1. Introduction

1.1 Introduction

A distributed denial-of-service (DDoS) attack is an attack in which multiple compromised computer systems attack a target, such as a server, website or other network resource, and cause a denial of service for users of the targeted resource. It is a cyber-attack in which the perpetrator seeks to make a machine or network resource unavailable to its intended users by temporarily or indefinitely disrupting services of a host connected to the Internet. Bots, or Internet robots, are also known as spiders, crawlers, and web bots. While they may be utilized to perform repetitive jobs, such as indexing a search engine, they often come in the form of malware. Malware bots are used to gain total control over a computer. Typically, bots perform tasks that are both simple and structurally repetitive, at a much higher rate than would be possible for a human alone. Botnets are the millions of systems infected with malware under hacker control in order to carry out DDoS attacks. These bots or zombie systems are used to carry out attacks against the target systems, often overwhelming the target system‟s bandwidth and processing capabilities. These DDoS attacks are difficult to trace because botnets are located in differing geographic locations.

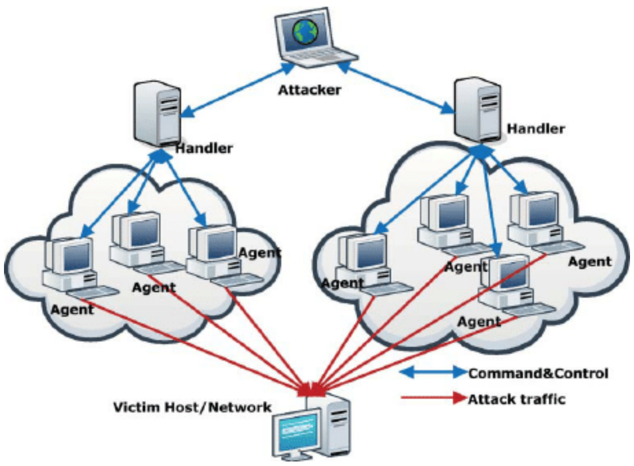


Fig 1.1- Overview of DDoS Attack

1.2 Overview Of the project

A distributed denial-of-service (DDoS) is a large-scale DoS attack where the perpetrator uses more than one unique IP address, often thousands of them. Thus, this detection of DDOS attack plans to detect the IP addresses and block the attack by working on various algorithms like Random Forest algorithm and packet sniffing. The packet sniffer module will capture the incoming packets for certain period, preprocess them, analyze them, and passes the dataset to analyzing module. The analysis module checks for the bandwidth flooding and connection flooding from the captured data. The graph or charts are displayed based on the analysis so that the administrator can easily identify the attacker. This system is thus very useful to the users and a network administrator in particular who is generally responsible for monitoring things on a network. Openstack is used to set-up the system which can have these modules for intrusion detection.

2. Literature Survey

2.1 Existing System

1. *"DDoS Attack Detection System: Utilizing Classification Algorithms with Apache Spark*", 9th IFIP International Conference, 2018 [1]
   a. Fuzzy logic system using parallel computing for classification.
   b. Accuracy of DT(Gini) 97.9 % and DT (Entropy) 97.6 %.

2. "*Detection DDoS attacks Based on Neural Network Using Apache Spark",* International Conference, 2016 [2]

   a. Back-propagation network and implemented in Apache Spark cluster.
   b. Used 2000 DARPA LLDOS 1.0 dataset in a real time environment and accuracy obtained was 94%.

3. "*DeepDefense: Identifying DDoS Attack via Deep Learning*", IEEE International Conference, 2017 [3]

   i. Recurrent deep neural network approach to identify DDoS attack.
   ii. UNB ISCX Intrusion Detection Evaluation 2012 Dataset is used.
   iii. This approach reduced the error rate from 7.517% to 2.103% compared with conventional machine learning method.
4. "*Machine-Learning-Based Online Distributed Denial-of-Service Attack Detection Using Spark Streaming*", IEEE International Conference, 2018 [4]

   i. Naive Bayes, Logistic Regression and Decision Tree algorithms are used in a cluster.
   ii. The system can detect 3 typical DDoS attacks TCP flooding, UDP flooding and ICMP flooding.
   iii. The classification was made with the Accuracy of 99.3%.
5. "*DDoS Detection System: Utilizing Gradient Boosting Algorithm and Apache Spark*", IEEE Canadian Conference, 2018 [5]

   a. Gradient Boosting classification algorithm (GBT) to identify DDoS attack.
   b. Apache Spark Processing Engine for processing data.
   c. The classification was made with the accuracy of 97%.

3.System Environment

3.1 Hardware Requirements

- Minimum 8GB RAM
- 512GB Hard disk
- Minimum core i3 processor

3.2 Software Requirements

- VMWare Workstation
- Ubuntu 16.04 LTS/Windows 8 & above
- Anaconda 3.6.0
- Python 3.6.0
- Jupyter notebook

4. System Requirement Specification

4.1 Introduction

A DDoS attack is an attack on system's resources, launched from a large number of other host machines that are infected by malicious software controlled by the attacker.

4.1.1 Purpose

Cyber security is the practice of defending computers and servers, mobile devices, networks, and data from malicious attacks. It is also known as information technology security. Cyber security relies on cryptographic protocols used to encrypt emails, files. This security protects information that transmits and guards against loss. In the end, user security software scans computers for malicious code and then removes it from the machine.

4.1.2 Scope

Our Project will be capable of detecting intrusion in network using benchmark and real time dataset.

4.1.3 Definitions, Acronyms & abbreviations

- DDoS- Distributed Denial of Service

4.2 General Description

4.2.1 Product Perspective

Aim is to provide a definite guideline on effective solution building and detailed solution requirements to help the cyber security research community in designing defense mechanisms. To the best of our knowledge, this work is a novel attempt to identify the need of DDoS mitigation solutions with effective resource management during the attack. We present a comprehensive solution with a detailed insight into the detection, and mitigation mechanisms of these attacks.

4.2.2 Product functions

- The system shall be able to process the incoming network traffic.
- The system shall be able to process the data in a distributed environment using Python IDE.
- The system shall be able to differentiate between normal and anomalous connections in the network.

