and VIF [68], considering accuracy and false positive rate as prime indicators of performance. In order to address the issues of multicollinearity, the threshold of VIF was set between 1 to 5.

The *module statsmodels* was imported onto Azure studio to use the function *variance_inflation_factor*. Eg: It was observed that features like dpkts, dbytes and dloss found in UNSW NB-15 dataset exhibit considerable degree of correlation that in all possibilities perturbs the final predictions.

Step 3: Design of the stacking ensemble

The stacking ensemble was designed using different classifiers that were found to be ideal for binary and multiclass classification tasks. We applied the grid search [61] to tune the hyper-parameters of all the classifiers since it offers a wide coverage of the search space while tuning the appropriate values.

a) Configuration of stacking ensemble

The input matrix D represents the training set used by the base classifiers at level-0 that consists of N samples and M features. Y is the resultant matrix.

$$\mathbf{N}\left\{\overbrace{\left(\mathbf{D}\right)}^{M}\left(\mathbf{Y}\right)\right.$$

With respect to binary classification, three base learners were taken into account and their hyper-parameters were tuned accordingly. F1-metric was used to assess the performance of each classifier since it is an ideal metric when unbalanced datasets are involved. Bayes point machine, logistic regression and locally deep SVM operated at level-0 to learn the samples from the training set. Due to its superior performance, decision jungle was selected as the level-1 classifier to generate final predictions.

b) Training the ensemble

We applied 10 fold stratified cross validation to obtain the predictions from each base learner ($CV_{1.....}CV_n$). Stratified cross validation is touted as a reliable estimate of a model's performance [64]. The meta-learner is trained using the level-1 data.

$$\mathbf{N}\left\{\left(\mathrm{CV}_1\right).....\left(\mathrm{CV}_n\right)\left(\mathrm{Y}\right) \longrightarrow \mathrm{N}\left\{\overbrace{\left(\mathrm{D}\right)}^{\mathrm{M}}\left(\mathrm{Y}\right)\right.\right.$$

c) Predictions

When the testing set is given to the meta-learner, it generates the final predictions.

We designed the proposed stacking ensemble by utilizing four heterogeneous classifiers namely locally deep SVM, logistic regression, bayes point machine and decision jungle to perform binary classification.

- **Binary Bayes point machine (BBPM)**

BBPM was selected owing to its Bayesian properties that decrease overfitting to a considerable extent. The bayes point signifies the posterior mean of the weights. BPM is an improvement over SVM. The application of expectation propagation message passing algorithm makes BPM quite robust [69]. Two-class BPM was chosen as the module to create a workflow. A significant hyper-parameter considered for BPM is no of iterations that signifies the number of times the message passing algorithm should iterate on the training data.

- **Binary Logistic Regression (BLR)**

Another popular classifier namely logistic regression was chosen as a base classifier to obtain accurate predictions. It works by applying a logistic function to the training set and thus predicting the probability for all the data points.

During the implementation of logistic regression algorithm, parameter optimization was performed using Limited memory-Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm that searches the variable space using inverse of the hessian matrix [70]. Two-class logistic regression was used to create a workflow in order to optimize the predictions.

- **Binary Locally deep SVM (BLDSVM)**

LDSVM is a non-linear SVM that operates using sigmoid function to gain consistent speed. As the name indicates, it searches the local decision boundary instead of the complete feature space for achieving faster predictions.

Locally deep SVM is quite efficient in dealing with computationally deep features and is known to be exponentially

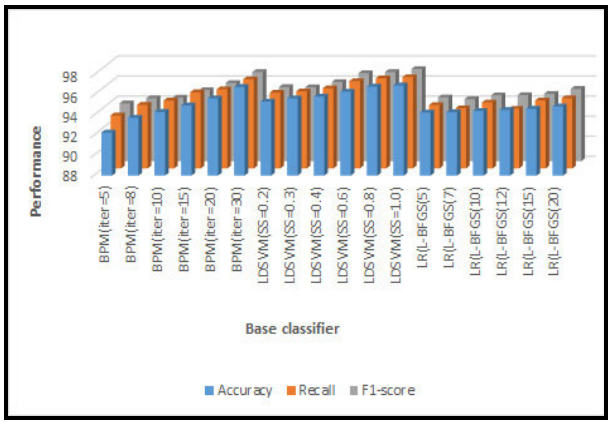

FIGURE 3. Performance comparison of base classifiers with varying hyper-parameters wrt UNSW NB-15 dataset iter = number of iterations, ss = sigmoid sharpness, L-BFGS = memory size for L-BFGS.

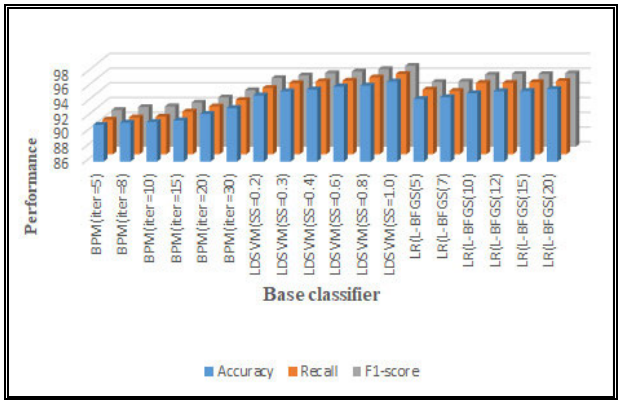

FIGURE 4. Performance comparison of base classifiers with varying hyper-parameters wrt CICIDS 2017 dataset iter = number of iterations, ss = sigmoid sharpness, L-BFGS = memory size for L-BFGS.

faster than other traditional implementations of SVM that further help in reducing computational costs [71].

RBF-SVM was also a contender for base level classification in the proposed work but locally deep SVM performed better in terms of achieving consistent speed and hence was chosen as one of the base classifier to build the stacking ensemble. An important factor that compelled us to choose LDSVM is due to the local learning that it exhibits.

Local learning is definitely preferred as it partitions a large problem at hand into sub problems and has the ability to choose training samples quite similar to test cases thus improving predictive accuracy [72].

A key parameter that was set to train locally deep SVM was sigmoid sharpness that offers a linear operating range to theta. A larger value if set introduces saturation that hinders the performance of the learner. Therefore, a diminutive value of 1 was assigned to sigmoid sharpness. Lambda refers to the regularization weight and was set as 0.1.

In order to illustrate the performance exhibited individually by the three binary classifiers, we have included bar charts as shown in **FIGURES 3 – 5**.

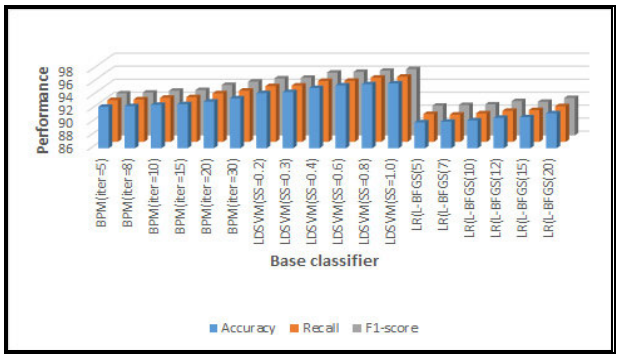

FIGURE 5. Performance comparison of base classifiers with varying hyper-parameters wrt CICDDOS 2019 dataset iter = number of iterations, ss = sigmoid sharpness, L-BFGS = memory size for L-BFGS.

The values of the critical hyper-parameters corresponding to the three base classifiers are considered along x-axis and the performance is considered along y-axis.

The best performance was exhibited by locally deep SVM, followed by bayes point machine. Although logistic regression emerged as an average performer, it was included in the study as one of the base-classifiers because a local learning algorithm like locally deep SVM compensates for the misclassifications done by a linear learner such as logistic regression.

Concerning empirical analysis, BPM(iter = 30), LDSVM(SS = 1.0) and LR(L-BFGS = 20) emerged as ideal base classifiers towards building the stacking ensemble that employed decision jungle as the meta-learner. We employed multiclass neural networks, multiclass logistic regression and multiclass decision forest as level-0 classifiers coupled with decision jungle as meta-learner to build a multiclass model.

- **Multiclass Neural Networks (MNN)**

Softmax was applied as the activation function to determine the multiple class labels pertaining to network instances. **Equation 6** can be applied to calculate the probability scores of various classes using softmax function. 'k' represents the number of classes, e_i^y refers to the exponential function applied for each network instance. The term 'e_j^y' ensures an appropriate probability distribution i.e., a valid range (0, 1) to all the output values generated by the function.

$$S(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{k} e^{y_j}} \tag{6}$$

The neural network classifier used in the proposed work performed 100 iterations during the learning process. The learning rate is a significant hyper-parameter since it is important to ascertain how efficiently the neural network model adapts itself to the learning process.

