MAJOR PROJECT REPORT

On

“Identification of DDoS attack using Machine Learning”

Submitted in partial fulfilment of the requirements for the award of

Bachelor of Technology (B. Tech)

In the Department of

Computer Science & Engineering

Submitted by:
Dibash Sarkar (UG/02/BTCSE/2019/011)

Simran Sapui (UG/02/BTCSE/2019/021)

Atish Chandra Ray (UG/02/BTCSE/2019/049)

Under the Guidance of
Mr. Sayantan Singha Roy
(Assistant Professor CSE)

School of Engineering & Technology
ADAMAS University, Kolkata, West Bengal
Jan 2023 – June 2023

CERTIFICATE

This is to certify that the report entitled “Identification of DDOS attack using Machine Learning”, submitted to the School of Engineering & Technology (SOET), **ADAMAS UNIVERSITY, KOLKATA** in partial fulfilment for the completion of a project of the degree of Bachelor of Technology in the department of Computer Science & Engineering, is a record of bona fide work carried out by Dibash Sarkar (UG/02/BTCSE/2019/011), Simran Sapui (UG/02/BTCSE/2019/021), Atish Chandra Ray (UG/02/BTCSE/2019/049), under our guidance.

All help received by us from various sources has been duly acknowledged.
No part of this report has been submitted elsewhere for the award of any other degree.

Mr. Sayantan Singha Roy
(Assistant Professor)

Dr. Pranav Kumar
(Project Coordinator)

Prof. Dr. Sajal Saha
(HOD CSE)

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without mentioning people whose constant guidance and encouragement made it possible. We take pleasure in presenting before you, our project, which is the result of a studied blend of both research and knowledge. We express our earnest gratitude to Mr Sayantan Singha Roy (Assistant Professor), Department of CSE, for their constant support, encouragement and guidance. We are grateful for their cooperation and valuable suggestions.

Finally, we express our gratitude to all other members who are involved either directly or indirectly in the completion of this project.

DECLARATION

We, the undersigned, declare that the project entitled 'Identification of DDoS attack using Machine Learning', being submitted in partial fulfilment for the award of a project certificate in Bachelor of Engineering Degree in Computer Science & Engineering, affiliated to ADAMAS University is the work carried out by us.

Dibash Sarkar
(UG/02/BTCSE/2019/011)

Simran Sapui
(UG/02/BTCSE/2019/021)

Atish Chandra Ray
(UG/02/BTCSE/2019/049)

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

In now a days world, technology has become an inevitable part of human life. During the Covid-19 pandemic, most industries noticeably the education and the corporate sector have migrated online. It leads to an escalation increase in intrusions and attacks over Internet-based technologies. One of the fetal threats surfacing is the Distributed Denial of Service (DDoS) attack that can ruin down Internet-based services and applications in no time. The attackers update their skill strategies constantly, avoiding existing detection mechanisms. Huge spikes in data generation and handling have proven that traditional methods of detecting DDOS need a huge upgrade. This paper standardized reviews the leading literature specifically on deep learning to detect DDoS. We have surveyed four considerably used digital libraries (IEEE, ACM, ScienceDirect, Springer) and one scholarly search engine (Google scholar) for searching the recent literature. We have analyzed the relevant studies and concluded a brief literature survey about it.

CHAPTER 1
INTRODUCTION

1.1 Background

Distributed Denial of Service attack is one of the most frightening attacks. It mainly targets a web server, network or website. A Distributed Denial of Service (DDoS) attack can easily flood a whole site by sending much more applications than the server can handle. Distributed Denial of Service (DDoS) attacks can be categorized into three groups [1].