4.2.3 User Characteristics

Clients-When a client outsources a server to a cloud vendor, he/she also relinquishes a large degree of control over that infrastructure, resulting in a risk to the client's details. Many organizations have rigorous regulatory security requirements, and cloud computing is not robust enough at the point to sufficiently satisfy those at the proper level. Protection from DDoS attacks is one of the provision provided to the clients.

4.3 Specific Requirements

4.3.1 Functional Requirements

Introduction:-DDoS attacks affect the normal functioning of system. And block the legitimate users from access to information. Hence detecting the Algorithm with highest accuracy of detecting the attacks.

Inputs:-In our project we are using Bench Mark Data Set KDDCup data set as input

Processing: - One-Hot-Encoding is used to transform all categorical features into binary features. The One-Hot-encoding takes a matrix of integers, denoting the values on by categorical features. The output will be a sparse matrix where each column corresponds to one possible value of one feature. Therefore the features first need to be transformed with Label Encoder, to transform every category to a number. In the second step, we will be applying various machine and deep learning algorithms. These algorithms analyze the large datasets and mechanism which show the intrusion in the given KDD Cup dataset with different accuracies.

Outputs

Sl.No	Algorithm Name	Accuracy
1.	Naïve Bayes Algorithm	98.78 %
2.	Decision Tree Algorithm	99.53 %
3.	Logistic Regression Algorithm	99.46 %

4.3.2 Non Functional Requirements

Type	Description
Performance	1.The system should be able to classify anomalies and normal packets with the accuracy of more than 95%. 2.The pre-processing time of the intrusion detection system should be within seconds.
Usability	1.The system should be available all the time.