A very minute value of 0.1 was set as learning rate with respect to UNSW NB-15 dataset whereas 0.2 was assigned as learning rate while training the network to learn samples pertaining to CICIDS 2017 and CICDDOS 2019 datasets.

TABLE 8. Friedman average rank of all binary classifiers by considering F1-metric.

Algorithm	UNSW NB-15	CICIDS 2017	CICDDOS 2019	Average	Friedman avg. rank	p-value
BLDSVM	97.2456	97.0134	95.2394	96.4994	2.25	
BBPM	94.9349	92.6429	92.3258	93.3012	3.5	0.1354
BLR	95.2484	96.2509	91.7862	94.4285	2.75	
Proposed	**99.8497**	**98.5801**	**97.9747**	**98.8015**	**1.5**	

TABLE 9. Friedman average rank of all binary classifiers by considering recall.

Algorithm	UNSW NB-15	CICIDS 2017	CICDDOS 2019	Average	Friedman avg. rank	p-value
BLDSVM	99.2479	96.8865	96.2355	97.4566	2.65	
BBPM	99.3689	96.2805	97.7349	97.7947	2.5	0.9256
BLR	99.5799	96.8903	96.4376	97.6359	2.65	
Proposed	**99.9086**	**98.3093**	**97.4065**	98.5414	**2.2**	

TABLE 10. Friedman average rank of all binary classifiers by considering specificity.

Algorithm	UNSW NB-15	CICIDS 2017	CICDDOS 2019	Average	Friedman avg. rank	p-value
BLDSVM	99.5609	97.4509	90.1097	95.7071	3.5	
BBPM	99.4986	96.9098	92.0965	96.1683	2.65	0.0423
BLR	99.7609	97.5324	91.9180	96.4037	2.35	
Proposed	**99.6196**	**97.3796**	**93.3018**	96.767	**1.5**	

TABLE 11. Results of Quade post-hoc test wrt specificity metric for binary classification.

Comparison	Quade post-hoc p-value
LDSVM v/s Proposed	0.321
BBPM v/s Proposed	0.097
BLR v/s Proposed	0.049

- **Multiclass Decision Forest (MDF)**

It is worthwhile to mention that only 8 trees were constructed to avoid training overheads. Replicate was used as the resampling method to develop the workflow to ensure the creation of diverse trees.

The maximum depth of the tree and number of random splits were set as 32 and 128 respectively that also happen to be the default values. There is always a risk of overfitting whenever we attempt to increase the depth of the tree.

- **Multiclass Logistic Regression (MLR)**

While creating a multiclass workflow, penalty terms are added to the loss function. L1 weight was assigned a value 1. Normally, L2 weights are used when the underlying data suffers from severe multicollinearity but to a large extent, we addressed multicollinearity before creating the workflow while selecting the features.

Moreover, L1 weights are generally used to improve the precision and recall of predictive models. L1 or Least Absolute Shrinkage and Selection Operator (LASSO) [73] is the preferred regularization method whenever there is a need to ignore most of the irrelevant features.

Step 4: Statistical significance

While building machine learning models, it often becomes imperative to compare the performance of classifiers and the best way to achieve this is to perform statistical significance tests.

Due to the presence of outliers and different characteristics exhibited by feature vectors, an algorithm may behave differently on each dataset.

In the proposed work, we have looked into Friedman test and Nemenyi post-hoc tests [74]. Friedman test is a non-parametric test for analyzing the performance of classifiers on multiple datasets. Upon rejecting null hypothesis, post-hoc test is conducted to determine the pairwise comparisons.

In this context, the null hypothesis (H_0) suggests that there is no performance difference among classifiers whereas an alternate hypothesis (H_1) indicates that at least one classifier performs differently.

Suppose, 'd' refers to the number of datasets and 'k' signifies the number of classifiers, Friedman test statistic can be calculated as shown in **equation (7)**.

$$\text{Friedman statistic} = \frac{(d-1)Q}{d(k-1)-Q} \tag{7}$$

'Q' can be calculated as follows, shown in equation (8).

$$Q = \frac{12}{dk(k+1)} \sum_{j=1}^{k} \left(Rj - \frac{d(k+1)}{2} \right)^2 \tag{8}$$

Q is distributed with α degrees of freedom.
Rj refers to the ranks of the classifiers from j = 1,2,3.... k.
Rj can be calculated as shown in **eqnarray** (9).

$$Rj = \sum_{i=1}^{d} R\left(X_{ij}\right) \tag{9}$$

X_{ij} refers to the performance results of classifiers involved in the study. The lowest Friedman rank is assigned to the best performing classifier. The threshold for p-value is 0.05 and if there is any significance found, then quade posthoc test is performed.

TABLE 12. Friedman average rank of all multiclass classifiers by considering macro-average recall.

Algorithm	UNSW NB-15	CICIDS 2017	CICDDOS 2019	Average	Friedman avg. rank	p-value
MNN	76.0785	83.9876	82.9076	80.9912	3.75	
MDF	74.9807	85.8015	83.9934	81.5918	2.650	0.3806
MLR	77.8034	84.9028	83.3673	82.0245	1.975	
Proposed	**79.8906**	**86.1956**	**84.0949**	83.3937	**1.625**	

TABLE 13. Friedman average rank of all multiclass classifiers by considering micro-average precision.

Algorithm	UNSW NB-15	CICIDS 2017	CICDDOS 2019	Average	Friedman avg. rank	p-value
MNN	90.0101	90.3487	83.9990	88.1192	3.25	
MDF	90.0288	91.9979	84.2014	88.7427	2.5	0.0396
MLR	90.1472	91.7654	84.4590	88.7905	2.5	
Proposed	**90.2950**	**93.9290**	**84.6508**	**89.6249**	**1.75**	

TABLE 14. Friedman average rank of all multiclass classifiers by considering overall accuracy.

Algorithm	UNSW NB-15	CICIDS 2017	CICDDOS 2019	Average	Friedman avg. rank	p-value
MNN	84.4579	91.5534	81.0957	85.7023	3.25	
MDF	89.6790	92.7809	82.6094	88.3564	2.25	0.2506
MLR	83.8604	90.6035	83.8823	86.1154	2.75	
Proposed	**91.2992**	**96.5679**	**84.9386**	**90.9352**	**1.75**	

TABLE 15. Results of Quade post-hoc test wrt micro-average precision for multiclass classification.

Comparison	Quade post-hoc p-value
MDF V/s proposed	0.3940
MLR v/s proposed	0.2607
MNN v/s proposed	0.0480

Tables 8-15 illustrate the performance of base classifiers and the proposed model by considering recall, f1-metric and specificity, macro-average, micro-average and overall accuracy. It is worthwhile to note that the proposed model outshines the other algorithms with respect to all standard metrics.

In terms of the specificity metric, the results are significant ($p<0.05$). Therefore, we reject the null hypothesis. Quade post-hoc test is thus performed. From **Table 11**, it can be inferred that the performance of the stacking ensemble is better than logistic regression ($p = 0.049$).

However, the performance difference between locally deep SVM and proposed model is not too significant ($p = 0.321$). Similarly, there is no significant difference between the performance of bayes point machine and proposed stacking ensemble ($p = 0.097$).

We have also compared the performance of multiclass algorithms against the proposed stacking ensemble to comprehend the performance differences. We have considered macro and micro average scores to compare the performance of multiclass classifiers.

Macro average scores are helpful when imbalance is found among classes. Micro average is normally calculated by considering TP, TN, FP and FN of all the classes involved in the study [75].

Micro-average precision is determined by taking into account the sum of TP of all classes divided by the positive predictions.

Macro-average recall can be calculated by considering the recall scores of all the classes. On the multiclass classification front, the results are significant w.r.t. micro-average precision (p-value = 0.0396). Thus null hypothesis is rejected.

Upon conducting Quade post-hoc test, we derived the following results as shown in **Table 15**. It can be inferred that the performance of the stacking ensemble is quite superior than the performance of multiclass neural networks with respect to multiclass classification.

However, the proposed model's performance as compared to multiclass logistic regression and multiclass decision forest is not too significant.

Step 5: Creation of Virtual Machines

On Virtual machine 1 (VM 1), the stacking ensemble operates to identify all the attack types found in UNSW NB-15 dataset. Similarly, to identify attack types found in CICIDS 2017 and CICDDOS 2019 datasets, we executed the stacking ensemble on VM 2 and VM 3 respectively as evident from **FIGURE 6**.

If cloud tenants need a security service to restrain only DDOS attacks (CICDDOS 2019 consists of only DDOS attack patterns), then it is suggested that such tenants subscribe to VM 3 to fulfill their requirements according to pay-as-you-go policy.

All three VMs were provisioned using the Azure portal with 8 GB RAM on Intel Xeon platinum platform that supports 260 GB/second memory bandwidth for efficient processing. We plan to upgrade the aforesaid VM configuration in future to meet the growing requirements.

