International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 14, No.1 (Sep-2023)

http://dx.doi.org/10.12785/ijcds/140184

REST-API based DDoS Detection Using Multi Feature Hybrid Classification in the Cloud Architecture

Beenish Habib 1[1] and Farida Khursheed 1[2]

[1]*Department of Electronics and Communication, National Institute of Technology Srinagar, India*
Received 5 Jun. 2022, Revised 27 Aug. 2023, Accepted 19 Sep. 2023, Published 22 Sep. 2023

Abstract: Cloud services are often delivered through HTTP protocol for ease and reduced cost for both service providers and users. The only drawback is that these protocols and the cloud itself are more prone to Distributed Denial of Service (DDoS) attacks. There is the need for a detection setup that is lightweight, robust and easily deployable on these architectures with an improved efficiency. We thus propose a novel multi-feature hybrid classification based DDOS detection setup that uses the Representational State Transfer (REST)-Application Programming Interface (API) for the attack prediction. The cloud architectures we are using are Heroku, one of the first developed platform as a service (paas) computing platform by salesforce and Amazon Elastic Compute Cloud, one of the most sought after and on-demand infrastructure as a service (iaas) computing platform by amazon. The GitHub repository holds the pre-trained hybrid classifier in its repository which is accessed using the REST-API by the cloud. The HTTP request and response commands are sent to the cloud architecture through the Postman API client where the final attack prediction is done. This makes our cloud burden free as the detection is done outside its domain. The hybrid classifier here is the integration of Random Forest classifier, Decision Tree classifier, Support Vector Machine and XGBoost classifier, all trained on the pre-processed Knowledge Discovery in Databases (KDD) Cup 99 and UNSW-NB15 datasets. Here we are also using two different feature ranking methods i.e., statistical feature ranking and machine learning ranking. It shows the best accuracy results with information gain on KDD Cup 99 dataset and decision tree classifier on UNSW-NB15 dataset as the feature selection technique.

Keywords: Network security, Denial of Service Attacks, Cloud Computing, SaaS, Rest-API, Machine learning

1. INTRODUCTION

As the use of internet cloud services has grown, there has been an increased need to secure these services from intruders and other threats. The Distributed Denial of Service (DDoS) attack is one of the main security concerns that has grown significantly over the past recent years [1]. The attack is achieved by sending a large volume of network traffic by exploited systems called bots or zombies [2]. These attacks exhaust the resources and makes the cloud services unavailable to users [3]. This even cause a huge financial loss with a loss of confidential and critical resources among the other concerns. By June 1, 2022 Google Cloud Armor was targeted with a series of HTTP DDoS attack peaking at 46 million request per second [4]. This was the largest layer 7 DDoS attack reported to date. The attack was at least 76 percent larger than the previously recorded. As per report of 2022, the HTTP DDoS attacks has increased by 111 percent than previous year with reports of ransom DDoS attack increased by 67 percent [5].

Cloud computing is the most popular platform that renders many essential services to the IT sector at a very low cost. A lot of computing and storage resources are transferred to users by large data centers and servers of a cloud [6]. To make cloud architecture robust and scalable the resources are exchanged via the internet. The internet in itself is a very vulnerable place for different attacks and threats especially the DDoS attack. For example, in 2018, Amazon Web Services (AWS) suffered a massive DDoS attack that affected the availability of its services in the US-East region [7]. In the same year, Rackspace, a leading cloud hosting provider, was also targeted by a DDoS attack causing a lot of financial loss. DDoS attack is one of the most serious issue a cloud is facing today. The DDoS attacks is a multi-vector attack and everyday a new attack comes under the headline. The Shrew attack is a type of man-in-the-middle attack that seeks to disrupt the Transmission Control Protocol (TCP) by injecting false packets into the communication stream. The RoQ (Reduction of Quality) attack is a type of DDoS attack that seeks to reduce the quality of service provided by a targeted website or service by overwhelming it with traffic. The LoRDAS (Low-Rate DDoS attack against application servers) is a type of DDoS attack that targets application servers with a low rate of traffic in an attempt to disrupt their function. EDoS (Economic Denial of Sustainability) is a type of DDoS attack that exploits the elasticity and auto-scaling capabilities of cloud computing to try to overwhelm a targeted website or

E-mail address: beenish_05phd17@nitsri.net, fklone@nitsri.net

service [8]. According to the recent survey a traditional web application faces about 800 concurrent user sessions under a DDoS attack in just a 15-minute timeframe with cloud hosted web application crashing at approx. 3000 concurrent user sessions under the 20-minute time frame [9].

Most of IDS (Intrusion Detection Systems) dealing with cloud security are knowledge based (signature based), anomaly-based or the hybrid-based detection systems. There is client puzzle, IP-marking, IP-Traceback, Filters (Sensor Filter, Hop-Count Filter, IP Frequency Divergence Filter, Puzzle resolver Filter, Double Signature Filter) and many other techniques for analysing these DDoS attacks . Machine learning (ML) and deep learning (DL) techniques have been widely used in recent years for detecting and mitigating DDoS attacks [10], [11]. The other being statistical methods, shallow machine learning, quantum computing and artificial intelligence [12], [13], [14], [15]. Most of the statistical approaches like entropy variations and correlation take a lot of time in execution throughout the DDoS attack detection process. They thus find less usage in real time set-ups due to their slow computations [16]. Shallow machine learning algorithms on the other hand take less time during training, but they take a lot of time in testing [17]. The quantum computing is promising in many areas like artificial intelligence, cyber-security and medical research and have proven to be efficient in solving decision problems, quantum search and game theory. The only limitation of it being less efficient in intrusion detections. Machine learning and deep learning approaches have proven to be far efficient in detection of DDoS attack with higher efficiency.

With the progress in science and technology, there is also increase in the complexity of contemporary decision problems and their solutions. One of the promising areas of research nowadays focuses on merging the classifiers as multiple classifiers, ensemble or as a hybrid system [18]. In multiple classifiers we have more than one classifier working collectively. If the architecture of the classifiers is same it is the ensemble system and if different architectures work collectively, we term it as hybrid classification. By combining the predictions of multiple classifiers, a hybrid classifier is able to achieve better performance than a single classifier, particularly in situations where the data is complex or noisy. Hybrid classifiers are also effective at detecting and mitigating DDoS attacks, as they can analyze large amounts of data in real-time and identify patterns and characteristics that may indicate an ongoing attack.

The goal of the proposed work is to design a DDoS attack detection system that is both effective and compatible with cloud architecture. Striking a balance between simplicity and effectiveness can be a challenge when designing a DDoS detection system, as it is important to have a system that is able to accurately distinguish between normal and attack traffic, but at the same time, it should not be too complex or resource-intensive to be practical for use in a cloud environment. In our proposed work the cloud only hosts the hybrid classifier while the detection is done by a third-party API client.

2. Related Work

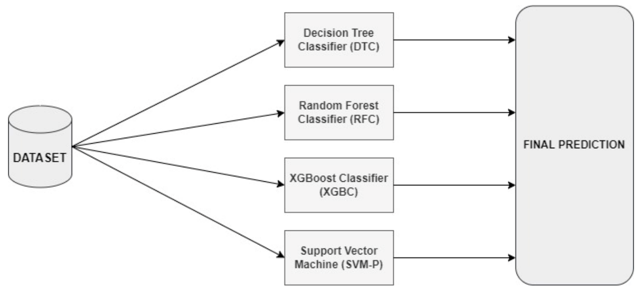
Tarun Karnwal in 2012 conference paper [19] “A Comber Approach to Protect Cloud Computing against XML DDoS and HTTP DDoS attack” proposes a virtual cloud defender that is designed to protect cloud computing environments against DDoS attacks that use HTTP, XML, and REST protocols. The service, known as the filtering tree, operates as a service broker within a Service-Oriented Architecture (SOA) model. G Leaden [20] proposes design and implementation of a honeypot disguised as a REST-API in the conference paper ”An API honeypot for DDoS and XSS analysis, 2017”. It presents analyses of both a distributed denial of service (DDoS) attack and a cross-site scripting (XSS) malware insertion attempt against this honeypot.

In 2019, IEEE World Congress on Services Obaid Rehman et al [21] gave the insight of how Software Defined Network (SDN) interfaces that involve application layer, control and data forwarding layers are connected via the API. The applications use several Application Programming Interfaces such as Java API to communicate locally with the controller or representational state transfer (REST) API for remote communication with the controller.

Raja Majid Ali Ujjan et al [22] in 2020 research article “Towards sFlow and adaptive polling sampling for deep learning-based DDoS detection in SDN” showed how SDN Ryu controller centrally controls VM2 to install sampling policies and other network manipulations via the REST-API configuration. In 2022 Madhura Shekhar Potnis et al [23], proposed “Hybrid Intrusion Detection System for Detecting DDoS Attacks on Web Applications Using Machine Learning”. They used the API tier of REST-APIs (one for each REST service) and Flask APIs to access the trained models for both anomaly and signature-based DDoS detection.

