VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Title: Malicious DDoS Attack detection using Machine Learning and Artificial Intelligence

A PROJECT REPORT

Team Members

Nakul Jadeja (19BCE0660)

Swayam Sharma (19BCE0523)

Sayan Saha (19BCE0510)

Shubh Gupta (20BCI0205)

School of Computer Science and Engineering

Course Code: CSE4003

Course Name: Cyber Security

Slot: A2 + TA2

Winter Semester 2021-22

Professor: Gayathri P

Acknowledgement

We are thankful to the Department because of whom, we have gained confidence in Innovative Thinking and it also enhanced our professional skills to become competent in this field.

In performing our project, we had to take the help and guideline of some respected persons, who deserve our greatest gratitude. The completion of this project gives us much pleasure. We would like to show our gratitude to **Prof. Gayathri P**, SCOPE, VIT University for giving us a good guideline for the project throughout numerous consultations. We would also like to expand our deepest gratitude to all those who have directly and indirectly guided us in this project.

Table of Contents

1. ABSTRACT

Distributed Denial of Service (DDoS) is a type of Cybersecurity threat which is one of many versions of Denial of Service (DoS) that uses IP addresses to attack a particular server/victim. DDoS threats are well-coordinated attacks that use compromised secondary victims to target the single or multiple victim systems, may it be a large firm server or a small-scale system. The DDoS threats are very costly in terms of bandwidth and power and they result in loss of confidential data as well. Therefore, it has become of much importance to devise better algorithms to detect different types of DDoS Cyber Threats with higher accuracy while considering the computation cost in detecting these threats. Most of the study conducted in literature assumes detection of DDoS threats as a binary classification problem and their results give out whether an attack was attempted or not. However, in order to effectively defend the network from causing extensive damage, it is of paramount importance to know which type of DDoS attack is targeting the network or the system.

2. KEYWORDS

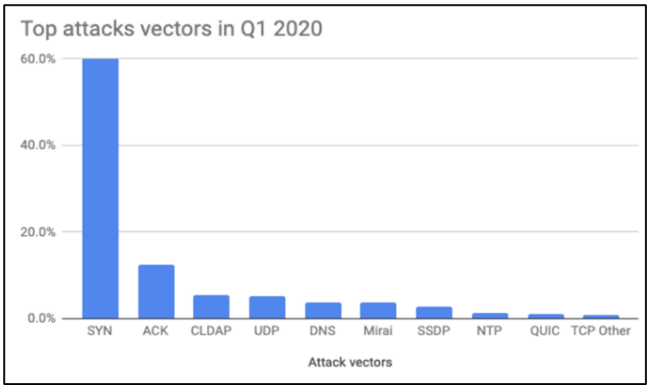
Distributed Denial of Service (DDoS); Transport Control Protocol (TCP); User Datagram Protocol (UDP); DDoS Evaluation Dataset (CICDDoS); Support Vector Machine (SVM); Decision Tree; Random Forest; Adaptive Boosting; Majority Vote Classifier;

3. INTRODUCTION

Distributed Denial of Service (DDoS) are flooding threats that deny a legitimate user from accessing its intended service. It is one of the most prevalent threats that the Cybersecurity industry is facing as per the recent market research. Therefore, it is of foremost importance to understand and detect different types of DDoS threats. There are mainly two major types of DDoS attacks Reflection Based and Exploitation Based as given in and each of the DDoS attacks falls in either of the two categories.

Reflection Based DDoS threats are those kinds of attacks in which the hacker hides its identity by using third-party tools and components. The attack is initiated by sending the data packets to a reflector server with the source and IP address of the victim/target. These attacks are executed through the Application layer protocol using either Transport Control Protocol (TCP), User Datagram Protocol or by using both of them simultaneously. The TCP based attacks include MSSQL, SSDP whereas UDP based attacks include NTP, TFTP and TCP/UDP based combined attacks include DNS, LDAP, NETBIOS and SNMP.

Exploitation based attacks are similar to that of Reflection based attacks and it also uses third party software and components to remain anonymous when initiating the attack. It has TCP based attacks and UDP based attacks, TCP based attack consists of SYN Flood whereas UDP based attacks consist of UDP-Flood (UDP) and UDP-lag. The hacker initiates the UDP threat by sending a huge amount of UDP packets at a very high rate on the random ports of the target machine which leads to the exhaustion of bandwidth and thus the system crashes.



Top Attack Vectors in 2021

3.1 Aim

We aim to develop an Ensemble Classifier that combines the performance of top 4 performing algorithms and compares it with different Artificial Intelligence and Machine Learning (AI and ML) algorithms to effectively detect the different types of DDoS threats by converting the problem to a multilabel classification problem.

3.2 Objective

The objective of our project lies in Analysis of Benign and DDoS Attack dataflows and also in creating machine learning models to detect malicious DDoS activity. This project will deal with several types of DDoS attacks which includes DNS, LDAP, MSSQL, NTP, NetBIOS, Portmap, SNMP, SSDP, Syn, TFTP and UDPLag. Our project includes 19 datasets which are combined into 11 DDoS and 1 Benign Dataset of over 70 common websites/ services. The target variables would be divided as Malicious (Binary Classification and Label (Multiclass Classification)

3.3 Scope and Applicability

Our project deals with the baseline models and advanced models. In addition, the project will present an Ensemble Classifier MV-4 which will combine the performance of the top 4 AI/ML models to give a good performing classifier for this dataset. Our project is divided into sections. One section will give an overview of relevant research and study conducted for DDoS Cyber Threat detection, another section will deal with the description of the methods and parameters used in this study, another section dealing with information about the Dataset and software used along with Metrics used for evaluating AI and ML models. At last, we discuss testing techniques and results and analysis for the used model, in the final section we give conclusions, and possible future works.

4. LITERATURE REVIEW

Reference No	Paper Title	Journal name and year of publication	Work done	Technique used	Disadvantages / Gaps identified	Student Reg. No. and name
1	Smart Detection: An Online Approach for DoS/DDoS Attack Detection Using Machine Learning	Francisco Sales de Lima Filho, 2019	Proposed system acts as a sensor that can be installed anywhere on the network and classifies online traffic using an MLA-based strategy that makes inferences utilising random traffic samples collected on network devices via stream protocol.	The software uses the Random Forest Tree algorithm to classify network traffic based on samples taken by the sFlow protocol directly from network devices.	Disadvantages would include analysis of DDoS attacks based on the vulnerabilities of services such as Heartbleed and web brute force attack, enhancement in the multiple-class classification, self-configuration of the system, developing methods for correlating triggered alarms, and formulating protective measures.	Nakul Jadeja (19BCE0660)
2	DDoS attack detection based on neural network	International Symposium on Aware Computing, 2020	We propose a DDoS detection method that uses the LVQ neural network for host anomaly detection. In this way, it can improve the recognition rate of detection systems. (99.732% Recognition Rate)	Learning Vector Quantisation (LVQ),	The application of neural network is limited to academic and research world. Therefore a practical attack detection system may be developed that have low error rate, high learning rate and quick attack detection by using this approach as well as with other neural networks in the	Nakul Jadeja (19BCE0660)

