Machine Learning approach for DoS/DDoS Attack Detection

A PROJECT REPORT

for

Information Security Management (CSE3502)

in

B. Tech – Information Technology and Engineering

by

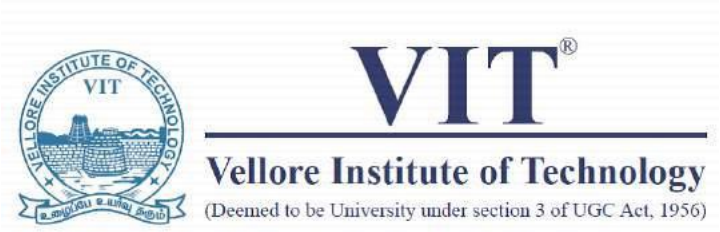
AYUSHI GUPTA (18BIT0367)

GAURANSH ARORA (18BIT0393)

Under the Guidance of

Dr. Sumaiya Thaseen

Associate Professor, SITE



School of Information Technology and Engineering

Winter Semester 2020-21

I. Abstract

Distributed Denial of Service attack (DDoS) is the most dangerous attack in the field of network security. DDoS attack halts normal functionality of critical services of various online applications. Systems under DDoS attacks remain busy with false requests (Bots) rather than providing services to legitimate users. These attacks are increasing day by day and have become more and more sophisticated. So, it has become difficult to detect these attacks and secure online services from these attacks. In this paper, we have used machine learning based approach to detect and classify different types of network traffic flows. The proposed approach is validated using a new dataset which is having mixture of various modern types of attacks such as HTTP flood, SID DoS and normal traffic. We also used many machine learning algorithms such as Logistic Regression, Neural Network, and Decision tree to detect abnormal activities such as DDOS features. In the experimental results, we found that random forest and decision tree achieved high accuracy to detect attacks.

II. Introduction

Internet services have become utmost important now days for business organizations and individuals. With the increase in demand of network-based services networks intruders have also increased their attacks on these services to halt the response of services to the legitimate users. The attacks which halt or slows down the services of network applications are known as DDoS attacks. A DDoS attack is achieved by attackers by controlling millions of freely available computer systems on the internet. Thus, causing servers to deny response to legitimate users and keeping busy with the requests generated by the attacks. Prominent websites like banking, social networking sites, universities etc. are more targeted by these attacks. Therefore, one or more security tools like antivirus software, firewall and intrusion detection

system (IDS) should be used in computer network to protect important data and services from the intruders.

One of the most used solutions to deal with the DDoS attacks is an intrusion detection system (IDS). An IDS preserves integrity, confidentiality and availability of computer network resources and web services. Machine learning techniques are used in IDS system to detect and classify various DDoS attacks and eliminate intrusion. However, it is hard to achieve hundred percent performance accuracy in classifying and detecting attacks.

Currently, there are various type of DDoS attacks are active in networks like Smurf attack, HTTP POST/GET and SQL Injection DOS (SIDDoS). All of them causes denial of service to network services by sending huge volume of useless traffic to web server. Various publicly available datasets are obsolete and don't have newer and latest type of attack traffic such as SIDDOS, Smurf and HITP flood in them. Therefore, there was a need of a dataset which includes various new attacks in it. So, earlier we used KDD Cup 1999 Data which includes four type of harmful attacks namely: UDP-flood, SIDDoS, H IQ-flood and Smurf.Since the dataset was older we moved on to NF-UQ-NIDS dataset available on cloudshare.

Classification of network traffic in done using machine learning techniques. Classification is done on the basis of some features like average packet size, bit rate, packet size, inter arrival time etc. These features are used to decide the class of network traffic i.e., normal or DDoS attack. Mostly DDoS attacks have same average packet size. So, by analyzing dataset machine learning techniques makes decision about the traffic type in the network either as attack traffic or as a normal traffic. We have used Jupyter Notebook and Google Colab as machine learning technique in our research to detect attacks in the network traffic.

UNIQUENESS: Since we faced a lot of problem in loading time and training time everytime when we train these type of models where datasets are so large , to avoid huge waiting time (loading time) in future we found an optimized way to reduce the loading time of the model in the future after it is trained once.

III. Literature Review

DDoS attacks are very common nowadays. These attacks cause a severe damage to network resources and cause denial of service to legitimate users.