Most of the research work till now, focuses on the usage of REST-API with SDN architecture and layers that co-ordinate through it. None of the research work is proposed on the application of REST-API for the detection of DDoS attack. We tried to co-ordinate the REST-API with the hybrid machine learning model for the cloud defense. This sets as the application of a trained machine learning model in a real-world system. Machine learning algorithms are now finding increased usage in data mining applications mainly to retrieve the important information for the decision making [24]. These methods may be classification, association, clustering, regression analysis based or rule, decision tree and neural network based among others.

Each machine learning model work independently and target different domains. Combining them gives better performance than the rest of the models. Hybrid models also reduce individual limitations of basic models and exploit their different generalization mechanisms [25]. An


DATASET
Decision Tree Classifier (DTC)
Random Forest Classifier (RFC)
XGBoost Classifier (XGBC)
Support Vector Machine (SVM-P)
FINAL PREDICTION
Figure 1. Proposed Hybrid-Machine Learning Classifier.

overwhelming majority of classifier ensembles are currently constructed based on the homogeneous base classifier model [26]. The ensemble learning model for homogeneous individual classifiers though cannot satisfy the needs of the higher ensemble performance while the heterogeneous multi-classifier ensemble learning owns more remarkable and more predominant generalization performance than the homogeneous multi-classifier ensemble learning [26]. The classifiers in an ensemble should be different from each other to increase the gain [27]. These differences cover diversity, orthogonality, and complementarity [28].

3. Proposed Architecture

Figure 2 gives the idea of the strategic framework of our proposed DDoS detector via the REST-API using a hybrid classifier. The hybrid classifier we have used is the amalgamation of four best machine learning classifiers i.e., Decision Tree classifier, Random Forest classifier, XGBoost classifier and Non-Linear SVM. The integration of these classifiers has the best attribute selection, best detection capabilities and best representation of the attack detection. The domain of the hybrid classifier involves both the classification as well as regression. The strength of weak classifiers is enhanced by the ensemble classifiers and the two well-known ensemble classifiers are Random Forest model (bagging model) and XGBoost model (extreme gradient boosting model).

There are many different ways to combine the predictions of multiple classifiers in a hybrid model. Some common methods include:

A. *Majority voting*

In this method, the final prediction is based on the majority vote of the base classifiers.

B. *Weighted voting*

In this method, each base classifier is assigned a weight, and the final prediction is based on the weighted sum of the votes of the base classifiers.

C. *Stacking*

In this method, the predictions of the base classifiers are used as features to train a second-level classifier, which makes the final prediction. In our proposed architecture figure 1, we stack two weak learners i.e., decision tree classifier and non-linear SVM with two strong ensemble learners i.e., Random Forest and eXtreme Gradient boosting (XGBoost) classifier. We use the term hybrid as this is a heterogeneous collection of both weak and strong learners. All the four machine learning models are trained individually and then collectively for improved performance. We take two different datasets; one is KDD Cup 99 and other as UNSW-NB15. Both the datasets are well known datasets used for intrusion detection. While KDD Cup 99 dataset is an old dataset, UNSW-NB15 is the most recent one. Both the datasets are pre-processed and balanced before being used for training the machine learning models. All the machine learning models are trained on two different feature selection techniques. The feature selection is done based on statistical ranking and machine learning ranking i.e., information gain and decision tree ranking respectively. The top four classifiers with maximum efficiency are then selected for designing our hybrid classifier.

The DDoS detection approach is robust and lightweight as it uses a pre-trained hybrid machine learning model. This makes it easily deployable on multiple platforms may that be a cloud, a traditional client-server or any networking device. In our proposed work we checked the performance of our DDoS detector using two different cloud platforms. We use a platform as a service (paas) cloud platform, Heroku developed by salesforce and infrastructure as a service (iaas) cloud platform, Elastic Compute Cloud (EC2) by Amazon Web Service (AWS). They emulate most of the attributes of high-end processor requirements that are functional through APIs accessed through HTTP.

Both the cloud platforms host the pre-trained hybrid classifier deployed on a GitHub repository and accesses it via the REST-API. The HTTP request and responses are sent via the third-party client, Postman, which predicts the attack based on the hybrid machine learning model algorithm. The proposed detection system consists of four main steps: data pre-processing, feature selection, classification based on hybrid trained machine learning model and predicting the DDoS attack on a cloud via the REST-API. The classifier is selected based on confusion matrix scores, i.e., accuracy, false alarm rate (FAR), sensitivity, specificity, false-positive rate (FPR), F1 score, and area under curve (AUC) analysis. The main contributions of this paper can be summarized as follows:

1. A lightweight DDoS attack detection system is proposed for DDoS defence in two different cloud architectures, Heroku (paas) and Amazon EC2 (iaas) stored in the GitHub repository. The detector is accessed by these cloud architectures through the REST-APIs.

2. We use ensemble feature selection technique i.e., both statistical feature ranking as well as machine learning ranking to get the best features for model training.

3. A highly efficient hybrid machine learning classifier is adopted comprising four highly efficient machine learning

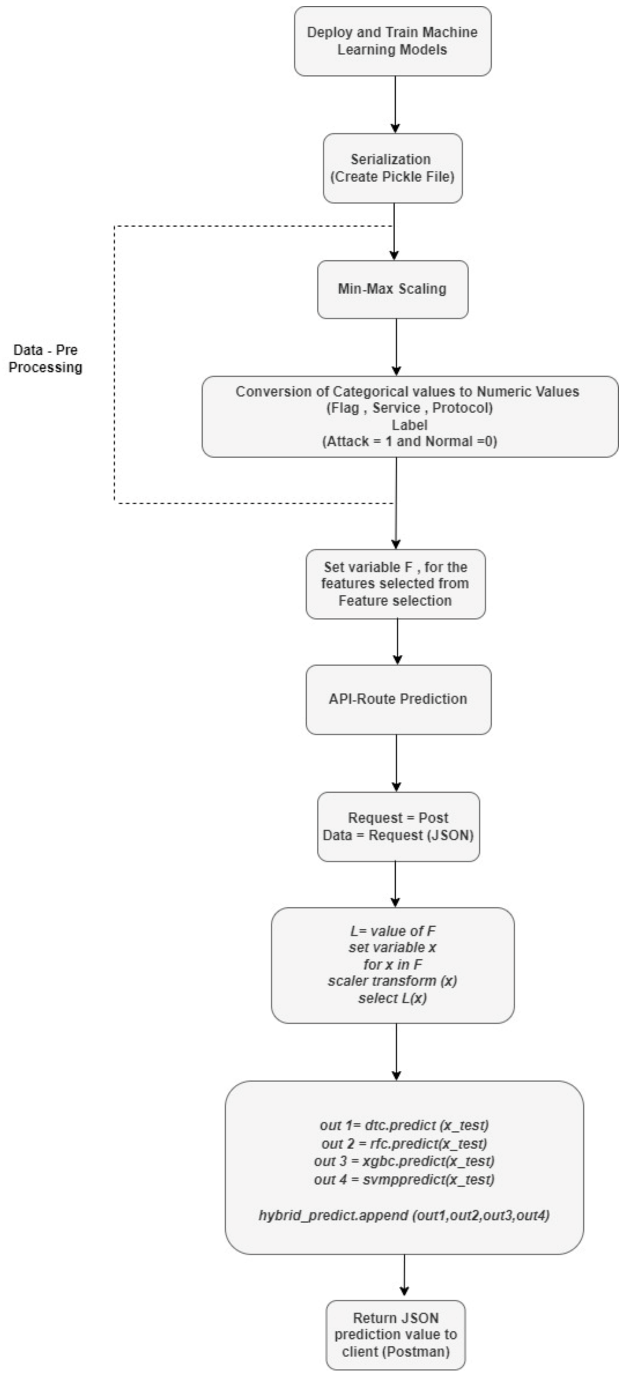

Deploy and Train Machine Learning Models
Serialization (Create Pickle File)
Data - Pre Processing
Min-Max Scaling
Conversion of Categorical values to Numeric Values (Flag , Service , Protocol) Label (Attack = 1 and Normal =0)
Set variable F , for the features selected from Feature selection
API-Route Prediction
Request = Post Data = Request (JSON)
L= value of F set variable x for x in F scaler transform (x) select L(x)
out 1= dtc.predict (x_test) out 2 = rfc.predict(x_test) out 3 = xgbc.predict(x_test) out 4 = svmppredict(x_test) hybrid_predict.append (out1,out2,out3,out4)
Return JSON prediction value to client (Postman)

Figure 2. Proposed Algorithm-Framework of REST-API based DDoS Detector Using Hybrid Machine Learning Classifier.

models i.e., Decision Tree, Random Forest, XGBoost and Support Vector Machine for the DDoS attack detection. The hybrid machine learning model is pre-trained on the pre-processed dataset and holds the maximum detection efficiency.