					form of hybrid architecture.	
3	DDoS Attack Detection and Mitigation Using SDN: Methods, Practises, and Solutions	Arabian Journal for Science and Engineering, February 2017	Software-Defined With this decoupling of control plane and data plane, network switches become simple forwarding devices whereas control logic and functionality are implemented in logically centralised controller	Entropy, Machine Learning, Traffic Learning Analysis, Connection Rate, Integration of SNORT and FLOW	SDN is not a silver bullet solution to all network security problems. In this paper, we also highlighted open research issues, challenges, and recommendations related to SDN-based DDoS attack detection and mitigation that require further research.	Nakul Jadeja (19BCE0660)
4	**DDoS attack detection method using cluster analysis**	Expert Systems with Applications, Volume 34, Issue 3, April 2008	In this paper, we apply cluster analysis to separate each phase of the DDoS attack and identify precursors for detection.	There are two main types of cluster algorithms; hierarchical and partitioning.	This paper analysed algorithm for DDoS attacks included in the 2000 DARPA data set only. It may be desirable to apply the proposed method to different types of DDoS attacks and data sets.	Nakul Jadeja (19BCE0660)
5	A DDoS Attack Detection Method Based on Machine Learning	Conference Series, Volume 1237, Issue 3	and the random forest algorithm is used to train the attack detection model	TFN2K (Tribe Flood Network 2000)	less interpretable than a single decision tree, more memory	Swayam Sharma (19BCE0523)
6	Performance evaluation of Botnet DDoS attack detection using machine learning	Evolutionary Intelligence volume 13, (2020)	Support Vector Machines, Artificial Neural Networks, Naïve Bayes, Decision Tree and Unsupervised Learning	The paper performed an experimental analysis of the machine learning methods for Botnet DDoS attack detection. The evaluation is done on the UNBS-NB 15 and KDD99.	Dataset not big enough	Swayam Sharma (19BCE0523)
7	Real-time DDoS attack detection using FPGA	Computer Communicati	real-time statistical solution to detect	FGPA	The power consumption is	Swayam Sharma

		ons Volume 110, 15 September 2017	DDoS attacks in hardware		more and programmers do not have any control on power optimization in FPGA	(19BCE0523)
8	DDoS Attack Detection and Wavelets	Telecommunication Systems 2005	utilise energy distribution based on wavelet analysis to detect DDoS attack traffic	Wavelet analysis	Developing a proper value for δ, T and W	Swayam Sharma (19BCE0523)
9	Towards Effective Detection of Recent DDoS Attacks: A Deep Learning Approach	Security and Communication Networks, 2021	a lightweight IDS framework, combining the ensemble of feature selection algorithm with DNN classifier, to improve the intrusion detection performance while reducing the system processing overhead	discriminative deep neural network mechanism that makes the binary classification of the traffic sample	the adoption of DL models in certain critical AI applications is limited due to its black-box nature	Sayan Saha (19BCE0510)
10	Early Detection of DDoS Attacks Against Software Defined Network Controllers	Journal of Network and Systems Management, 2018	detect a DDoS attack in its early stages by measuring the randomness or entropy of the incoming packets	concept of entropy to detect abnormal variations in the traffic. Components used: window size and a threshold	an attack against an entire network or multi-controllers' environment may not be detected	Sayan Saha (19BCE0510)
11	Study of detection method for spoofed IP against DDoS attacks	Personal and Ubiquitous Computing, 2018	method of detecting spoofed IPs for IoT devices detects the reliable TCP packets from the normal traffic flowing into the DDoS shelter	controlled flooding, IP traceback, Spoofing Prevention Method (SPM), and Stack Push identificatio	the effectiveness of our proposed scheme to detect a spoofed IP using a DDoS shelter can be further improved	Sayan Saha (19BCE0510)

				n (StackPI)		
12	Detection of DDOS Attack using Deep Learning Model in Cloud Storage Application	Wireless Personal Communications, 2021	A novel and well organized framework is designed for executing DDoS attack detection method by classifying normal and attack packet to get better the DDoS attack detection rate	min–max normalisation, feature selection-whale optimization algorithm deep neural network (FS-WOA–DNN)	IDS schemes can detect individual instantiations but not novel attacks	Sayan Saha (19BCE0510)
13	Detection of DDoS Attack on the Client Side Using Support Vector Machine	Published on 4 March 2019	The DDoS attack will be detected on the SDN network by using the Advanced Support Vector Machine (ASVM) method which generates DDoS attack alerts by considering the application's security requirements.	Leverages the programming and dynamic nature of SDN and implements an adaptive DDoS protection mechanism.	The overall accuracy of the proposed model is at 97%.	Shubh Gupta (20BCI0205)
14	Detection of DNS DDoS Attacks with Random Forest Algorithm on Spark	1 January 2018	The model proposed is finally applied to central recursive DNS servers in ISPs which are closer from the attack sources to reduce the attack traffic on servers of CN. The model could be used to deal with large-scale DNS query flows, which is fast enough to be used in practice.	The classification model is built on spark and performs with 0.0% FPR and 4.36% FNR, which means both the accuracy and performance demands in practice are met.	Feature extraction costs more time than classification. Not very suitable for real-time rules made by the firewall.	Shubh Gupta (20BCI0205)

15	Detection of DDOS For open stack based private cloud.	1st January 2020	Network traffic capture, preparation of training dataset and different machine learning algorithms and Deep Neural Network (DNN) algorithm proposed to detect the DDoS attacks in OpenStack based cloud. Socket programming has been used for network traffic capturing, three machine learning algorithms, namely Naive Bayes, Random Forest, Decision Tree and a DNN algorithm to implement and test the IDS in OpenStack based cloud.	A protocol based packet sniffer is considered which will track packets by attributing itself to a socket in promiscuous mode, packets will be tracked according to the filtering standards set by the network administrator.	The existing firewall in OpenStack did not handle DDoS attacks specifically. Thus a DDoS detection module was required along with the firewall to have a proper security system in OpenStack.	Shubh Gupta (20BCI0205)
16	DDoS Attack Detection Algorithms Based on Entropy Computing	12 december 2007	This entropy detection method is mainly used to calculate the distribution randomness of some attributes in the network packets' headers. These attributes could be the packet's source IP address, TTL value, or some other values indicating the packet's properties.	The formula of entropy calculation is as follows : $H = -\sum_{i=1} p_i \log_2 p$	Cumulative entropy detection method has good detection capability.	Shubh Gupta (20BCI0205)

5. OBSERVATIONS MADE AND GAPS

5.1 Observation made from Literature Review

The criticality of the situation makes this one of the major problems of internet security and many statistical detection methods can be found in our literature papers like wavelet-based, port entropy-based, destination entropy-based etc. However, all these methodologies are very time consuming and ineffective as the internet is a widely dynamic field that is constantly changing. Therefore, to solve these issues, many researchers resorted to Machine Learning and Artificial Intelligence methodologies to detect the DDoS attack.

5.2 Gaps from Literature Review

There is a lack in the literature papers to classify the DDoS attacks into the various types which motivated us to use efficient Machine Learning and Artificial Intelligence techniques to capture the variability of the changing internet domains as it is easy to update the ML and AI models. However, as the dataset is large and has 87 features the computational complexity and the prediction time increase. To tackle that we applied feature selection – Extra Trees classifier technique to select the best relevant 20 features.

6. PROPOSED MODEL

6.1. Software Requirements

- **Operation System:** Linux- Ubuntu 16.04 to 17.10, or Windows 7 to 10
- **Chrome Version:** 79.0 or above or Safari Version 11.1.2 or upgraded
- **IDLE:** Python v3.7 (preferable) along with the use of popular libraries such as Numpy and Pandas Python modules
- **AI Models:** Keras is used as the application layer and Tensorflow library is used as backend support on Python

6.2. Hardware Requirements

- **Processor:** x86 64-bit CPU (Intel / AMD architecture)
- **Ram:** 2 GB or more

6.3 Selected Algorithms

At first, we will select different algorithms on the basis of their Computational Complexity in order to detect DDoS Cyber Threats.

The table below shows the Calculated Complexity of each of the algorithms that we will be using in this project.

Algorithm	Training Complexity	Prediction Complexity
SVM	$\mathcal{O}(n^2\ f\ +\ n^3)$	$\mathcal{O}(n_{sv}\ f)$
Decision Tree	$\mathcal{O}(n^2\ f)$	$\mathcal{O}(f)$
XGBoost	$\mathcal{O}(n\ f\ n_{trees})$	$\mathcal{O}(f\ n_{trees})$
Random Forest	$\mathcal{O}(n^2\ f\ n_{trees})$	$\mathcal{O}(f\ n_{trees})$
AdaBoost	$\mathcal{O}(n\ f)$	$\mathcal{O}(f\ n_{trees})$
Neural Network	-	$\mathcal{O}(fn_{l_1} + n_{l_1}n_{l_2} + ...)$

Here n is the number of training samples, f being the total number of features in each of the training samples. n_{sv} is the number of support vectors used in the SVM algorithm. n_{li} is the number of neurons in layer i for a given Neural Network.