X. Yuan, C. Li and X. Li, "DeepDefense: Identifying DDoS Attack via Deep Learning,"

This paper proposes a deep learning-based DDoS attack detection approach (DeepDefense). The experimental results demonstrate a better performance of the model compared with conventional machine learning models. The error rate is reduced from 7.517% to 2.103% compared with conventional machine learning method in the larger data set.

Tuan, T.A., Long, H.V., Son, L.H. *et al.* Performance evaluation of Botnet DDoS attack detection using machine learning.

To over the problem of DDoS attack, various machine learning methods typically Support Vector Machine (SVM), Artificial Neural Network (ANN), Naïve Bayes (NB), Decision Tree (DT), and Unsupervised Learning (USML) (K-means, X-means etc.) were proposed. This 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 which are well-known publicity datasets for Botnet DDoS attack detection.

Zekri, M., El Kafhali, S., Aboutabit, N., & Saadi, Y. (2017, October). DDoS attack detection using machine learning techniques in cloud computing environments.

In this work, a DDoS detection system based on the C.4.5 algorithm to mitigate the DDoS threat is designed. This algorithm, coupled with signature detection techniques, generates a decision tree to perform automatic, effective detection of signatures attacks for DDoS flooding attacks. This paper also selected other machine learning techniques and compared the obtained results in order to validate the system.

Aamir, M., Zaidi, S.M.A. DDoS attack detection with feature engineering and machine learning: the framework and performance evaluation.

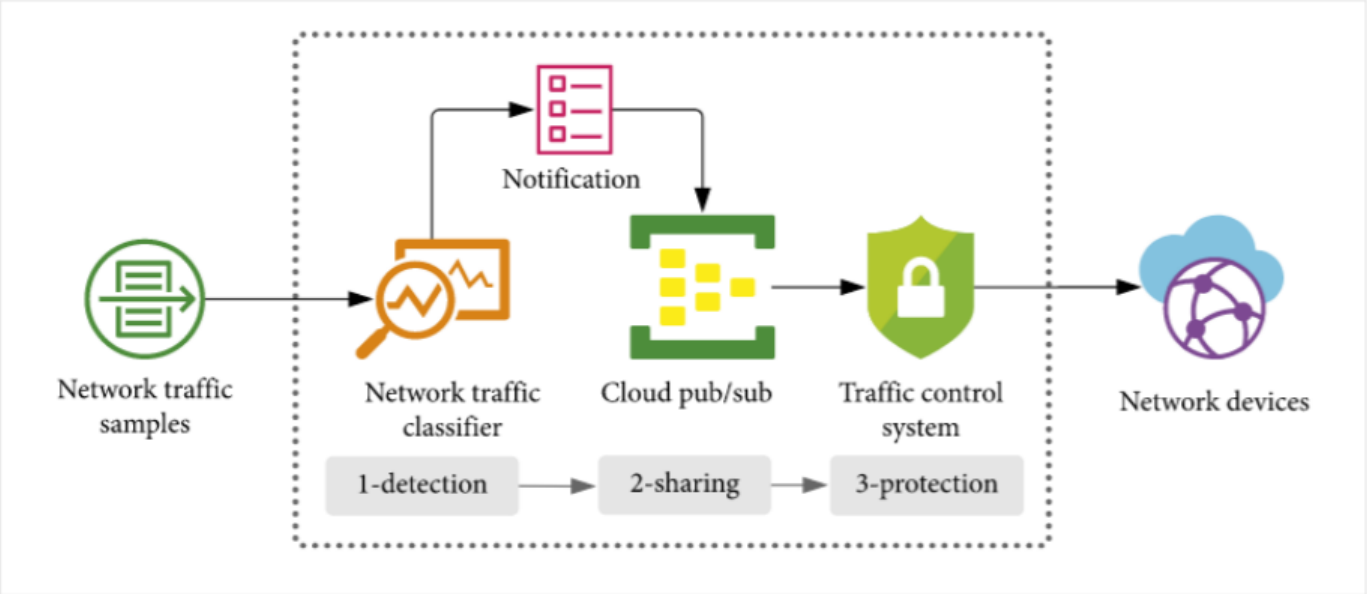
This paper applies an organized flow of feature engineering and machine learning to detect distributed denial-of-service (DDoS) attacks. The results show that substantial feature reduction is possible to make DDoS detection faster and optimized with minimal performance hit. The paper proposes a strategic-level framework which incorporates the necessary elements of feature engineering and machine learning with a defined flow of experimentation. The experiments show that approximately 68% reduction in the feature space is possible with an impact of only about 0.03% on accuracy.

