An Ensemble Machine Learning Approach For Network Intrusion Detection On An Ensemble Of CIC-IDS2017 And CSE-CIC-IDS2018 Datasets

Vishal Sharma*, Ramanujam N†, Shubham Dhingra‡, Paras Prakash§ and Lakshay Arora¶
Department of Computer Science and Engineering, Guru Gobind Singh Indraprastha University
New Delhi,India
Email: *vishal.sharma@bharatividyapeeth.edu.in, †pnramanujam9@gmail.com,
‡dhingra.shubham37@gmail.com,
§parasprakash98@gmail.com,
§aroralakshay2014@gmail.com

***Abstract*—Security and privacy threats are an ever-rising trend nowadays. Malicious Hackers are trying to infiltrate into toughest of defences by using advanced threat vectors. In a hostile environment, an IDS can be the best detection and prevention option. The dataset which we used, is a combination of CIC-IDS-2017 and CIC-IDS-2018. We performed both multiclass and binary classification to identify whether there is a malicious attempt or attack, and if there is then to identify the type of attack. Statistical features are generated from the dataset and their correlation is measured after extensive visualization. The implemented model uses a stacking ensemble machine learning model that dwells and develops its learning process on the results provided by the previous algorithm. The algorithms used in stacking are Logistic Regression, Decision Tree, Random Forest, Support Vector Machine and the XGBoost. This tool can work as a standalone protection mechanism as a fully comprehensive IDS system. The implemented model achieved an accuracy of 98.2% for binary classification and 98.1% for multi-class classification with very few false positive instances.**

***Index Terms*—NIDS, ML, IoT, Cyber-Security**

I. Introduction

Network anomaly prevention and detection are widely discussed topic in the field of cybersecurity. Network traffic is formed by a lot of varied and different devices like firewalls, mobiles, IOT devices, switches, proxies, honeypots, corporate desktops, etc. These system not only differ to each other architecturally and internally but they also generate logs and traffic which looks very different in the egress point. Also parsing of logs from these disparate devices can be a serious problem and threat to the security.

With the substantial increase of IoT devices and availability of the internet, the situations have turned out to be even more worse. A lot of companies have allowed their employees to work from home thus defeating a single point of egress methodology that is always preferred in cybersecurity (that increases visiblility for the IDS or network administrator).

Normal pure static packet analysis approach has turned out be a very ineffective way of dealing with malicious traffic and often fails when a new or considerable change in attackers approach is done. Machine Learning has also been used for this purpose, but the success remains limited. The deep learning algorithms suffer from system load drawback.

The proposed tool in this research is based on stacking ensembling model, using algorithms – Logistic Regression, Decision Tree, Random Forest, Support Vector Classifier and the XGBoost. The raw data from the 2 selected datasets is passed through heavy and extensive EDA and preporcessing. After proper feature selection, the ensembling machine learning model is used. Furthermore, the implemented ensemble solution extrapolates a high detection rate and a low false positive rate compared with each classification technique included in the framework.

A. The Problem

Security failures and intrusion has been a major problem since the commercialization of the internet. It's a more valid issue now more than ever due to the ever increasing number of devices being linked to networks. While this has indeed led to big possibilities online storage and others, it also has led to huge security risks which needs to be sufficiently countered. Network Intrusion Detection Systems have been around for decades now but traditional detection were more reliant on static analysis [1] of the packets and extraction of simple attributes of data packets [2] like header length, body size, flags used, etc, have been replaced by more robust and efficient IDS systems that have capability of deep packet inspection and uses Machine Learning algorithms to find correlations and extrapolate new attributes and relations out of the given data to predict the result. But most of these solutions suffer from high false positive rate problem [3] or extensive consumption of computational power problem [4].

Our approach tries to redress this weakness by using an ensembling machine learning algorithms which can learn the nature of attacks from the data flow statistics of some channel and gauge whether it is malicious in nature or not. The

implemented approach has a very low false positive rate and a very high true positive rate.

B. Network Intrusion Detection Software

Earliest generation of IDS were completely based on rule-set and needed frequent updates. These rulesets were highly specific to malware families like Gozi, Phynx, Robbox, etc and needed to be completely modified to detect some different malware family. Furthermore, it was really difficult for network administrators to maintain a log-chart of all these rule-sets and mostly these rules failed on zero-day attacks. Also even minor change in malware behaviour like polymorphic malware(which automatically changes its codes and stub sequences) bypassed these statistics based IDS [5]. These IDS couldn't do packet inspection on the network flow, and also failed to decode encrypted traffic. The next generation of IDS were based on heuristics and used more sophisticated analysis chains to detect malicious traffic. Hadi in [6] used a heuristic chain to make visualization charts for the network administrators, which could help them to save their time and energy wasted on scrolling through long logs stored by the IDS. Their proposed solution could also correlate the irregular behaviour logs to multiple IP's and thus could even detect the Distributed Denial of Service(DDOS) attacks. The output graphs generated by their tool could easily amalgamate gigabytes of log data onto a page and reveal the most important information out of it. Similar attempts were made by Jagadeesh in [7], wherein a heuristics based IDS was made for the Internet of Things(IOT) devices, that could couple with the most of the highly variant IOT hardware and could still detect IOT based malware like Marai malware. Also the proposed solution used very less computational power and thus could be properly integrated in any low-powered IOT device. The heuristic rules of the proposed solution were based on network flow dynamics and statistics. The latest addition to the legacy of IDS has been Artificial Intelligence oriented solutions that do learn from past experiences and could handle even the zero-day threats. These solutions can made through supervised or unsupervised algorithms.