6.4 Dataset

The dataset that we use in this study is **CICDDoS**, this dataset is an improvement on its predecessor and improves most of the shortcomings. The main benefit of using this dataset is that it has analysed new attacks that are mainly carried out using the TCP/UDP protocols using the Application layer.It is divided into different categories based on the protocols used. It has modern reflective attacks such as PortMap, NetBios, LDAP, MSSQL, UDP, UDP-lag, SYN, NTP, DNS and SNMP.

Using this dataset, we will perform multi-class classification for AI and ML algorithms. For reducing the dataset while maintaining the integrity of data along with the distribution, **Scikit-learn** Python Library is used.

The multiclass classification for DDoS Cyber Threat will be performed by using different AI and ML algorithms and each of the threats was individually identified and validated by using different metrics.

We will then analyze and calculate the accuracy score of each algorithm on the given dataset for DDoS detection using **Receiving Operating Characteristic Curve** (RoC).

6.5 List of Modules

Machine Algorithms:

- **Support Vector Machines (SVMs)** are a set of supervised learning algorithms that are majorly used for classification and regression. SVM separates different classes by a hyperplane and produces a classifier that works well on unseen examples which is why it generalizes very well. The hyperparameters were tuned in order to prevent overfitting. Square-hinge was used as the loss function because this function uses simple mathematics which gives computationally effective results. A large value of C was not taken to prevent overfitting and if C was taken too low that is lesser than 1 then it was leading to soft margin. For penalty, l2 normalization was used in penalization as l1 normalization resulted in too sparse coefficients.
- **Decision Tree Decision Trees** are supervised learning algorithms that sort the tree starting from the root node to a leaf node. The leaf node gives the classification that is the label name. Decision trees use hyperplanes/ axis-parallel rectangles that divide the feature space into the classification. Decision Trees are not prone to outliers so lesser data processing is needed. It is used as a baseline benchmark for other algorithms. While implementing Decision Trees we used Gini Index as the splitting criteria and 3 as the sample split value. The best split is used as the splitting strategy at each node. Pruning was not performed to maintain the cost complexity. The features considered to look for the best split were taken to be equal to the maximum number of features.

Ensemble Learning Algorithms:

- **Random Forest** classifier is a set of decision trees that are randomly selected from a subset of the training set and then the vote is aggregated from all the decision trees randomly and the final class of the object tested is given. This classifier is mainly used as it is quite efficient with big datasets, it can also handle a large number of input variables without removing any variables. Besides, it also avoids overfitting by improving the accuracy score while training on the dataset. Additionally, as the forest building happens it produces unbiased generalization error estimates. The parameters which give the best accuracy score

for this classifier are: 100 number of estimators, minimum sample leaves as 1, minimum sample split as 2 and the Gini criterion is used to measure the quality of the split.

- **Majority Vote Classifier** is an Ensemble learning technique that selects the class label that has been predicted by the majority of the classifiers if a class label has received more than 50% of the votes. We combine the top 4 performing classifiers and get the best approximation for the different class labels for different DDoS threats. This combination of the top 4 classifiers is called MV-4 classifier in this study mainly to separate it from other classifiers. We combine these four Classifiers using Majority Voting Technique to come up with an MV-4 classifier. To achieve the combination of 4 different algorithms the code was developed in Python3.7 from scratch.

7. RESULTS AND DISCUSSION

7.1 Implementation Details

For implementing our project we will follow the following steps:-

1. **DATA PREPROCESSING**
    a) DATA CLEANING:

```
In [9]: data_real = data.replace(np.inf, np.nan)

In [10]: data_real.isnull().sum().sum()
Out[10]: 23612

In [11]: data_df = data_real.dropna(axis=0)

In [12]: data_df.isnull().sum().sum()
Out[12]: 0

In [13]: data_df
```

- data.replace(np.inf, np.nan):- dropping infinite values from our dataset
- dataframe.isnull().sum().sum() returns the number of missing values in the data set.
- Drop rows which contain missing values.=data_real.dropna(axis=0)
- Then again return the null values and We see that the output obtained is 0
- Next, we remove the columns with similar HTTP

```
In [15]: data_X = data_df.drop([' Label', 'SimillarHTTP'], axis = 1)

In [16]: data_X.columns
```

b) LABEL ENCODING:

Label Encoding refers to converting the labels into a numeric form so as to convert them into the machine-readable form. Machine learning algorithms can then decide in a better way how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.

Label Encoding for the Dataset

```
In [23]: from sklearn.preprocessing import LabelEncoder

In [24]: le = LabelEncoder()

In [25]: data_y_trans = le.fit_transform(data_y)

In [26]: data_y_trans
Out[26]: array([ 1,  1,  1, ..., 10, 10, 10])

In [27]: le_fid = LabelEncoder()

In [28]: le_fid.fit(data_X['Flow ID'])
         data_X['Flow ID'] = le_fid.fit_transform(data_X['Flow ID'])

In [29]: le_SIP = LabelEncoder()

In [30]: le_SIP.fit(data_X[' Source IP'])
         data_X[' Source IP'] = le_SIP.fit_transform(data_X[' Source IP'])

In [31]: le_DIP = LabelEncoder()

In [32]: le_DIP.fit(data_X[' Destination IP'])
         data_X[' Destination IP'] = le_DIP.fit_transform(data_X[' Destination IP'])

In [33]: le_timestamp = LabelEncoder()
         le_timestamp.fit(data_X[' Timestamp'])
         data_X[' Timestamp'] = le_timestamp.fit_transform(data_X[' Timestamp'])

In [34]: data_X
```

```
In [34]: data_X
```

Out[34]:

	Flow ID	Source IP	Source Port	Destination IP	Destination Port	Protocol	Timestamp	Flow Duration	Total Fwd Packets	Total Backward Packets	Total Length of Fwd Packets	Total Length of Bwd Packets	Fwd Packet Length Max	Fwd Packet Length Min	Fwd Packet Length Mean	Pa Le
0	205418	17	556	129	16923	17	153807	1.0	2.0	0.0	2944.0	0.0	1472.0	1472.0	1472.0	
1	356277	17	690	129	38772	17	149038	1.0	2.0	0.0	2944.0	0.0	1472.0	1472.0	1472.0	
2	390438	17	761	129	24941	17	149918	1.0	2.0	0.0	2944.0	0.0	1472.0	1472.0	1472.0	
3	172900	17	526	129	22632	17	139638	1.0	2.0	0.0	2944.0	0.0	1472.0	1472.0	1472.0	
4	194820	17	546	129	5185	17	143350	1.0	2.0	0.0	2944.0	0.0	1472.0	1472.0	1472.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
585965	38260	17	26917	129	29675	6	555528	104.0	2.0	2.0	0.0	0.0	0.0	0.0	0.0	
585966	54237	17	33856	129	60933	17	514638	2.0	2.0	0.0	750.0	0.0	375.0	375.0	375.0	
585967	11899	17	10655	129	18473	6	552548	50.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	
585968	309616	17	63789	129	3373	6	534101	33957023.0	6.0	0.0	0.0	0.0	0.0	0.0	0.0	
585969	292092	17	62675	129	62675	6	549955	16621605.0	6.0	0.0	0.0	0.0	0.0	0.0	0.0	

574164 rows × 85 columns

We still have a lot of rows and columns in our so we will now perform **Data Reduction using Feature Classifier**

c) FEATURE CLASSIFIER:

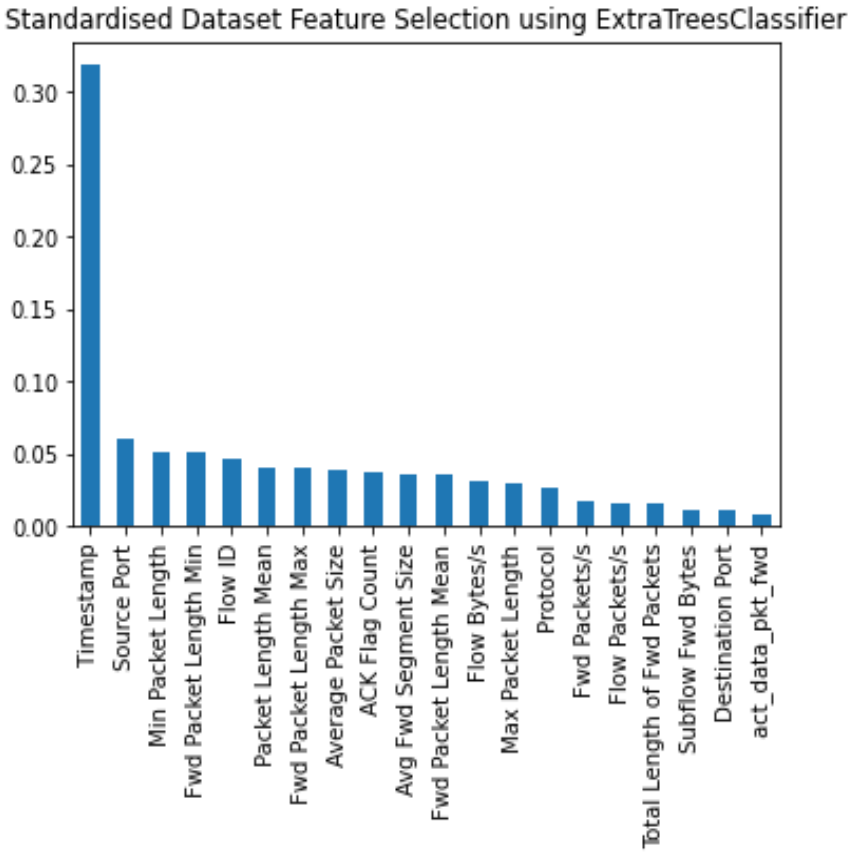
- Feature selection includes the use on ExtraTressClassifier.
- sklearn.ensemble.ExtraTreesClassifier:- This class implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

Feature Selection

In [36]:
```
from sklearn.feature_selection import chi2
from sklearn.feature_selection import SelectKBest
from sklearn.ensemble import ExtraTreesClassifier

#selecting 20 best features
# select_best= SelectKBest(chi2, k=20)
# X_feat_20 = select_best.fit_transform(data_X, data_y_trans)
# X_feat_20.shape

model = ExtraTreesClassifier(random_state=42)
model.fit(data_X, data_y_trans)
```

Out[36]:
```
ExtraTreesClassifier(bootstrap=False, ccp_alpha=0.0, class_weight=None,
                     criterion='gini', max_depth=None, max_features='auto',
                     max_leaf_nodes=None, max_samples=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators=100,
                     n_jobs=None, oob_score=False, random_state=42, verbose=0,
                     warm_start=False)
```

- Parameters:
- 1. n_estimators
- 2. gini=gain
- 3.split: min no of samples to split an internal node



- Now we split the dataset into 3 halves namely: training dataset, validation dataset and test set.
- One of the key aspects of supervised machine learning is model evaluation and validation. When you evaluate the predictive performance of your model, it's essential that the process be unbiased. Using train_test_split() from the data science library scikit-learn, you can split your dataset into subsets that minimize the potential for bias in your evaluation and validation process.

2. RANDOM FOREST CLASSIFICATION: -

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max_samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree.

```
In [52]: from sklearn.ensemble import RandomForestClassifier
         rf = RandomForestClassifier()
         rf.fit(X_train_std_20, y_train_20)

Out[52]: RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                                criterion='gini', max_depth=None, max_features='auto',
                                max_leaf_nodes=None, max_samples=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, n_estimators=100,
                                n_jobs=None, oob_score=False, random_state=None,
                                verbose=0, warm_start=False)

In [53]: rf_y_pred = rf.predict(X_test_std_20)

In [54]: rf_y_pred

Out[54]: array([ 4,  6,  4, ..., 10,  2, 10])

In [55]: from sklearn.metrics import accuracy_score
         from sklearn.metrics import classification_report
         from sklearn.metrics import confusion_matrix
```

3. DECISION TREE: -

It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

```
In [62]: from sklearn.tree import DecisionTreeClassifier
         dt = DecisionTreeClassifier()
         dt.fit(X_train_std_20, y_train_20)

Out[62]: DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                                max_depth=None, max_features=None, max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, presort='deprecated',
                                random_state=None, splitter='best')

In [63]: dt_y_pred = dt.predict(X_test_std_20)
```

4. SUPPORT VECTOR MACHINES(SVMS): -

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

```
In [68]: from sklearn.svm import LinearSVC

In [69]: svm = LinearSVC(multi_class = 'ovr')
         svm.fit(X_train_std_20, y_train_20)

         /usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:947: ConvergenceWarning: Liblinear failed to converge, increase the
         number of iterations.
           "the number of iterations.", ConvergenceWarning)

Out[69]: LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
                   intercept_scaling=1, loss='squared_hinge', max_iter=1000,
                   multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
                   verbose=0)

In [70]: y_pred_svm = svm.predict(X_test_std_20)

In [71]: svm.score(X_test_std_20, y_test_20)

Out[71]: 0.9275355587808418
```

5. NAIVE BYES: -

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features,

all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as ‘Naive’.

```
In [77]: from sklearn.naive_bayes import GaussianNB

In [78]: gnb = GaussianNB()
         gnb.fit(X_train_std_20, y_train_20)
         gnb_y_pred = gnb.predict(X_test_std_20)
```

6. ENSEMBLE METHOD OF MACHINE LEARNING: -

An ensemble is a machine learning model that combines the predictions from two or more models. The models that contribute to the ensemble, referred to as ensemble members, may be the same type or different types and may or may not be trained on the same training data.

Adaboost:-

What this algorithm does is that it builds a model and gives equal weights to all the data points. It then assigns higher weights to points that are wrongly classified. Now all the points which have higher weights are given more importance in the next model. It will keep training models until and unless a lowe error is received.

```
In [115]: from sklearn.pipeline import Pipeline
          from sklearn.ensemble import AdaBoostClassifier
          from sklearn.model_selection import cross_val_score

In [116]: # ADABOOST
          adaboost = AdaBoostClassifier(base_estimator= dt, n_estimators=100)

In [117]: adaboost.fit(X_train_std_20, y_train_20)
```

7. MAJORITY VOTING CLASSIFIER: -

A voting ensemble (or a “majority voting ensemble”) is an ensemble machine learning model that combines the predictions from multiple other models.

It is a technique that may be used to improve model performance, ideally achieving better performance than any single model used in the ensemble.

A voting ensemble works by combining the predictions from multiple models. It can be used for classification or regression. In the case of regression, this involves calculating the average of the predictions from the models. In the case of classification, the predictions for each label are summed and the label with the majority vote is predicted.

```
In [ ]: mv_clf2 = MajorityVoteClassifier(classifiers = [rf, adaboost, dt, gradinet_boost], vote='classlabel')

In [ ]: # Train the classifiers
        mv_clf2.fit(X_train_std_20, y_train_20)
```

7.2 Performance Evaluation

The metrics used for evaluating the Artificial Intelligence and Machine Learning models are as mentioned below:

sklearn.metrics.classification_report:-Build a text report showing the main classification metrics

sklearn.metrics.confusion_matrix:- By definition a confusion matrix C(i,j) is such that is equal to the number of observations known to be in group i and predicted to be in group j

sklearn.metrics.accuracy_score:- In multilabel classification, this function computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in y_true.

Accuracy Score: The accuracy score gives the measure of closeness to a specific value. The formula for Accuracy score as given in Sci-Kit learn manual is as given below:

$$\text{Accuracy (y, y')} = \frac{1}{nsamples} \sum_{i=0}^{nsamples-1}$$

In the equation, y is the true label whereas y' is the predicted label which is given by an algorithm after prediction. For multilabel classification, the Accuracy Score gives us the accuracy of the subset. When the entire set of the predicted values given by the algorithm matches the true label the accuracy score is given as 1.0 otherwise it is 0.0.