5. System Definition

5.1 System Design

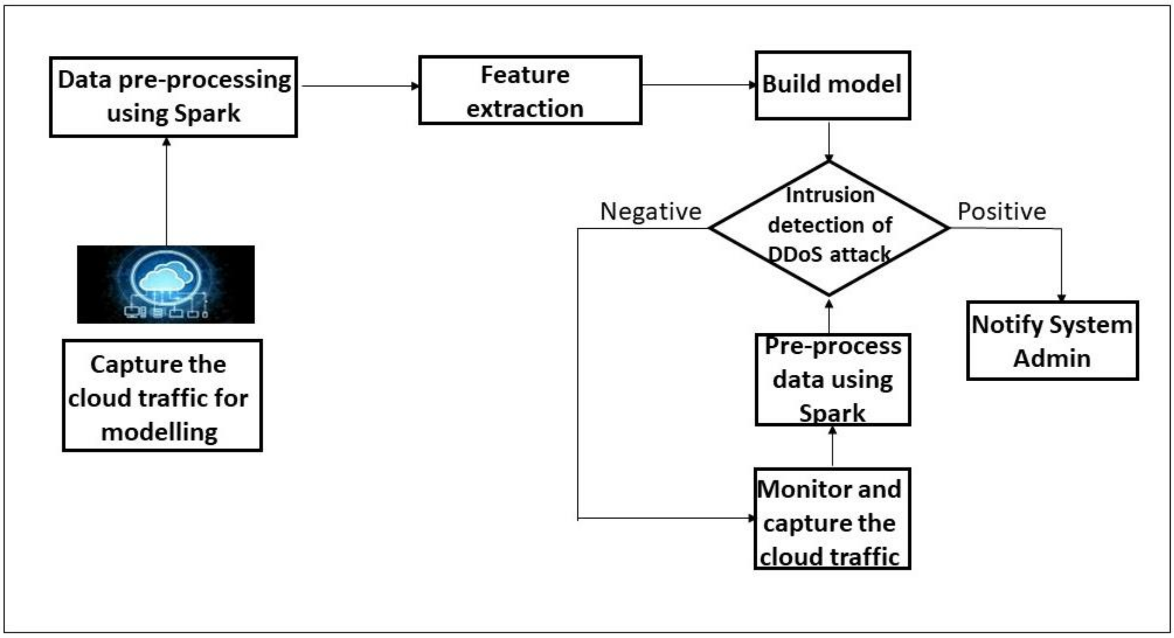


Figure 5.1 Architecture diagram

5.2 Data Flow Diagram

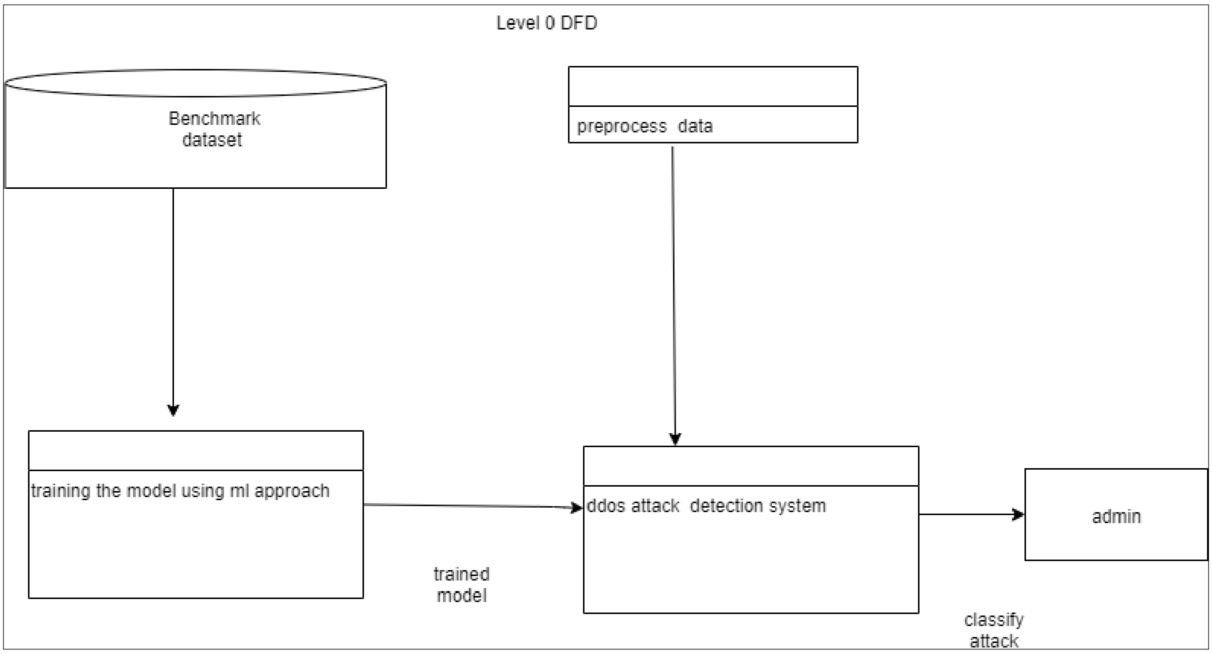


Figure 5.2(a) Level 0 Data Flow Diagram

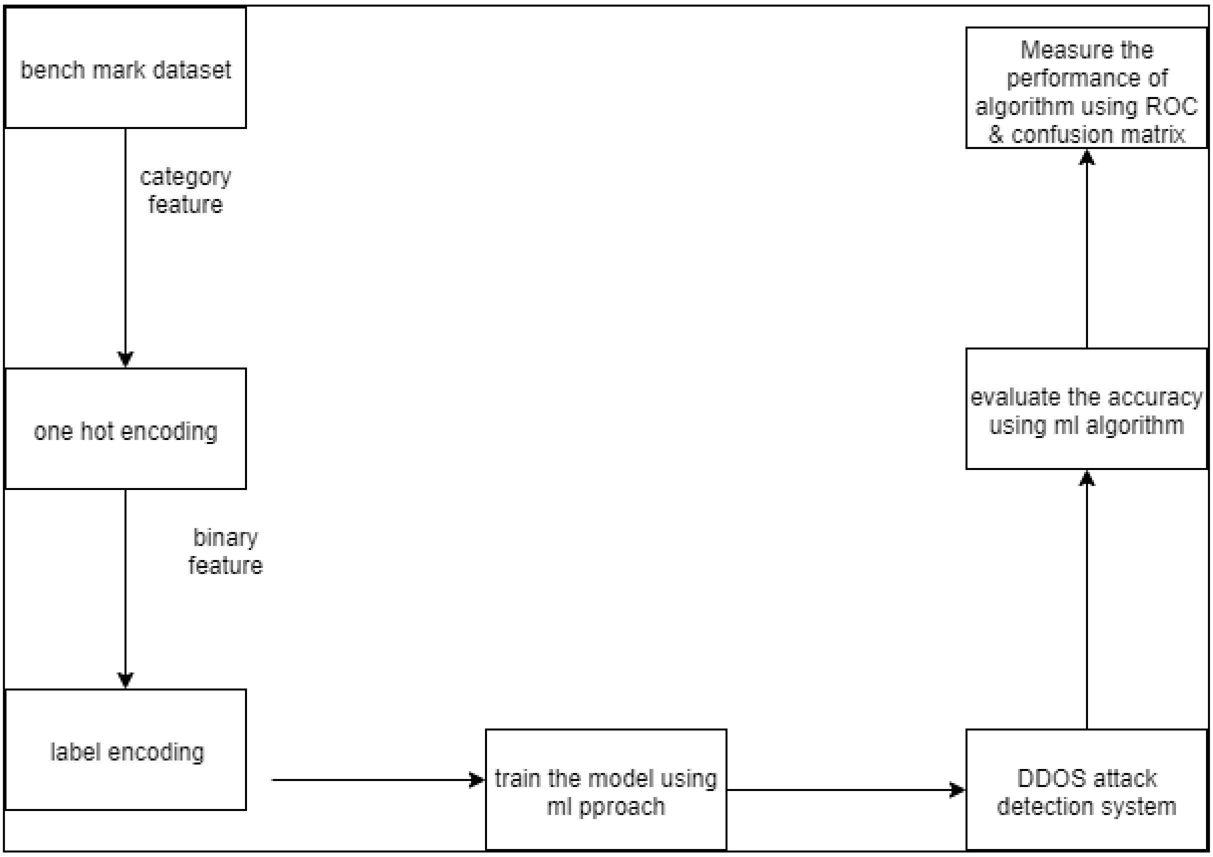


Figure 5.2(b) Level 1 Data Flow Diagram

5.3 Use Case Diagram

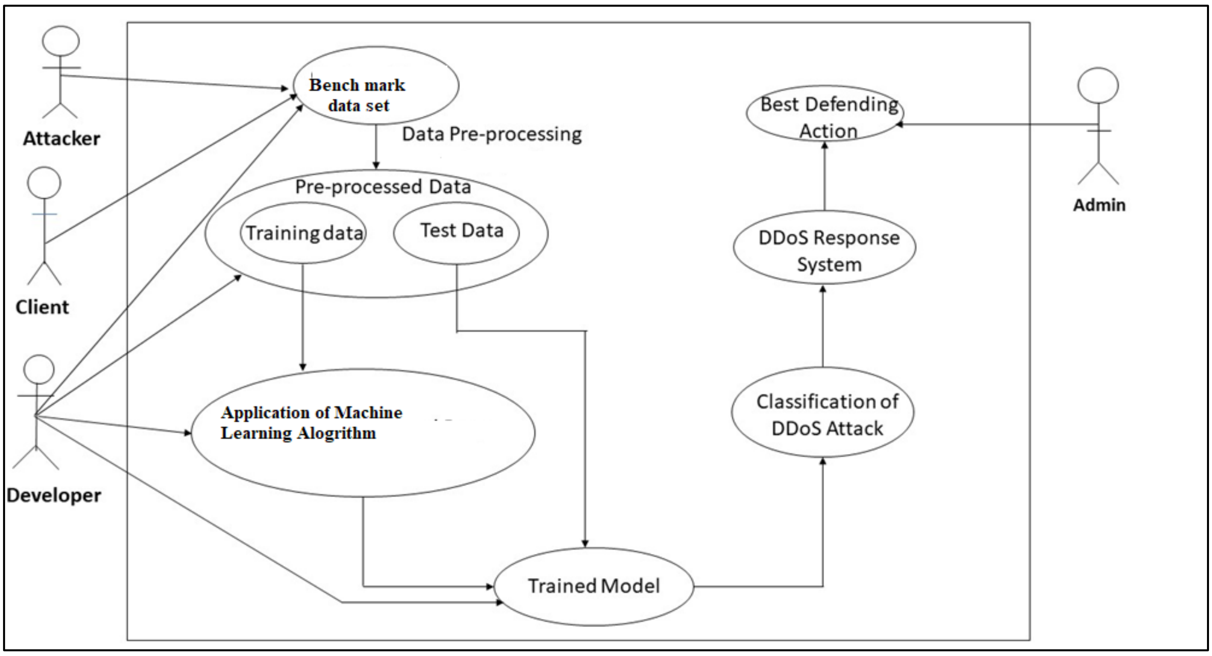


Figure 5.3 Use Case Diagram

5.4 Data-set Contents:

Feature No.	Feature Name	Description
1.	Count	No. of connections to the same host as the current connection in the last two seconds
2.	destination bytes	Bytes sent from destination to source
3.	diff srv rate	percentage of connections to different services
4.	dst host count	count of connections having the same destination hosts
5.	dst host diff srv rate	percentage of different services on the current host
6.	dst host rerror rate	percentage of connections to the current host that has an RST error
7.	dst host same src port rate	percentage of connections to the current host having the same src port
8.	dst host same srv rate	percentage of connections having the same destination host and using the same service
9.	dst host serror rate	percentage of connections to the current host that have an S0 error
10.	dst host srv count	count of connections having the same destination host and using the same service
11.	dst host srv diff host rate	percentage of connections to the same service coming from different hosts
12.	dst host srv rerror rate	percentage of connections to the current host and specified service that have an RST error
13.	dst host srv serror rate	percentage of connections to the current host and specified service that have an S0 error
14.	Duration	Duration of the active connection.
15.	Flag	Status flag of the connection
16	Hot	No. of "hot" indicators
17.	is guest login	One if the login is a "guest." login; Otherwise 0
18.	is host login	One if the login belongs to the "host"; otherwise 0
19.	Land	One if the connection is from/to the same host/port; Otherwise 0
20.	logged in	One if successfully logged in; otherwise 0
21.	num access files	No. of operations on

		access control files
22.	num compromised	No. of compromised conditions
23	num failed logins	No. of failed logins
24.	num file creations	No. of file creation operations
25.	num outbound cmds	No. of outbound commands in an ftp session
26.	num root	No. of "root" accesses
27.	num shells	No. of shell prompts
28.	protocol type	Connection protocol (e.g. tcp, udp).
29.	rerror rate	percentage of connections that have “REJ" Errors
30.	root shell	One if the root shell is obtained; otherwise 0
31.	same srv rate	percentage of connections to the same service
32.	serror rate	percentage of connections that have “SYN" Errors
33.	Service	Destination service (e.g. telnet, ftp)
34.	src bytes	Bytes sent from source to destination
35.	srv count	No. of connections to the same service as the current connection in the last two seconds
36.	srv diff host rate	percentage of connections to different hosts
37.	srv rerror rate	percentage of connections that have “REJ" errors
38.	srv serror rate	percentage of connections that have “SYN" Errors