4. We have an easy access and prediction of attack through a simple request and response command sent via the client-server architecture.

4. Model pipeline

The model pipeline defines the course of data processing done before it is being used and analyzed for training the machine learning model. Data we use for model training is first pre-processed to be used for exploratory data analysis (EDA). The analysis finds the appropriate relationship between the required variables. This step is crucial for selecting the correct machine learning models for our hybrid classifier and further evaluating the prediction done by it. The top 10 ranking features of both datasets used for training the machine learning model are selected using Information Gain (statistical) and Decision Tree classifier (machine learning) based scores. We then divide the dataset and label set into training and testing set with a 70:30 margin. The variables, weights and bias are initialized before proceeding to training the machine learning models. The maximum performing parameters are determined using the GridSearchCV technique. Figure 2 illustrates the algorithm framework for API based DDoS detection using the trained hybrid machine learning model.

A. Data Pre-Processing

Both datasets are imbalance datasets and requires pre-processing [29]. For KDD Cup 99, about 108,927 packets are attack packets of 1,048,575 which corresponds 10 percent as attack class and 30 percent as normal class. For UNSW-NB15, Normal attacks are represented using 2,218,761 records while Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms signatures include 24246, 2677, 2329, 16535, 44525, 215481, 13987, 1511, and 174 records, respectively [30]. Consequently, there is considerable lack of balance for the dataset as 87% of the dataset comprises normal records whereas only 0.007% of the dataset consists of worms' records [31]. This skewness causes biased results. Though this is actual representation of real-world traffic but causes the machine learning training to be biased figure 3. We use oversampling and undersampling technique to balance both the datasets. We also go through the following pre-processing stages before being trained by machine learning models which involve:

B. Data cleaning and transformation

Handling missing undefined data is essential in data pre-processing part as they could lead to faulty results. The null values are eliminated and undefined data is equaled to zero [32]. Also, some features are transformed to numeric, float or integer values depending on the format that works well with a particular machine learning model.

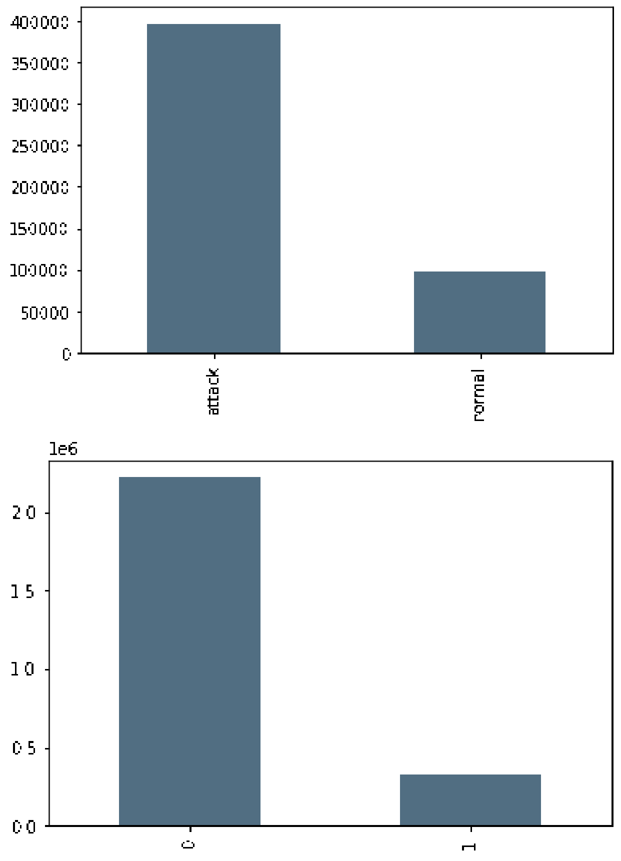


Figure 3. Class Imbalance in KDD Cup 99 and UNSW-NB15 .

C. Data Normalization

To scale the features in the same range, we use normalization. The continuous values on the dataset have large difference in their values and can lead to prediction errors. We use min-max scaling as our normalization technique [33]. This method rescales the range of feature value to 0 and 1. The standardization method (z-score) scales the features to have a mean value 0 and standard deviation 1. This method makes the approach less susceptible to outliers.

$$\chi_{sc} = \frac{\chi - \chi_{min}}{\chi_{max} - \chi_{min}}. \tag{1}$$

The standardization is given by,

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{N-1}}. \tag{2}$$

Where σ denotes the standard deviation, x_i denotes the feature value, N denotes the number of training samples and μ represents the mean which is given by,

$$\mu = \frac{\sum_{i=1}^{n} x_i}{N}. \tag{3}$$

D. Class Balance

We distribute the classes to have them both in same distribution using over-sampling or undersampling techniques. This way no class is biased with respect to the other.

E. Label Conversion

We have the labels set for the binary classification as normal network traffic and DDoS/Attack traffic. Both these labels can mathematically be represented by a function f(y), where y denotes the data label.

$$F(y) = Normal, fory \tag{4}$$

F. Exploratory Data Analysis

Here we check the features and their dependency by visualizing the data patterns. We use either statistical or graphical analysis to summarize the main characteristics. The data visualization techniques we can use include principal component analysis (PCA) or t-Distributed Stochastic Neighbor Embedding (t-SNE) technique. They both check the patterns between the classes to predict without modeling which algorithms are going to perform better on the data. Here we have used T-SNE for dimensionality reduction in both datasets figure 4.

G. Feature and Model Selection

Feature selection is the first step of any model training. It selects the features in the dataset that contributes most to the prediction variable or output in which we are interested [34]. Having irrelevant features in the dataset can contribute to reduced accuracy scores and we need to remove them beforehand. There are many benefits of doing feature selection before modeling machine learning models. It reduces over-fitting, improves modeling accuracy and decreases the training time.

1) Information-Gain Test

Features are selected based on Information Gain scores. Information Gain score-based features give better accuracy than other statistical measures for it directly measures the entropy or randomness measures. The top ten scoring features for both datasets are given in the table I. Equation 5 is the formula for calculating the entropy of a feature X. Equation 6 is the formula for calculating the conditional entropy of a feature X given a class label Y.

$$(X) = -\Sigma_i P(x_i) \log(2P(x_i)) \tag{5}$$

$$H(X|Y) = -\Sigma_j \Sigma_i P(x_j|y_i) \log(2P(x_j|y_i)) \tag{6}$$

$$IG(X|Y) = H(X) - H(X|Y) \tag{7}$$

2) Decision Tree Test

The Decision Tree (DT) is a machine learning algorithm that is often used for feature selection in classification problems [35]. It works by constructing a tree-like model of decisions based on the features of the data. The internal nodes of the tree represent decisions based on the values of the features, and the leaf nodes represent the class labels of

TABLE I. Top 10 significant features and their scores with IG testing.

KDD99		UNSW-NB15	
Features	IG scores	Features	IG scores
src_bytes	0.445262	ct_state_ttl	0.661290
count	0.435637	sttl	0.653078
service	0.397823	dttl	0.640302
dst_bytes	0.368835	dbytes	0.611305
dst_host_same_src_port_rate	0.276318	smeansz	0.602808
srv_count	0.247668	Stime	0.562286
dst_host_count	0.223360	dmeansz	0.561209
protocol_type	0.221617	Dpkts	0.555514
dst_host_srv_diff_host_rate	0.185189	dmeansz	0.485649
dst_host_srv_count	0.129100	Dpkts	0.437280

the data (Figure 5). DTs can be used to identify the most important features in a dataset by ranking them based on their importance in the decision-making process. DTs are often used as part of an ensemble classifier, which combines the predictions of multiple classifiers to make a final prediction. In the proposed DDT detection setup, DTs are used as part of the hybrid classifier, which combines multiple classifiers to improve the accuracy of attack predictions. The top 10 features with the highest importance score are identified using DTs on both the KDD Cup 99 and UNSW-NB15 datasets table II.

5. Model Deployment and Experimental Set-Up

The proposed architecture is set up in Anaconda-Jupyter-Python framework with a GPU base. We are using two cloud architectures, Heroku (as platform as service model) and Amazon web services (as infrastructure as service model) with Postman as third-party REST-API client. For training of our machine learning models, we are using pre-trained hybrid classifier trained on two datasets, KDD cup 99 and UNSW-NB 15 with different attack size, features and performance metrics. The other configuration details are given in table III below.