C. Objectives

- A single standalone tool framework that can detect as well as classify network attacks with high accuracy and low false positive rate.
- Use novel training and sampling techniques that formulates to remove the shortcomings of the previously used machine learning approaches.
- Combining 2 similar datasets(CICIDS2017 and CICIDS2018 dataset): By using several balancing and sampling techniques, finding an optimum way to combine two similar datasets.
- Develop a machine learning architecture that fits perfectly on the data and provides robust security even against zero-day attacks.
- Develop a proof of concept of a live performance of IDS, while checking its compatibility with modern systems.
- Test several sampling techniques to merge and preprocess the dataset

D. Cyber Threats

Hackers use tons of techniques, tactics and procedures(TTP) to compromise a machine or a network. There are several steps like infiltration, lateral movement that a malicious person needs to do before executing the malware. It can be as simple as causing a denial of service or a trickier web-based attack like XXE injection, XSS attack, etc. Some of the attacks are as follows:

DOS: Denial of Service situation arises when malware is able to temporarily disable any service on a platform. It may be done via flooding of an open port, using vulnerability of a service running on a machine or flooding the network devices RAM like MAC flooding attacks.

DDOS: Distributed Denial of service is an advanced form of DOS attacks, wherein several(even in millions) of machines try to connect to the services of the target machine, thus flooding all its computational resources and bringing down the machine to its knees. This type of attack is difficult to defeat as IP banning does not work in this case.

Infiltration: Infiltration is the second step of any successful hacking attempt, wherein a bad guy tries to attain a shell or command prompt on a machine. The hacker may or may not run any malicious file in this step but this step ensures that internal machines become in-reach to the bad guy.

Botnet: Botnet is a zombie host that remains infected for a long period of time until its master bot(controller bot) issues a command to it. These botnets can be used to conduct other types of attacks like DDOS or can be used as front-end in any legal breach attempt. Botnets can also be anytime converted into trojan infected systems and thus can leak credentials, information about the user of that machine,

PortScan: Port Scan is the basic and most primary step in any penetration testing. In PortScan, the malicious hacker tries to send different types of normal as well as deformed packets to the vulnerable target, so that the target leaks out some information about its configurations, services or ports. Port Scanning is illegal in the USA, but it is within legal boundaries in countries like India.

E. Previous Research on the issue

The first generation of Intrusion Detection systems were purely rule and signature based. They had a very resource-intensive processor and analyzer unit which scanned all network traffic by matching all the pre-defined rule set with the network statistics. This type of systems required frequent updates and were prone to new zero-day attacks. As an earlier NIDS example, Crotti [8] proposed that apart from network statistics, packet inspection at IP level can also reveal some useful artifacts that can help in segregation of traffic. But these attempts failed miserably infornt of newer attacks like DNS tunneling [9], wherein forged DNS queries are transferred to

the destination server which on decryption, results in execution of HTTP protocol. To counter these problems, newer and more robust rules had to be created like [10], which can inspect deep into the network packets and calculate indirect statistical features like inter-arrival time, among many others. Tian in [11], proposed a novel model named smooth K-Windows architecture which recorded a users normal behaviour over a period of time and then compared it to future situations. Muhammed in [12] compared all statistical approaches that were used in NIDS and gave out conclusions that signature-based NIDS are the most accurate defense system but this less false-positive rate comes at cost of high resource consumption and higher operational costs.The software used in this research was Weka, which was used to compile all research data and find the resultant outcomes.

The idea of using Machine Learning techniques in developing intrusion detection systems isn't a replacement idea. Knowing that we decided to look for papers and research on the particular issue. We focused on the work of two particular papers, both from research conducted by the University of Mexico. The first, by Mahdi Zamani and Mahnush Movahedi [13] delves into the utilization of Machine Learning techniques in the intrusion Detection field of study, taking a more general approach, which familiarised us with the matter we wanted to unravel. The second paper we studied, by Mukkamala, Janoski and Sung [14] takes a better perspective on the difficulty, trying to implement solutions using Support Vector Machines and Neural Network on a DARPA dataset, and comparing the 2 methods' results. Based on these respected professors' work, we decided to use a number of these methods in our project. Abhishek in [15] used basic euclidean distance based algorithms like K-means algorithm to segregate pristine network behaviour patterns to the abnormal patterns.

In paper [16], an Anomaly-based Intrusion Detection System was implemented using DNN (Deep Neural Network) and their model made use of Synthetic Minority Over-sampling Technique(SMOTE) algorithm to deal with imbalanced classification problem and to extract high-level features. To achieve their goals they made use of the latest and diverse real-traffic Flow-based dataset named CIC-IDS 2018 containing varied types of attacks.

The authors of [17] address various issues such as training data inequality and improper selection of methods in the latest and varied datset CIC IDS 2017. To achieve high accuracy and robustness use Principal Component Analysis (PCA) and Ensemble Feature Selection (EFS) to select key attributes from the database, along with SMOTE for an AdaBoost-based IDS.

