INSTITUTE FOR ADVANCED COMPUTING AND SOFTWARE DEVELOPMENT AKURDI, PUNE

DOCUMENTATION ON

"DDoS Attack Detection and Analysis"

PG-DBDA May-2021

Submited by:

Group No. :10

Mane Snehal D.(1330)

Bandapalle Mahadevi B.(1326)

Mr. Prashant Karhale

Centre Coordinator

Mr.Akshay Tilekar

Project Guide

INDEX

Table of Contents

List of Figures

List of Tables

List of Screenshots

1 Abstract

Distributed denial-of-service (DDoS) attack is a rapidly growing threat to today's Internet. Significant research works have been done in this area. It is vital to incorporate the latest research works in academic program to provide training and education to students and professionals for cyber security. In this paper, we present the design and implementation of a senior design project named DDoS Attack, Detection and Defense Simulation. We aim to build a test bed and configure the network environment to simulate the "real-world" DDoS attack, detection and defense. We study several DDoS attack tools, as well as some commonly-used DDoS detection and defense software. We perform extensive tests, collect and analyze the experimental data, and draw our conclusions. This is an on-going project. Some preliminary results have been reported here. The purpose of this project is to help students to apply their technical skills and knowledge on a "real world" project, and gain better understanding and more hands-on experience on Internet security, especially DDoS attack, detection and defense mechanisms.

2 Introduction

Network security is a topic gaining tremendous interests in today's Information Technology world. The increasing frequency and severity of network attacks in recent years reveal some fundamental security issues of Internet environment. Significant efforts from university and industry have been made to improve computer and network security. It is vital to incorporate the latest research results in higher education and academic programs to provide training and education to college students and cyber security professionals. College seniors in Computer Network & System Administration (CNSA) program [1] at Michigan Technological University are required to complete a capstone senior design project during their final year. The senior design project affords students the opportunity to apply their individual technical skills and knowledge on a real world project, as well as develop their problem solving skills, communication skills, and teamwork skills. DDoS attack has become a rapidly growing threat to today's Internet. A large number of DDoS detection and defense mechanisms have been proposed to combat the problem. In this paper, we present the design and implementation of an Information Technology senior design project named DDoS Attack, Detection and Defense Simulation. In this project, we aim to set up test bed and configure the network environment to simulate the "real-world" DDoS attack, detection and defense mechanism. We test several DDoS attack software, as well as some leading DDoS detection and defense products and tools. The purpose of this project is to help student gain better understanding and more hands-on experience on Internet security, especially DDoS attack, detection and defense mechanisms.

2.1. Background

A DDoS attack is one in which a multitude of compromised computer systems attack a selected target, thereby causing denial of service for legitimate users of the targeted system. The flood of incoming traffic to the target system essentially forces it to shut down, thereby denying service to users. Figure 1 shows a typical DDoS attack [2]. A hacker begins a DDoS attack by exploiting vulnerability in a computer system and making it the DDoS "master". From the master system, the intruder identifies and communicates with other systems that can be compromised also. The intruder loads DDoS attack tools on those compromised systems. The intruder can instruct the controlled machines to launch one of many flood attacks against a specified target. The inundation of packets to the target causes a denial of service. Some DDoS attacks utilize Internet worms to automate the process of exploiting and compromising computer systems, as well as launching DDoS attacks. In general, DDoS defense research can be roughly categorized into four areas: intrusion prevention, intrusion detection, intrusion response, and intrusion tolerance. Intrusion prevention focuses on stopping attacks before attack packets reach the target victim. Intrusion detection explores the various techniques used to detect attack incidents as they occur. Intrusion response investigates various

techniques to handle an attack once the attack is discovered. Intrusion tolerance responds to attacks by minimizing the attack impact. Handler (Middleman) Agent.