Link for Review 1 document: https://github.com/Goldy1906/CSE3502-Project_review/blob/master/REVIEW%201/Review-1%20.pdf

IV. Proposed Model

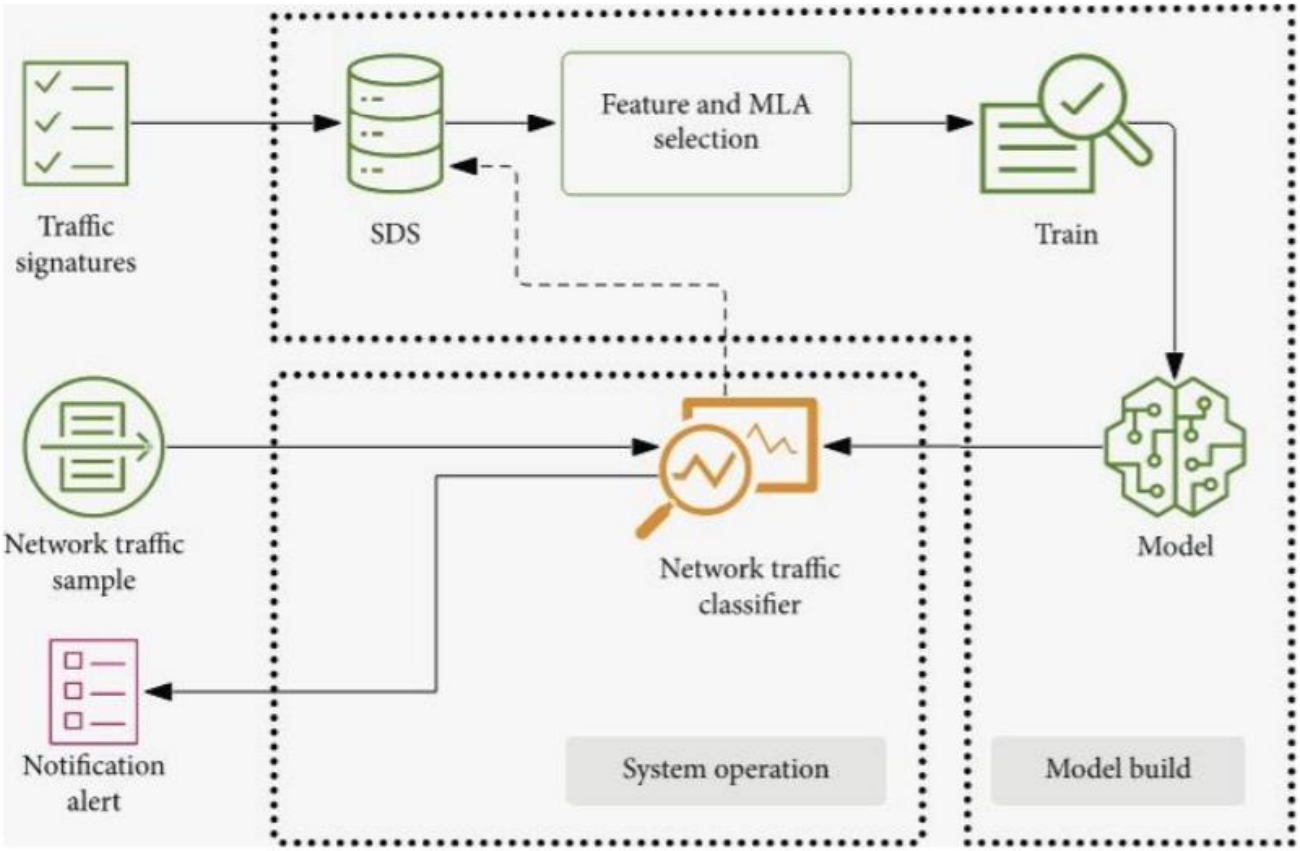
Low Level Diagram

Smart Detection is designed to combat DDoS attacks on the Internet in a modern collaborative way. In this approach, the system collects network traffic samples and classifies them. Attack notification messages are shared using a cloud platform for convenient use by traffic control protection systems.