This paper [18] pointed out the various shortcomings faced while working on CIC IDS 2017 dataset and provided some alternatives to resolve the class imbalance problem. They made use of some preprocessing measures and techniques to resolve the class imbalance problem and inherent shortcomings of the dataset. Along with this they proposed a combined dataset approach to deal with such shortcomings and make detection and classification of IDS better.

The authors of [19] talks about how the web applications

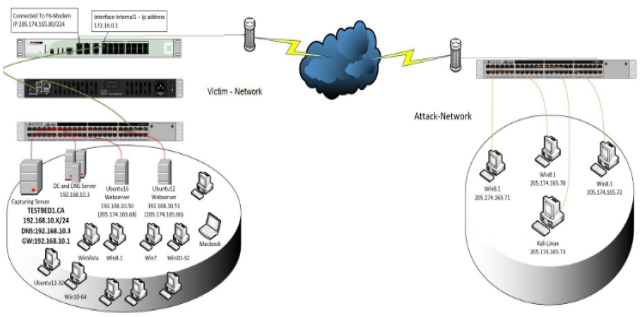

Fig. 1: CIC-IDS-2017 Testbed Visualized

can be exploited by malicious attackers through internet by utilizing hidden vulnerabilities and risks, and proposed a Firewall-based ADS. They used the one-stage SVM process and information about the HTTP application structure to create better ways to extract features to improve recovery rates. This paper [20] proposes a new algorithm, called the Energy-based Flow Classifier (EFC). This anomaly-based separator uses opposite statistics to include a statistical model based on risk models. It has been shown that EFC is able to accurately perform binary flow segments and is able to adapt to different data distribution than ML-based formats. They used their model in the CIDDS-001, CICDDoS19 and CICIDS17 datasets and concluded that EFC is a promising algorithm for performing robust segments based on traffic flow.

Janarthanan and Zargari [21] used a number of algorithms for the purpose of feature selection on the UNSW-NB15 dataset. They used Weka tool and used the Kappa Statistic measure for evaluating the effectiveness of different subsets of features. They proposed that the Random Forest Classifier outperformed other classifiers after feature selection. However, forward feature selection method was not employed here. We use this method of feature selection for iteratively selecting the best feature which provides an increase in the performance of our model.

Husain et al. [22] performed Extreme Gradient Boosting as a classifier along with feature selection. It was reported to outperform other classifiers. An ensemble approach of combining different base learners and the XGBoost model was not taken here. Our experimental results show that it outperforms the traditional approach.

II. Experimental Analysis

A. Dataset Description

The datasets we chose for the task of training an ML model were the CSE-CIC-IDS-2018 and CIC-IDS-2017. These datasets have been constructed such that they effectively capture a real-world network attack. The toplogy of these are shown in Fig. 1 and Fig.4 respectively. The data creators in their seminal paper [23] made use of the concept called profiles to generate cyber-security datasets. In their paper, they describe the use of M profiles in an attempt to make the attack setting simple. A straightforward way to accomplish this is

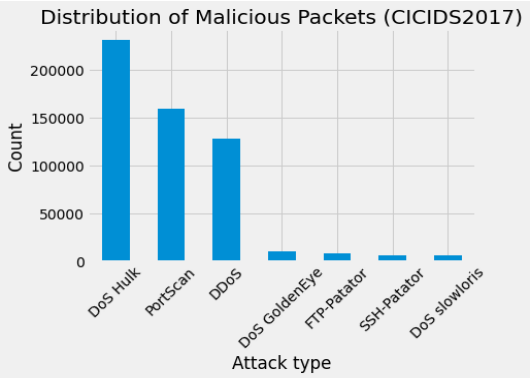

Fig. 2: Attack distribution CIC-IDS-2017

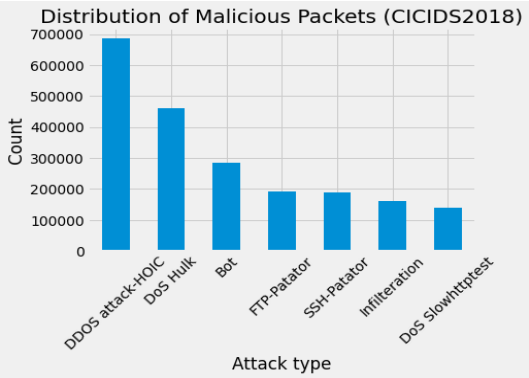

Fig. 3: Attack distribution CSE-CIC-IDS-2018

by having humans interpret and act it out. In a more ideal setting, we would have an independent agent with compilers to carry them out. Some examples of the attacks carried out include Heartleech, Botnet, DDos, Dos, Infiltration. On the other hand B profile can be understood as having ML and Statistical models capture the behaviour of the malicious user and have it perform attacks.

In the end, their dataset constituted seven different attack scenarios and their attacking infrastructure was composed of fifty independent machines. The victim possessed five departments made up of four hundred twenty machines and thirty servers. The dataset has 80 features derived from CICFlowMeter , the record of logs and network traffic of each system.