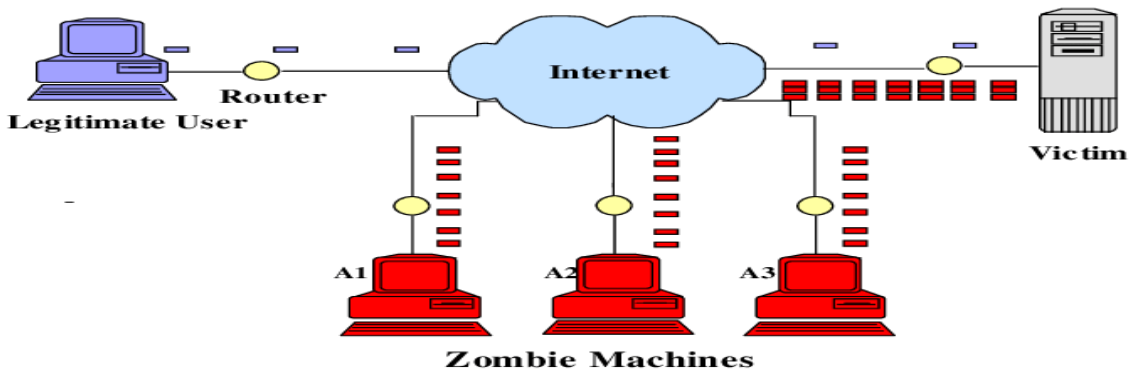


Fig 1.1 ddos introduction

2.2 Scope of the project

2.2.1 Initial functional requirement will be:-

- Selecting the algorithm meeting requirement.
- Choosing the optimum algorithm form set of algorithm.
- Testing it on the datasets.
- After getting the result if the result is low change the hyperparameters.
- Out of all result get best of all.

2.2.2 Initial nonfunctional requirement will be:-

- Getting the large datasets which can provide developer enough data to train the model.
- Maintain the minimum variance& bias so the model is successfully work.
- Avoid the underfitting and overfitting.

Terms	Definitions
Dataset	Data for training and testing for the model
Variance	Difference between the training and testing accuracy
Bias	Both learning and training accuracy is low
Overfitting	Model is very complex
Underfitting	Model is bias
Developer	Who is developing the model
Review	A written recommendation about the appropriateness of an Product for

	selling and buying may include suggestions for improvement.
Reviewer	A person that examines an Product and has the ability to recommend approval Product for buying or to request that changes be made in the Product.
Software Requirement Specification	A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. For example, this document
User	Reviewer

Table 2.2.1: Terms and Definitions.

2.3 Software Life Cycle Model:

In order to make this Project we are going to use Classic LIFE CYCLE MODEL .Classic life cycle model is also known as WATER FALL MODEL. The life cycle model demands a Systematic sequential approach to software development that begins at the system level and progress through analysis design coding, testing and maintenance.

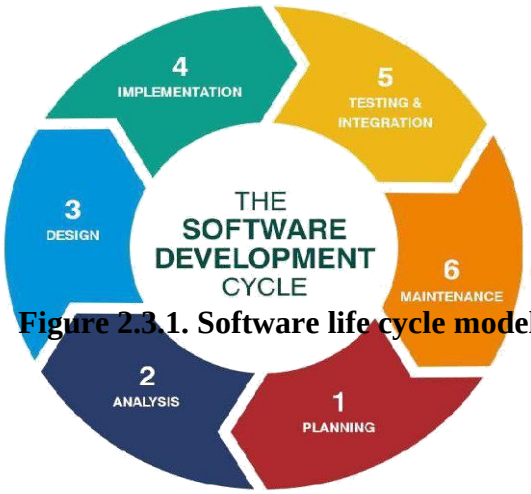


Figure 2.3.1. Software life cycle model

The Classic Life Cycle Model

The waterfall model is sequential software development process, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of conception initiation, Analysis, Design (validation), construction. Testing and maintained.

1) **System Engineering and Analysis:**

Because software is always a part of larger system work. Begins by establishing requirement for all system elements and Then allocating some subset of these requirement to the software system Engineering and analysis encompasses the requirement gathering at the system level with a small amount of top level design and analysis.

2) Software requirement Analysis:

The requirement gathering process is intensified and focused specifically on the software Requirement for the both system and software are discussed and reviewed with the customer. The customer specifies the entire requirement or the software and the function to be performed by the software.

3) Design:

Software design is actually a multi-step process that focuses on distinct attributes of the program data structure, software architecture, procedural detail and interface of the software that can be assessed or quality before coding begins .Like requirement the design is documented and becomes part of the software.

4) Coding:

The design must be translated into a machine readable form. The coding step performs this task. If design is programmed in a detailed manner, coding can be accomplished mechanically.

5) Testing:

Once code has been generated programmed testing begins. The testing process focuses on the internals of the software ensuring that all statement have been tested and on the functional externals hat is conducting tests to uncover the errors and ensure that defined input will produce the results that agree with the required results.

5.1 Unit testing:

In computer programming, Unit testing is software Verification and validation method where the programmer gains confidence that individual units of source code are fit to use A unit is the smallest testable part of an application. In procedural programming a unit may be an individual programmed, function, procedure, etc. while in object-oriented programming, the smallest Unit is a class, which may belong to a base/super class abstract class or derived/child class.