A. *Setting Hyper-Parameter Values and Training Hybrid Machine Learning Model*

For the model selection we need to check on highest performing machine learning model based on various performance analysis measures. We have trained and tested all the major machine learning models, optimized their performance by using various hyper-parameter optimization techniques. The accuracy score of Decision Tree, Random Forest, XGBoost and non-linear SVM is better than the rest and we use these four models for our hybrid machine learning classification. The hybrid classifier selects the best performance among the four best performing models to be used for our proposed DDoS detection set-up. The detection parameters of each machine leaning models is detailed in table IV below.

From table IV, V, and VI, we can see that different machine learning models have different set of hyper-parameters and accuracy scores on different datasets and feature selection techniques. From the results we infer that hybrid classifier selects the best of the classifier performance for the final prediction. For KDD Cup 99 the hybrid classifier has the same performance as that of XGBoost classifier and for UNSW-NB15, it has same performance as that of Random Forest classifier. Both XGboost and Radom Forest classifier have the highest accuracy score among all the other trained machine learning models. In hybrid learning we use amalgamation of heterogenous classifiers for improved performance using either majority voting or simply appending the common highest performance scores of individual classifiers while training. We use the latter in our proposed work.

B. *Serialization*

We serialize and de-serialize python object structures (our machine learning models) using picking process also called as marshaling or flattening [36]. We convert these machine learning classifiers as python object in memory to a byte stream that can be stored on a disk or could be sent over a network. Thus, is different from the JSON format where we convert the data to a human readable format. We then retrieve the python object from the character stream using de-serialization. This is thus the transformation of python object which is stored as data in Random Access Memory (RAM) to the byte stream and vice versa. This is an important part of rendering the availability machine learning models as per our requirement for any real time DDoS detection.

6. Cloud architectures

For attack prediction of our proposed DDoS detection model, we use two different cloud architectures i.e., Heroku and Amazon EC2. Heroku is a platform as a service (PaaS) cloud platform, which means that it provides a complete platform for developing, running, and managing applications, including the infrastructure, runtime environment, and other services such as databases and messaging [37]. Heroku is designed to be easy to use and requires minimal configuration, making it a popular choice for developers who want to get their applications up and running quickly [37]. Amazon EC2 (Elastic Compute Cloud) is an infrastructure as a service (IaaS) cloud platform, which

TABLE II. Top 10 significant features and their scores with Decision Tree testing.

KDD99		UNSW-NB15	
Features	Decision Tree scores	Features	Decision Tree scores
count	0.881905	ct_state_ttl	0.976662
flag	0.030078	is_sm_ips_ports	0.006275
src_bytes	0.024445	sbytes	0.004351
dst_bytes	0.021507	synack	0.002840
dst_host_srv_diff_host_rate	0.015716	Ltime	0.002473
service	0.010178	Stime	0.002241
wrong_fragment	0.007232	smeansz	0.001604
dst_host_same_src_port_rate	0.002349	dsport	0.000728
dst_host_diff_srv_rate	0.001665	Spkts	0.000600
rerror_rate	0.001184	ct_srv_dst	0.000577

TABLE III. Configuration Details of Proposed DDoS Detection Set-up.

Project	**Environment**
Operating System	Ubuntu 18.04 LTS
Python	Python 3.10.9
Anaconda	4.10.3
Jupiter Notebook	Version 6.3.0
Sci-kit Learn	0.24.1
Mlxtend	0.18.0
Cloud Architectures	Heroku and Amazon EC2 for hosting our proposed DDoS detector and Google Colab Pro with GPU accelerator and high RAM for training our machine learning models.
API Platform	Postman v9.4
Repository	GitHub

means that it provides raw computing resources that can be used to build and run applications and services [38]. With EC2, users have more control over the underlying infrastructure, including the choice of operating system, runtime environment, and other software components. EC2 is more flexible than PaaS platforms like Heroku, but also requires more setup and configuration [38].

Both Heroku and Amazon EC2 can be used to host applications and services for DDoS attack prediction, depending on the specific requirements of the application and the needs of the userTop of FormBottom of Form.

A. Heroku and HTTP Routing

Heroku uses both public and private APIs to render services to various users and customers [37]. APIs can also be used for defence too for the Distributed Denial of Service attacks that happen on a cloud architecture especially HTTP, XML or even REST based DDoS attacks. We use Heroku as our platform as a service cloud platform. It was the first cloud platform developed by salesforce and dates back to 2007. What makes it a robust cloud platform is that the developer that it uses to build and run the applications, supports multiple programming languages. Thus, it is also known as polyglot cloud platform. Applications that run on Heroku have a unique domain which routes the HTTP requests to the correct application container which we call as dyno. We type the application URL or makes a request to it via an API, to route to the persistent location that hosts our DDoS detection application (figure 6). This is done using Heroku router which is actually a software and a gateway between the users and dynos hosting an application.

B. Amazon EC2 deployment with DDoS detector

Amazon Elastic Compute offers the most efficient and multifaceted platform with over 500 instances of computing resources (figure 7). The resources range from storage, networking, operating systems to machine learning and mobile developing tools [38]. AWS is the major cloud service provider that supports INTEL, AMD and arm processors [39]. It has an on demand EC2 instances with over 400 Gbps ethernet networking. It provides on demand cloud platforms and APIs to individuals, companies and government agencies on a metered and pay-as-you-go basis.

The EC2 provides the scalable deployment of multiple applications through web-services which one can boot as Amazon Machine Image (AMI) to finally configure a virtual machine or an instance. The services are functional through APIs and that are accessed over HTTP protocol using the REST architectural style (with SOAP protocol for older APIs and JSON for new). These APIs are accessed by the client in various ways including AWS console (a website), SDKs (python/java) or making direct REST calls. Here in our proposed architecture, we access the AWS resources through the REST calls. To deploy our DDoS detection

TABLE IV. Hyper-Parameters of all the major machine learning models.

Classifier	Hyper Parameters CV Attributes	Grid Search	Maximum Performing Features	
			Information Gain	Decision Tree Classifier
Logistic Regression	Maximum Iterations Penalty Parameter	Classifier, Hyper-, Parameters Cross validation fold Scoring metric = accuracy	KDD: ('C': 1, 'max_iter' 100) UNSW-NB15: ('C' : 1, 'max_iter': 100)	KDD: ('C': 0.1, 'max_iter': 100) UNSW-NB15: ('C': 1, 'max_iter': 100)
Decision Tree classifier	Maximum Depth and Minimum Sample Split	Classifier, Hyper-Parameters, Cross validation fold, Scoring metric = accuracy	KDD: ('max_depth': None, 'min_samples_split' : 2) UNSW-NB15: ('max_depth': None, 'min_samples_split': 2)	KDD: ('max_depth': None, 'min_samples_split' : 2) UNSW-NB15: ('max_depth': None, 'min_samples_split': 2)
SVM	Kernel function [linear, poly, rbf]	Classifier, Hyper-, Parameters Cross validation fold, Scoring metric = accuracy	KDD: ('kernel': 'rbf') UNSW-NB15: ('kernel': 'poly')	KDD: ('kernel': 'rbf') UNSW-NB15: ('kernel': 'rbf')
KNN	Number of neighbors	Classifier, Hyper-Parameters, Cross validation fold, Scoring metric = accuracy	KDD Cup 99: k=3 , UNSW-NB15: k=3	
Random Forest Classifier	Number of estimators, maximum and criterion [Gini, Entropy]	Classifier, Hyper-Parameters, Cross Scoring metric = accuracy	KDD:('criterion': 'entropy' 'max_depth': 20, 'n_estimators': 100) UNSW-NB15: ('criterion' : 'max_depth': 50, 'n_estimators': 100)	KDD:('criterion': ' gini', 'max _depth': 30, 'n_estimators': 100 UNSW -NB15: ('criterion' : 'entropy' 'max _depth' : 50, 'n_estimators': 100)
XGBoost	Maximum Depth	Classifier, Hyper-Parameters, Cross Scoring metric = accuracy	KDD: ('max_depth': 7) UNSW-NB15: (' max_depth': 7)	KDD: ('max_depth': 7) UNSW-NB15: ('max _depth': 7)

app on an Amazon EC2 instance, we need to follow certain steps. Here is a brief summary of each step:

1) *Create an AWS account and set up an EC2 instance*

This is a virtual server in the cloud that we can use to host your app.

2) *Choose an AMI and add storage*

An AMI (Amazon Machine Image) is a pre-configured virtual machine image that we can use to launch an EC2 instance. We need to choose an AMI that meets the requirements of our app, and then add the necessary storage to the instance.

3) *Add tags and configure security groups*

We can use tags to identify and organize our EC2 instances. We will also need to configure security groups, which specify rules for accessing the instance.

4) *Clone the app from the GitHub repository to the EC2 instance and access the instance through the SSH client*

Once we have set up the EC2 instance, we can clone the app from its GitHub repository onto the instance. We can then use an SSH client to access the instance and run the app. Here 43.206.226.128 is the public IPv4 address of the instance.