F1-Score: F1-Score which is also known as F-Score or F-Measure. It is a measure of the Test set accuracy. F-Measure calculates the harmonic mean of precision(P) and recall(R) to find the score. The formula for F1-Score is given by:

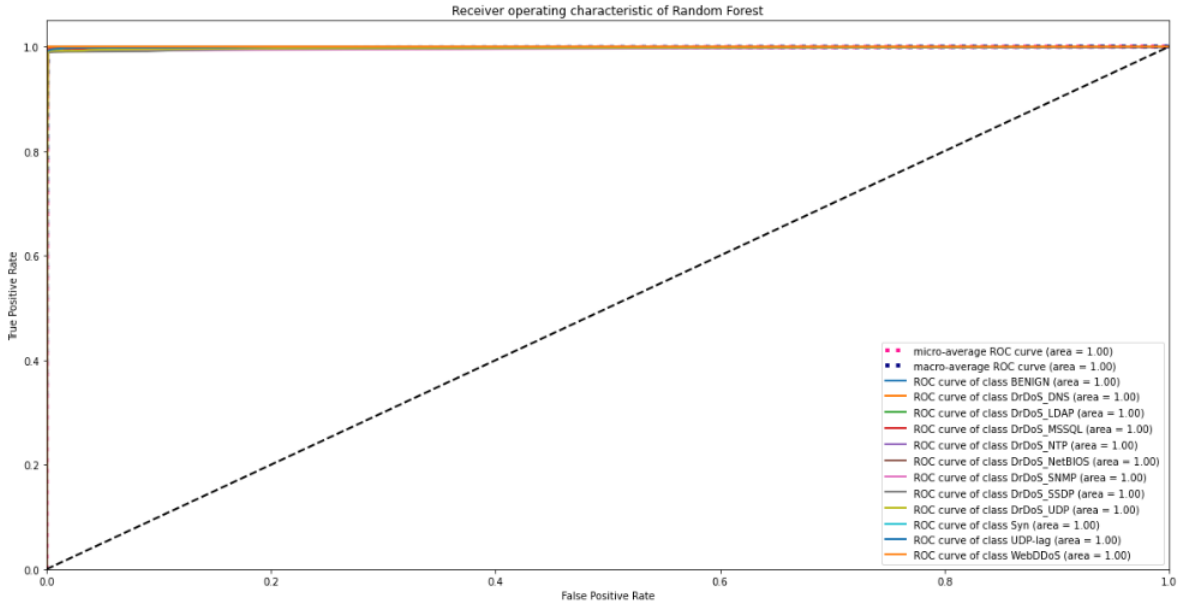
$$\text{F1 Score} = 2 * \frac{P * R}{P + R}$$

The maximum value attained by F1-score is 1 and it is also known as the Dice similarity coefficient.

Receiver Operating Characteristic Curve: The Receiver Operating Characteristic curve (RoC) is a useful tool to evaluate any models for classification based on their performance by considering the False Positive Rate (FPR) and True Positive Rate (TPR). The values of the TPR and FPR are computed by shifting the decision threshold of the classifier. On the RoC curve, the TPR feature lies on the Y-axis whereas the FPR feature lies on the X-axis. The ideal point for the classifier is at the top left corner where the FPR is zero and TPR is 1 making the classifier ideal for the problem. The larger the area under the curve the better the performance of the classifier.

7.3 Results and Analysis

1. Random Forest Classifier:




```
[[  615     1     0     0     0     0     0     0     0     0     0     0]
[    1 14656     0     0    36     6     0     0     0     0     0     0]
[    3   207 15779     3     0     0     0     0     0     0     0     0]
[    2     0    82 16932     1     0     0     0     0     0     0     0]
[   24     0     0     0 17862     0     0     0     0     0     0     0]
[    2    34     0   158     1 15333     1     0     0     0     0     0]
[    5     2     0     0     0   184 19942     0     0     0     0     0]
[    1     0     0     0     0     2   182 15376     1     0     0     0]
[    2     0     0     0     0     1     0   170 18391     0     1     0]
[    4     0     0     0     0     0     0     0     0 16336     0     0]
[    5     0     0     0     0     0     0     0   185     0 19693     0]
[   17     0     0     0     0     0     0     0     0     0     1    10]]
```

```
In [58]: acc_score = accuracy_score(y_test_20, rf_y_pred)
         print("Accuracy Score for Random_Forest: \n", acc_score*100)

         Accuracy Score for Random_Forest:
          99.23076923076923
```

2. Decision Tree:

In [64]:
```
print("Classification Report for Decision Tree: \n", classification_report(le.inverse_transform(y_test_20), le.inverse_transform(
```

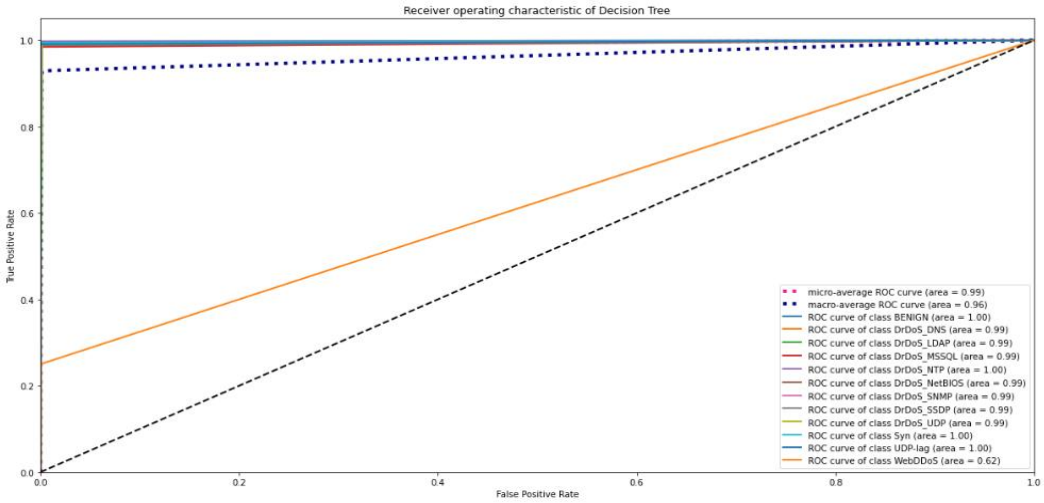
```
Classification Report for Decision Tree:
                precision    recall  f1-score   support

        BENIGN       0.82      1.00      0.90       616
     DrDoS_DNS       0.99      0.99      0.99     14699
    DrDoS_LDAP       0.98      0.99      0.99     15992
   DrDoS_MSSQL       0.99      0.98      0.99     17017
     DrDoS_NTP       0.99      1.00      0.99     17886
DrDoS_NetBIOS       0.99      0.99      0.99     15529
    DrDoS_SNMP       0.99      0.99      0.99     20133
    DrDoS_SSDP       0.99      0.99      0.99     15562
     DrDoS_UDP       0.99      0.99      0.99     18565
           Syn       1.00      0.99      1.00     16340
       UDP-lag       1.00      0.99      0.99     19883
       WebDDoS       0.78      0.25      0.38        28

      accuracy                           0.99    172250
     macro avg       0.96      0.93      0.93    172250
  weighted avg       0.99      0.99      0.99    172250
```

In [65]:
```
dt_conf_mat = confusion_matrix(y_test_20, dt_y_pred)
print("Decision Tree Confusion: \n", dt_conf_mat)
```

```
Decision Tree Confusion:
 [[  614     0     0     0     0     0     0     0     0     0     2     0]
 [    0 14502     0     0   197     0     0     0     0     0     0     0]
 [    0   204 15788     0     0     0     0     0     0     0     0     0]
 [    0     0   268 16749     0     0     0     0     0     0     0     0]
 [   38     0     0     0 17831     0     0     0     0    17     0     0]
 [    1     0     0   159     0 15369     0     0     0     0     0     0]
 [    0     0     0     0     0   186 19947     0     0     0     0     0]
 [    0     0     0     0     0     0   181 15381     0     0     0     0]
 [    0     0     0     0     0     0     0   168 18397     0     0     0]
 [   96     0     0     0     0     0     0     0     0 16244     0     0]
 [    0     0     0     0     0     0     0     0   183     0 19698     2]
 [    0     0     0     0     0     0     0     0     0     0    21     7]]
```