39.	su attempted	One if "su root" command attempted; otherwise 0
40.	Urgent	No. of urgent packets
41.	Wrong fragment	No. of wrong fragments

5.7 Detailed Design (Flow Chart/Algorithm/Pseudo code)

6. Implementation

6.1 Introduction

Anaconda

Anaconda is a freemium open source distribution of the Python and R programming languages for large-scale data processing, predictive analytics, and scientific computing, that aims to simplify package management and deployment. Its package management system is conda. Anaconda distribution comes with more than 1,000 data packages as well as the Conda package and virtual environment manager, called Anaconda Navigatorso it eliminates the need to learn to install each library independently.

Jupyter

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebooks documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context.

Python

Python is an interpreted, high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. In July 2018, Van Rossum stepped down as the leader in the language community. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms. It includes object oriented, imperative, functional and procedural, and has a large and comprehensive standard library. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of Python's other implementations. Python and CPython are managed by the non-profit Python Software Foundation.

6.2 Algorithms used for Implementation

ALGORITHMS USED:

1. Linear Regression

It is used to estimate real values (cost of houses, number of calls, total sales etc.) based on continuous variable(s). Here, we establish relationship between independent and dependent variables by fitting a best line. This best fit line is known as regression line and represented by a linear equation:

$Y = a * X + b$.

In this equation:

- Y – Dependent Variable
- a – Slope
- X – Independent variable
- b – Intercept

The best way to understand linear regression is to relive this experience of childhood. Let us say, you ask a child in fifth grade to arrange people in his class by increasing order of weight, without asking them their weights! What do you think the child will do? He / she would likely look (visually analyze) at the height and build of people and arrange them using a combination of these visible parameters. This is linear regression in real life! The child has actually figured out that height and build would be correlated to the weight by a relationship, which looks like the equation above.

2. Decision Tree

This is one of my favorite algorithm and I use it quite frequently. It is a type of supervised learning algorithm that is mostly used for classification problems. Surprisingly, it works for both categorical and continuous dependent variables. In this algorithm, we split the population into two or more homogeneous sets. This is done based on most significant attributes/ independent variables to make as distinct groups as possible

3.Naive Bayes Algorithm

It is a classification technique based on Bayes' theorem with an assumption of independence between predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Naive Bayesian model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c).