FIGURE 6. A view of three virtual machines dedicated to three datasets.

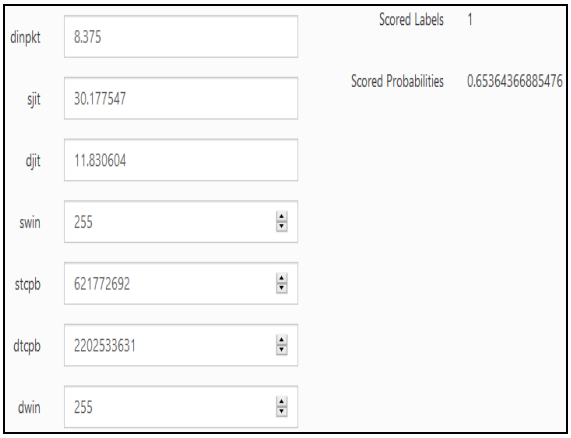

FIGURE 7. Predictive solution transformed into a web service (binary).

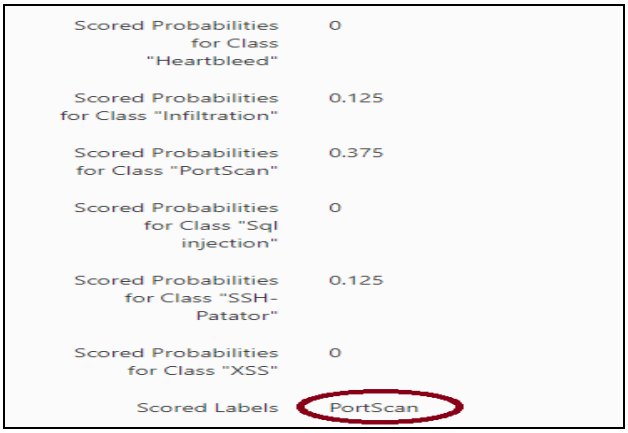

FIGURE 8. Predictive solution transformed into a web service (multiclass).

Each virtual machine facilitates a dedicated operating environment to users for mitigating various attack types belonging to heterogeneous data sources. Predictive solutions when automated can be consumed by cloud users.

Since we conceptualized a framework that involved automation, a comparatively larger test set was used to validate the effectiveness of the web service so that we could testify the requirements of tenants adeptly.

We have demonstrated the efficiency of the implementation by considering both binary and multiclass classification tasks with respect to three datasets as elaborated in **Section VII**.

Step 6: Deployment of web service

Typically, a normal instance is designated a probability score greater than 0.5 and the scored label assigned as 1 further indicates that the network instance under consideration is normal.

In a real-time environment, it becomes crucial to implement such automated models to reduce the time and effort of network security professionals thereby enabling rapid responses.

FIGURE 7 represents a use case transformed into a web service corresponding to a binary classification task. Upon testing the web service, it can be ascertained whether the output score pertains to attack or normal.

Given a random sample to testify the request-response scenario, the web service generates probability scores as end result.

FIGURE 8 represents a multiclass use case to identify specific category of attack. A particular network instance was identified as portscan attack with a probability score of 0.375 (as shown in **Fig.8**).

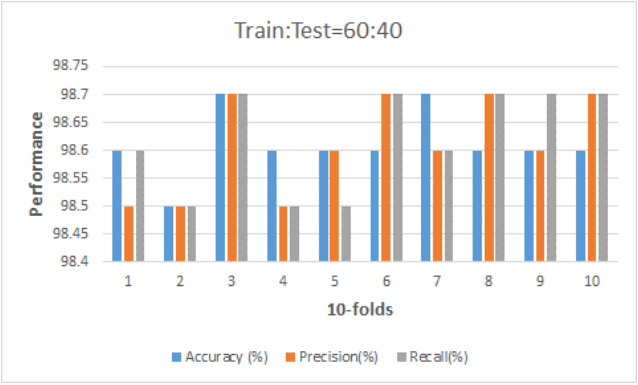

FIGURE 9. 10-fold cross validation results using 60:40 as train-test ratio w.r.t. UNSW NB-15 dataset.

Whenever a network sample is seen as nefarious, it is quite obvious that the probability scores would be less than 0.5 as noticeable in **FIGURE 8** and it can be observed that the scored probability for infiltration is 0.125. The scored probability is 0 for attack types heartbleed, SQL injection and Cross-site scripting (XSS).

It is evident that the highest probability score is assigned to *portscan attack* type due to which the scored label is *portscan* as predicted by the stacking ensemble.

VII. RESULTS AND DISCUSSION

We started our experimentation by taking into account all the features found in the three datasets.

TABLE 16. Prediction results on testing set of UNSW NB-15 dataset (60%) considering all features and pertinent features.

No of features	Accuracy (%)	FPR(%)	Testing time (seconds)
42 (All)	92.9	5.2	30
9	**99.8**	**0.38**	**4**

TABLE 17. Prediction results on testing set of CICIDS 2017 dataset (60%) considering all features and pertinent features.

No of features	Accuracy (%)	FPR(%)	Testing time (seconds)
78 (All)	94.3	5.16	23
12	**98**	**2.6**	**3**

TABLE 18. Prediction results on testing set of CICDDOS 2019 dataset (60%) considering all features and pertinent features.

No of features	Accuracy (%)	FPR(%)	Testing time (seconds)
87 (All)	90.3	9.2	29
13	**97**	**6.7**	**5**

TABLE 19. Confusion Matrix.

		Predicted	
		Attack	Normal
Actual	Attack	TP	FN
	Normal	FP	TN

However, only the pertinent features seemed to contribute substantially towards improving the accuracy and reducing execution time.

As a matter of fact, the proposed stacking ensemble performed in a slightly different manner given two scenarios namely, the presence of all features and pertinent ones as illustrated in **Tables 16-18**.

Table 19 provides a general view of the actual versus predicted outputs.

We conducted binary classification task to determine the efficiency of the proposed model to differentiate between attack and normal instances. At the initial step, binary classification was performed and subsequently true positives, false positives, true negatives and false negatives were recorded before pursuing the multiclass classification task.

UNSW NB-15 is a dataset that has both binary and multiclass labels in its composition but the remaining two datasets namely CICIDS 2017 and CICDDOS 2019 contain only attack-specific labels. Therefore, we labeled all the samples found in these two datasets as either benign or malicious to perform binary classification.

Concerning a perceptive analysis, it is evident that the false positives and false negatives are quite less corresponding to the three datasets that contributed majorly towards achieving superior predictions.

All the testing sets considered in the proposed work are independent. **Tables 20 – 25** depict the results obtained so far taking into account the testing composition of three datasets.

It is worthwhile to mention that the least execution time taken by the stacking ensemble to generate optimal predictions is 3 seconds with respect to CICIDS 2017 dataset.

TABLE 20. Confusion matrix on testing set of UNSW NB-15.

	Attack	Normal
Attack	119232	109
Normal	213	55787

TABLE 21. Results obtained on testing set of UNSW NB-15.

Accuracy	Precision	Recall	F1-score	FPR(%)
0.998	0.998	0.999	0.998	0.38

TABLE 22. Confusion matrix on testing set of CICIDS 2017.

	Attack	Normal
Attack	46985	808
Normal	511	18990

TABLE 23. Results obtained on testing set of CICIDS 2017.

Accuracy	Precision	Recall	F1-score	FPR(%)
0.98	0.989	0.983	0.984	2.6

TABLE 24. Confusion matrix on testing set of CICDDOS 2019.

	Attack	Normal
Attack	68695	1829
Normal	496	6909

TABLE 25. Results obtained on testing set of CICDDOS 2019.

Accuracy	Precision	Recall	F1-score	FPR(%)
0.97	0.99	0.97	0.978	6.7

When we considered 13 features for validating the effectiveness of the proposed classification framework pertaining to CICDDOS 2019 dataset, the execution time was reported as 5 seconds.

The theoretical importance of feature selection was discussed earlier but rational relevance of pertinent features is highlighted by focusing on the testing time that decreased considerably upon selecting a subset of features. Nonetheless, variation in accuracy with respect to all the three datasets, considering all features or pertinent ones has been quite trivial but a drastic decrease in false positive rate was obvious upon selecting only pertinent features.

It can be noted that the performance of the stacking ensemble has been quite desirable as the accuracy is quite high corresponding to all three datasets included in the study. Apart from accuracy and false positive rate, we considered some critical evaluation parameters to assess the performance of the meta-classification approach and the equations **(10)** to **(15)** represent the standard performance metrics.

True positives (TP) indicate the number of attack samples predicted as attacks by the model whereas normal instances predicted as normal is represented by True negatives (TN). False positives (FP) and False negatives (FN) deteriorate the performance of the intrusion detection model. False positives are generated when normal samples are predicted wrongly as attack by the model whereas the attack samples predicted as normal denote false negatives.