5.2 Benefits:

The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. A unit test provides a strict written contract that the piece of code must satisfy.

5.3 Documentation:

Unit testing provides a sort of living documentation of the system. Developers looking to learn what functionality is provided by a unit and how to use it can look at the tests to gain a basic understanding of the unit API.

5.3.1 Limitation of unit testing:

Testing cannot be expected to catch error in the program –It is impossible to evaluate all execution paths for all but the most trivial programs. The same is true for unit testing. Additionally, by unit testing only types the functionality if the units themselves.

6) Maintenance:

Software will undoubtedly undergo change after it is Delivered to the customer .Change will occur because errors have been encountered because the software must be able adopted to accommodate changes in its external environment because the customer requires functional or performance enhancement enhancements. The classic life cycle is the oldest and most widely used paradigm or software engineering.

2.3 Aims & Objectives

The primary goal of this project is to extract quality patterns from the stackoverflow question dataset and use the trained models to predict any question given by the user on stackoverflow. Before giving the question the training of the models will be done using a bunch of different ML models and after the training is done the ML models will be compared based on their accuracy score and f1-score and the best model will be selected which will then be used to make prediction's for the question given by the user. The prediction which is given by our system will give the user the quite insight of the question which he/she wants to post on stackoverflow.com.

3 Overall Description

3.1 Workflow of Project:

The diagram below shows the workflow of this project.

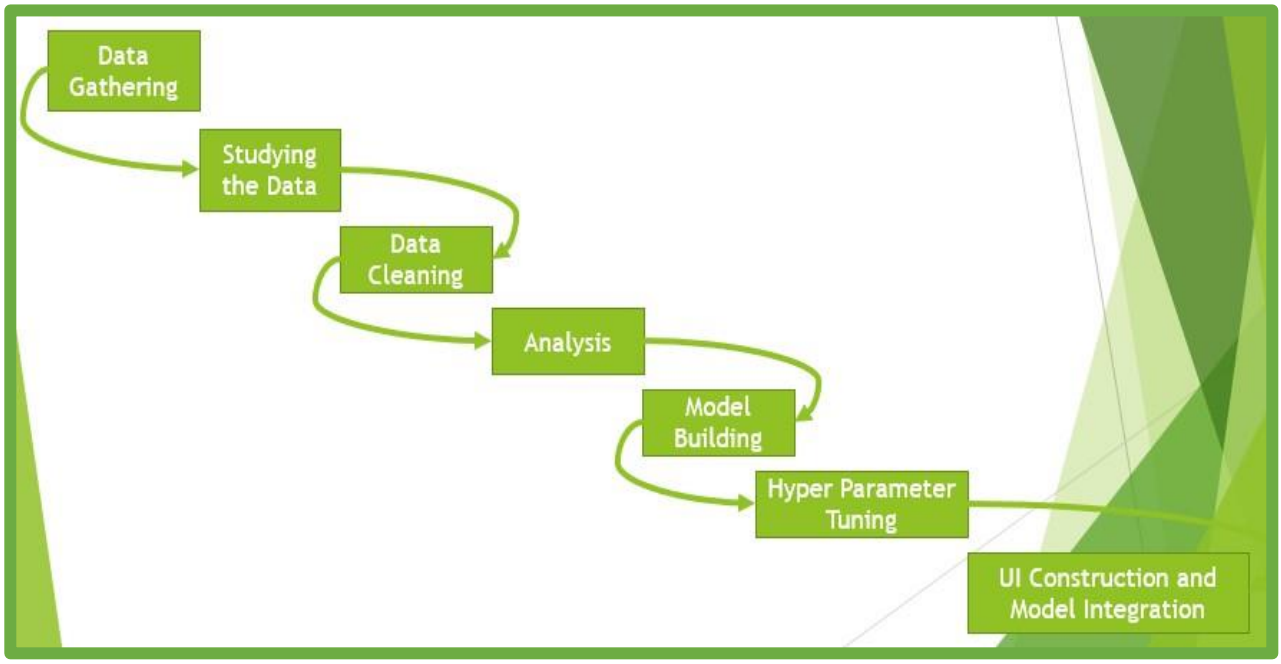


Figure 1Workflow Diagram

3.2 Data Preprocessing and Cleaning:

3.2.1 Data Cleaning:

The data can have many irrelevant, missing parts, null values, NaN and infinity values. To handle this part, data cleaning is done.