- **Volume-based attacks:**

  In volume-based Distributed Denial of Service (DDoS) attacks, the abnormality is generated by absorbing the target's usable bandwidth between the target and the internet. It is mainly accomplished by sending a very large amount of traffic to a target. Examples include UPD Floods, ICMP Floods, and Spoofed-Packet Floods. [2]

- **Protocol-based attacks:**

  Protocol-based attacks may bring a service to a halt by using all the capacity of the server. Additionally, it is referred to as a state-exhaustion attack. SYN Flood, Ping-Of-Death, Smurf DDoS are some of the examples [2]

- **Application layer attacks:**

  In this case, the attacker establishes a link with the target and then monopolizes processes on the server, exhausting its resources. Slow Loris, Apache are examples of this attack [2]

The network has become an important business organization. The increased demand for network-based services increases the attack which stops response for users. The attack slows down all the network services which tends to application failure. The latest and most important technology used by many users is computer networks therefore more security is required to prevent computer networks. The DDOS attack is carried out by controlling computer systems which are freely available and consist of the internet. An attack is done on servers basically to make the source unavailable to the user. Here it blocks a single source of the user. Multiple digital devices which are connected are more vulnerable. The Attack Detection should be likely to be smart and should battle successfully hackers. Availability, Confidentiality, and integrity play the main role in security purposes.

A. Confidentiality

Confidentiality is also known as secrecy. The main work of confidentiality is to protect information and personal data from unauthorized access. The Sharing information is valid for legitimate users with proper assurance. Cryptography is more powerful which can protect privacy with more security.

B. Integrity

The term Integrity ensures that the data and system are uncorrupted. The integrity protection also protects applications, operating systems, and hardware for unauthorised Users.

C. Availability

Availability is accessing information consistently to authorized users. It also consists of technical infrastructure and maintaining hardware that can display the information. To perform this attack firstly internet is required, through different hacking tools like botnets, this attack is performed mainly focusing on the targeted source IP addresses. It asks for a request first and makes the user overwhelmed and then cuts the valid user from the service. This attack is performed smoothly. The tools used to attack are made by hackers. A trojan virus is made by the attacker first, whereas a trojan is nothing but a message sent to the authorized user in the form of an ad or any other way it is sent by the attacker.

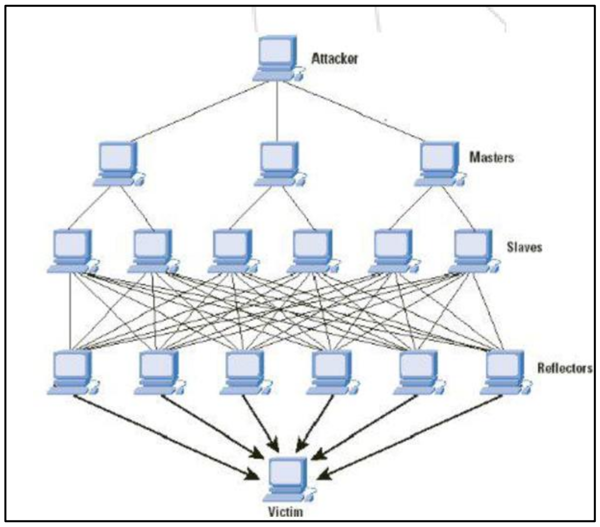


Fig 1.1-DDoS Attack Procedure Flowchart

- **TYPES OF DDOS ATTACK**

A. <u>ICMP (Ping) Flood</u>

ICMP echo request packets are used to send requests to a valid user by the attacker. It slows down the entire system by sending several packets consuming the incoming and out bandwidth.

B. <u>SYN Flood</u>

Any or every device that gets connected to the internet can get venerable to this type of attack. In the SYN flood sender sends ample No. of SYN request but it denies a response to host SYN-ACK and unsensed SYN requests with a spoofed IP address. The delay in receiving the acknowledgement results in the user's denial of services.

C. <u>**Ping of Death**</u>

It delivers the targeted source a packet which is bigger than the allowable size of the system, 65535 bytes is the maximum length of a packet of an IP packet. By this process, it can overflow the memory buffer which is allocated for the packet.

D. <u>**Slow Loris**</u>

It takes place by sending a request to connect a single machine and a server. It sends an HTTP header but in an incomplete format and partial request just after it establishes a connection with the targeted server. This leads to all connections open failing thus the maximum concurrent connection results in the denial of connection.