The binary classification results achieved by the proposed ensemble have been quite promising with respect to UNSW

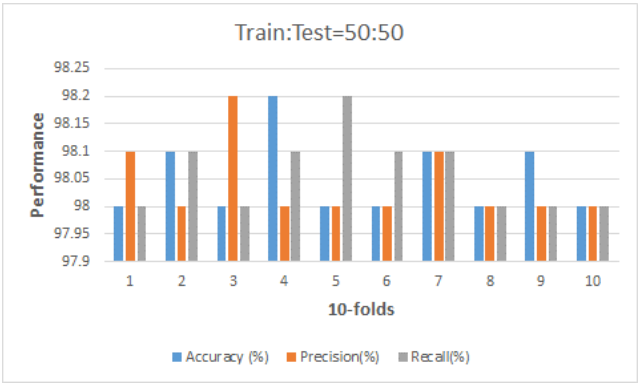

FIGURE 10. 10-fold cross validation results using 50:50 as train-test ratio w.r.t. UNSW NB-15 dataset.

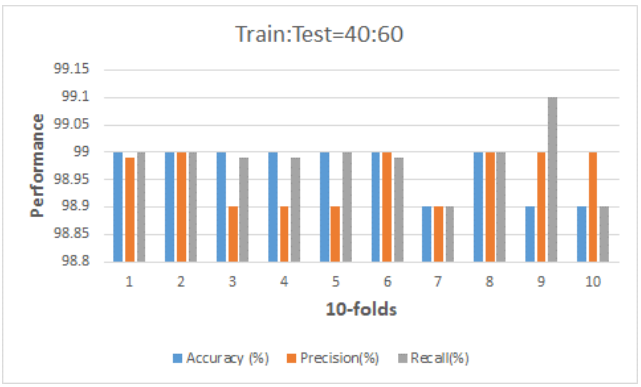

FIGURE 11. 10-fold cross validation results using 40:60 as train-test ratio w.r.t. UNSW NB-15 dataset.

NB-15 dataset since the attack detection rate is 0.999. The false positive rate is marginal corresponding to both UNSW NB-15 and CICIDS 2017 i.e., 0.38 and 2.6 respectively.

A slightly higher false positive rate, 6.7% was recorded pertaining to CICDDOS 2019 dataset although scores of other evaluation metrics have been quite promising.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

$$\text{Pr}ecision = \frac{TP}{TP + FP} \tag{11}$$

$$FPR = \frac{FP}{FP + TN} \tag{12}$$

$$\text{Re}call = \frac{TP}{TP + FN} \tag{13}$$

$$F1 - score = 2\left(\frac{\text{Pr}\,ecision * \text{Re}call}{\text{Pr}\,ecision + \text{Re}call}\right) \tag{14}$$

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{15}$$

We applied stratified cross validation to obtain a reliable estimate of the proposed ensemble. The idea behind using stratified cross validation is to ensure that each subset contains equal number of samples from each class so that there is a proper balance established [76].

It is very time consuming to administer pre-processing on the entire dataset. UNSW NB-15 is already available

FIGURE 12. 10-fold cross validation results using 50:50 as train-test ratio w.r.t. CICDDOS-2019 dataset.

FIGURE 13. 10-fold cross validation results using 40:60 as train-test ratio w.r.t. CICDDOS-2019 dataset.

in a pre-determined train-test ratio but CICIDS 2017 and CICDDOS 2019 are huge datasets with redundant samples. In order to extract only unique samples from these two datasets, we used a module found in Azure Studio called *Remove Duplicate Rows* thereby ensured the usage of independent training and test sets to obtain reliable estimates. We have illustrated the performance evaluation of the proposed model based on different population sizes as shown in **FIGURES 9-11** pertaining to UNSW NB-15 dataset.

Although there are only slight variations between accuracy, recall and precision scores obtained using three different train-test ratios corresponding to three datasets, it is worthwhile to emphasize upon the credibility of our proposed model.

The most reliable results pertain to the train-test ratio of 40:60 that further establishes the proficiency of our approach thereby corroborating that the proposed ensemble can indeed be a reliable candidate for real-time network intrusion detection.

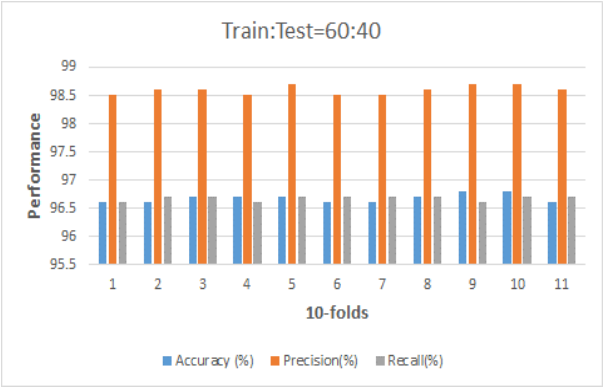

FIGURE 14. 10-fold cross validation results using 60:40 as train-test ratio w.r.t. CICDDOS-2019 dataset.

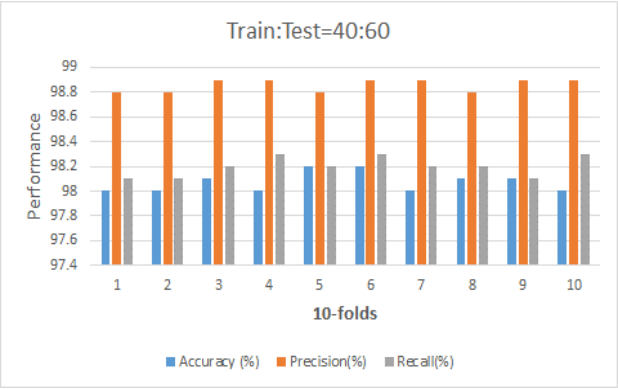

FIGURE 15. 10-fold cross validation results using 40:60 as train-test ratio w.r.t. CICIDS-2017 dataset.

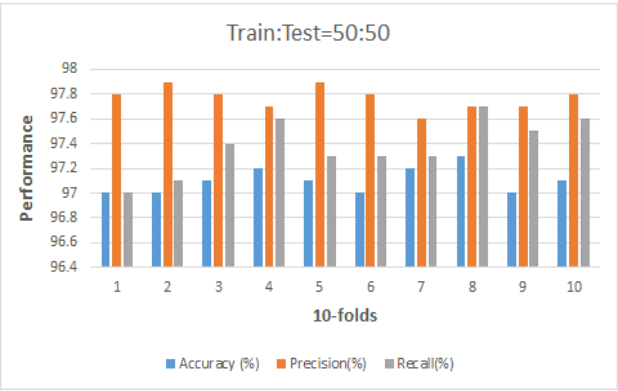

FIGURE 16. 10-fold cross validation results using 50:50 as train-test ratio w.r.t. CICIDS-2017 dataset.

FIGURES 12–14 is a demonstration of the performance w.r.t. CICDDOS-2019 dataset by considering different proportions of train-test ratios.

FIGURES 15-17 shows the performance of the proposed approach by considering three different proportions of train-test ratio wherein the number of folds (10-folds) and performance are taken along x-axis and y-axis respectively for CICIDS-2017 dataset.

In order to discern the capability of the proposed stacking ensemble in identifying different attack types, we broadened our study and performed multiclass classification task.

FIGURE 17. 10-fold cross validation results using 60:40 as train-test ratio w.r.t. CICIDS-2017 dataset.

The overall accuracy achieved during binary classification task is not enough to determine the performance of the classifier and thus we relied on a multiclass classification task to gauge the attack-wise detection of the proposed ensemble.

We have emphasized on recall and precision in our study of multiclass classification task since the former is a measure of classifier's completeness and the latter represents the exactness of the classifier.

The bar graph shown in **FIGURE 18** depicts the performance of the proposed model by considering recall and precision scores of all thirty-three modern attack types involved in the study.

The recall scores are quite superior pertaining to attack classes like generic (99.1%), exploits (93.6%), fuzzers (91%), reconnaissance (82.5%), shellcode (80.1%) and worms (89.2%).

The results affirm that the proposed model achieved the highest precision score with respect to generic attack type i.e., 99.9%.

Eventhough the number of samples found in analysis and backdoor classes are comparatively less, the proposed model has been quite competent enough to achieve a good precision score with respect to the two classes i.e., 95% and 92.18%.

Shellcode and worms are the other two minority classes found in the testing set and the precision scores with respect to these two minority classes are 96.28% and 94.3% respectively.

The least precision score obtained so far pertains to DOS (61.51%) but the comparative study reveals that Dendron [17] and GA-LR [43] approaches also reported the precision scores of DOS to be quite less that corresponds to 20.26% and 36.090% respectively.