B. Specifics of the data

We in attempts to have more data to work with and to build a more robust model concatenated IDS 2017 with IDS 2018 to get a total of 2.8M + 8.2M samples ie. approximately an 11.1M samples. As mentioned earlier the dataset consisted of 7 distinct scenarios of attack Heartbleed, DoS, Botnet, Brute-force, DDoS, infiltration, and web attacks. The distribution of these attacks in the dataset is shown in Fig. 2 and Fig.3 respectively. The combined dataset had the following labels consisting of both malignant and benign samples. These are:- Brute Force, DoS, slow loris, PortScan, SQL Injection, DDoS, GoldenEye, FTP-Patator, SSH-Patator,XSS, Infiltration, Slowhttptest, Bot, and Heartbleed.

C. Architectural Framework

As mentioned earlier the authors of the datasets used an infrastructure consisting of 50 independent machines which operated the 7 different attack scenarios. The attacks were separately conducted and logged. To make the data more realistic each attack's dataset consists of both benign and the attack's samples.

For capturing of packets and feature extraction CICFlowMeterV3 was used. It is a network traffic flow generator coded in Java. It provides the user with a high degree of freedom in choosing the features he/she wants to include. The flow generator renders Bidirectional Flows where source sends the

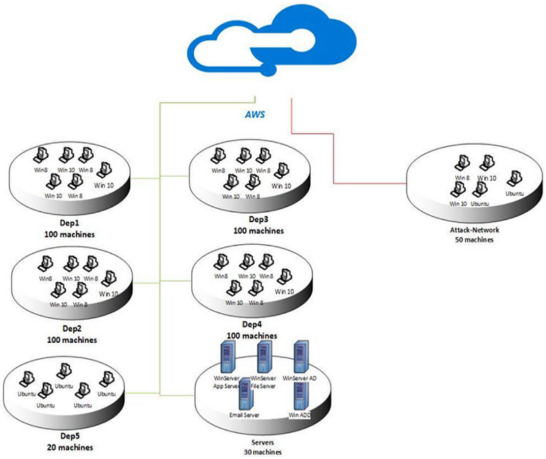

Fig. 4: CSE-CIC-IDS-2018 Network Topology

first packet (the forward flow) and the other from destination back to source (the backward flow). 80 or so features such as Number of bytes,Length of packets,Duration, Number of packets, etc. were calculated separately both from sender to receiver and vice versa.

D. Preprocessing

As preprocessing we removed the features which had a high correlation to an existing feature; this helps reduce the dataset's dimensionality while preserving information. [24] and [25] had some niche points which we also considered when preprocessing the dataset. Furthermore, we dropped all duplicate samples and removed samples of labels whose counts were too low to take into consideration. Further, we dropped samples that had Null fields. Finally, we encoded our labels to fit them in ML models. During preprocessing we also experimented with different types of Up and Down Sampling but overall they either gave us very poor performance or prediction scores no better than what we achieved without them. However, Edited Nearest Neighbours sampling technique proved to be useful over unsampled data and gave the best results. The comparison between different sampling techniques was done using the Decision Tree classifier, with a smaller depth. These results are given in Table 1.

Scores	Random_Over_Sampler	ADASYN	SMOTE	Near_Miss	Random_Under_Sampler	Edited_Nearest _Neighbours
Accuracy	0.968300	0.794019	0.968255	0.501347	0.969826	**0.978565**
Precision	0.986113	**0.993346**	0.985594	0.970989	0.986187	0.983466
Recall	0.976604	0.763725	0.977080	0.428426	0.978343	**0.991550**
F1	0.981335	0.863532	0.981318	0.594529	0.982249	**0.987492**
ROC _AUC	0.948299	0.866983	0.947000	0.676982	**0.949315**	0.947291

TABLE I: Changes in scores due to different sampling techniques

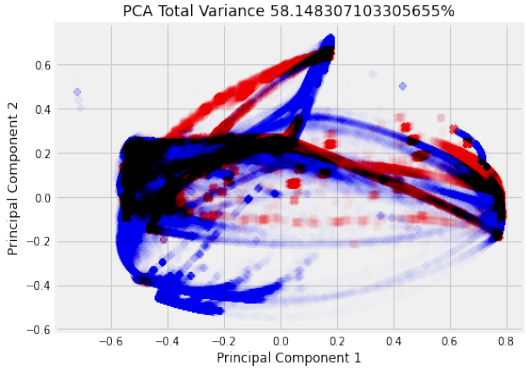

Fig. 5: Dimensionality reduction

E. General Approach and Algorithms used

The dataset provides two columns as targets one with normal-not normal values and one with attack type values. We found that classifying by attack type and aggregating the results to normal and abnormal traffic gave us better results than classifying on normal/abnormal traffic. Thus, our problem is categorized as a supervised multiclass classification problem. supported the papers we mentioned we selected the utilization of three algorithms to specialize in the issue:

- Logistic Regression
- Support Vector Machine
- Decision Tree Classifier
- Random Forest
- XGBoost Stacking Ensemble

We visualized the data after applying preprocessing and using Principal Component Analysis with two features to see the separability of data. This is shown in Fig. 5, where the total variance by these two principal components is only 58% and the data does not look easily separable by a linear classifier.