URL: ssh - i "vs-flask-hybrid.pem" @ ec2-43-206-226-128.ap-northeast -1.compute.amazon aws.com

C. API testing using Postman

APIs provide a set of standardized methods for accessing and interacting with a system or service, and they can be used to facilitate communication between different applications or components. Web services are a way for different systems or devices to communicate over the internet or other networks. SOAP and REST are two common architectures for implementing web services. SOAP is based on XML and is typically used for more complex and secure communication, while REST is based on HTTP and is usually used for simpler, more lightweight communication [40].

Postman is a tool that allows developers to test and work with web services, particularly APIs (Application Programming Interfaces) [41]. It allows developers to send HTTP requests and view the responses from the server (figure 8). It's a useful tool for testing and debugging APIs, as well as for documenting and sharing APIs with

TABLE V. Performance scores of machine learning models and hybrid classifier on KDD Cup 99 dataset with Information Gain as feature selection technique.

S. No	Classifier	Accuracy (%)	FAR (%)	Sensitivity (%)	Specificity (%)	FPR (%)	AUC (%)	Precision	Recall	f1
0	Logistic Regression	99.734865	0.265135	99.124088	100.000000	0.000000	99.562044	1.000000	0.991241	0.995601
1	Decision Tree Classifier	99.955811	0.044189	100.000000	99.936629	0.063371	99.968314	0.998542	1.000000	0.999271
2	Linear SVM	99.734865	0.265135	99.270073	99.936629	0.063371	99.603351	0.998532	0.992701	0.995608
3	GaussianNB	99.779054	0.220946	99.270073	100.000000	0.000000	99.635036	1.000000	0.992701	0.996337
4	KNN	99.779054	0.220946	99.270073	100.000000	0.000000	99.635036	1.000000	0.992701	0.996337
5	Random Forest Classifier	99.911622	0.088378	99.854015	99.936629	0.063371	99.895322	0.998540	0.998540	0.998540
6	XGB Classifier	99.955811	0.044189	100.000000	99.936629	0.063371	99.968314	0.998542	1.000000	0.999271
7	SVM (polynomial)	99.911622	0.088378	99.708029	100.000000	0.000000	99.854015	1.000000	0.997080	0.998538
8	Hybrid Model	99.955811	0.044189	100.000000	99.936629	0.063371	99.968314	0.998542	1.000000	0.999271

TABLE VI. Performance scores of machine learning models and hybrid classifier on UNSW-NB15 dataset with Decision Tree as feature selection technique.

S. No	Classifier	Accuracy (%)	FAR (%)	Sensitivity (%)	Specificity (%)	FPR (%)	AUC (%)	Precision	Recall	f1
0	Logistic Regression	98.591593	1.408407	98.169898	99.014748	0.985252	98.592323	0.990097	0.981699	0.985880
1	Decision Tree Classifier	99.866162	0.133838	99.871054	99.861254	0.138746	99.866154	0.998617	0.998711	0.998664
2	Linear SVM	98.165959	1.834041	97.066866	99.268855	0.731145	98.167860	0.992550	0.970669	0.981487
3	GaussianNB	97.019002	2.980998	95.028586	99.016307	0.983693	97.022446	0.989789	0.950286	0.96963
4	KNN	97.019002	2.980998	95.028586	99.016307	0.983693	97.022446	0.989789	0.950286	0.969635
5	Random Forest Classifier	99.928412	0.071588	99.950286	99.906463	0.093537	99.928375	0.999068	0.999503	0.999286
6	XGB Classifier	99.819475	0.180525	99.843090	99.795778	0.204222	99.819434	0.997966	0.998431	0.998198
7	SVM (polynomial)	99.896509	0.103491	99.951839	99.840988	0.159012	99.896414	0.998417	0.999518	0.998967
8	Hybrid Model	99.928412	0.071588	99.950286	99.906463	0.093537	99.928375	0.999068	0.999503	0.999286

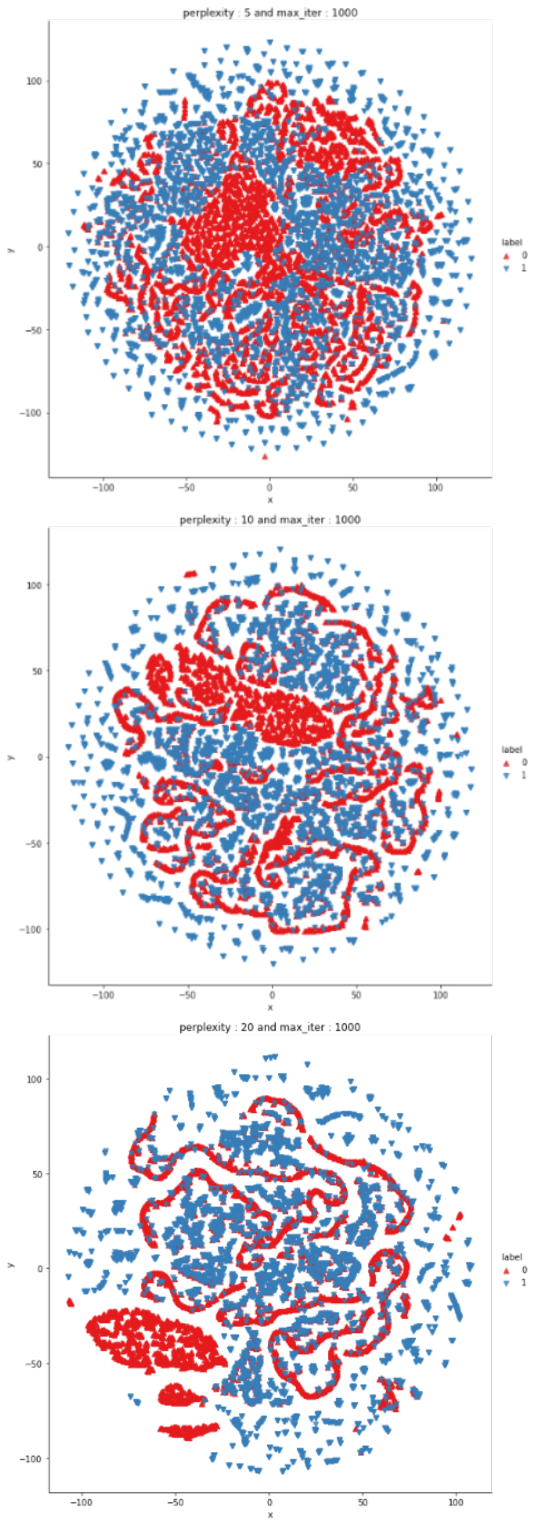


Figure 4. Data Reduction using T-SNE.

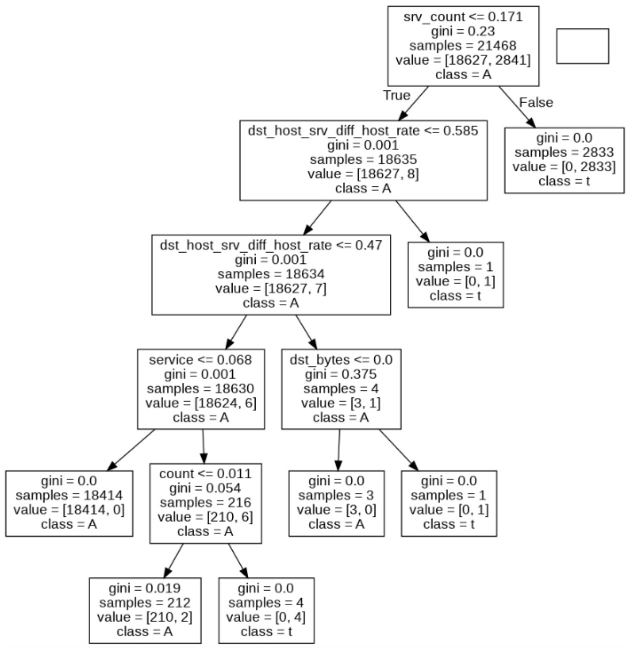


Figure 5. Decision Tree Architecture.

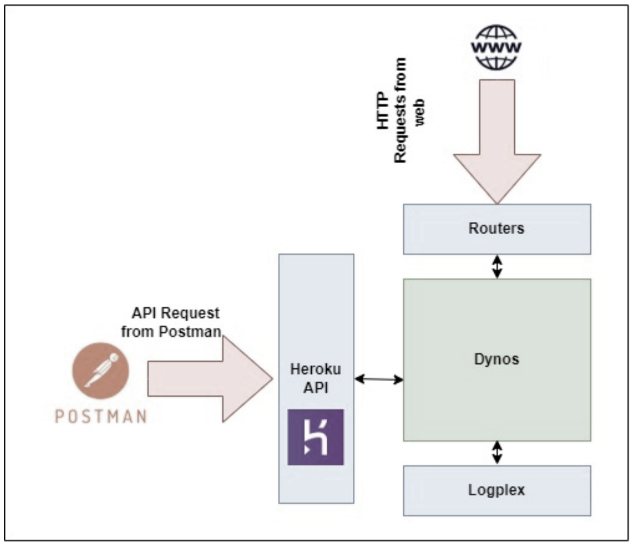


Figure 6. Heroku-Postman Architecture.