The detection rate of the proposed ensemble approach with respect to backdoor is 32.4% and also the least recall score.

As compared to GA-LR [43] approach wherein the recall score was reported to be 6.925%, the proposed meta-classifier has fared better.

The testing samples considered for backdoor attacks correspond to 306 in [17] which is way lesser than the proposed

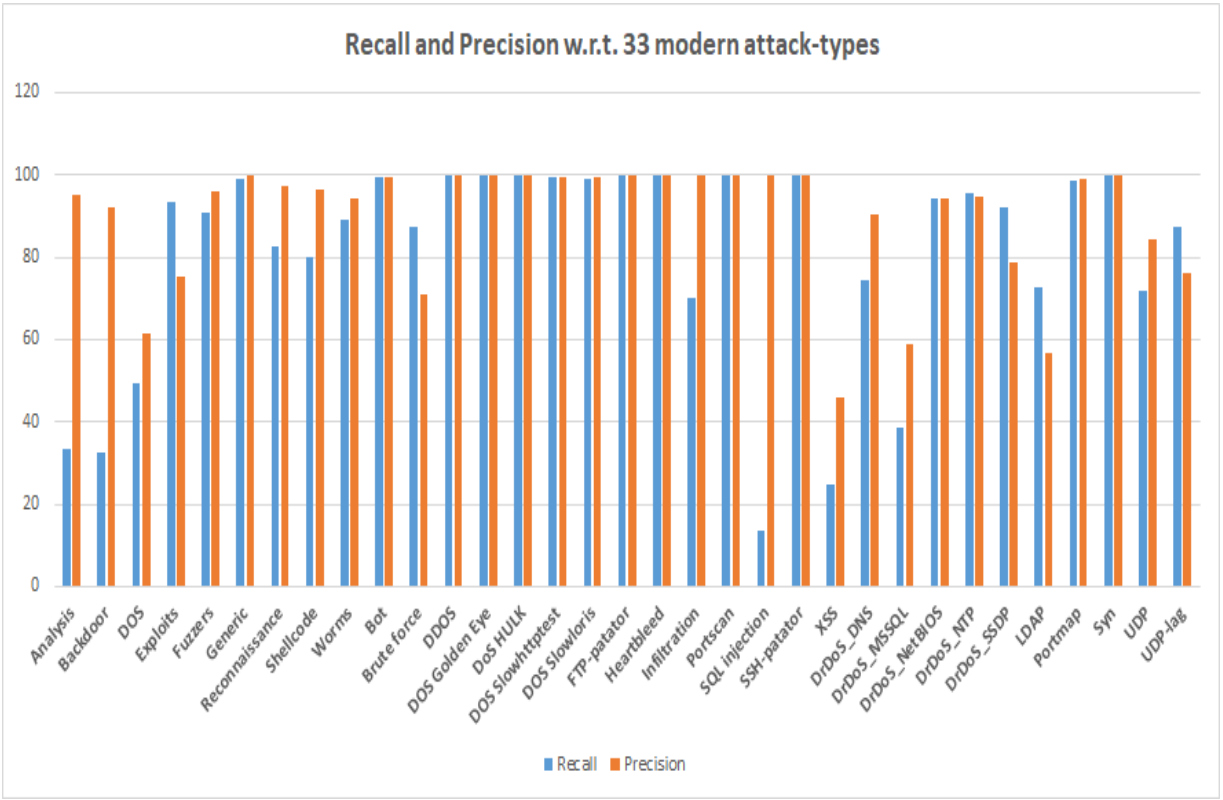

FIGURE 18. Performance of the proposed approach w.r.t. thirty-three modern attack types.

work that considers 1746 samples of backdoor attack type for testing the efficiency of the design.

A lower value of precision denotes higher degree of false positives and thus reducing the false positives is one of the primary objective of any intrusion detection system.

Similarly, decreasing false negatives is another critical concern and a high recall score reveals that the number of false negatives are lesser.

The proposed ensemble exhibited a superior performance while predicting the samples of CICIDS 2017 dataset and also saved computational time and resources considerably.

The findings reported in this work further indicate that the proposed model achieved a very good performance with respect to the detection of all DOS and DDOS attack types.

An exceptional recall score of 100% was reported by the proposed model pertaining to heartbleed attack type. Similarly, the best precision score of 100% was achieved while detecting the samples belonging to heartbleed, infiltration, SQL injections and SSH-patator too.

Although strong modeling capabilities were exhibited by the proposed model with respect to twelve attack types out of fourteen, the detection rate pertaining to web application attacks like SQL injections and cross-site scripting (XSS) were on the lower side that may require tweaking of hyper-parameters in future to achieve desirable results.

CICDDOS 2019 is relatively a newer dataset and we have not come across ensemble approaches that have been validated using this dataset. The highest recall and precision scores achieved so far are 100% and 99.8% respectively that corresponds to Syn attack.

The least recall score reported by the proposed multiclass model is 38.6% with respect to DrDoS-MSSQL attack type and the least precision score is 56.89% pertaining to LDAP attack type. However, it can be recounted that the proposed ensemble approach has achieved a feat since the size of the testing set considered in the study is larger than the training set.

- **Comparison with recent state-of-the-art approaches**

We have compared our proposed ensemble against some recent techniques that used UNSW NB-15 dataset as enumerated in **Table 26**. Our approach outperforms some state of the art methods concerning both accuracy and false positive rate.

The false positive rate generated by the proposed approach is definitely less that further emphasizes the adeptness of both base learners and meta-learner considered in the study. It is noteworthy that the Dendron [17] approach achieved a very low false alarm rate but its accuracy was lesser than the proposed model.

A recently proposed approach [77] that used decision tree for classification also achieved a very low false positive rate but the accuracy achieved by the model was relatively lesser than the proposed model. It is evident from **Table 27** that the proposed ensemble approach surpasses some recent techniques that used CICIDS 2017 dataset.

TABLE 26. Comparison with the baselines on UNSW NB-15 dataset.

METHOD	*YEAR*	*ACCURACY(%)*	*FALSE POSITIVE RATE(%)*
Decision Tree [52]	2016	85.56	15.78
Logistic regression [52]	2016	83.15	18.48
Naive Bayes [52]	2016	82.07	18.56
Neural Network [52]	2016	81.34	21.13
Expectation-Maximization [52]	2016	78.47	23.79
Decision Tree [43]	2017	81.42	6.39
Dendron [17]	2018	84.33	2.61
Majority voting [78]	2018	95.44	5.82
Gradient Boosted Machine [79]	2019	91.31	8.60
Ensemble approach [80]	2020	92.45	11.3
Stacking [81]	2020	94.3	4.4
Decision Tree [77]	2020	94.90	0.72
Proposed approach	**2021**	**99.8**	**0.38**

TABLE 27. Comparison with the baselines on CICIDS 2017 dataset.

METHOD	*YEAR*	*PRECISION(%)*	*RECALL(%)*	*F1-SCORE(%)*
K Nearest Neighbor [56]	2018	96	96	96
Random Forest [56]	2018	98	97	97
ID3 [56]	2018	98	98	98
Adaboost [56]	2018	77	84	77
Multilayer perceptron [56]	2018	77	83	76
Naive Bayes [56]	2018	88	04	04
Quadratic Discriminant Analysis [56]	2018	97	88	92
Deep belief network and ensemble SVM [82]	2018	90.40	95.65	92.95
Autoencoder conditional generative adversarial networks [83]	2019	98.46	93.29	95.38
Proposed approach	**2021**	**98.9**	**98.3**	**98.4**

TABLE 28. Comparison with the baselines on CICDDOS 2019 dataset.

TECHNIQUE	YEAR	PRECISION(%)	RECALL(%)	F1-SCORE(%)
ID3 [57]	2019	78	65	69
Random Forest [57]	2019	77	56	62
Naïve Bayes [57]	2019	41	11	05
Logistic Regression [57]	2019	25	02	04
Convolutional Neural Networks [84]	2020	93.3	92.4	92.8
Dense multi-layer perceptron [84]	2020	83.4	95.7	89.0
Logistic regression [84]	2020	86.8	77.1	79.4
Multi-layer perceptron [84]	2020	84.4	94.2	89.0
Proposed approach	**2021**	**99**	**97**	**97.8**

Sharafaldin *et al.* [56], founders of CICIDS 2017 applied various classifiers and also extracted some significant features using RandomForestRegressor to achieve the aforementioned results but they evaluated the performance of classifiers using three standard metrics but false positive rate was not specified.

It can be observed that tree based classifiers [56] have yielded the best possible classification outcome and our proposed model was also built along the same lines. The meta-classifier employed in the study for both binary and multiclass classification is also a tree-based classifier called decision jungle.

It is noteworthy that DDOS attack detection is also carried out in a robust manner by the propsed ensemble as compared to some existing approaches mentioned in **Table 28**.