E. <u>**NTP Amplification**</u>

Here the publicly available sources are being attacked using UDP traffic. High volume and high bandwidth are generated once the attacker obtains the list of open NTP servers.

F. <u>**HTTP Flood**</u>

With HTTP request this attack is performed. It is done by manipulation of and generation of unwanted requests in HTTP or posts at the user end.

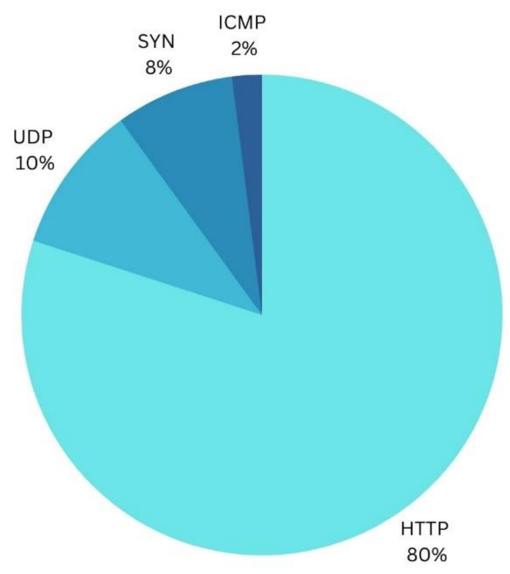


Fig 1.2- Types of DDoS Attack

1.2 Purpose of Project

The purpose of this internship is to fulfil a literature survey on DDoS attacks on different network layers. The purpose is to find a suitable way to detect these attacks. The goal was to study the various ways the DDoS attacks. The target was to find a technique that can detect and identify the attack traffic as quickly as possible, and try to mitigate it so that the server does not lose its functionalities for a long amount of time. To make sure that the server is back and working at its full potential soon after a DDoS attack.

CHAPTER 2
LITERATURE SURVEY

2.1 Flow Aggregation

Flow is a unidirectional series of IP packets of a given protocol travelling between a source and a destination IP/port pair within a certain period. Flow aggregation techniques are used to aggregate flows into a single flow with a larger coarseness of classification giving a flow count for each connection with a unique combination of attributes given below for a packet.

– Source IP
– Destination IP
– Source Port
– Destination Port
– Protocol

Aggregated flows have a larger number of packet information that fiercely reduces the amount of monitoring data. Hence, Internet traffic flow profiling has become a useful technique in the passive measurement and analysis field. Instead of considering the packet count of each connection, the recommend method calculates the flow count of each connection at a particular time interval for detecting the flooding attacks, increasing the speed of analysis and reducing the time complexity of our algorithm. [3]

2.2 Entropy Approach

In this detection approach, the entropy of the flow count is calculated for each connection using the formula given below

$H_{i,t} = -\log(x_{i,t}/(x_{1,t} + x_{2,t} + ... + x_{n,t})) + R_{i,t}$

If $(x_{i,t} \geq x_{i,t+1})$ then $r \leftarrow |\log(x_{i,t+1}/x_{i,t})|$ else $r \leftarrow |\log(x_{i,t}/x_{i,t+1})|$.