In [66]:
```
acc_score_dt = accuracy_score(y_test_20, dt_y_pred)
print("Accuracy Score for Decision Tree: \n", acc_score_dt*100)
```

```
Accuracy Score for Decision Tree:
 98.99970972423803
```



3. SVM:

In [72]:
```
print("Classification Report for: \n", classification_report(le.inverse_transform(y_test_20), le.inverse_transform(y_pred_svm)))
```

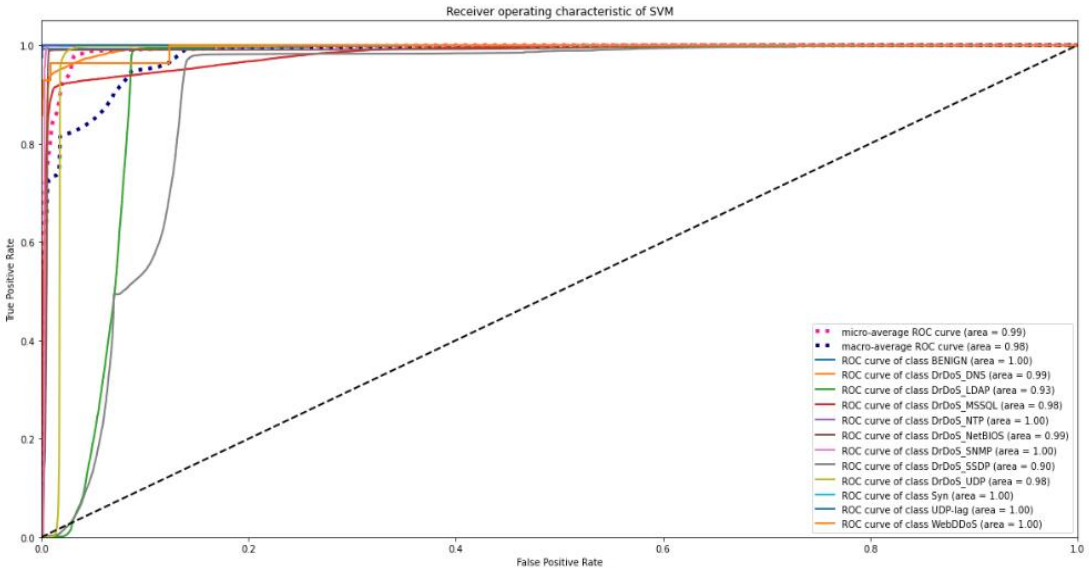
```
Classification Report for Random Forest:
                precision    recall  f1-score   support

        BENIGN       0.95      0.86      0.90       616
     DrDoS_DNS       0.90      0.89      0.90     14699
    DrDoS_LDAP       0.94      0.95      0.95     15992
   DrDoS_MSSQL       0.86      0.93      0.89     17017
     DrDoS_NTP       1.00      0.96      0.98     17886
 DrDoS_NetBIOS       0.87      0.99      0.93     15529
    DrDoS_SNMP       0.98      0.96      0.97     20133
    DrDoS_SSDP       0.99      0.68      0.80     15562
     DrDoS_UDP       0.80      1.00      0.89     18565
           Syn       1.00      1.00      1.00     16340
      UDP-lag        1.00      0.90      0.95     19883
       WebDDoS       0.48      0.57      0.52        28

      accuracy                           0.93    172250
     macro avg       0.90      0.89      0.89    172250
  weighted avg       0.94      0.93      0.93    172250
```

In [73]:
```
svm_conf_mat = confusion_matrix(y_test_20, y_pred_svm)
print("SVM Confusion Matrix: \n", svm_conf_mat)
```

```
SVM Confusion Matrix:
 [[  530     0     0    44     6     0     0     0     0    21    13     2]
 [    0 13108   805   348    41   397     0     0     0     0     0     0]
 [    1   730 15226    19     2    12     0     2     0     0     0     0]
 [    1     0   178 15853     0   967     1    16     0     0     0     1]
 [    7   685     2    39 17136     7     0     2     0     8     0     0]
 [    0     0     2   143     0 15380     0     3     0     0     0     1]
 [    0     0     0     3     0   883 19242     1     0     0     0     4]
 [    1     0     0  2059     0    11   273 10518  2700     0     0     0]
 [    1     0     0     0     0     4    32    49 18477     0     0     2]
 [    9     0     0    17     0     0     0     0     0 16314     0     0]
 [    1     0     0     0     0     0    10     0  1897     0 17968     7]
 [    8     0     0     0     0     0     0     0     0     0     4    16]]
```

In [74]:
```
acc_score_svm = accuracy_score(y_test_20, y_pred_svm)
print("Accuracy Score for SVM: \n", acc_score_svm*100)
```

```
Accuracy Score for SVM:
 92.75355587808419
```



4. Naive Bayes:

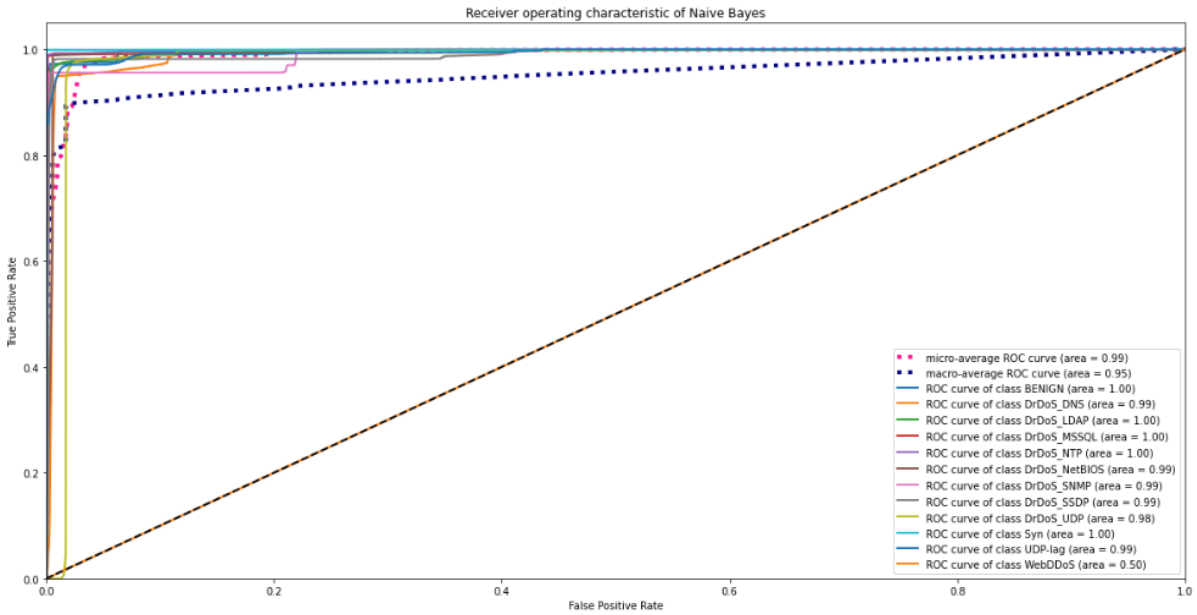
In [79]:
```
print("Classification Report for Naive Bayes: \n", classification_report(le.inverse_transform(y_test_20), le.inverse_transform(gr
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedMetricWarning: Precision and F-score a
re ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavio
r.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
Classification Report for Naive Bayes:
                precision    recall  f1-score   support

        BENIGN       0.61      0.96      0.75       616
     DrDoS_DNS       0.63      0.02      0.04     14699
    DrDoS_LDAP       0.54      0.99      0.70     15992
   DrDoS_MSSQL       0.93      0.99      0.96     17017
     DrDoS_NTP       0.99      0.98      0.99     17886
DrDoS_NetBIOS       0.95      0.99      0.97     15529
    DrDoS_SNMP       0.98      0.96      0.97     20133
    DrDoS_SSDP       0.95      0.31      0.47     15562
     DrDoS_UDP       0.58      0.99      0.73     18565
           Syn       1.00      1.00      1.00     16340
       UDP-lag       1.00      0.85      0.92     19883
       WebDDoS       0.00      0.00      0.00        28

      accuracy                           0.82    172250
     macro avg       0.76      0.75      0.71    172250
  weighted avg       0.86      0.82      0.79    172250
```