DDOS attack detection on Software Defined Networking (SDN) is another research avenue for which CICDDOS 2019 dataset is being used lately [84].

In order to demonstrate the efficiency of the proposed ensemble more appropriately, we have presented the multiclass classification results using three confusion matrices pertaining to three datasets as shown in **Tables 29 – 31**. [Refer Appendix].

VIII. CONCLUSION

This article has emphasized on a meta-classification approach to propose a network intrusion detection model on a cloud environment. We have combined robust classifiers like bayes point machine, logistic regression, locally deep svm and decision jungle for binary classification and strong learners like decision forest, neural networks, logistic regression and decision jungle to perform multiclass classification.

We have highlighted the efficiency exhibited by the meta-learner called decision jungle that is common to both binary and multiclass classification.

In order to affirm the generalization perspective of our implementation, we have used three datasets in our study namely UNSW NB-15, CICIDS 2017 and CICDDOS 2019. Initially, the optimal feature subsets were identified pertaining to each dataset and subsequently models were built on Azure cloud to train the algorithms.

In order to corroborate the critical findings of the proposed work, we have also emphasized on statistical significance tests. Unlike traditional train-test strategies adopted in machine learning wherein a relatively larger training set is often employed, we have emphasized on a larger test set to determine the effectiveness of the meta-classifier.

The novelty of our approach lies in the fact that it is automated and is transformed as a web service. Whenever the web service is provisioned, it can serve as a mechanism to mitigate security incidents in complex large scale networks. Furthermore, in order to offer better customization capabilities to cloud users upon subscription, we configured three virtual machines, each dedicated to the detection of specific attack types found in the three datasets.

In future, we plan to consider additional datasets in our study and also incorporate Azure Data Lakes so that massive network data can be furnished to the algorithms for pattern matching and storage constraints can be also addressed to a large extent. An interesting research avenue would be to execute deep learning algorithms on MLaaS paradigm.

APPENDIX

TABLE 29. Confusion matrix w.r.t UNSW NB-15 dataset.

	A	B	D	E	F	G	N	R	S	W	Recall (%)
A	**666**	26	358	945	3	0	2	0	0	0	33.3
B	21	**566**	367	787	0	0	0	5	0	0	32.4
D	5	2	**6046**	6088	76	8	3	31	5	0	49.2
E	6	1	1921	**31267**	138	5	6	38	6	5	93.6
F	2	8	408	1054	**16549**	2	86	55	20	0	91.0
G	0	1	196	114	41	**39643**	0	2	1	2	99.1
N	1	0	1	17	299	0	**55667**	12	3	0	99.4
R	0	10	529	1287	6	2	0	**8657**	0	0	82.5
S	0	0	2	7	123	1	7	85	**908**	0	80.1
W	0	0	0	7	7	0	0	0	0	**116**	89.2
Precision (%)	95	92.18	61.51	75.2	95.9	99.9	99.8	97.43	96.28	94.3	

A=Analysis, B=Backdoor, D=Denial of Service, E=Exploits, F=Fuzzers, G=Generic, N=Normal, R=Reconnaissance, S=Shellcode, W=Worms

TABLE 30. Confusion matrix w.r.t CICIDS-2017 dataset.

	B	Bot	BF	DDOS	DG	DH	DS1	DS2	FP	H	I	P	S	SP	XSS	Recall (%)
B	**19395**	3	5	4	3	5	0	0	1	0	0	1	0	0	0	99.88
Bot	8	**1186**	0	0	0	0	0	0	0	0	0	0	0	0	0	99.32
BF	6	0	**780**	0	0	0	0	0	0	0	0	0	0	0	107	87.34
DDOS	6	0	0	**7105**	0	0	0	0	0	0	0	0	0	0	0	99.9
DG	1	0	0	0	**3545**	0	2	0	0	0	0	0	0	0	0	99.9
DH	3	0	0	3	0	**12463**	0	0	0	0	0	0	0	0	6	99.9
DS1	4	0	4	0	2	0	**3168**	9	0	0	0	0	0	0	0	99.4
DS2	2	0	2	0	0	0	13	**1692**	0	0	0	0	0	0	1	98.94
FP	0	0	0	0	0	0	0	0	**4704**	0	0	0	0	0	1	99.97
H	0	0	0	0	0	0	0	0	0	**6**	0	0	0	0	0	100
I	7	1	0	0	0	0	0	0	0	0	**19**	0	0	0	0	70.3
P	2	0	5	0	0	1	0	0	0	0	0	**9047**	0	0	0	99.91
S	3	0	8	0	2	0	0	0	0	0	0	0	**2**	0	0	13.33
SP	0	0	0	0	0	2	0	0	0	0	0	0	0	**3548**	0	99.94
XSS	0	0	297	0	1	0	0	0	0	0	0	0	0	0	**98**	24.74
Precision (%)	99.78	99.6	70.84	99.9	99.77	99.93	99.52	99.47	99.97	100	100	99.98	100	100	46	

B=Benign, BF= Brute force, DG= DOS Golden Eye, DH= DOS Http Unbearable Load King (Hulk), DS1=DOS Slowhttptest, DS2=DOS Slowloris, FP=Ftp-patator, H=Heartbleed, I=Infiltration, P=Portscan, S=SQL injection, SP=SSH-patator, XSS=Cross-site scripting

TABLE 31. Confusion matrix w.r.t CICDDOS-2019 dataset.

	B	DD	DM	DN1	DN2	DS	L	P	S	U	UL	Recall (%)
B	**8138**	4	0	0	13	0	0	4	4	0	8	99.5
DD	22	**8607**	266	131	289	1157	1049	32	0	9	1	74.4
DM	0	111	**1738**	19	1	1009	1622	0	0	0	0	38.6
DN1	0	79	22	**4525**	11	73	52	13	0	25	8	94.1
DN2	10	266	0	9	**6183**	0	0	9	0	0	1	95.4
DS	0	238	351	44	1	**11363**	312	0	0	0	1	92.3
L	0	117	564	16	0	803	**4006**	0	0	0	0	72.7
P	1	70	0	10	24	0	0	**7170**	0	4	0	98.5
S	0	0	0	0	0	0	0	0	**3195**	0	0	100
U	0	8	0	35	0	0	0	1	0	**5021**	1911	71.9
UL	0	1	0	8	1	0	0	0	0	898	**6235**	87.2
Precision (%)	99.5	90.5	59	94.32	94.78	78.88	56.89	99.18	99.8	84.28	76.36	

B=Benign, DD= Distributed Denial of Service domain name system, DM=DDOS MSSQL, DN1=DDOS Network Basic Input/Output System (DDOS Netbios), DN2= DDOS Network Time Protocol (DDOS NTP), DS= DDOS Simple Service Discovery Protocol (DDOS SSDP), L= Lightweight Directory Access Protocol (LDAP), P=Portscan, S=SYN, U=UDP, UL=UDP-lag.

REFERENCES

[1] S. Zeadally, E. Adi, Z. Baig, and I. A. Khan, "Harnessing artificial intelligence capabilities to improve cybersecurity," *IEEE Access*, vol. 8, pp. 23817–23837, 2020.

[2] M. Taddeo, "Three ethical challenges of applications of artificial intelligence in cybersecurity," *Minds Mach.*, vol. 29, no. 2, pp. 187–191, Jun. 2019.

[3] Z. Pan, W. Yu, X. Yi, A. Khan, F. Yuan, and Y. Zheng, "Recent progress on generative adversarial networks (GANs): A survey," *IEEE Access*, vol. 7, pp. 36322–36333, 2019.

[4] N. Ye, X. Li, Q. Chen, S. M. Emran, and M. Xu, "Probabilistic techniques for intrusion detection based on computer audit data," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 31, no. 4, pp. 266–274, Jul. 2001.

[5] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Comput. Netw.*, vol. 51, no. 12, pp. 3448–3470, Aug. 2007.

[6] G. Kou, Y. Peng, Z. Chen, and Y. Shi, "Multiple criteria mathematical programming for multi-classification and application in network intrusion detection," *Inf. Sci.*, vol. 179, no. 4, pp. 371–381, 2009.

[7] S. Rastegari, P. Hingston, and C.-P. Lam, "Evolving statistical rulesets for network intrusion detection," *Appl. Soft Comput.*, vol. 33, pp. 348–359, Aug. 2015.

[8] N. Keegan, S.-Y. Ji, A. Chaudhary, C. Concolato, B. Yu, and D. H. Jeong, "A survey of cloud-based network intrusion detection analysis," *Hum.-Centric Comput. Inf. Sci.*, vol. 6, no. 1, p. 19, Dec. 2016.

[9] V. Balamurugan and R. Saravanan, "Enhanced intrusion detection and prevention system on cloud environment using hybrid classification and OTS generation," *Cluster Comput.*, vol. 22, no. S6, pp. 13027–13039, Nov. 2019.