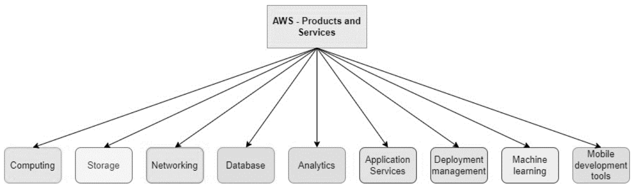


Figure 7. Amazon Web Service – Products and Services.

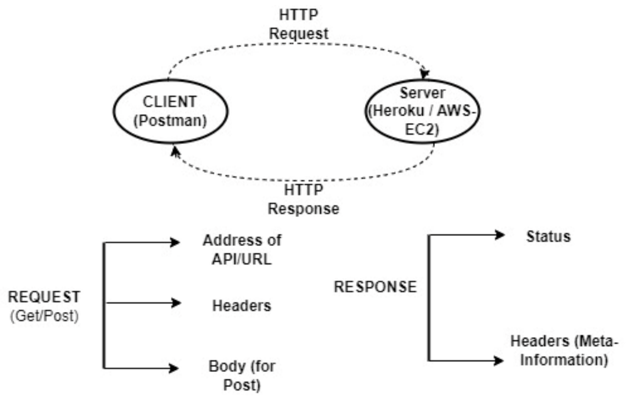


Figure 8. Client-Server Request and Response Structure.

other developers.

1) *Request*

When a request is sent to an API, it includes the complete URL, HTTP headers, and a body or payload [42]. The API processes the request and sends a response back to the client, which includes an HTTP status code indicating whether the request was successful, and a JSON response that contains the data requested by the client. The JSON response can also include additional information such as error messages or metadata (figure 8).

2) *Response*

HTTP responses contain the status line, headers, and message-response body. The status line includes the HTTP status code, which indicates the result of the request [43] (figure 8). There are several categories of HTTP status codes, each with a specific meaning:
1. 1XX codes: These are informational codes that are used to communicate intermediate steps in the request process.
2. 2XX codes: These are success codes that indicate that the request was successful. This includes the 200 OK code, which is the most common success code.
3. 3XX codes: These are redirection codes that are used when a client's request is redirected to a different URL.
4. 4XX codes: These are error codes that are returned when the client's request has an error. The most common 4XX codes are 400 Bad Request, 401 Unauthorized, 403 Forbidden, and 404 Not Found.
5. 5XX codes: These are server error codes that are returned when there is an error on the server side. The most common 5XX codes are 500 Internal Server Error and 503 Service Unavailable. It's important to handle these different HTTP status codes appropriately in your API to ensure that the client understands the result of their request (table VII).

7. Model Performance Evaluation

The main aim of our proposed approach is to classify the network traffic coming to a cloud as attack and normal. The performance of the approach is measured with respect to various metrics which include accuracy, false acceptance rate, sensitivity, specificity, false positive rate, precision, recall, F1 score and AUC curve analysis [44]. The hybrid classifier shows highest accuracy scores when Information Gain scored features are taken for feature selection with the highest of 99.955811% on KDD Cup 99 dataset and 99.928412% using UNSW-NB15 dataset with decision tree as feature selection method (after optimization and balancing both the datasets) (table VIII)

A. *Measuring performance through AUROC analysis*

The Receiver Operator Characteristic (ROC) and Area Under ROC (AUC) is another metric to analyse the performance of the classifier [45]. They are the best predictors of the performance of binary decision problem classifiers. The ROC curve plots the true positive rate (sensitivity) against the false positive rate (1 - specificity) at different classification thresholds, while the AUC represents the area under the ROC curve. AUC is a good metric to use when you want to compare the performance of different classifiers, as it is independent of the classification threshold. A classifier with an AUC of 1.0 is considered to be a perfect classifier, while a classifier with an AUC of 0.5 is considered to be a random classifier. The hybrid classifier ROC curve analysis shows that our hybrid classifier is highly efficient among all the machine learning models (Figure 9). It has the same performance characteristics as that of XGBoost classifier for KDD Cup 99 and Random Forest classifier for UNSW-NB15 as both have highest accuracy score among the rest of the trained machine learning models. For KDD Cup 99 dataset, the hybrid classifier shows highest performance when information gain is used as feature selection technique and for UNSW-NB15 dataset, the hybrid classifier shows highest performance when decision tree is used as the feature selection technique.

B. *Results and Discussion*

In this section we give the insight of obtained results of our REST-API based DDoS detector. We use Jupyter-Python framework to code the post requests. We send the POST request detailing the normal and attack parameters (of a particular packet) via the REST-API using JSON format on the cloud URL (Figure 10 and 11). In both POST requests we get the response of 200 which is a success code. This means it has been successfully uploaded on the cloud architecture.

The trained classifier which is stored as a pickle file on the GitHub repository is accessed and run as application on the cloud using the REST-API. To check whether our cloud can predict the packet as being attack or normal we use the Postman API Testing. Giving it the access to POST the cloud URL to predict, we get the response in the JSON format as “Prediction: Attack” which determines that the packet we used was that of an attack or “Prediction: Normal” which determines that there is normal traffic on the cloud (Figure 12).

Same is the case with API testing of Amazon EC2 architecture via the Postman. The response we get predicts

TABLE VII. HTTP responses, URL access and Status Codes values (43.206.226.128 as public IPv4-address of Amazon EC2)

Operation	URL	Method	Success/Failure	Status/Code
GET	**Heroku** https://ddosattack.herokuapp.com/pred **AWS** http://43.206.226.128/pred	**GET**	Success Not Found Failure	200 404 500
DELETE	**Heroku** https://ddosattack.herokuapp.com/pred **AWS** http://43.206.226.128/pred	**DELETE**	Success Not Found Failure	200 or 204 404 500
UPDATE	**Heroku** https://ddosattack.herokuapp.com/pred **AWS** http://43.206.226.128/pred	**PUT**	Success Wrong Format/Data Not Found Failure	200 400 or 415 404 500
CREATE	**Heroku** https://ddosattack.herokuapp.com/pred **AWS** http://43.206.226.128/pred	**POST**	Success Wrong Format/Data Failure	200 400 or 415 500

TABLE VIII. Performance metric scores of Hybrid Classifier

Dataset	Feature selection method	Accuracy (%)	FAR (%)	Sensitivity (%)	Specificity (%)	FPR (%)	AUC (%)	Precision	Recall	f1
KDD	IG Features	99.955811	0.044189	100.000000	99.936629	0.063371	99.968314	0.998542	1.000000	0.999271
UNSW	DT Features	99.928412	0.071588	99.950286	99.906463	0.093537	99.928375	0.999068	0.999503	0.999286

whether the cloud architecture is under DDoS attack or not (Figure 13 and 14)

Our proposed model thus makes the use of trained hybrid machine learning model in DDoS attack detection on the cloud architectures using the REST-API. By making the trained classifier available on a GitHub repository, it can be easily accessed and used by cloud architectures when needed. This makes the model lightweight, robust, and easily accessible, with maximum efficiency.

C. Conclusion and Future Work

A very important role of network security is the detection of DDoS attack. Traditional networks can be complex and may require more complex set-ups for attack detection. Our proposed model aims to address this issue by using a lightweight REST API-based DDoS detector that can be deployed on platform as a service (PaaS) and infrastructure as a service (IaaS) cloud platforms using a hybrid machine learning classifier. The hybrid machine learning model we used is a combination of four trained heterogeneous classifiers (Decision Tree, Random Forest, XGBoost, and Non-Linear Support Vector Machine) that have been optimized and tested on two different datasets. The trained hybrid classifier is stored in a GitHub repository, and the cloud architecture accesses it through a third-party client (Postman) via a REST API. This allows the cloud architecture to use the hybrid classifier for DDoS attack detection without having to host the detection set-up within the cloud itself. This can help to reduce the burden on the cloud architecture and ensure maximum performance.

For the future work we can check for further possibilities of using pre-trained machine learning models or deep neural networks in other real-world applications. Right now, a stable cloud architecture and its defense are the most sought-after research problems around the world which need an immediate addressal.

References

[1] S. Bhatia, S. Behal, and I. Ahmed, "Distributed denial of service attacks and defense mechanisms: current landscape and future directions," *Versatile Cybersecurity*, pp. 55–97, 2018.