- P(c|x) is the posterior probability of class (target) given predictor (attribute).
- P(c) is the prior probability of class.
- P(x|c) is the likelihood which is the probability of predictor given class.
- P(x) is the prior probability of predictor**.**

6.3 Source Code:

```
### Reading the dataset
import pandas as pd
import matplotlib.pyplot as plt
import csv
import numpy as np
import matplotlib.pyplot as plt
%pylab inline
data =pd.read_csv("KDDCup99.csv")
data.head()
data.transpose()

#Finding missing data
def num_missing(x):
    return sum(x.isnull())

#Applying per column:
print("Missing values per column:")
print(data.apply(num_missing, axis=0)) #axis=0 defines that
function is to be applied on each column

#Different types of attacks in the database
data['label'].value_counts()



#creating a CSV file with attribute names
with open('KDDCup99-preprocessed.csv', 'w',newline='') as out:
    writer=csv.writer(out)

writer.writerow(['duration','protocol_type','service','flag','src_bytes','dst_bytes
','land','wrong_fragment','urgent',

'hot','num_failed_logins','logged_in','lnum_compromised','lroot_shell','lsu_attempt
ed','lnum_root','lnum_file_creations',

'lnum_shells','lnum_access_files','lnum_outbound_cmds','is_host_login','is_guest_lo
gin','count','srv_count','serror_rate',

'srv_serror_rate','rerror_rate','srv_rerror_rate','same_srv_rate','diff_srv_rate','
srv_diff_host_rate','dst_host_count',

'dst_host_srv_count','dst_host_same_srv_rate','dst_host_diff_srv_rate','dst_host_sa
me_src_port_rate','dst_host_srv_diff_host_rate',

'dst_host_serror_rate','dst_host_srv_serror_rate','dst_host_rerror_rate','dst_host_
srv_rerror_rate','label'])
```

```
#Extracting DoS Attacks from dataset
a=['back','land','neptune','pod','smurf','teardrop','normal']
with open('KDDCup99.csv', 'r') as inp, open('KDDCup99-preprocessed.csv',
'a',newline='') as out:
    writer = csv.writer(out)
    for row in csv.reader(inp):
        if row[41] in a:
            writer.writerow(row)
#Creating new dataframe with modified dataset
%pylab inline
datapre=pd.read_csv("KDDCup99-preprocessed.csv")
datapre.head()
datapre.transpose()
datapre.dtypes
datapre['label'].value_counts()
Mapping of Categorical data in the dataset
cleanup_nums={"protocol_type":{"tcp":1,"icmp":2,"udp":3},"serv
ice": {"vmnet": 1, "smtp": 2, "ntp_u":3, "shell":4,
"kshell":5, "aol":6, "imap4":7, "urh_i":8, "netbios_ssn":9,
                                  "tftp_u":10, "mtp":11, "uucp":12,
"nnsp":13, "echo":14, "tim_i":15, "ssh":16, "iso_tsap":17,
"time":18,
                                  "netbios_ns":19,"systat":20,
"hostnames":21, "login":22, "efs":23, "supdup":24,
"http_8001":25, "courier":26,

"ctf":27,"finger":28,"nntp":29,"ftp_data":30,"red_i":31,"ldap"
:32,"http":33,"ftp":34,"pm_dump":35,"exec":36,

"klogin":37,"auth":38,"netbios_dgm":39,"other":40,"link":41,"X
11":42,"discard":43,"private":44,"remote_job":45,

"IRC":46,"daytime":47,"pop_3":48,"pop_2":49,"gopher":50,"sunrp
c":51,"name":52,"rje":53,"domain":54,"uucp_path":55,

"http_2784":56,"Z39_50":57,"domain_u":58,"csnet_ns":59,"whois"
:60,"eco_i":61,"bgp":62,"sql_net":63,"printer":64,

"telnet":65,"ecr_i":66,"urp_i":67,"netstat":68,"http_443":69,"
harvest":70},

"flag":{"RSTR":1,"S3":2,"SF":3,"RSTO":4,"SH":5,"OTH":6,"S2":7,
"RSTOS0":8,"S1":9,"S0":10,"REJ":11},
                    "label":{"normal":0, "back":1, "land":2,
"neptune":3, "pod":4, "smurf":5, "teardrop":6}}
#Replacing the encoded data in the dataset
datapre.replace(cleanup_nums, inplace=True)
datapre.head()
datapre.transpose()

datapre.dtypes
cor_mat=datapre.corr()
cor_mat
```

```
import seaborn as sns
plt.rcParams['figure.figsize'] = [18, 12]
corr_heat = sns.heatmap(cor_mat,annot=True)
plt.title('Variable Correlations')

datapre.drop('duration', axis = 1, inplace = True)
print("Dropping duration")
datapre.drop('flag', axis = 1, inplace = True)
print("Dropping flag")
datapre.drop('land', axis = 1, inplace = True)
print("Dropping land")
datapre.drop('wrong_fragment', axis = 1, inplace = True)
print("Dropping wrong_fragment")
datapre.drop('urgent', axis = 1, inplace = True)
print("Dropping urgent")
datapre.drop('hot', axis = 1, inplace = True)
print("Dropping hot")
datapre.drop('num_failed_logins', axis = 1, inplace = True)
print("Dropping num_failed_logins")
datapre.drop('logged_in', axis = 1, inplace = True)
print("Dropping logged_in")
datapre.drop('lnum_compromised', axis = 1, inplace = True)
print("Dropping lnum_compromised")
datapre.drop('lroot_shell', axis = 1, inplace = True)
print("Dropping lroot_shell")
datapre.drop('lsu_attempted', axis = 1, inplace = True)
print("Dropping lsu_attempted")
datapre.drop('lnum_root', axis = 1, inplace = True)
print("Dropping lnum_root")
datapre.drop('lnum_file_creations', axis = 1, inplace = True)
print("Dropping lnum_file_creations")
datapre.drop('lnum_shells', axis = 1, inplace = True)
print("Dropping lnum_shells")
datapre.drop('lnum_access_files', axis = 1, inplace = True)
print("Dropping lnum_access_files")
datapre.drop('lnum_outbound_cmds', axis = 1, inplace = True)
print("Dropping lnum_outbound_cmds")
datapre.drop('is_host_login', axis = 1, inplace = True)
print("Dropping is_host_login")
datapre.drop('is_guest_login', axis = 1, inplace = True)
print("Dropping is_guest_login")
datapre.drop('serror_rate', axis = 1, inplace = True)
print("Dropping serror_rate")
datapre.drop('srv_serror_rate', axis = 1, inplace = True)
print("Dropping srv_serror_rate")
datapre.drop('rerror_rate', axis = 1, inplace = True)
print("Dropping rerror_rate")
datapre.drop('srv_rerror_rate', axis = 1, inplace = True)
print("Dropping srv_rerror_rate")
datapre.drop('diff_srv_rate', axis = 1, inplace = True)
print("Dropping diff_srv_rate")
datapre.drop('srv_diff_host_rate', axis = 1, inplace = True)
print("Dropping srv_diff_host_rate")
datapre.drop('dst_host_count', axis = 1, inplace = True)
print("dst_host_count")
datapre.drop('dst_host_srv_count', axis = 1, inplace = True)
print("Dropping dst_host_srv_count")
datapre.drop('dst_host_diff_srv_rate', axis = 1, inplace = True)
print("Dropping dst_host_diff_srv_rate")
datapre.drop('dst_host_srv_diff_host_rate', axis = 1, inplace = True)
print("Dropping dst_host_srv_diff_host_rate")
datapre.drop('dst_host_serror_rate', axis = 1, inplace = True)
print("Dropping dst_host_serror_rate")
datapre.drop('dst_host_srv_serror_rate', axis = 1, inplace = True)
print("Dropping dst_host_srv_serror_rate")