In [81]:
```
acc_score_gnb = accuracy_score(y_test_20, gnb_y_pred)
print("Accuracy Score for Naive: \n", acc_score_gnb*100)
```

```
Accuracy Score for Naive:
 82.47721335268506
```



5. Adaboost:

In [80]:
```
gnb_conf_mat = confusion_matrix(y_test_20, gnb_y_pred)
print("Naive Bayes Confusion Matrix: \n", gnb_conf_mat)
```

```
Naive Bayes Confusion Matrix:
 [[  594     0     0     0     0     0     0     0     0    14     8     0]
 [  184   332 13324   694   163     2     0     0     0     0     0     0]
 [    3   176 15770    43     0     0     0     0     0     0     0     0]
 [    3     5   153 16832     1     0     9    14     0     0     0     0]
 [  143     9     0   162 17572     0     0     0     0     0     0     0]
 [    4     0     1   167     0 15342     1    14     0     0     0     0]
 [   11     5     0    10     0   870 19233     4     0     0     0     0]
 [    6     0     0     2     2     0   273  4831 10448     0     0     0]
 [    5     0     0     1     0     0    28   206 18321     0     4     0]
 [    9     0     0    17     0     0     0     0     0 16314     0     0]
 [    4     0     0   152     0     0   123    30  2648     0 16926     0]
 [    0     0     0     0     0     0     0     0     0     0    28     0]]
```

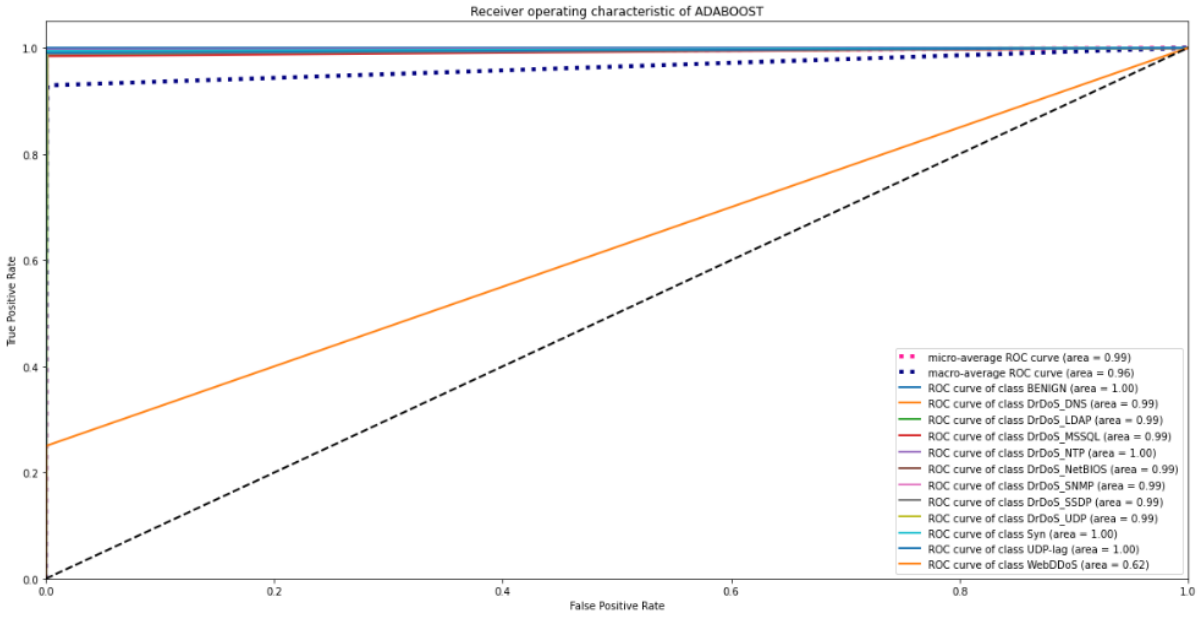
In [120]:
```
print("Classification Report for Adaboost: ",classification_report(le.inverse_transform(y_test_20), le.inverse_transform(y_pred_a
```

```
Classification Report for Adaboost:                precision    recall  f1-score   support

       BENIGN       0.82      1.00      0.90       616
    DrDoS_DNS       0.99      0.99      0.99     14699
   DrDoS_LDAP       0.98      0.99      0.99     15992
  DrDoS_MSSQL       0.99      0.98      0.99     17017
    DrDoS_NTP       0.99      1.00      0.99     17886
DrDoS_NetBIOS       0.99      0.99      0.99     15529
   DrDoS_SNMP       0.99      0.99      0.99     20133
   DrDoS_SSDP       0.99      0.99      0.99     15562
    DrDoS_UDP       0.99      0.99      0.99     18565
          Syn       1.00      0.99      1.00     16340
      UDP-lag       1.00      0.99      0.99     19883
      WebDDoS       0.44      0.25      0.32        28

     accuracy                           0.99    172250
    macro avg       0.93      0.93      0.93    172250
 weighted avg       0.99      0.99      0.99    172250
```

In [122]:
```
# Confusion Matrix
adaboost_conf_mat = confusion_matrix(y_test_20, y_pred_adaboost)
print("Adaboost Confusion: \n", adaboost_conf_mat)
```

```
Adaboost Confusion:
 [[  616     0     0     0     0     0     0     0     0     0     0     0]
 [    0 14502     0     0   197     0     0     0     0     0     0     0]
 [    0   204 15788     0     0     0     0     0     0     0     0     0]
 [    0     0   268 16749     0     0     0     0     0     0     0     0]
 [   38     0     0     0 17831     0     0     0     0    17     0     0]
 [    1     0     0   159     0 15369     0     0     0     0     0     0]
 [    0     0     0     0     0   186 19947     0     0     0     0     0]
 [    0     0     0     0     0     0   181 15381     0     0     0     0]
 [    0     0     0     0     0     0     0   168 18397     0     0     0]
 [   96     0     0     0     0     0     0     0     0 16244     0     0]
 [    0     0     0     0     0     0     0     0   183     0 19691     9]
 [    0     0     0     0     0     0     0     0     0     0    21     7]]
```

```
In [119]: print("Accuracy Score for Adaboost: ", accuracy_score(y_test_20, y_pred_adaboost))

          Accuracy Score for Adaboost:  0.9899680696661829
```