We will analyse the work implemented in all, in the order of increase in performance:

1) **Logistic Regression**- Logistic Regression is a generalized linear model which uses the 'logistic' function. It performed the worst of all classifiers used by us, for both binary and multiclass classification. We assume the inability of the model to fit non-linearly separable data was the reason and decided to apply non-linear classifiers.
2) **Support Vector Machine**- Support Vector Machine maps data samples as points in space in an attempt to maximize the distances between the classification targets. New samples of data are then mapped inside the same space where they get predicted to lie in one of the many different categories based on which side space they belong to. The thing that differentiates SVM is its ability to use kernel tricks to implicitly map points into higher dimensional spaces.
   Linear SVM seems to mimic logistic regression's performance as they both possess similar structures. A slight increase of scores observed in logistic regression might be due to its probabilistic nature.
3) **Decision Tree**- A Decision Tree Classifier is a tree-based model whose internal nodes represent a label with an input feature. When a sample flows down a tree decisions are taken on each node with regards to the value that the sample possesses. The end nodes or leaf nodes represent probabilities of the sample having a particular class in the case of classification trees and continuous in the case of regression trees. Decision Tree Classifier performed relatively well compared to Logistic Regression, which motivated us to experiment with more tree-based models. With it having relatively shorter training time to boosted algorithms it truly gave a proper insight into whether tree-based modelling was proper for this particular problem
4) **Random Forest**- Random Forests are based on an ensemble learning approach where multiple decision trees are constructed at training time. Essentially decision trees can be made to learn unique patterns by allowing them to grow to high depths. This makes the decision tree models overfit on the dataset which means they have high variance and low bias. Random Forest remedies the problem as it averages over these deep trees which are trained on different portions of the training set. This leads to a reduction in variance in better performance overall. After observing a better result with the Decision Tree, we applied the Random Forest Classifier which is just an ensemble of many Decision Trees. We used 100 estimators and kept the maximum depth the same as what we used for training a single Decision Tree.
5) **XGBoost Classifer**- XGBoost is an open-source software library that allows the user to make use of a regularizing gradient boosting framework. Gradient Boosting trees also are a form of ensemble learning where weak learners are used for prediction. When the weak learners are Decision Trees then they are called gradient boosted trees which in most cases outperforms Random Forest. XGBoostClassifier is an optimized distributed gradient

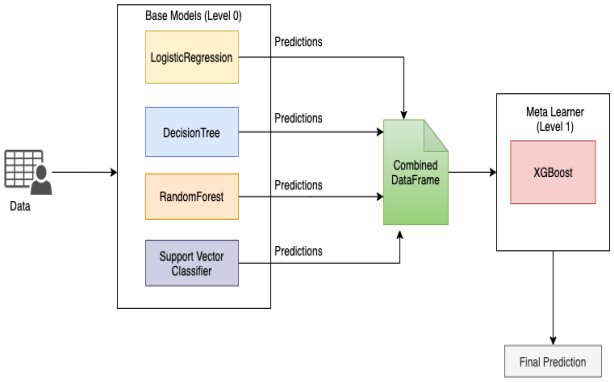

Fig. 6: XGBoost Stacking Ensemble Model

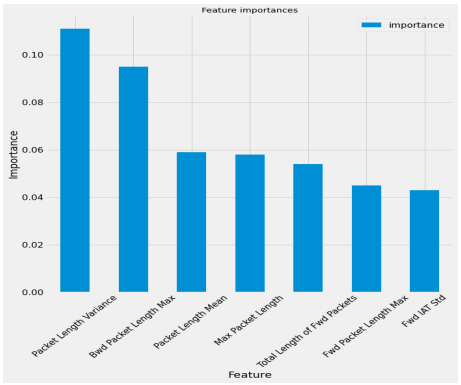

Fig. 7: Feature importance

boosting classifier designed to be highly efficient, flexible, and portable. XGBoost provides a parallel tree boosting which provides fast and accurate algorithms for predictive analysis purposes.

6) **Ensemble Stacking**- Stacked Generalization (also known as Stacking) is a type of ensemble ML algorithm. We combine the predicted class probabilities from multiple ML models on the same dataset, like bagging or boosting. Unlike bagging, the models we make use of are typically different but trained on the same dataset. Unlike boosting, a single model has been used to fit over the predicted class probabilities from the models used as base. The stacking model is made up of two or more models, known as Zero-layer models, and a model that fits over them is known as a First-layer model.
   - *Zero-layer Models (Base-Models)*: Models fit on the train set and predictions are made into a dataset for the meta model.
   - *First-layer Model (Meta-Model)*: Model that fits on predictions of the base models.

   Our base models [Fig. 6] included Logistic Regression, Support Vector Machine,Decision Tree and Random Forest Classifier. We used the Meta-Model as the XGBoost Classifier.