datapre.drop('dst_host_rerror_rate', axis = 1, inplace = True)
print("Dropping dst_host_rerror_rate")
datapre.drop('dst_host_srv_rerror_rate', axis = 1, inplace = True)
```

```
print("Dropping dst_host_srv_rerror_rate")
datapre.drop('dst_host_same_srv_rate', axis = 1, inplace = True)
print("Dropping dst_host_same_srv_rate")
datapre.drop('dst_host_same_src_port_rate', axis = 1, inplace = True)
print("Dropping dst_host_same_src_port_rate")
datapre.label.unique()
datapre.to_csv('finalKDD.csv')
datapre.head()
#del(datapre)
import pandas as pd
datapre=pd.read_csv('./finalKDD.csv')
sampleset1=datapre
sampleset2=sampleset1
sampleset2=sampleset2.append(datapre)
sampleset2.shape
sampleset1.shape
#Sampling 50% of the data with replacement
#from sklearn.cross_validation import train_test_split
# Generate the sampleset from training set.  Set random_state
to be able to replicate results.
sampleset1 = datapre.sample(frac=0.4, random_state=1)
sampleset1

#We will work with this dataset with the x feature-
object matrix and values of the y target variable.
array=sampleset2.values
x=array[:,0:7]
print(x)
y=array[:,7]
print(y)
x.shape
y.shape
#Cross validation
from sklearn import model_selection
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
## Spliting of training dataset into 70% training data and 30% testing data
randomly
features_train, features_test, labels_train, labels_test =
model_selection.train_test_split(x, y, test_size=0.2, random_state=0)
features_train.shape
labels_test.shape
#Array for storing classifier Models and their respective scores
models=[]
scores=[]
#Naive Bayes Classifier
from sklearn import metrics
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(features_train)

features_train = scaler.transform(features_train)
features_test = scaler.transform(features_test)
labels_test
```

```
#Naive Bayes Classifier
accu=[]
prec=[]
Rcal=[]
F_Sco=[]
from sklearn.metrics import precision_recall_fscore_support as score
from sklearn import metrics
from sklearn import tree
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import multilabel_confusion_matrix
from sklearn.metrics import mean_squared_error
clf = GaussianNB()
features_train=features_train.astype(int)
labels_train=labels_train.astype(int)
labels_test=labels_test.astype(int)
#training the model using training set
clf.fit(features_train , labels_train)
#predicting the label using test set on trained Model
prediction = clf.predict(features_test)
#calculating accuracy
acc=accuracy_score(prediction, labels_test)
accu.append(acc*100)
print("-------------------------------------------")

accuracy=float(acc*100)
print("Accuracy  : ",accuracy," %")
mat = multilabel_confusion_matrix(labels_test, prediction)
#print(mat)
print("Report is : ")

precision,recall,fscore,support=score(labels_test, prediction)
prec.append(precision[1])
Rcal.append(recall[1])
F_Sco.append(fscore[1])
print("Precision :",(precision[1]*100),"%")
print("Recall    :",(recall[1]*100),"%")
print("F-Score   :",(fscore[1]*100),"%")
print("Support   :",support[1])
models.append("Naive Bayes Classifier")
scores.append(acc*100)
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
num_folds = 7
seed = 1
kfold = KFold(n_splits=num_folds, random_state=seed)
model = clf
results = cross_val_score(model, features_test , labels_test, cv=kfold)
print(results)
rs1=results.mean()*100.0
print("The results mean of cross_val_score is",rs1)
rs2= results.std()*100.0
print("The results std of cross_val_score is",rs2)
```

```
#logical regression
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import multilabel_confusion_matrix
classifier = LogisticRegression(random_state = 0)
features_train=features_train.astype(int)
labels_train=labels_train.astype(int)
classifier.fit(features_train, labels_train)
features_test=features_test.astype(int)
labels_test=labels_test.astype(int)
y_pred = classifier.predict(features_test)
acc=accuracy_score(y_pred, labels_test)
#accu.append(acc*100)
logical_acc=acc*100
print("Accuracy  : ",logical_acc," %")
from sklearn.metrics import confusion_matrix
cm = multilabel_confusion_matrix(labels_test, y_pred)
print(cm)
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.metrics import precision_recall_fscore_support as score
precision,recall,fscore,support=score(labels_test,y_pred)
#prec.append(precision[1])
#Rcal.append(recall[1])
#F_Sco.append(fscore[1])
print("Presicion :",(precision[1]*100),"%")
print("Recall    :",(recall[1]*100),"%")
print("F-Score   :",(fscore[1]*100),"%")
print("Support   :",support[1])
models.append("Logistic Regression")
scores.append(acc*100)
```

```
#decision tree
accu=[]
prec=[]
Rcal=[]
F_Sco=[]
from sklearn.tree import DecisionTreeClassifier
features_train=features_train.astype(int)
labels_train=labels_train.astype(int)
classifier=DecisionTreeClassifier(criterion='entropy',random_state=0)
dct=classifier.fit(features_train,labels_train)

# Predicting the Test set results
labels_test=labels_test.astype(int)
y_pred = dct.predict(features_test)
y_pred=y_pred.astype(int)
acc=accuracy_score(y_pred,labels_test)
accu.append(acc*100)
dession_tree=acc*100
print("Accuracy  : ",dession_tree," %")
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = multilabel_confusion_matrix(labels_test, y_pred)
#print(cm)
models.append("Decision Tree Classifier")
#core.append(acc*100)
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.metrics import precision_recall_fscore_support as score
precision,recall,fscore,support=score(labels_test,y_pred)
prec.append(precision[1])
Rcal.append(recall[1])
F_Sco.append(fscore[1])
print("--------------------------------------")
print("|Presicion :",(precision[1]*100),"%                  |")
print("|Recall    :",(recall[1]*100),"%     |")
print("|F-Score   :",(fscore[1]*100),"%     |")
print("|Support   :",support[1],"%                  |")
print("--------------------------------------")
scores.append(acc*100)




#WRITE A CODE USING PYTHON TO SENDs A MAIL USING SMTP LIBRARY
import smtplib
s=smtplib.SMTP('smtp.gmail.com',587)
#start TLS for security
s.starttls()
#authentication
s.login("xxx.sn@gmail.com","xxxxx")
#message to be sent
subject="DDoS Attack Detaction using Machine learning"
TEXT="Accuracy of Navive Bayes =",accuracy ,"Accuracy of logical ligration
=",logical_acc,"Accuracy of diession tree", dession_tree


message='Subject:{}\n\n{}'.format(subject,TEXT)
#sending the mail
s.sendmail("xxx@gmail.com","xxx.sn@gmail.com",message)

#terminating the session
s.quit()
```