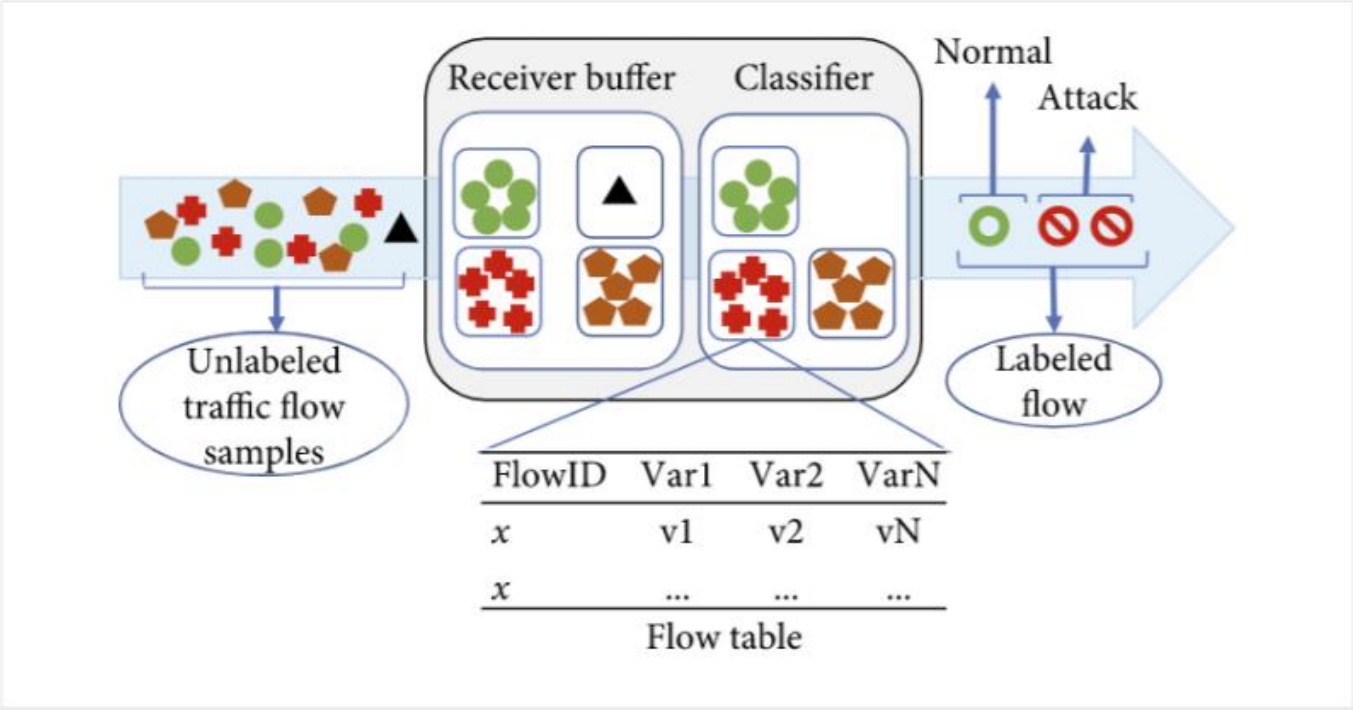
High Level Diagram

The core of the detection system consists of a Signature Dataset (SDS) (dataset being used form https://cloudstor.aarnet.edu.au/plus/apps/onlyoffice/s/N0JTc8JFNtZtUE4?fileId=5025371984 and a machine learning algorithm (MLA).



First, normal traffic and DDoS signatures were extracted, labeled, and stored in a database. SDS was then created using feature selection techniques. Finally, the

most accurate MLA was selected, trained, and loaded into the traffic classification system.



The architecture of the detection system was designed to work with samples of network traffic provided by industrial standard traffic sampling protocols, collected from network devices. The unlabeled samples are received and grouped in flow tables in the receiver buffer. Thus, when the table length is greater than or equal to the reference value, they are presented to the classifier responsible for labeling them. If the flow table expires, it may be processed one more time. The occurrence of small flow tables is higher at lower sampling rates or under some types of DoS attacks, e.g., SYN flood attacks.