[10] C. N. Modi, D. R. Patel, A. Patel, and M. Rajarajan, "Integrating signature apriori based network intrusion detection system (NIDS) in cloud computing," *Procedia Technol.*, vol. 6, pp. 905–912, Jan. 2012.

[11] S. Gupta, P. Kumar, and A. Abraham, "A profile based network intrusion detection and prevention system for securing cloud environment," *Int. J. Distrib. Sensor Netw.*, vol. 9, no. 3, Mar. 2013, Art. no. 364575.

[12] B. Hajimirzaei and N. J. Navimipour, "Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm," *ICT Exp.*, vol. 5, no. 1, pp. 56–59, Mar. 2019.

[13] D.-Y. Yeung and Y. Ding, "Host-based intrusion detection using dynamic and static behavioral models," *Pattern Recognit.*, vol. 36, no. 1, pp. 229–243, Jan. 2003.

[14] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.

[15] M. Tavallaee, N. Stakhanova, and A. A. Ghorbani, "Toward credible evaluation of anomaly-based intrusion-detection methods," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 40, no. 5, pp. 516–524, Sep. 2010.

[16] S. Kumar, A. Viinikainen, and T. Hamalainen, "Machine learning classification model for network based intrusion detection system," in *Proc. 11th Int. Conf. Internet Technol. Secured Trans. (ICITST)*, Dec. 2016, pp. 242–249.

[17] D. Papamartzivanos, F. G. Mármol, and G. Kambourakis, "Dendron: Genetic trees driven rule induction for network intrusion detection systems," *Future Gener. Comput. Syst.*, vol. 79, pp. 558–574, Feb. 2018.

[18] R. Singh, H. Kumar, and R. K. Singla, "An intrusion detection system using network traffic profiling and online sequential extreme learning machine," *Expert Syst. Appl.*, vol. 42, no. 22, pp. 8609–8624, Dec. 2015.

[19] J. Kim, N. Shin, S. Y. Jo, and S. Hyun Kim, "Method of intrusion detection using deep neural network," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Feb. 2017, pp. 313–316.

[20] B. Lu, J. Yang, L. Y. Chen, and S. Ren, "Automating deep neural network model selection for edge inference," in *Proc. IEEE 1st Int. Conf. Cognit. Mach. Intell. (CogMI)*, Dec. 2019, pp. 184–193.

[21] G. Folino and P. Sabatino, "Ensemble based collaborative and distributed intrusion detection systems: A survey," *J. Netw. Comput. Appl.*, vol. 66, pp. 1–16, May 2016.

[22] A. J. Malik and F. A. Khan, "A hybrid technique using binary particle swarm optimization and decision tree pruning for network intrusion detection," *Cluster Comput.*, vol. 21, no. 1, pp. 667–680, Mar. 2018.

[23] B. Pavlyshenko, "Machine-learning models for sales time series forecasting," *Data*, vol. 4, no. 1, p. 15, Jan. 2019.

[24] B. Zhai and J. Chen, "Development of a stacked ensemble model for forecasting and analyzing daily average PM2. 5 concentrations in Beijing, China," *Sci. Total Environ.*, vol. 635, pp. 644–658, Sep. 2018.

[25] B. A. Tama, M. Comuzzi, and K.-H. Rhee, "TSE-IDS: A two-stage classifier ensemble for intelligent anomaly-based intrusion detection system," *IEEE Access*, vol. 7, pp. 94497–94507, 2019.

[26] R. M. Elbasiony, E. A. Sallam, T. E. Eltobely, and M. M. Fahmy, "A hybrid network intrusion detection framework based on random forests and weighted K-means," *Ain Shams Eng. J.*, vol. 4, no. 4, pp. 753–762, Dec. 2013.

[27] A. H. Hamamoto, L. F. Carvalho, L. D. H. Sampaio, T. Abrão, and M. L. Proença, "Network anomaly detection system using genetic algorithm and fuzzy logic," *Expert Syst. Appl.*, vol. 92, pp. 390–402, Feb. 2018.

[28] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 303–336, 1st Quart., 2014.

[29] G. Kim, S. Lee, and S. Kim, "A novel hybrid intrusion detection method integrating anomaly detection with misuse detection," *Expert Syst. Appl.*, vol. 41, no. 4, pp. 1690–1700, Mar. 2014.

[30] S. Rayana and L. Akoglu, "Less is more: Building selective anomaly ensembles," *ACM Trans. Knowl. Discovery Data*, vol. 10, no. 4, pp. 1–33, 2016.

[31] S. Rajagopal, P. P. Kundapur, and K. S. Hareesha, "A stacking ensemble for network intrusion detection using heterogeneous datasets," *Secur. Commun. Netw.*, vol. 2020, pp. 1–9, Jan. 2020.

[32] W. Chen, F. Kong, F. Mei, G. Yuan, and B. Li, "A novel unsupervised anomaly detection approach for intrusion detection system," in *Proc. IEEE IEEE 3rd Int. Conf. Big Data Secur. Cloud (BigDataSecurity), Int. Conf. High Perform. Smart Comput. (HPSC), IEEE Int. Conf. Intell. Data Secur. (IDS)*, May 2017, pp. 69–73.

[33] S. Kaur and I. Garg, "Ensemble technique based on supervised and unsupervised learning approach for intrusion detection," in *Proc. Int. Conf. Adv. Comput. Data Sci.* Singapore: Springer, 2018, pp. 228–238.

[34] S. Ruoti, S. Heidbrink, M. O'Neill, E. Gustafson, and Y. R. Choe, "Intrusion detection with unsupervised heterogeneous ensembles using cluster-based normalization," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jun. 2017, pp. 862–865.

[35] N. Chand, P. Mishra, C. R. Krishna, E. S. Pilli, and M. C. Govil, "A comparative analysis of SVM and its stacking with other classification algorithm for intrusion detection," in *Proc. Int. Conf. Adv. Comput., Commun., Autom. (ICACCA) (Spring)*, Apr. 2016, pp. 1–6.

[36] A. A. Aburomman and M. B. I. Reaz, "A novel SVM-kNN-PSO ensemble method for intrusion detection system," *Appl. Soft Comput.*, vol. 38, pp. 360–372, Jan. 2016.

[37] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Comput. Secur.*, vol. 28, nos. 1–2, pp. 18–28, Feb. 2009.

[38] S. A. Aljawarneh and R. Vangipuram, "GARUDA: Gaussian dissimilarity measure for feature representation and anomaly detection in Internet of Things," *J. Supercomput.*, vol. 76, no. 6, pp. 4376–4413, Jun. 2020.

[39] Y. Y. Chung and N. Wahid, "A hybrid network intrusion detection system using simplified swarm optimization (SSO)," *Appl. Soft Comput.*, vol. 12, no. 9, pp. 3014–3022, Sep. 2012.

[40] A. S. Eesa, Z. Orman, and A. M. A. Brifcani, "A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems," *Expert Syst. Appl.*, vol. 42, no. 5, pp. 2670–2679, Apr. 2015.

[41] S. S. S. Sindhu, S. Geetha, and A. Kannan, "Decision tree based light weight intrusion detection using a wrapper approach," *Expert Syst. Appl.*, vol. 39, no. 1, pp. 129–141, Jan. 2012.

[42] M. S. Mok, S. Y. Sohn, and Y. H. Ju, "Random effects logistic regression model for anomaly detection," *Expert Syst. Appl.*, vol. 37, no. 10, pp. 7162–7166, Oct. 2010.

[43] C. Khammassi and S. Krichen, "A GA-LR wrapper approach for feature selection in network intrusion detection," *Comput. Secur.*, vol. 70, pp. 255–277, Sep. 2017.

[44] W. Zong, Y.-W. Chow, and W. Susilo, "A two-stage classifier approach for network intrusion detection," in *Proc. Int. Conf. Inf. Secur. Pract. Exper.* Cham, Switzerland: Springer, 2018, pp. 329–340.

[45] D. Chaboya, R. Raines, R. Baldwin, and B. Mullins, "Network intrusion detection: Automated and manual methods prone to attack and evasion," *IEEE Secur. Privacy Mag.*, vol. 4, no. 6, pp. 36–43, Nov. 2006.

[46] C. Augenstein, N. Spangenberg, and B. Franczyk, "Applying machine learning to big data streams: An overview of challenges," in *Proc. IEEE 4th Int. Conf. Soft Comput. Mach. Intell. (ISCMI)*, Nov. 2017, pp. 25–29.

[47] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated Machine Learning: Methods, Systems, Challenges*. Cham, Switzerland: Springer, 2019.