[2] R. Kesavamoorthy, P. Alaguvathana, R. Suganya, and P. Vigneshwaran, "Classification of ddos attacks–a survey," *Test Eng. Manag*, vol. 83, pp. 12 926–12 932, 2020.

[3] B. Sudharsan, D. Sundaram, P. Patel, J. G. Breslin, and M. I. Ali, "Edge2guard: Botnet attacks detecting offline models for resource-constrained iot devices," in *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*. IEEE, 2021, pp. 680–685.

[4] Google Cloud Blog. (2023) How google cloud blocked largest layer 7 ddos attack at 46 million rps. [Online]. Available: https://cloud.google.com/blog/products/identity-security/how-google-cloud-blocked-largest-layer-7-ddos-attack-at-46-million-rps

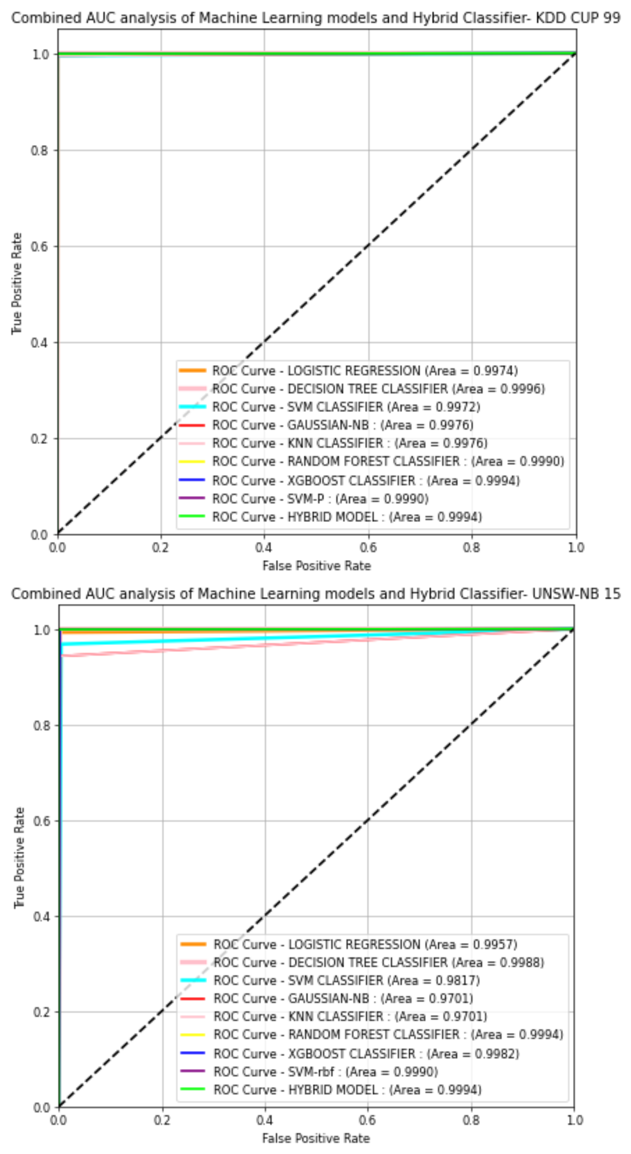
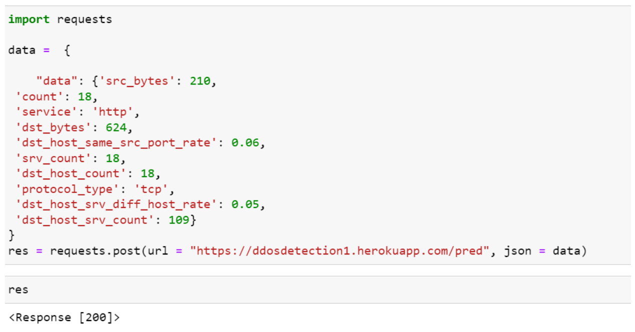
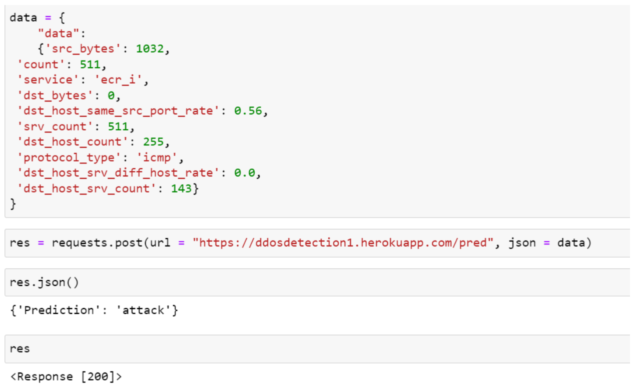


Figure 9. ROC curves for the Hybrid Classifier



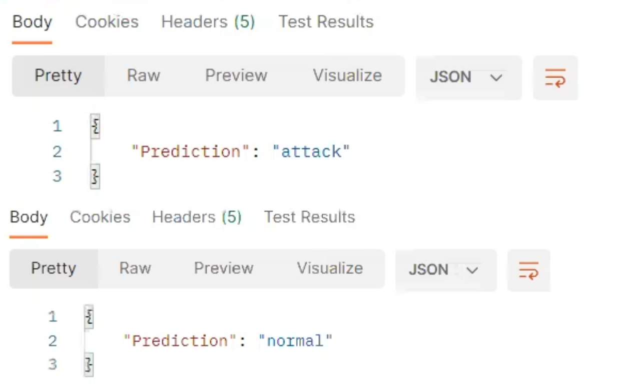
```
import requests

data =  {

    "data": {'src_bytes': 210,
 'count': 18,
 'service': 'http',
 'dst_bytes': 624,
 'dst_host_same_src_port_rate': 0.06,
 'srv_count': 18,
 'dst_host_count': 18,
 'protocol_type': 'tcp',
 'dst_host_srv_diff_host_rate': 0.05,
 'dst_host_srv_count': 109}
}
res = requests.post(url = "https://ddosdetection1.herokuapp.com/pred", json = data)

res

<Response [200]>
```

Figure 10. Normal Data Post Request sent via REST-API using JSON format on the Cloud URL



```
data = {
    "data":
    {'src_bytes': 1032,
 'count': 511,
 'service': 'ecr_i',
 'dst_bytes': 0,
 'dst_host_same_src_port_rate': 0.56,
 'srv_count': 511,
 'dst_host_count': 255,
 'protocol_type': 'icmp',
 'dst_host_srv_diff_host_rate': 0.0,
 'dst_host_srv_count': 143}
}

res = requests.post(url = "https://ddosdetection1.herokuapp.com/pred", json = data)

res.json()

{'Prediction': 'attack'}

res

<Response [200]>
```

Figure 11. Attack Data Post Request sent via REST-API using JSON format on the Cloud URL



```
{
    "Prediction": "attack"
}
```

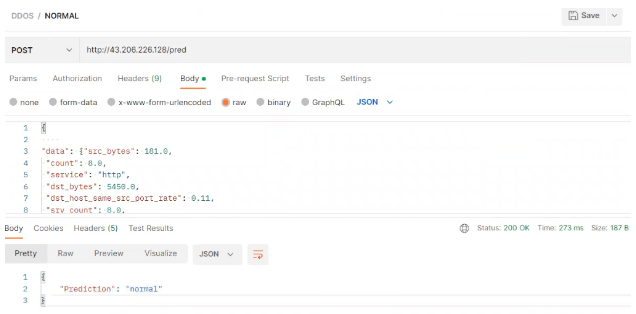
```
{
    "Prediction": "normal"
}
```

Figure 12. Final API based DDoS detection testing via Postman

[5] Cloudflare Blog. (2022) Cloudflare ddos threat report 2022 q3. [Online]. Available: https://blog.cloudflare.com/cloudflare-ddos-threat-report-2022-q3/

[6] A. A. Khan and M. Zakarya, "Energy, performance and cost efficient cloud datacentres: A survey," *Computer Science Review*, vol. 40, p. 100390, 2021.