F. Metrics Used

1) ROC AUC Score : ROC-AUC Score also known as the area under the curve of Receiver Operating Characteristics. This score ends up showing the relative measure of distinguish-ability. A machine learning model with larger AUC can predict True Positives and True Negatives better [26]. AUC is independent of the scale and threshold.
   Probabilistically AUC score is equal to the ability of the model to give higher rank to a random positive instance than a negative one.
2) Accuracy Score: Accuracy is the measure of True Positives and True Negatives over the entire sample space it was measured on.

$$\frac{TP+TN}{TP+TN+FP+FN} \tag{1}$$

3) Precision and Recall Score : Precision is the ratio between the actual number of true prediction which were accurately predicted over all true predictions. Recall score is the ratio between actual number of true predictions which were accurately predicted over the entire true samples.

$$\frac{TP}{TP+FP} \tag{2}$$

$$\frac{TP}{TP+FN} \tag{3}$$

4) F1 Score : The F1 score is the harmonic mean of precision and recall.

$$\frac{TP}{TP+\frac{1}{2}(FP+FN)} \tag{4}$$

G. Data Visualization techniques used

1) Confusion Matrix- It is a table used to visualize the performance of an algorithm. It describes the performance of a classification model on data for which true values are known. They are used to visualize important predictive analysis and provide direct comparisons between True Positive, True Negative, False Postive and False Negative.
2) ROC(Receiver Operating Characteristic)-An ROC curve is a tool used to understand the performance of a binary classifier at various discrimination thresholds. AUC of ROC is an important measure to know how the model is able to rank a positive instance higher than a negative one.Its value goes from 0 to 1. This curve plots two parameters: True Positive Rate and False Positive Rate.

III. Results and Analysis

The overview of the results obtained for binary and multiclass classification tasks can be seen from Table 2 and Table 3 for the different metrics: Accuracy, Precision, Recall, F1 and ROC Score.

1) **Random Forest Classifier**: Keeping the maximum depth of the tree the same as that used in Decision Tree and keeping several estimators, the performance of Random Forest is found to be better than Decision Tree. The gain in accuracy score is 0.5% over Decision

Model	Accuracy	Precision Score	Recall Score	F1 Score	ROC Score
Logistic Regression	0.909	0.919	0.979	0.948	0.741
Decision Tree	0.975	0.979	0.992	0.985	0.935
Random Forest	0.980	0.979	0.998	0.988	0.937
Support Vector Classifier	0.909	0.918	0.980	0.948	0.738
XGBoost Stacking Ensemble	**0.982**	**0.980**	**0.999**	**0.989**	**0.941**

TABLE II: Metrics for binary classification

Model	Accuracy	Precision Score	Recall Score	F1 Score	ROC Score
Logistic Regression	0.869	0.268	0.197	0.214	0.564
Decision Tree	0.963	0.747	0.692	0.682	0.839
Random Forest	0.980	0.912	0.877	0.893	0.933
Support Vector Classifier	0.871	0.277	0.195	0.208	0.563
XGBoost Stacking Ensemble	**0.981**	**0.953**	**0.897**	**0.905**	**0.943**

TABLE III: Metrics for multiclass classification

Tree in binary classification (98%), but a large margin of 1.5% in the case of multi-class classification (98%). The ROC score is better as well which indicates that it is able to differentiate between the different classes better. Out of all the level 0 base learners, this is by far the best in terms of performance. It took considerably a large amount of time to train it, but less than what it took for training the SVM model. Feature importance plot as shown in Fig. 7 was generated using Random Forest Classifier and depicts that the variance in Packet Length is the most important and useful feature.

2) **Logistic Regression**: As logistic regression was one of the simpler models to train, it ended training comparatively quickly and gave an accuracy of 90.9% for binary classification problems and 86.9% for multi-class classification. Out of the four classification techniques used it was one of the worst-performing in terms of all the metrics which we listed.
3) **Decision Tree Classifier**: As one of the most simple tree-based classifiers, we trained it for a maximum depth of 10, increasing which further we observed no improvement in the accuracy score. An accuracy of 97.5% for binary classification and 96% for multi-class classification is observed. It outperforms the linear models but still is not fitting the data well. This model has high bias and low variance and seems to be underfitting but not as much as the linear classifiers. The difference between Decision Tree and Random Forest is not that huge in binary classification, but in multi-class classification, there is a very large difference in the performance of these two models.
4) **Support Vector Classifier**: The results obtained for the Support Vector Classifier did not justify the amount of time it took to train the model. Although we used a linear kernel which is the most simplistic kernel and assumes that the data can be linearly separated by a hyperplane, the results were still as bad as Logistic Regression. This model took the most amount of training time and had the worst performance.

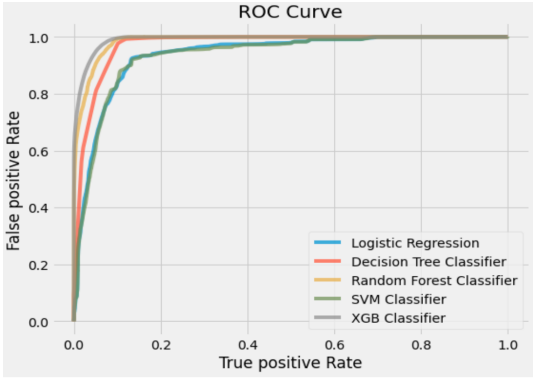

Fig. 8: ROC AUC curves for all models.

5) **XGBoost Stacking Ensemble**: Using XGBoost as the level 1 meta learner, which doesn't actually look at the training data but the prediction probabilities (or the predictions in case the implementation does not have a method to predict the probability such as LinearSVC) as input and the correct expected labels as output. Keeping default parameters, combining the above base learners and applying to stack using XGBoost observed the best score out of all. The performance was a little better than the best performing level 0 model but not by a huge gap. Although the model here is more confident as evident by a better F1 score and ROC score.