Review 2

Modules' description and Implementation Modules:

Icmp_attack.ipynb: This is the script which implements algorithm for detection of DDoS attack in ICMP packet which is written by Gauransh.
Tcp_syn_attack.ipynb: This is the script which implements algorithm for detection of DDoS attack in TCP packet which is written by Ayushi.
Udp_attack.ipynb: This is the script which implements algorithm for detection of DDoS attack in UDP packet which is written by Gauransh.
Models used:

i. Logistic regression: Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression). It is used to obtain odds ratio in the presence of more than one explanatory variable. The procedure is quite similar to multiple linear regression, with the exception that the response variable is binomial. The result is the impact of each variable on the odds ratio of the observed event of interest.

ii. KNeighboursClassifier: KNeighborsClassifier implements classification based on voting by nearest k-neighbors of target point, t, while RadiusNeighborsClassifier implements classification based on all neighborhood points within a fixed radius, r, of target point, t. It works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes

for the most frequent label (in the case of classification) or averages the labels (in the case of regression).

iii. MLPClassifier: MLPClassifier stands for Multi-layer Perceptron classifier which in the name itself connects to a Neural Network. Unlike other classification algorithms such as Support Vectors or Naive Bayes Classifier, MLPClassifier relies on an underlying Neural Network to perform the task of classification. It is a class of feedforward artificial neural network (ANN). MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.

iv. Decision Tree Classifier: Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, decision tree algorithm can be used for solving regression and classification problems too. The general motive of using Decision Tree is to create a training model which can use to predict class or value of target variables by learning decision rules inferred from prior data (training data). The understanding level of Decision Trees algorithm is so easy compared with other classification algorithms.

Link for Review 2:

https://github.com/Goldy1906/CSE3502-Project_review/blob/master/REVIEW%202/Review%202.pdf

V. Plan for Review 3

Modules

DOS_detection_train.ipynb : This module is used to train the models that are created to find DDoS attack through different packages. It required around 70 percent of the dataset and stored the instant trained model for further use where time required will be the least. This module was used to extract the pretrained model and test the model on the parameters given with the 30 percent of the dataset. After training and testing, finally trained model will be saved in the trained_final_model.sav file.

trained_final_model.sav : This module contains the current state of trained model, which will run directly with calling this module with least possible loading time.

Since our dataset was old in initial review
http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
So we changed our dataset to
https://cloudstor.aarnet.edu.au/plus/apps/onlyoffice/s/N0JTc8JFNtZtUE4?fileId=5025371984
And atlast we finally added pickle library which stores the trained model in a new file and when we run the following file, it runs faster as it skips the training process because it is trained before only.

VI. Performance Analysis

For testing and verification of the trained model we used 30% of our dataset. (we divided our dataset in 70:30 ratio where 70 % of the data is for training and other 30 % is for testing and result verification)

```
Complete dataset:
Number of rows: 1199490
Number of columns: 11

Number of rows in training set: 803658
Number of rows in testing set: 395832
```

DATASET: NF-UQ-NIDS

Results

```
# Create dataframe to display results
results = {'Algorithms': ['Random Forest', 'Decision Tree', 'Logistic Regression', 'Gaussian Naive Bayes'],
           'Training time (in seconds)': [rf_train_time, dtree_train_time, logreg_train_time, gnb_train_time],
           'Testing time (in seconds)': [rf_test_time, dtree_test_time, logreg_test_time, gnb_test_time],
           'Training accuracy (%)': [rf_train_acc, dtree_train_acc, logreg_train_acc, gnb_train_acc],
           'Testing accuracy (%)': [rf_test_acc, dtree_test_acc, logreg_test_acc, gnb_test_acc]}
results_DF = pd.DataFrame(results)
results_DF
```

	Algorithms	Training time (in seconds)	Testing time (in seconds)	Training accuracy (%)	Testing accuracy (%)
0	Random Forest	70.408894	2.127437	99.6994	99.2108
1	Decision Tree	2.050298	0.028388	95.8311	95.8594
2	Logistic Regression	9.553495	0.013102	88.6418	88.6348
3	Gaussian Naive Bayes	0.254029	0.101007	65.2794	65.2860

As we can see here that accuracy for Random Forest is the best as compared to other algorithms but it's training and testing time is very large. So, from the above result table we took Decision Tree as our base model due to it's second highest accuracy and better training and testing time.

```
[15] filename='trained_final_model.sav'
     pickle.dump(dtree_model,open(filename,'wb'))
     print("Decision Tree's trained model Saved!!")

     Decision Tree's trained model Saved!!
```

Finally, pickle library uses Decision Tree model and saved its current state of training in the .sav file.