3.2.2 Label encoding:

To make the data understandable or in human readable form, the training data is often labeled in words. Label Encoding refers to converting the labels into numeric form so as to convert it into the machine-readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated.

3.3 Exploratory Data Analysis:

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

Fig. correlation

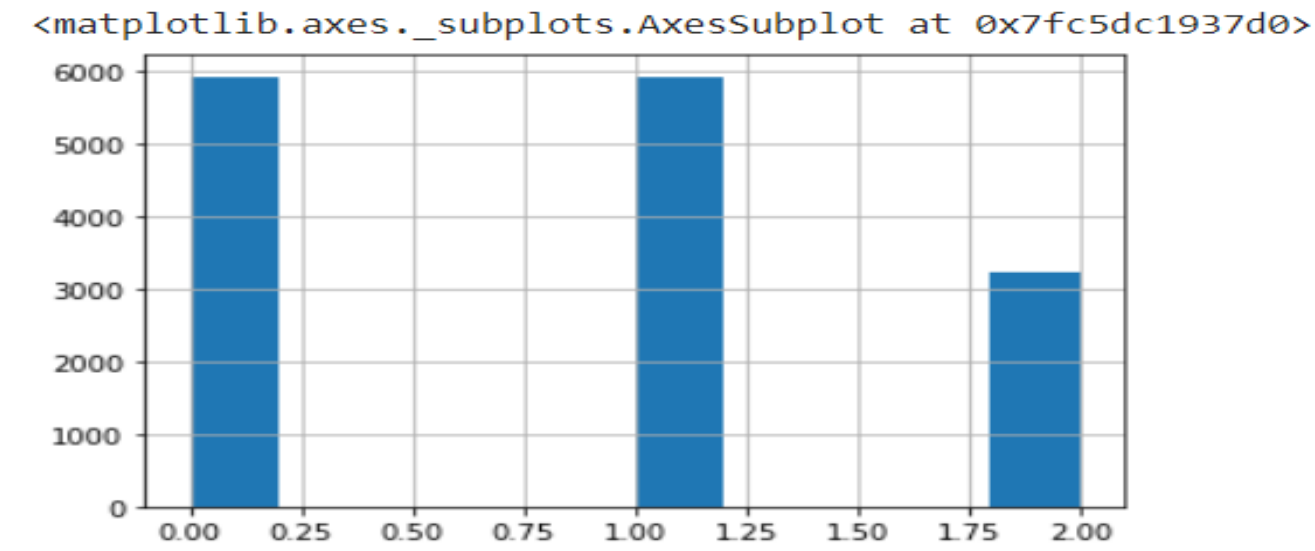


Fig. label visualization.

3.4 Model Building:

1. Train/Test split:

One important aspect of all machine learning models is to determine their accuracy. Now, in order to determine their accuracy, one can train the model using the given dataset and then predict the response values for the same dataset using that model and hence, find the accuracy of the model. A better option is to split our data into two parts: first one for training our machine learning model, and second one for testing our model.

- Split the dataset into two pieces: a training set and a testing set.
- Train the model on the training set.
- Test the model on the testing set, and evaluate how well our model did.

Advantages of train/test split:

- Model can be trained and tested on different data than the one used for training.
- Response values are known for the test dataset, hence predictions can be evaluated
- Testing accuracy is a better estimate than training accuracy of out-of-sample performance.

Machine learning consists of algorithms that can automate analytical model building. Using algorithms that iteratively learn from data, machine learning models facilitate computers to find hidden insights from Big Data without being explicitly programmed where to look.

We have used the following three algorithms to build predictive model.

2. Logistic Regression:

Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Logistic regression is used to describe data and to

explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

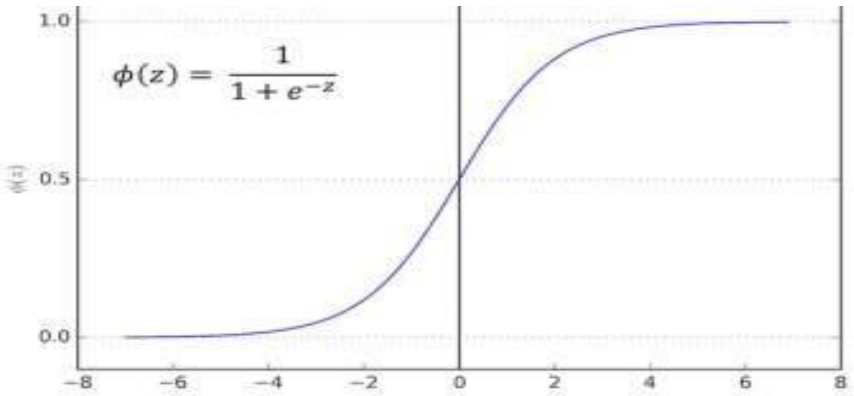


Fig. logistic regression

K nearest neighbor:

K Nearest Neighbor Algorithm. K nearest neighbor algorithm is very simple. It works based on minimum distance from the query instance to the training samples to determine the K-nearest neighbors. The data for KNN algorithm consist of several multivariate attributes name that will be used to classify.

k' in KNN is a parameter that refers to the number of nearest neighbors to include in the majority of the voting process.

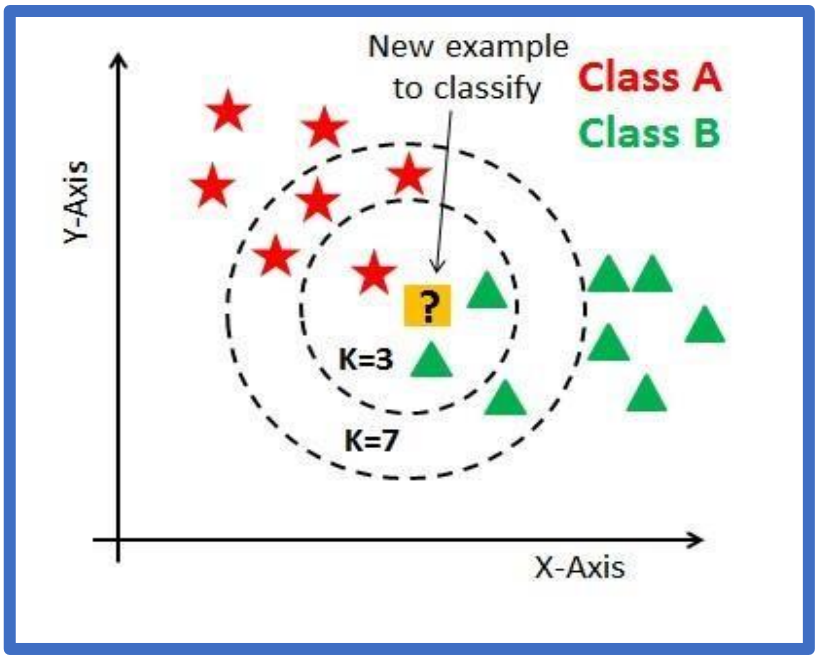
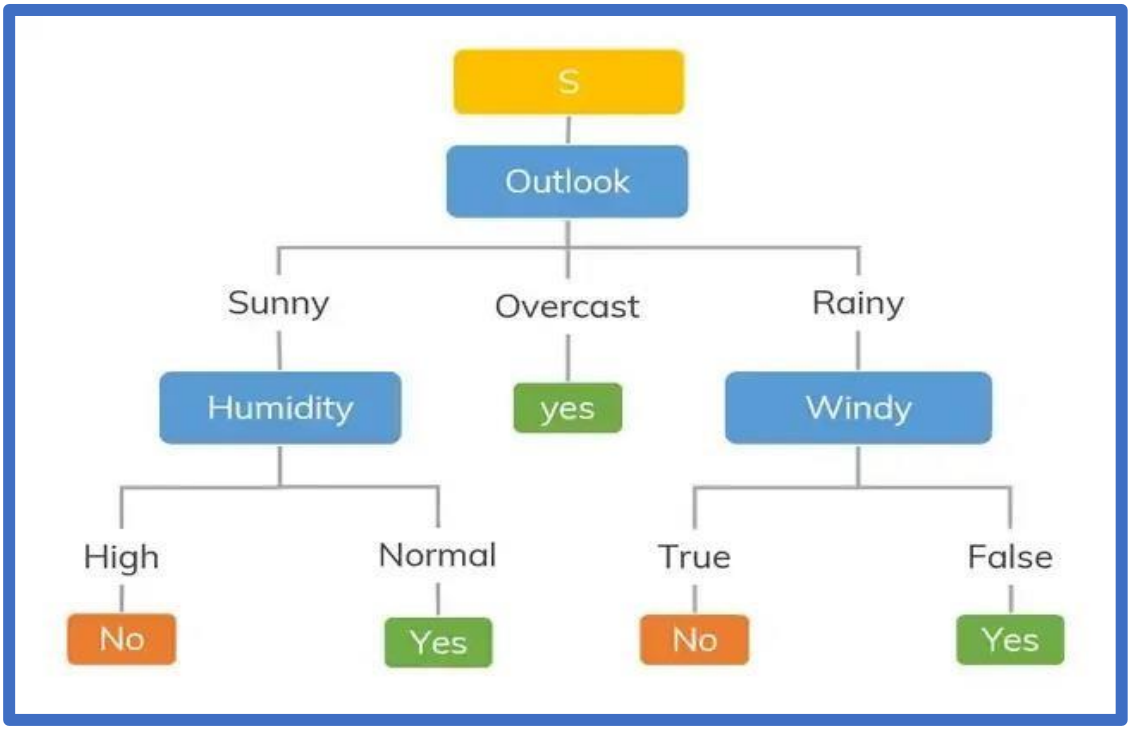