It can be observed that Xgboost Classifier seems to perform better under both the cases. This proves that weaker learners can be combined into strong learners [27] Logistic Regression and Support Vector Classifier seems to lag behind other classifiers in both binary and multiclass classification. Random Forest based ensemble seems to have an edge over other types of algorithms.

It is seen that on binary classification other than XGBoost Stacking Ensemble, Random Forest Classifier provided the best Recall and F1 score. Also it is very evident from the table that XGBoost Stacking Ensemble came out with the highest scores in all the metrics in both binary and multiclass

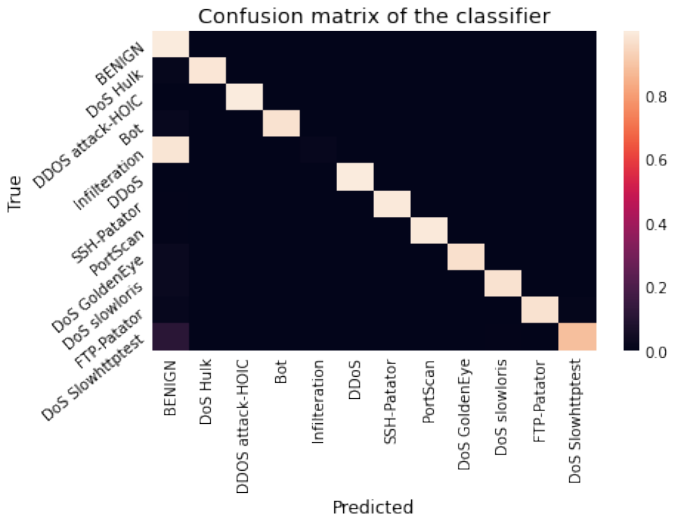

Fig. 9: Confusion Matrix for XGBoost Classifier (Multi-class).

classification. Its Accuracy, Precision, Recall, F1 and ROC scores are- .982, .980, .999, .989, .943, respectively for binary classification and .981, .953, .897, .905, .943 respectively for multiclass classification.

The Receiver Operating Characteristics (ROC) curve shown in Fig. 8 depicts that XGB Ensemble has the best ROC-AUC score which is an improvement over Random Forest Classifier which itself (an ensemble of Decision Trees) is an improvement over Decision Tree ROC-AUC score. ROC-AUC Score is a comparatively better metric to use here than just accuracy score because of imbalance in dataset [28]. The linear classifiers (LogisticRegression and Support Vector Machine with a Linear kernel) have a similar unsmooth curve with an area under the curve which is although a little better than random naive classifier would, but still outperformed by a huge margin by the non-linear tree-based classifier and the final level 0 models stacked ensemble.

The readings shown by confusion matrix from Fig. 9 shows that infiltration was highly incorrectly classified by the model even though there are more than 150000 samples for infiltration available in CICIDS2018. This is a matter of concern and needs to be further investigated and it may lead us to believe that some different statistical tools may be needed to understand infiltration. The classifier does not confuse this attack for any other attack but completely as BENIGN.

IV. Future Scope

The learning module of our model can be expanded to incorporate more types of Intrusion attacks and system vulnerabilities. Increasing the number of base models for the meta learning can result in better results [29] by combining more weak learners into a strong learner. Other methods for feature selection can be explored. One can perform more rigorous hyper-parameter tuning on base models used in the meta learning. Increasing the size of data corpus and augmenting the data-set with more real world data for a generalized model. Better handling of imbalanced data using different sampling techniques can be performed.

Our model primarily works on flows generated by data packets so it can be integrated along with a classical signature based intrusion detection software, deployed together into the real world. This can enable real-time data collection along with automatic online learning, thus making the model much dynamic in nature. Combining signature based intrusion detection with machine learning based intrusion detection [30] will also allow better processing and analysis of the results. This not only allows the possibility of detecting attacks in real-time accurately but also to thwart such attempts at an early stage, thus maintaining system integrity.

References

[1] Hu Zhengbing, Li Zhitang, and Wu Junqi. A novel network intrusion detection system (nids) based on signatures search of data mining. In *First International Workshop on Knowledge Discovery and Data Mining (WKDD 2008)*, pages 10–16. IEEE, 2008.

[2] Martuza Ahmed, Rima Pal, Md Mojammel Hossain, Md Abu Naser Bikas, and Md Khalad Hasan. Nids: A network based approach to intrusion detection and prevention. In *2009 International Association of Computer Science and Information Technology-Spring Conference*, pages 141–144. IEEE, 2009.

[3] Sailesh Kumar. Survey of current network intrusion detection techniques. *Washington Univ. in St. Louis*, pages 1–18, 2007.

[4] Prashantkumar M Rathod, Nilesh Marathe, and Amarsinh V Vidhate. A survey on finite automata based pattern matching techniques for network intrusion detection system (nids). In *2014 International Conference on Advances in Electronics Computers and Communications*, pages 1–5. IEEE, 2014.

[5] Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24, 2013.

[6] Hadi Shiravi, Ali Shiravi, and Ali A Ghorbani. Ids alert visualization and monitoring through heuristic host selection. In *International Conference on Information and Communications Security*, pages 445–458. Springer, 2010.

[7] M Jagadeesh Babu and A Raji Reddy. Sh-ids: Specification heuristics based intrusion detection system for iot networks. *Wireless Personal Communications*, pages 1–23, 2020.