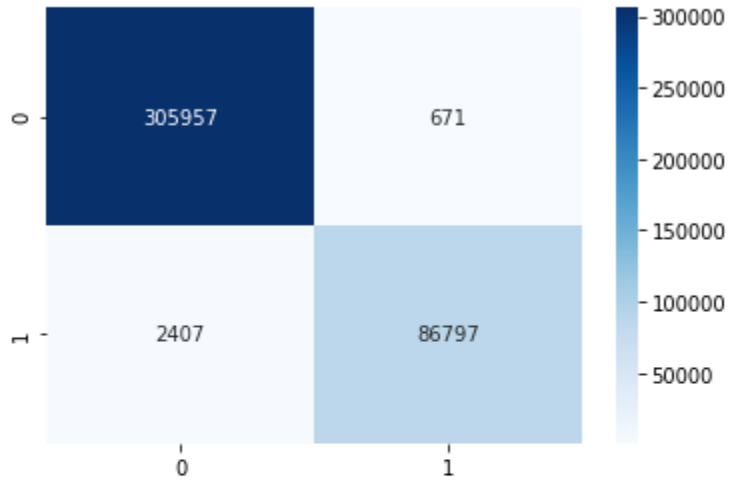
```
[20] test_start_time = time.time()
     loaded_model=pickle.load(open(filename,'rb'))
     test_end_time = time.time()
     model_loading_time = test_end_time-test_start_time
     result=loaded_model.score(X_test, Y_test)
     #Test Acuuracy should same as of Decision Tree
     #Loading time will be decreased
     print("Test accuracy : ")
     print(result)
     print("Total Loading Time: ")
     print(model_loading_time)

     Test accuracy :
     0.9585935447361507
     Total Loading Time:
     0.0036504268646240234
```

And then finally when we test the saved file it gave us the accuracy same as of Decision Tree, but the loading time was incredibly very less.

CLASSIFICATION REPORT,CONFUSION MATRIX AND F1 SCORE OF ALL THE ALGORTIHMS

RANDOM FOREST MODEL:

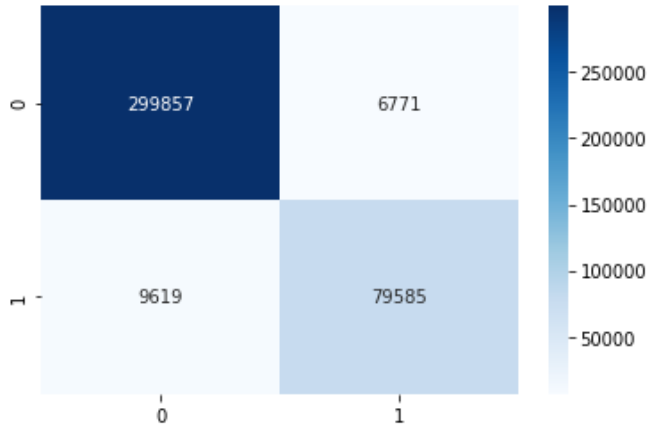
```
[12] Confusion Matrix for Random Forest model:
```