Figure 14

Decision Tree:

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). Each internal node of the tree corresponds to an attribute, and each leaf node corresponds to a class label.

Outlook	Temperature	Humidity	Windy	Play Golf
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No



Random Forest Model:

Random forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

Put simply: random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

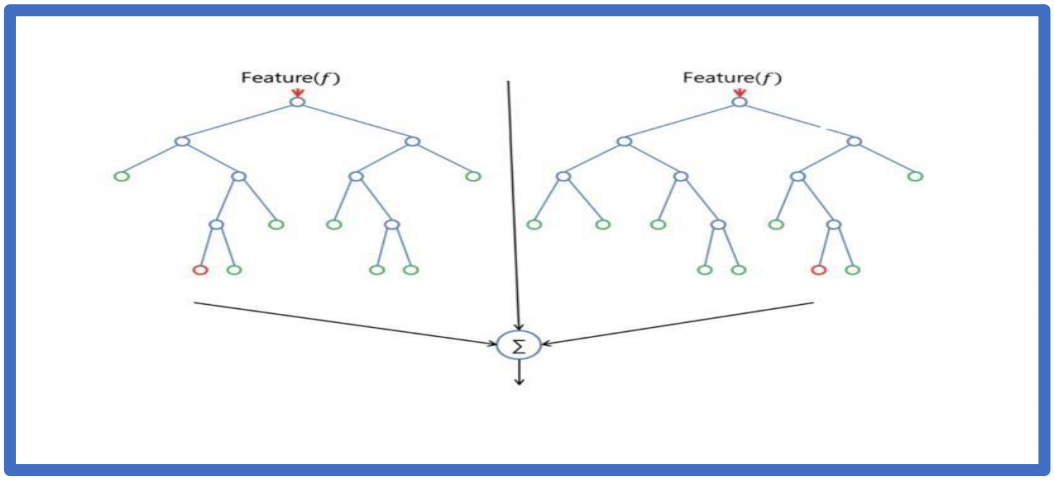


Figure. Shows the working of the random forest model

Gradient Boost:

Gradient boosting is a greedy algorithm and can overfit a training dataset quickly. It can benefit from regularization methods that penalize various parts of the algorithm and generally improve the performance of the algorithm by reducing overfitting..

Adaboost:

Adaptive Boosting, or most commonly known AdaBoost, is a Boosting algorithm. This algorithm uses the method to correct its predecessor. It pays more attention to under fitted training instances by the previous model. Thus, at every new predictor the focus is more on the complicated cases more than the others.

It fits a sequence of weak learners on different weighted training data. It starts by predicting the original data set and gives equal weight to each observation. If prediction is incorrect using the

first learner, then it gives higher weight to observation which have been predicted incorrectly. Being an iterative process, it continues to add learner(s) until a limit is reached in the number of models or accuracy.

Voting classifier:

Voting classifier is a machine learning model that trains on an ensemble of numerous models and predicts an output (class) based on their highest probability of chosen class as the output.

It simply aggregates the findings of each classifier passed into Voting Classifier and predicts the output class based on the highest majority of voting. The idea is instead of creating separate dedicated models and finding the accuracy for each them, we create a single model which trains by these models and predicts output based on their combined majority of voting for each output class.