[8] Manuel Crotti, Francesco Gringoli, Paolo Pelosato, and Luca Salgarelli. A statistical approach to ip-level classification of network traffic. In *2006 IEEE International Conference on Communications*, volume 1, pages 170–176. IEEE, 2006.

[9] Daan Raman, Bjorn De Sutter, Bart Coppens, Stijn Volckaert, Koen De Bosschere, Pieter Danhieux, and Erik Van Buggenhout. Dns tunneling for network penetration. In *International Conference on Information Security and Cryptology*, pages 65–77. Springer, 2012.

[10] Manuel Crotti, Maurizio Dusi, Francesco Gringoli, and Luca Salgarelli. Detecting http tunnels with statistical mechanisms. In *2007 IEEE International Conference on Communications*, pages 6162–6168. IEEE, 2007.

[11] Ming TIAN and Yi ZHUANG. Building the nids models with smooth k-windows statistical model. *Journal of Changchun University of Science and Technology (Natural Science Edition)*, 2009.

[12] Muhammad Junaid Muzammil, Sameer Qazi, and Taha Ali. Comparative analysis of classification algorithms performance for statistical based intrusion detection system. In *2013 3rd IEEE International Conference on Computer, Control and Communication (IC4)*, pages 1–6. IEEE, 2013.

[13] Mahdi Zamani and Mahnush Movahedi. Machine learning techniques for intrusion detection. *arXiv preprint arXiv:1312.2177*, 2013.

[14] Srinivas Mukkamala, Guadalupe Janoski, and Andrew Sung. Intrusion detection: support vector machines and neural networks. In *Proceedings of the IEEE international joint conference on neural networks (ANNIE)*, pages 1702–1707, 2002.

[15] Abhishek Verma and Virender Ranga. Statistical analysis of cidds-001 dataset for network intrusion detection systems using distance-based machine learning. *Procedia Computer Science*, 125:709–716, 2018.

[16] Rawaa Ismael Farhan, Abeer Tariq Maolood, and NidaaFlaih Hassan. Performance analysis of flow-based attacks detection on cse-cic-ids2018 dataset using deep learning. *Indonesian Journal of Electrical Engineering and Computer Science*, 20(3):1413–1418, 2020.

[17] Arif Yulianto, Parman Sukarno, and Novian Anggis Suwastika. Improving adaboost-based intrusion detection system (ids) performance on cic ids 2017 dataset. In *Journal of Physics: Conference Series*, volume 1192, page 012018. IOP Publishing, 2019.

[18] Ranjit Panigrahi and Samarjeet Borah. A detailed analysis of cicids2017 dataset for designing intrusion detection systems. *International Journal of Engineering & Technology*, 7(3.24):479–482, 2018.

[19] Nico Epp, Ralf Funk, Cristian Cappo, and San Lorenzo-Paraguay. Anomaly-based web application firewall using http-specific features and one-class svm. In *Workshop Regional de Segurança da Informação e de Sistemas Computacionais*, 2017.

[20] Camila Pontes, Manuela Souza, João Gondim, Matt Bishop, and Marcelo Marotta. A new method for flow-based network intrusion detection using the inverse potts model. *IEEE Transactions on Network and Service Management*, 2021.

[21] Tharmini Janarthanan and Shahrzad Zargari. Feature selection in unsw-nb15 and kddcup'99 datasets. In *2017 IEEE 26th international symposium on industrial electronics (ISIE)*, pages 1881–1886. IEEE, 2017.

[22] Anwar Husain, Ahmed Salem, Carol Jim, and George Dimitoglou. Development of an efficient network intrusion detection model using extreme gradient boosting (xgboost) on the unsw-nb15 dataset. In *2019 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pages 1–7. IEEE, 2019.

[23] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *ICISSp*, pages 108–116, 2018.

[24] Ziadoon Kamil Maseer, Robiah Yusof, Nazrulazhar Bahaman, Salama A Mostafa, and Cik Feresa Mohd Foozy. Benchmarking of machine learning for anomaly based intrusion detection systems in the cicids2017 dataset. *IEEE Access*, 9:22351–22370, 2021.

[25] V Kanimozhi and T Prem Jacob. Artificial intelligence based network intrusion detection with hyper-parameter optimization tuning on the realistic cyber dataset cse-cic-ids2018 using cloud computing. In *2019 International Conference on Communication and Signal Processing (ICCSP)*, pages 0033–0036. IEEE, 2019.

[26] Peter A Flach. Roc analysis. In *Encyclopedia of Machine Learning and Data Mining*, pages 1–8. Springer, 2016.

[27] Bohdan Pavlyshenko. Using stacking approaches for machine learning models. In *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)*, pages 255–258. IEEE, 2018.

[28] Jin Huang and Charles X Ling. Using auc and accuracy in evaluating learning algorithms. *IEEE Transactions on knowledge and Data Engineering*, 17(3):299–310, 2005.

[29] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249, 2018.

[30] Ibraheem Aljamal, Ali Tekeoğlu, Korkut Bekiroglu, and Saumendra Sengupta. Hybrid intrusion detection system using machine learning techniques in cloud computing environments. In *2019 IEEE 17th International Conference on Software Engineering Research, Management and Applications (SERA)*, pages 84–89. IEEE, 2019.