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

           0       0.99      1.00      0.99    306628
           1       0.99      0.97      0.98     89204

    accuracy                           0.99    395832
   macro avg       0.99      0.99      0.99    395832
weighted avg       0.99      0.99      0.99    395832
```

DECISION TREE MODEL

Confusion Matrix for Decision Tree model:

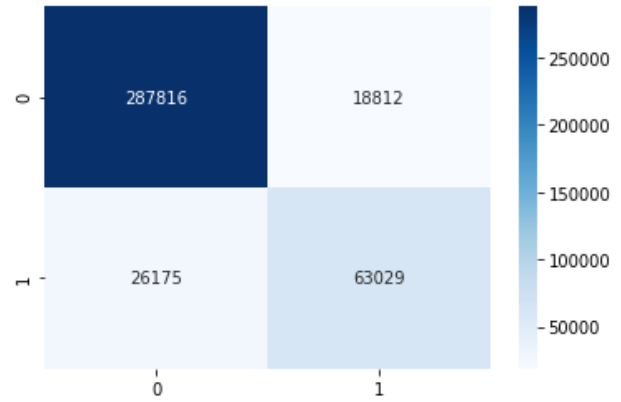


Classification report for Decision Tree model:

	precision	recall	f1-score	support
0	0.97	0.98	0.97	306628
1	0.92	0.89	0.91	89204
accuracy			0.96	395832
macro avg	0.95	0.94	0.94	395832
weighted avg	0.96	0.96	0.96	395832

LOGISTIC REGRESSION MODEL

Confusion Matrix for Logistic Regression model:

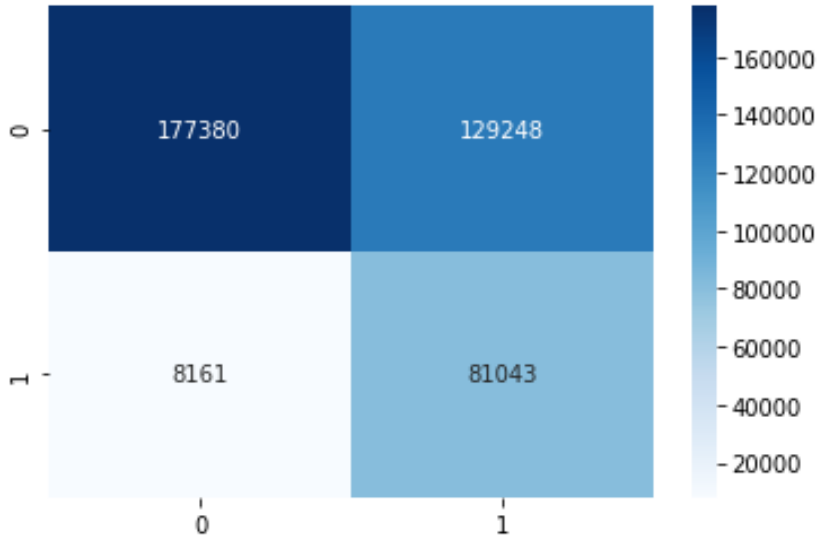


Classification report for Logistic Regression model:

	precision	recall	f1-score	support
0	0.92	0.94	0.93	306628
1	0.77	0.71	0.74	89204
accuracy			0.89	395832
macro avg	0.84	0.82	0.83	395832
weighted avg	0.88	0.89	0.88	395832

GAUSSIAN NAÏVE BAYES MODEL

[15] Confusion Matrix for Gaussian Naive Bayes model:



Classification report for Gaussian Naive Bayes model:

	precision	recall	f1-score	support
0	0.96	0.58	0.72	306628
1	0.39	0.91	0.54	89204
accuracy			0.65	395832
macro avg	0.67	0.74	0.63	395832
weighted avg	0.83	0.65	0.68	395832

FROM THE ABOVE MATRCES WE CAN SEE HOW WORST RESULTS ARE GIVEN BY GAUSSIAN NAÏVE BAYES MODEL AS COMPARED TO OTHER MODELS LIKE RANDOM FOREST AND DECISION TREE.

Code Link:
https://colab.research.google.com/drive/1ZhQSwwD3hjlWMlmMMvMN-EDi2lU4E1Fn#scrollTo=ZNw0zmjDlMlE

VII. Conclusion

We used the machine learning algorithms to detect the DoS/DDoS attacks. But it was taking too much time to run the model. Therefore, we used pickle library to save the trained model. This way anyone who uses this model will not be needed to train the model and can directly use it.

Finally, we see that the Random Forest model achieves the highest accuracy on both training and tesing datasets.

However, the Decision Tree model is the quickest in terms of training and testing time. It also gives fairly high accuracy. Hence, the best model out of the 4 is the Decision Tree model as it achieves high accuracy while also minimizing training and testing time.

Such a model can be used in intrusion detection systems to make these systems more robust and intelligent in detecting malicious attacks such as Denial of Service attacks.

GITHUB LINK:

https://github.com/Goldy1906/CSE3502-Project_review

REVIEW 2 CODE (GOOGLE COLAB LINK):

https://colab.research.google.com/drive/16vIZkPfsnFc9Bvvqx_5Gm8tPlDs-m6A5?usp=sharing

REVIEW 2 CODE (GOOGLE COLAB LINK):

https://colab.research.google.com/drive/1ZhQSwwD3hjlWMImMMvMN-EDi2lU4E1Fn?usp=sharing

VIII. References

[1] X. Yuan, C. Li and X. Li, "DeepDefense: Identifying DDoS Attack via Deep Learning," 2017 IEEE International Conference on Smart Computing (SMARTCOMP), Hong Kong, China, 2017, pp. 1-8, doi: 10.1109/SMARTCOMP.2017.7946998.