[48] A. P. Tafti, E. LaRose, J. C. Badger, R. Kleiman, and P. Peissig, "Machine learning-as-a-service and its application to medical informatics," in *Proc. Int. Conf. Mach. Learn. Data Mining Pattern Recognit.* Cham, Switzerland: Springer, 2017, pp. 206–219.
[49] A. Evangelidis, D. Parker, and R. Bahsoon, "Performance modelling and verification of cloud-based auto-scaling policies," *Future Gener. Comput. Syst.*, vol. 87, pp. 629–638, Oct. 2018.
[50] D. Loreti, M. Lippi, and P. Torroni, "Parallelizing machine learning as a service for the end-user," *Future Gener. Comput. Syst.*, vol. 105, pp. 275–286, Apr. 2020.
[51] N. Moustafa and J. Slay, "The significant features of the UNSW-NB15 and the KDD99 data sets for network intrusion detection systems," in *Proc. 4th Int. Workshop Building Anal. Datasets Gathering Exper. Returns Secur. (BADGERS)*, Nov. 2015, pp. 25–31.
[52] N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Inf. Secur. J. A, Global Perspective*, vol. 25, nos. 1–3, pp. 18–31, Apr. 2016.
[53] N. Fallahi, A. Sami, and M. Tajbakhsh, "Automated flow-based rule generation for network intrusion detection systems," in *Proc. 24th Iranian Conf. Electr. Eng. (ICEE)*, May 2016, pp. 1948–1953.
[54] Y. Yu, J. Long, and Z. Cai, "Network intrusion detection through stacking dilated convolutional autoencoders," *Secur. Commun. Netw.*, vol. 2017, pp. 1–10, Nov. 2017.
[55] H. Wang, Z. Cao, and B. Hong, "A network intrusion detection system based on convolutional neural network," *J. Intell. Fuzzy Syst.*, vol. 38, no. 6, pp. 7623–7637, Jun. 2020.
[56] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, 2018, pp. 108–116.
[57] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in *Proc. Int. Carnahan Conf. Secur. Technol. (ICCST)*, Oct. 2019, pp. 1–8.
[58] G. Kumar and K. Kumar, "The use of artificial-intelligence-based ensembles for intrusion detection: A review," *Appl. Comput. Intell. Soft Comput.*, vol. 2012, pp. 1–20, Jul. 2012.
[59] A. A. Aburomman and M. B. I. Reaz, "A survey of intrusion detection systems based on ensemble and hybrid classifiers," *Comput. Secur.*, vol. 65, pp. 135–152, Mar. 2017.
[60] J. Shotton, T. Sharp, P. Kohli, S. Nowozin, J. Winn, and A. Criminisi, "Decision jungles: Compact and rich models for classification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 234–242.
[61] R. Barga, V. Fontama, W. H. Tok, and L. Cabrera-Cordon, *Predictive Analytics With Microsoft Azure Machine Learning*. Berkely, CA, USA: Apress, 2015.
[62] J. Barnes, "Azure machine learning," in *Microsoft Azure Essentials*, 1st ed. Redmond, WA, USA: Microsoft, 2015.
[63] S. Mund, *Microsoft Azure Machine Learning*. Birmingham, U.K.: Packt, 2015.
[64] A. V. Joshi, "Azure machine learning," in *Machine Learning and Artificial Intelligence*. Cham, Switzerland:'Springer, 2020, pp. 207–220.
[65] Y. Liu, K. Akram Hassan, M. Karlsson, Z. Pang, and S. Gong, "A data-centric Internet of Things framework based on azure cloud," *IEEE Access*, vol. 7, pp. 53839–53858, 2019.
[66] E.-G. Learned-Miller, "Entropy and mutual information," Univ. Massachusetts Amherst, Amherst, MA, USA, Tech. Rep., 2013, vol. 4, pp. 1–4.
[67] P. Schober, C. Boer, and L. A. Schwarte, "Correlation coefficients: Appropriate use and interpretation," *Anesthesia Analgesia*, vol. 126, no. 5, pp. 1763–1768, 2018.
[68] R. Tamura, K. Kobayashi, Y. Takano, R. Miyashiro, K. Nakata, and T. Matsui, "Mixed integer quadratic optimization formulations for eliminating multicollinearity based on variance inflation factor," *J. Global Optim.*, vol. 73, no. 2, pp. 431–446, Feb. 2019.
[69] R. Herbrich, T. Graepel, and C. Campbell, "Bayes point machines," *J. Mach. Learn. Res.*, vol. 1, pp. 245–279, Aug. 2001.
[70] R. Bollapragada, D. Mudigere, J. Nocedal, H.-J. M. Shi, and P. T. P. Tang, "A progressive batching L-BFGS method for machine learning," 2018, *arXiv:1802.05374*. [Online]. Available: http://arxiv.org/abs/1802.05374
[71] C. Jose, P. Goyal, P. Aggrwal, and M. Varma, "Local deep kernel learning for efficient non-linear svm prediction," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 486–494.
[72] M.-I. Georgescu, R. T. Ionescu, and M. Popescu, "Local learning with deep and handcrafted features for facial expression recognition," *IEEE Access*, vol. 7, pp. 64827–64836, 2019.
[73] S. L. Kukreja, J. Löfberg, and M. J. Brenner, "A least absolute shrinkage and selection operator (LASSO) for nonlinear system identification," *IFAC Proc. Volumes*, vol. 39, no. 1, pp. 814–819, 2006.
[74] S. Garcia and F. Herrera, "An extension on' statistical comparisons of classifiers over multiple data sets' for all pairwise comparisons," *J. Mach. Learn. Res.*, vol. 9, pp. 2677–2694, Dec. 2008.
[75] P. Phoungphol, Y. Zhang, and Y. Zhao, "Robust multiclass classification for learning from imbalanced biomedical data," *Tsinghua Sci. Technol.*, vol. 17, no. 6, pp. 619–628, Dec. 2012.
[76] G. Forman and M. Scholz, "Apples-to-apples in cross-validation studies: Pitfalls in classifier performance measurement," *ACM SIGKDD Explorations Newslett.*, vol.12, no. 1, pp. 49–57 , 2010.
[77] C. Khammassi and S. Krichen, "A NSGA2-LR wrapper approach for feature selection in network intrusion detection," *Comput. Netw.*, vol. 172, May 2020, Art. no. 107183.
[78] D. Kadam, R. Patil, and C. Modi, "An enhanced approach for intrusion detection in virtual network of cloud computing," in *Proc. 10th Int. Conf. Adv. Comput. (ICoAC)*, Dec. 2018, pp. 80–87.
[79] B. A. Tama and K.-H. Rhee, "An in-depth experimental study of anomaly detection using gradient boosted machine," *Neural Comput. Appl.*, vol. 31, no. 4, pp. 955–965, Apr. 2019.
[80] B. A. Tama, L. Nkenyereye, S. M. R. Islam, and K.-S. Kwak, "An enhanced anomaly detection in Web traffic using a stack of classifier ensemble," *IEEE Access*, vol. 8, pp. 24120–24134, 2020.
[81] S. Rajagopal, P. P. Kundapur, and H. K. Siddaramappa, "A predictive model for network intrusion detection using stacking approach," *Int. J. Electr. Comput. Eng.*, vol. 10, no. 3, p. 2734, Jun. 2020.
[82] N. Marir, H. Wang, G. Feng, B. Li, and M. Jia, "Distributed abnormal behavior detection approach based on deep belief network and ensemble SVM using spark," *IEEE Access*, vol. 6, pp. 59657–59671, 2018.
[83] J. Lee and K. Park, "AE-CGAN model based high performance network intrusion detection system," *Appl. Sci.*, vol. 9, no. 20, p. 4221, Oct. 2019.
[84] M. V. O. de Assis, L. F. Carvalho, J. J. P. C. Rodrigues, J. Lloret, and M. L. Proença, Jr., "Near real-time security system applied to SDN environments in IoT networks using convolutional neural network," *Comput. Electr. Eng.*, vol. 86, Sep. 2020, Art. no. 106738.

SMITHA RAJAGOPAL received the M.C.A. degree from the BMS College of Engineering, Bengaluru. She is currently pursuing research with the Department of Computer Applications, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal. Her research interests include cloud computing, machine learning, software engineering, big data, design patterns, blockchain technology, and cyber security.

POORNIMA PANDURANGA KUNDAPUR is currently an Associate Professor with the Department of Computer Applications, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal. Her research interests are knowledge management, system dynamics, cloud computing, and machine learning. She is a Fellow under the MAHE-FAIMER IPE/PP program.

HAREESHA K. S. (Senior Member, IEEE) is currently a Professor with the Department of Computer Applications, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal. He is into academic administration. His research areas of interests include machine learning, bio-informatics, soft computing, and digital image processing. He is also a member of the Indian Society for Technical Education and International Association of Computer Science and Information Technology. He is credited toward holding funded projects from DST, New Delhi, and a startup funded by DST-BIRAC-BIG10.