Confusion Matrix:

- The confusion matrix provides us a matrix/table as output and describes the performance of the model.
- It is also known as the error matrix.
- The matrix consists of predictions result in a summarized form, which has a total number of correct predictions and incorrect predictions. The matrix looks like as below table:

○	**Actual Positive**	**Actual Negative**
Predicted Positive	True Positive	False Positive
Predicted Negative	False Negative	True Negative

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{Total Population}}$$

4 Requirement specification

4.1 Hardware Requirement:

500 GB hard drive (Minimum requirement)

8 GB RAM (Minimum requirement)

PC x64-bit CPU

4.2 Software Requirement:

Windows/Mac/Linux

Python-3.9.1

Google Colab

Libraries:

- Numpy 1.18.2
- Pandas 1.2.1
- Matplotlib 3.3.3
- Scikit-learn 0.24.1

Google Colab:
Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser,with
Zero configuration required
Free access to GPUs
Easy sharing
Colab notebooks are Jupyter notebooks that are hosted by Colab.

Pandas:
Pandas is a popular Python library for data analysis. It is not directly related to Machine Learning. As we know that the dataset must be prepared before training. In this case, Pandas comes handy as it was developed specifically for data extraction and preparation. It provides high-level data structures and wide variety tools for data analysis. It provides many inbuilt methods for groping, combining and filtering data.

NumPy :
NumPy is a very popular python library for large multi-dimensional array and matrix processing, with the help of a large collection of high-level mathematical functions. It is very useful for fundamental scientific computations in Machine Learning. It is particularly useful for linear algebra, Fourier transform, and random number capabilities. High-end libraries like TensorFlow uses NumPy internally for manipulation of Tensors.

Seaborn:
Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures.
Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them.

Matplotlib:
Matplotlib is a very popular Python library for data visualization. Like Pandas, it is not directly related to Machine Learning. It particularly comes in handy when a programmer wants to visualize the patterns in the data. It is a 2D plotting library used for creating 2D graphs and plots. A module named pyplot makes it easy for programmers for plotting as it provides features to control line styles, font properties, formatting axes, etc. It provides various kinds of graphs and plots for data visualization, viz., histogram, error charts, bar chats, etc,

ScikitLearn:
Scikit-learn is one of the most popular ML libraries for classical ML algorithms. It is built on top of two basic Python libraries, viz., NumPy and SciPy. Scikit-learn supports most of the supervised and unsupervised learning algorithms. Scikit-learn can also be used for data-mining and data-analysis, which makes it a great tool who is starting out with ML.
The OS module in python provides functions for interacting with the operating system. OS, comes under Python's standard utility modules. This module provides a portable way of using operating system dependent functionality

5. System Design

Project Design and Implementation This project is divided into three phases. The first phase of this project is to build a DDoS attack network and simulate the DDoS attacks. We test some existing DDoS attack tools, like StacheldrahtV4 . We also develop several simple DDoS attack programs by ourselves. For example, programs that can launch ping flood attack or UDP attack. On the victim network, we use Apache web server and RealPlayer Multimedia Server . The students can observe how DDoS attacks affect the normal users and normal traffic.

The second phase of this project is to set up DDoS Intrusion Detection and Defense systems. We use Snort , as well as several other leading products in market. We perform extensive tests and compare these DDoS defense products. We design our own defense mechanism by integrating Snort with Firewall and router to enable effective rate-limiting and QoS provisioning. This requires students to have good understanding on TCP/IP, iptable , firewall and router. The last part of this project is to further the studies on DDoS attacks. For example, there is a new type of DDoS attack called degrading DDoS attacks, or non-disruptive DDoS attacks. This type of DDoS attack consumes a large portion of victim network resources but does not stop the network services completely. The traditional DDoS defense mechanisms react poorly to degrading DDoS attacks. We also try to combine Internet worms with DDoS .

FLOW:

6.Results

Output:

Logistic regression:

```
============================== LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='multinomial', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False) Model Evaluation ==============================
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
Accuracy of  model is:  82.87248856176646
Confusion Matrix:
 [[1851  116   10]
 [ 656 1321    0]
 [   9   70  994]]
Report:
              precision    recall  f1-score   support

           0       0.74      0.94      0.82      1977
           1       0.88      0.67      0.76      1977
           2       0.99      0.93      0.96      1073

    accuracy                           0.83      5027
   macro avg       0.87      0.84      0.85      5027
weighted avg       0.85      0.83      0.83      5027
```

Screenshot : logistic regression

2.KNN:

```
============================== KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform') Model Evaluation ==============================
Accuracy of  model is:  98.66719713546847
Confusion Matrix:
 [[1944    8   25]
 [  16 1957    4]
 [  10    4 1059]]
Report:
              precision    recall  f1-score   support

           0       0.99      0.98      0.99      1977
           1       0.99      0.99      0.99      1977
           2       0.97      0.99      0.98      1073

    accuracy                           0.99      5027
   macro avg       0.98      0.99      0.99      5027
weighted avg       0.99      0.99      0.99      5027
```