[2] He, Z., Zhang, T., & Lee, R. B. (2017, June). Machine learning based DDoS attack detection from source side in cloud. In *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)* (pp. 114-120). IEEE.

[3] Zekri, M., El Kafhali, S., Aboutabit, N., & Saadi, Y. (2017, October). DDoS attack detection using machine learning techniques in cloud computing environments. In *2017 3rd International Conference of Cloud Computing Technologies and Applications (CloudTech)* (pp. 1-7). IEEE.

[4] Zhang, B., Zhang, T., & Yu, Z. (2017, December). DDoS detection and prevention based on artificial intelligence techniques. In *2017 3rd IEEE International Conference on Computer and Communications (ICCC)* (pp. 1276-1280). IEEE.

[5] Zhou, B., Li, J., Wu, J., Guo, S., Gu, Y., & Li, Z. (2018, May). Machine-learning-based online distributed denial-of-service attack detection using spark streaming. In *2018 IEEE International Conference on Communications (ICC)* (pp. 1-6). IEEE.

[6] Aamir, M., Zaidi, S.M.A. DDoS attack detection with feature engineering and machine learning: the framework and performance evaluation. *Int. J. Inf. Secur.* 18, 761–785 (2019).

[7] Smys, S. (2019). DDOS attack detection in telecommunication network using machine learning. *Journal of Ubiquitous Computing and Communication Technologies (UCCT)*, *1*(01), 33-44.

[8] Tuan, T.A., Long, H.V., Son, L.H. *et al.* Performance evaluation of Botnet DDoS attack detection using machine learning. *Evol. Intel.* 13, 283–294 (2020)

[9] Sahoo, K. S., Tripathy, B. K., Naik, K., Ramasubbareddy, S., Balusamy, B., Khari, M., & Burgos, D. (2020). An evolutionary SVM model for DDOS attack detection in software defined networks. *IEEE Access*, *8*, 132502-132513.

[10] Wang, M., Lu, Y., & Qin, J. (2020). A dynamic MLP-based DDoS attack detection method using feature selection and feedback. *Computers & Security*, *88*, 101645.

[11] Sen, S., Gupta, K. D., & Ahsan, M. M. (2020). Leveraging machine learning approach to setup software-defined network (SDN) controller rules during DDoS attack. In *Proceedings of International Joint Conference on Computational Intelligence* (pp. 49-60). Springer, Singapore.

[13] Tuan, N. N., Hung, P. H., Nghia, N. D., Tho, N. V., Phan, T. V., & Thanh, N. H. (2020). A DDoS attack mitigation scheme in ISP networks using machine learning based on SDN. *Electronics*, *9*(3), 413.

[14] Pande, S., Khamparia, A., Gupta, D., & Thanh, D. N. (2021). DDOS Detection Using Machine Learning Technique. In *Recent Studies on Computational Intelligence* (pp. 59-68). Springer, Singapore.

[15] Dehkordi, A. B., Soltanaghaei, M., & Boroujeni, F. Z. (2021). The DDoS attacks detection through machine learning and statistical methods in SDN. *The Journal of Supercomputing*, *77*(3), 2383-2415.

[16]Swami, R., Dave, M., & Ranga, V. (2021). DDoS attacks and defense mechanisms using machine learning techniques for SDN. In *Research Anthology on Combating Denial-of-Service Attacks* (pp. 248-264). IGI Global.