7.Testing and Result

7.1 Test Cases

7.2 Unit Testing

7.3 System Testing

7.4 Test Report (Give Link to the user manual)

8. Conclusion

We use the KddCup data set and applyed this algorithm and find the accuracy .

Sl.No	Algorithm Name	Accuracy
1.	Naïve Bayes Algorithm	98.78205980746212 %
2.	Decision Tree Algorithm	99.99539627814664 %
3.	Logistic Regression Algorithm	99.4695489375633 %

9. Future Enhancement

With growing internet traffic, traditional network monitoring methods working on a single machine is not efficient method. To solve the problem which algorithm to be used at run time. Our system can be used to decide which algorithm to be used. Our system can be deployed on Cloud Environment, to detect the attacks happening in the cloud. Our system can be integrated with Big data frameworks to improve the processing efficiency. Attacks in live network traffic can use this system to quickly detect the attacks.

10. User Manual

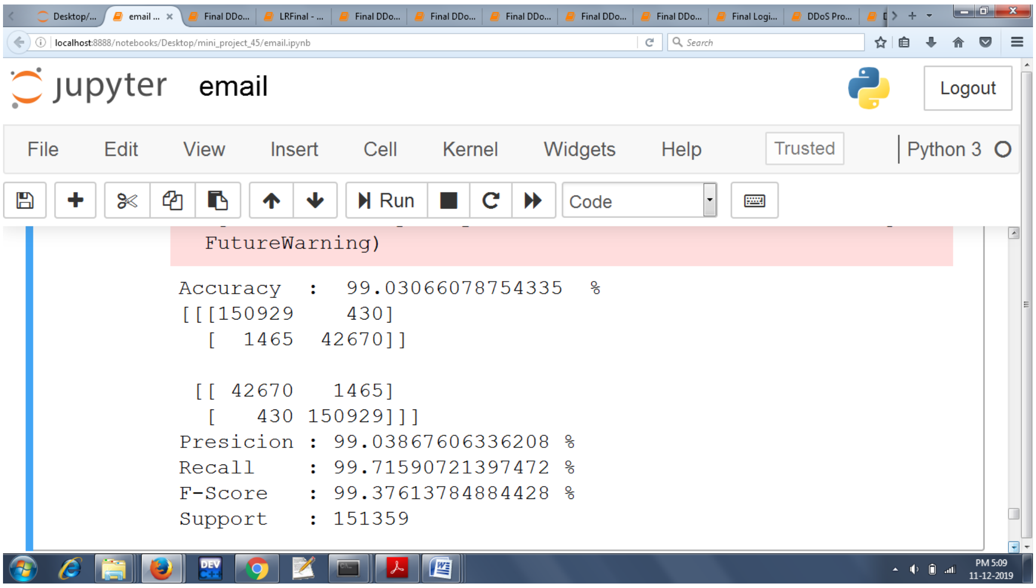


Figure 10.1 Logistic Regression Algorithm

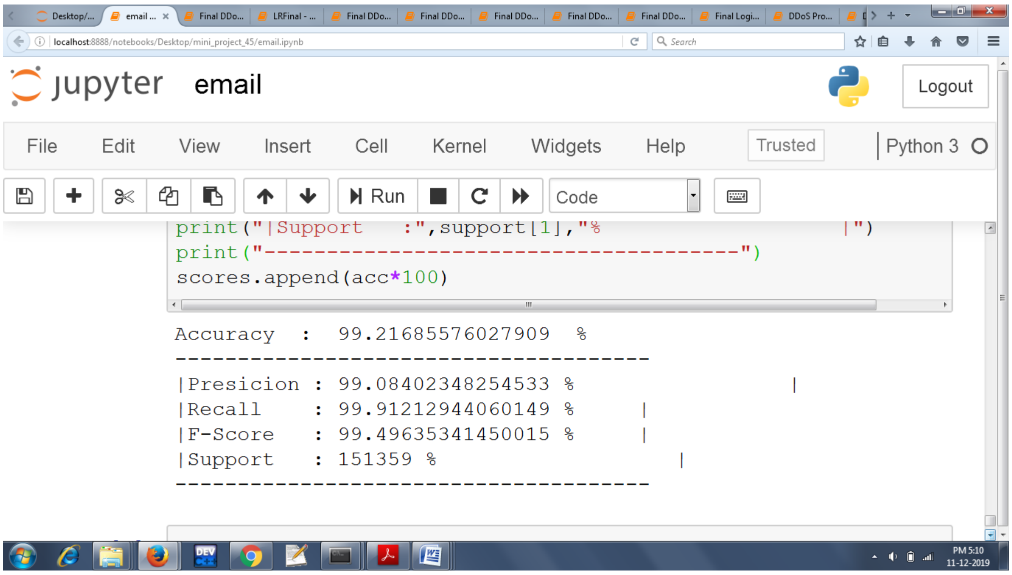
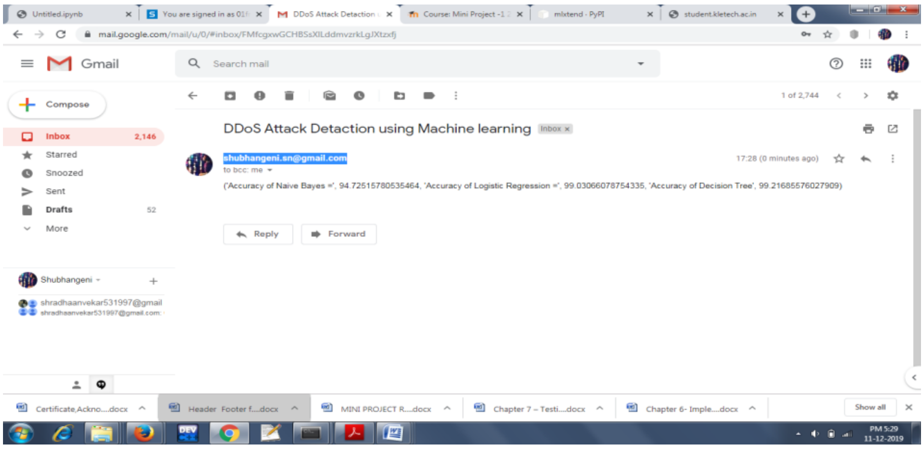