[7] D. Radain, S. Almalki, H. Alsaadi, and S. Salama, "A review on defense mechanisms against distributed denial of service (ddos) attacks on cloud computing," in *2021 International Conference of*



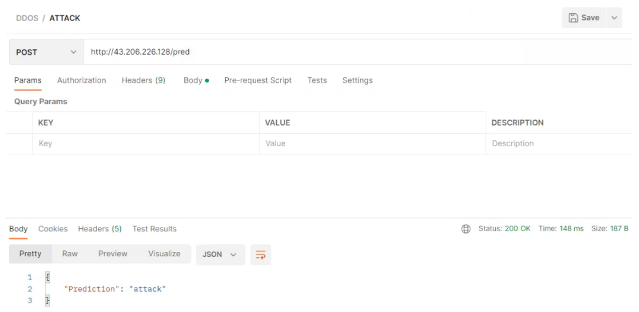
POST http://43.206.226.128/pred

```
{

"data": {"src_bytes": 181.0,
 "count": 8.0,
 "service": "http",
 "dst_bytes": 5450.0,
 "dst_host_same_src_port_rate": 0.11,
 "srv_count": 8.0,
```

Status: 200 OK Time: 273 ms Size: 187 B

```
{
    "Prediction": "normal"
}
```

Figure 13. REST-API based DDoS detection on Amazon EC2 testing via Postman - NORMAL prediction



POST http://43.206.226.128/pred

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Status: 200 OK Time: 148 ms Size: 187 B

```
{
    "Prediction": "attack"
}
```

Figure 14. REST-API based DDoS detection on Amazon EC2 testing via Postman – ATTACK prediction

Women in Data Science at Taif University (WiDSTaif). IEEE, 2021, pp. 1–6.

[8] V. Vedula, P. Lama, R. V. Boppana, and L. A. Trejo, "On the detection of low-rate denial of service attacks at transport and application layers," *Electronics*, vol. 10, no. 17, p. 2105, 2021.

[9] H. Khandare, S. Jain, and R. Doriya, "A survey on http flooding—a distributed denial of service attack," in *Pervasive Computing and Social Networking: Proceedings of ICPCSN 2022*. Springer, 2022, pp. 39–52.

[10] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey," *applied sciences*, vol. 9, no. 20, p. 4396, 2019.

[11] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, M. Gao, H. Hou, and C. Wang, "Machine learning and deep learning methods for cybersecurity," *Ieee access*, vol. 6, pp. 35 365–35 381, 2018.

[12] M. Nooribakhsh and M. Mollamotalebi, "A review on statistical approaches for anomaly detection in ddos attacks," *Information Security Journal: A Global Perspective*, vol. 29, no. 3, pp. 118–133, 2020.

[13] Y. Wei, J. Jang-Jaccard, F. Sabrina, A. Singh, W. Xu, and S. Camtepe, "Ae-mlp: A hybrid deep learning approach for ddos detection and classification," *IEEE Access*, vol. 9, pp. 146 810–146 821, 2021.

[14] E. Payares and J. C. Martínez-Santos, "Quantum machine learning for intrusion detection of distributed denial of service attacks: a comparative overview," *Quantum Computing, Communication, and Simulation*, vol. 11699, pp. 35–43, 2021.

[15] B. A. Khalaf, S. A. Mostafa, A. Mustapha, M. A. Mohammed, and W. M. Abduallah, "Comprehensive review of artificial intelligence and statistical approaches in distributed denial of service attack and defense methods," *IEEE Access*, vol. 7, pp. 51 691–51 713, 2019.

[16] N. Hoque, H. Kashyap, and D. K. Bhattacharyya, "Real-time ddos attack detection using fpga," *Computers & Communications*, vol. 110, pp. 48–58, 2017.

[17] M. Woźniak, M. Grana, and E. Corchado, "A survey of multiple classifier systems as hybrid systems," *Information Fusion*, vol. 16, pp. 3–17, 2014.

[18] N. N. Abdulla and R. K. Hasoun, "Review of detection denial of service attacks using machine learning through ensemble learning," *Iraqi Journal for Computers and Informatics*, vol. 48, no. 1, pp. 13–20, 2022.

[19] T. Karnwal, T. Sivakumar, and G. Aghila, "A comber approach to protect cloud computing against xml ddos and http ddos attack," in *2012 IEEE Students' Conference on Electrical, Electronics and Computer Science*. IEEE, 2012, pp. 1–5.

[20] G. Leaden *et al.*, "An api honeypot for ddos and xss analysis," in *2017 IEEE MIT Undergraduate Research Technology Conference (URTC)*. IEEE, 2017.

[21] O. Rahman, M. A. G. Quraishi, and C. H. Lung, "Ddos attacks detection and mitigation in sdn using machine learning," in *2019 IEEE World Congress on Services (SERVICES)*, 2019, pp. 184–189.

[22] R. M. A. Ujjan *et al.*, "Towards sflow and adaptive polling sampling for deep learning-based ddos detection in sdn," *Future Generation Computer Systems*, vol. 111, pp. 763–779, 2020.

[23] M. S. Potnis *et al.*, "Hybrid intrusion detection system for detecting ddos attacks on web applications using machine learning," *ICT Analysis and Applications*, pp. 797–805, 2022.

[24] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions," *SN Computer Science*, vol. 2, no. 3, pp. 1–21, 2021.

[25] R. K. Batchu and H. Seetha, "A generalized machine learning model for ddos attacks detection using hybrid feature selection and hyperparameter tuning," *Computer Networks*, vol. 200, p. 108498, 2021.

[26] B. Jia *et al.*, "A ddos attack detection method based on hybrid heterogeneous multiclassifier ensemble learning," *Journal of Electrical and Computer Engineering*, p. 2017, 2017.

[27] V. Miškovic, "Machine learning of hybrid classification models for decision support," *Sinteza 2014-Impact of the Internet on Business Activities in Serbia and Worldwide*, pp. 318–323, 2014.

[28] L. I. Kuncheva, "That elusive diversity in classifier ensembles," in *Proceedings of the 1st Iberian Conference on Pattern Recognition and Image Analysis (ibPRIA '03)*. Springer, 2003, pp. 1126–1138.

[29] I. Obeidat *et al.*, "Intensive pre-processing of kdd cup 99 for network intrusion classification using machine learning techniques," pp. 70–84, 2019.

[30] Z. Zoghi and G. Serpen, "Unsw-nb15 computer security dataset: Analysis through visualization," *arXiv preprint arXiv:2101.05067*, 2021.

[31] Z. Zoghi, "Ensemble classifier design and performance evaluation for intrusion detection using unsw-nb15 dataset," 2020.

[32] B. Habib and F. Khursheed, "Performance evaluation of machine learning models for distributed denial of service attack detection using improved feature selection and hyper-parameter optimization techniques," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 26, p. e7299, 2022.

[33] S. Das *et al.*, "Empirical evaluation of the ensemble framework for feature selection in ddos attack," in *2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*. IEEE, 2020.

[34] H. Polat, O. Polat, and A. Cetin, "Detecting ddos attacks in software-defined networks through feature selection methods and machine learning models," *Sustainability*, vol. 12, no. 3, p. 1035, 2020.

[35] K. Sahin and I. C. Emrehan, "Performance analysis of advanced decision tree-based ensemble learning algorithms for landslide susceptibility mapping," *Geocarto International*, vol. 36, no. 11, pp. 1253–1275, 2021.

[36] "pickle — python object serialization," https://docs.python.org/3/library/pickle.html, accessed: 2023-08-17.

[37] N. Middleton and R. Schneeman, *Heroku: up and running: effortless application deployment and scaling*. O'Reilly Media, Inc., 2013.

[38] M. Wittig and A. Wittig, *Amazon web services in action*. Simon and Schuster, 2018.

[39] W. Khan, "Ddos mitigation analysis of aws cloud network," 2017.

[40] M. I. Beer and M. F. Hassan, "Adaptive security architecture for protecting restful web services in enterprise computing environment," *Service Oriented Computing and Applications*, vol. 12, no. 2, pp. 111–121, 2018.

[41] J. Jain, *Learn API Testing*. Berkeley, CA: Apress, 2022.

[42] A. Rodriguez, "Restful web services: The basics," *IBM developerWorks*, no. 33, p. 18, 2008.

[43] "Api testing using postman," https://www.softwaretestinghelp.com/api-testing-using-postman/, accessed: 2023-08-17.

[44] A. A. Salih and A. M. Abdulazeez, "Evaluation of classification algorithms for intrusion detection system: A review," *Journal of Soft Computing and Data Mining*, vol. 2, no. 1, pp. 31–40, 2021.

[45] N. Bindra and M. Sood, "Detecting ddos attacks using machine learning techniques and contemporary intrusion detection dataset," *Automatic Control and Computer Sciences*, vol. 53, no. 5, pp. 419–428, 2019.

Beenish Habib Beenish Habib is a PhD candidate in the Department of Electronics and Communication Engineering in NIT Srinagar. Her field of expertise is Cloud and Network Security and is mainly working on detecting and mitigating DDoS attacks. She has done her Bachelors in Technology in ECE from IUST Awantipora and Masters in Technology in Communication and IT from NIT Srinagar. She has 3 years of teaching experience and 4 years of Research Experience.

Farida Khursheed Dr. Farida Khurshid is currently providing services as Associate Professor in ECE department in NIT Srinagar. She has authored and co-authored multiple peer-reviewed scientific papers and presented works at many national and international conferences. Her research interests include Digital Image Processing, Security and Biometrics.