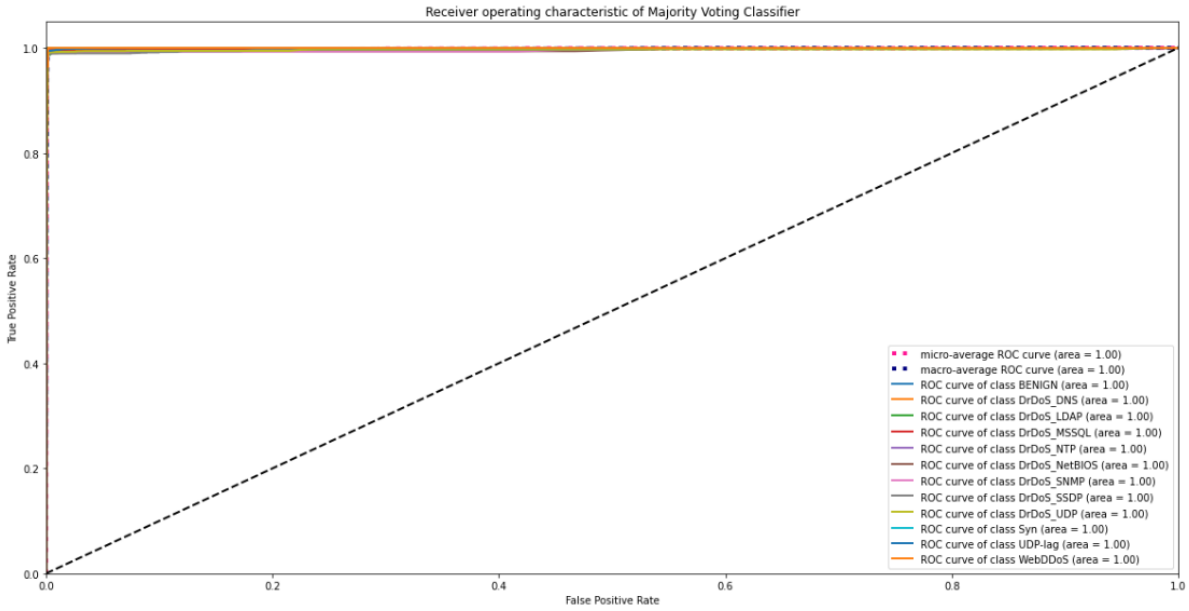
6. Majority Voting Classifier:

In [ ]:
```
# Confusion Matrix
majorityvoting_conf_mat = confusion_matrix(y_test_20, y_pred_mjv)
print("Majority Voting Confusion: \n", majorityvoting_conf_mat)
```

```
Majority Voting Confusion:
 [[  616     0     0     0     0     0     0     0     0     0     0     0]
 [    0 14502     0     0   197     0     0     0     0     0     0     0]
 [    0   204 15788     0     0     0     0     0     0     0     0     0]
 [    0     0   268 16749     0     0     0     0     0     0     0     0]
 [   38     0     0     0 17844     0     0     0     0     4     0     0]
 [    1     0     0   159     0 15369     0     0     0     0     0     0]
 [    0     0     0     0     0   186 19947     0     0     0     0     0]
 [    0     0     0     0     0     0   181 15381     0     0     0     0]
 [    0     0     0     0     0     0     0   168 18397     0     0     0]
 [   96     0     0     0     0     0     0     0     0 16244     0     0]
 [    3     0     0     0     0     0     0     0   183     0 19695     2]
 [    3     0     0     0     0     0     0     0     0     0    21     4]]
```

In [ ]:
```
# Classification Accuracy for Majority Voting
print("Accuracy Score for Majority Voting : ", accuracy_score(y_test_20, y_pred_mjv))
```

```
Accuracy Score for Majority Voting :  0.9900493468795356
```



8. CONCLUSION AND FUTURE WORK

The multiclass classification for DDoS Cyber Threat was performed by using different AI and ML algorithms and each of the threats was individually identified and validated by using different metrics. An Ensemble Classifier MV-4 was presented for multiclass DDoS Cyber Threat detection which has an accuracy score of 99.01%. In addition, a comprehensive study of different AI and ML algorithms was performed for DDoS multiclass Cyberthreat detection and among all the algorithms Random Forest Classifier has the highest accuracy score of 99.24% followed by MV-4 and AdaBoost Classifier both of which have an accuracy score of 99.01%. However, the F1-Score and results from the RoC curve show that AdaBoost lags behind Random Forest and MV-4 in terms of not detecting a few threats correctly. The F1-Score of Random Forest and MV-4 is very similar to Random Forest having a slight advantage as it detects 3 threats perfectly.

As the detection of different types of DDoS threats is successfully implemented which is the main aim of this paper. For future work, our goal is to deploy different solutions for each of the attack types to defend the network from such attacks by using AI and ML algorithms. This work will further be extended to develop a system that can successfully detect DDoS Cyberthreats and deploy countermeasures to prevent critical CyberSecurity threats.

9. REFERENCES

1) Filho, L. F. S. D. (2019, October 13). *Smart Detection: An Online Approach for DoS/DDoS Attack Detection Using Machine Learning*. https://www.hindawi.com/journals/scn/2019/1574749/

2) *DDoS attack detection based on neural network*. (2010, November 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/5670479

3) Bawany, N. Z. (2017, February 2). *DDoS Attack Detection and Mitigation Using SDN: Methods, Practices, and Solutions*. SpringerLink. https://link.springer.com/article/10.1007/s13369-017-2414-5?error=cookies_not_supported&code=56bc52bf-d0eb-42dd-84b9-3cf636966493

4) *DDoS attack detection method using cluster analysis*. (2008, April 1). ScienceDirect. https://www.sciencedirect.com/science/article/abs/pii/S0957417407000395#

5) *Journal of Physics: Conference Series*. (2019, January 1). Machine Leanin: Conference Series. https://iopscience.iop.org/article/10.1088/1742-6596/1237/3/032040

6) Tuan, T. A. (2019, November 20). *Performance evaluation of Botnet DDoS attack detection using machine learning*. SpringerLink. https://link.springer.com/article/10.1007/s12065-019-00310-w?error=cookies_not_supported&code=d3bcba53-499f-4155-9cb5-33739db7de59

7) *Real-time DDoS attack detection using FPGA*. (2017, September 15). ScienceDirect. https://www.sciencedirect.com/science/article/abs/pii/S0140366416306442

8) Li, L. (2005, March 1). *DDoS Attack Detection and Wavelets*. SpringerLink. https://link.springer.com/article/10.1007/s11235-004-5581-0?error=cookies_not_supported&code=16bc9d94-4a32-432d-a62c-bbbabc7cd9f2

9) Lopes, O. I. (2021, November 30). *Towards Effective Detection of Recent DDoS Attacks: A Deep Learning Approach*. Security and Communication Network. https://www.hindawi.com/journals/scn/2021/5710028/

10) Mousavi, S. M. (2017, September 30). *Early Detection of DDoS Attacks Against Software Defined Network Controllers*. SpringerLink. https://link.springer.com/article/10.1007/s10922-017-9432-1?error=cookies_not_supported&code=08d5daf5-2a58-405b-8152-59bc3064bf4b

11) Lee, Y. (2017, December 29). *Study of detection method for spoofed IP against DDoS attacks*. SpringerLink. https://link.springer.com/article/10.1007/s00779-017-1097-y?error=cookies_not_supported&code=3d195a80-c086-44da-8fe9-b041d9163525

12) Agarwal, A. (2021, March 13). *Detection of DDOS Attack using Deep Learning Model in Cloud Storage Application*. SpringerLink. https://link.springer.com/article/10.1007/s11277-021-08271-z?error=cookies_not_supported&code=8b7656be-562e-4bc2-b311-6a6b1d7058b2

17) Oo, M. M. (2019b, March 4). *Advanced Support Vector Machine- (ASVM-) Based Detection for Distributed Denial of Service (DDoS) Attack on Software Defined Networking (SDN)*. Hinawi.

https://www.hindawi.com/journals/jcnc/2019/8012568/

18) *Detection of DNS DDoS Attacks with Random Forest Algorithm on Spark*. (2018c, January 1). ScienceDirect. https://www.sciencedirect.com/science/article/pii/S1877050918311426

19) *Distributed Denial of Service (DDoS) Attacks Detection System for OpenStack-based Private Cloud*. (2020, January 1). ScienceDirect. https://www.sciencedirect.com/science/article/pii/S1877050920307481

20) Li, L. (2007, December 12). *DDoS Attack Detection Algorithms Based on Entropy Computing*. SpringerLink. https://link.springer.com/chapter/10.1007/978-3-540-77048-0_35?error=cookies_not_supported&code=dcec3ae0-863d-4478-9a5d-a48c7ae91115#:%7E:text=The%20entropy%20detection%20method%20is,in%20the%20network%20packets'%20headers.&text=Experiment%20results%20show%20that%20these,accurate%20and%20effective%20DDoS%20detection.