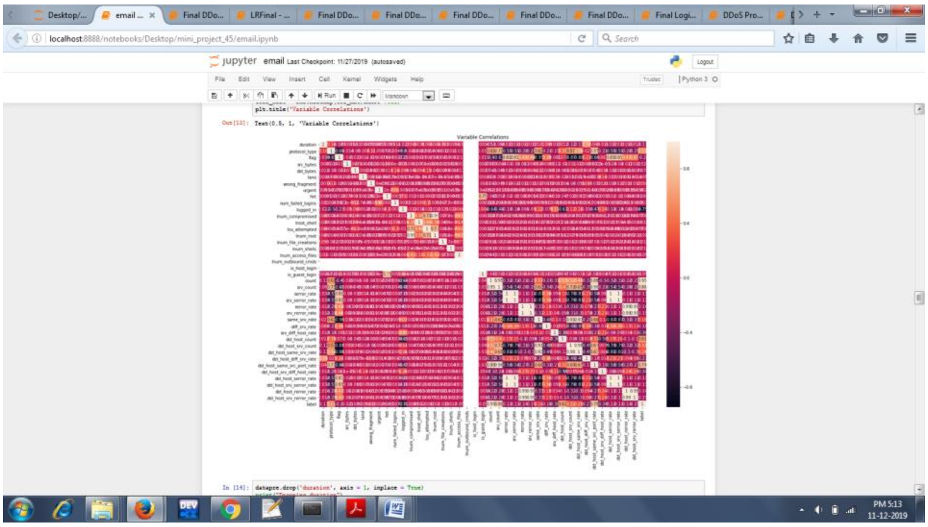
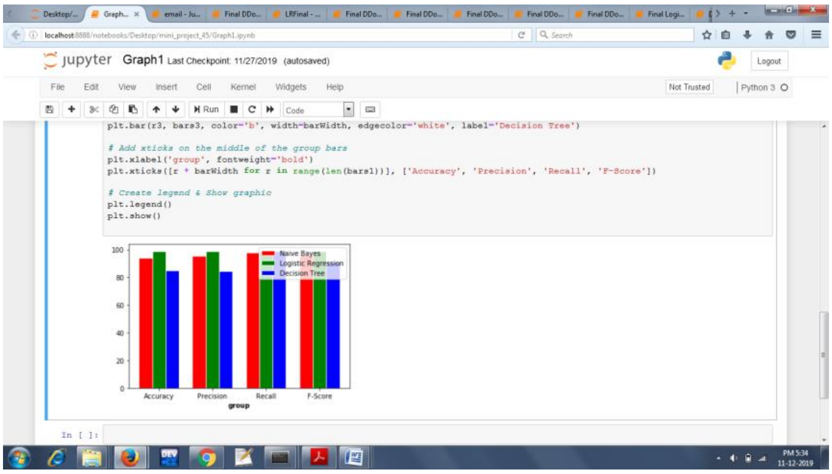


Figure 10.2 Decision Tree Algorithm







11.Bibliography

1. A. Alsirhani, S. Sampalli and P. Bodorik, "DDoS Attack Detection System: Utilizing Classification Algorithms with Apache Spark," 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Paris, 2018

2. C. Hsieh and T. Chan, "Detection DDoS attacks based on neural-network using Apache Spark," 2016 International Conference on Applied System Innovation (ICASI), Okinawa, 2016
3. X. Yuan, C. Li and X. Li, "DeepDefense: Identifying DDoS Attack via Deep Learning," 2017 IEEE International Conference on Smart Computing (SMARTCOMP), Hong Kong, 2017
4. B. Zhou, J. Li, J. Wu, S. Guo, Y. Gu and Z. Li, "Machine-Learning-Based Online Distributed Denial-of-Service Attack Detection Using Spark Streaming," 2018 IEEE International Conference on Communications (ICC), Kansas City, MO, 2018
5. A. Alsirhani, S. Sampalli and P. Bodorik, "DDoS Detection System: Utilizing Gradient Boosting Algorithm and Apache Spark," 2018 IEEE Canadian Conference on Electrical & Computer Engineering (CCECE), Quebec City, QC, 2018