Screenshot : KNN

3.Decision tree:

```
============================== 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') Model Evaluation ==============================
Accuracy of  model is:  99.98010741993237
Confusion Matrix:
 [[1977    0    0]
 [   1 1976    0]
 [   0    0 1073]]
Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1977
           1       1.00      1.00      1.00      1977
           2       1.00      1.00      1.00      1073

    accuracy                           1.00      5027
   macro avg       1.00      1.00      1.00      5027
weighted avg       1.00      1.00      1.00      5027
```

Screenshot: decision tree

4.Gradient boost:

```
============================== GradientBoostingClassifier(ccp_alpha=0.0, criterion='friedman_mse', init=None,
                           learning_rate=1.0, loss='deviance', max_depth=3,
                           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, n_estimators=100,
                           n_iter_no_change=None, presort='deprecated',
                           random_state=7, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0,
                           warm_start=False) Model Evaluation ==============================
Accuracy of  model is:  99.96021483986472
Confusion Matrix:
 [[1976    0    1]
 [   1 1976    0]
 [   0    0 1073]]
Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1977
           1       1.00      1.00      1.00      1977
           2       1.00      1.00      1.00      1073

    accuracy                           1.00      5027
   macro avg       1.00      1.00      1.00      5027
weighted avg       1.00      1.00      1.00      5027
```

Screenshot :gradient boost

5.adaboost:

```
============================== AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None, learning_rate=1.0,
                   n_estimators=100, random_state=7) Model Evaluation ==============================
Accuracy of  model is:  99.9403222597971
Confusion Matrix:
 [[1975    1    1]
 [   1 1976    0]
 [   0    0 1073]]
Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1977
           1       1.00      1.00      1.00      1977
           2       1.00      1.00      1.00      1073

    accuracy                           1.00      5027
   macro avg       1.00      1.00      1.00      5027
weighted avg       1.00      1.00      1.00      5027
```

Screenshot : adaboost

6.Random Forest:

```
============================== 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=7, verbose=0,
                       warm_start=False) Model Evaluation ==============================
Accuracy of  model is:  99.98010741993237
Confusion Matrix:
 [[1977    0    0]
 [   1 1976    0]
 [   0    0 1073]]
Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1977
           1       1.00      1.00      1.00      1977
           2       1.00      1.00      1.00      1073

    accuracy                           1.00      5027
   macro avg       1.00      1.00      1.00      5027
weighted avg       1.00      1.00      1.00      5027
```

Screenshot: Random forest

7.voting classifier:

```
============================== VotingClassifier(estimators=[('dc',
                               DecisionTreeClassifier(ccp_alpha=0.0,
                                                      class_weight=None,
                                                      criterion='gini',
                                                      max_depth=33,
                                                      max_features=7,
                                                      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=20,
                                                      splitter='random')),
                              ('knn',
                               KNeighborsClassifier(algorithm='auto',
                                                    leaf_size=30,
                                                    metric='minkowski',
                                                    metric_params=None,
                                                    n_jobs=None, n_neighbors=3,
                                                    p=2, weights='uniform'))],
                 flatten_transform=True, n_jobs=None, voting='hard',
                 weights=[2, 2]) Model Evaluation ==============================
Accuracy of  model is:  99.36343743783569
Confusion Matrix:
 [[1977    0    0]
 [  16 1959    2]
 [  10    4 1059]]
Report:
               precision    recall  f1-score   support

           0       0.99      1.00      0.99      1977
           1       1.00      0.99      0.99      1977
```

7.Conclusion

Denial of Service is currently the most expensive computer crime for victim organizations.

- Strategic firewall placement allows companies to use the Internet during a DDoS attack, and it allows them to continue receiving the packets they want.
- Distributed Denial of Service Attacks could be Detected by Monitoring the Source IP.
- It is easy to generate a successful DDoS attack that bypasses these defenses

8.Future Scope

- As the data is being analyzed the prediction of a recorded system been attacked gets furthermore accurate.
- And also we can predict which system should be blacklisted on the basis of attacks it has performed.

9.References

- https://www.comptia.org/content/guides/what-is-a-ddos-attack-how-it-works
- http://www.usenix.org/events/lisa2000/full_papers/dietrich/dietrich_html/
- https://ieeexplore.ieee.org/document/5401280
- https://www.hindawi.com/journals/scn/2018/9804061/