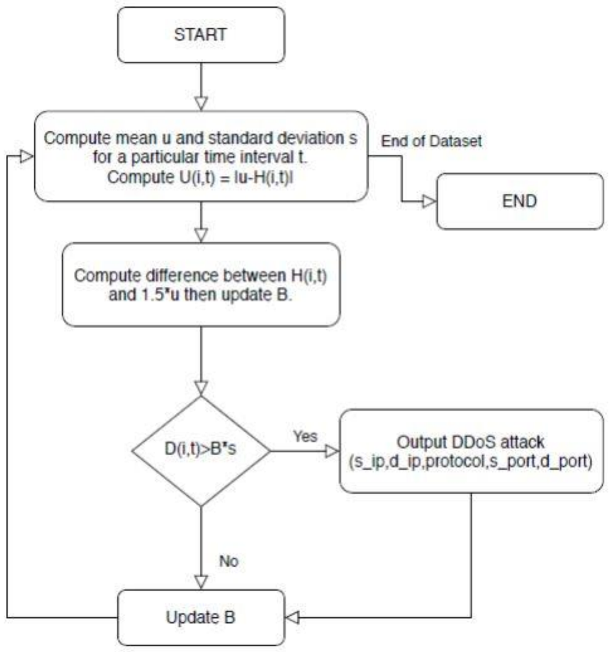

START
Compute mean u and standard deviation s for a particular time interval t. Compute U(i,t) = |u-H(i,t)|
End of Dataset
END
Compute difference between H(i,t) and 1.5*u then update B.
D(i,t)>B*s
Yes
Output DDoS attack (s_ip,d_ip,protocol,s_port,d_port)
No
Update B

Figure 2.1-Entropy Approach Flow Chart

It is essentially a standard way of calculating any entropy in a system. When there is an attack, entropy drops highly, because there is one flow count that is dominating. In the non-attack case, the entropy will be in a sustained range. Let a random variable xi, t represent the flow count of a particular connection I over a given time interval t. [3]

2.3 A Proposed Technique for Simultaneously Detecting DDoS and SQL Injection Attacks

Intrusion detection:

The intrusion detection systems detect Distributed denial of service (DDoS) and inform the admins to take necessary measures. These systems are developed by training and testing existing datasets of distributed denial of service (DDoS) attacks. There are a few existing datasets like KddCUP'99, NSL-KDD, UCLA, and CARDA. These datasets contain the features of various Distributed denial of service (DDoS) attacks. Classification algorithms are used to train and test intrusion detection systems to classify whether a request is legit or an anomaly. Random Forest, KNN, decision tree, and Support Vector Machine are some of the classification algorithms that are frequently used. [4] [5] [6] [7] [8]

SQL Injection Attack:

Structured Query Language injection (SQLi) is a form of web hacking technique in which an attacker inserts code to manipulate a SQL query to obtain unauthorized access to a database. As exchanging information over the Internet through various channels and web applications became a fairly widespread phenomenon, these applications and their related databases are vulnerable to all sorts of threats to information security since they can be accessed through the Internet. To better understand SQL injection attacks, a website that allows users to log in by entering their username and password can be used as an example. Following is the query constructed in case of an authorized login attempt where the username is 'user' and password '123'. SELECT * FROM users WHERE name = 'user' and password = '123' However, it is also possible to type the following input into the website's username with 'user' and password field with" or '1’=’1' by a user with malicious intent

SELECT * FROM users WHERE name = 'user' and password = ''or '1'='1'. Here, this user will always be logged into the website because 1=1 will always be valid. Hence, unauthorized access to the account details of authentic users is gained by the attacker and ownership of this data may have significant implications for the individual who is the authentic account owner. This is known as fraud and data privacy infringement. This is a straightforward example of a SQL injection attack just for understanding. [9]

Proposed Idea:

In the study, a mechanism is proposed which capable of detecting DDoS attacks and SQL injection simultaneously. It is a two-part system that's the first part is competent to detect DDoS attacks include of the network layer, transport layer and application layer of the OSI model. Its second part is competent to detect SQL injection which is consisted of application-layer functionality. Though the purposes of those two tiers are the same which is to prevent suspicious activity on the server-side their main functionalities of detecting attacks are not equivalent. This recommend system should be present on a network server where those attacks have occurred. Whenever the system is deployed on the server, it starts monitoring. So, in the first part to detect a DDoS attack a machine learning-based solution is used to find network traffic in normal and attack state, TCP-SYN or ICMP flood attack, how many times a user sends a pdf get request, HTTP get request or HTTP post request and their request time. In the network layer for finding TCP-SYN or ICMP flood attacks, here is an SDN-based gateway that receives traffic coming in and out. When a new incoming data flow arrives at the network server it performs an ML-based detection algorithm to decide if the flow is incorrect or attack based on the port's entropy for TCP vehicles or logarithm of the number of packets for ICMP vehicles. For abnormal flow, it blocks the attack flow. At the same time, the information of each IP address that arrives in the server network is collected to calculate their entropy [10] [11]. In the application layer, the system checks how many requests a server gets from a specific IP address. If the number of requests exceeds the threshold value set by the system, it detects that a specific source is being used to attack the server. In addition, when a request is coming from an address system checks the requested time, when it found any anomaly then the system blocks that IP address for a while [10] In the second part to detect SQL injection the system checks whether there is any malicious query is sent as a request from a user. The proposed system is trained with a large number of SQL injection attack query patterns. So, when a user sent their input to the server first of all, that input is going to be run down into a sequence of characters then each of the sequences is passed to the system model to check whether there is any unsure pattern found or not. If any suspicious input pattern is found then the system halts the input request of that user and notifies the administrator [9]. In fig-4 it is shown that when our system starts monitoring it checks incoming traffic in the network layer collecting information on each IP address to calculate their port entropy. After that using an ML-based algorithm checks whether a TCP-SYN or ICPM flood attack occurs or not. At the same time when collecting information on each IP address, it also monitors the input request behaviour means what types of request a user send, how many times the network server gets that same user request, determining the time of request between host and server. If the system found any suspicious activity, then that specific IP address is inserted into the block list. In addition, with the help of tokenization, each input value that comes to the server is compared with a malicious query pattern.

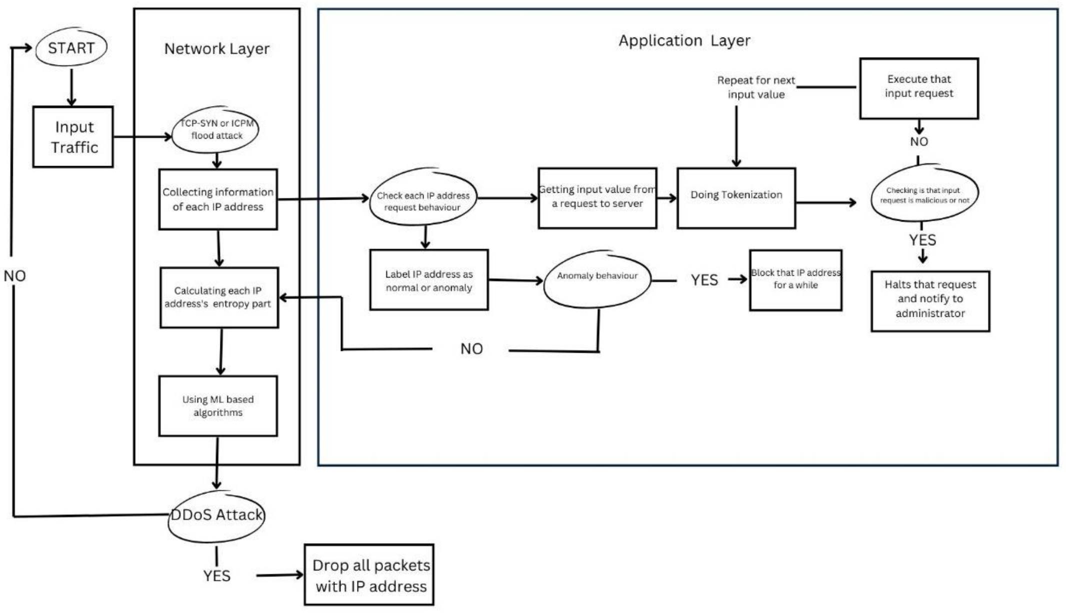

START
Input Traffic
Network Layer
TCP-SYN or ICPM flood attack
Collecting information of each IP address
Calculating each IP address's entropy part
Using ML based algorithms
DDoS Attack
YES
Drop all packets with IP address
NO
Application Layer
Check each IP address request behaviour
Getting input value from a request to server
Doing Tokenization
Repeat for next input value
Execute that input request
NO
Checking is that input request is malicious or not
YES
Halts that request and notify to administrator
Label IP address as normal or anomaly
Anomaly behaviour
YES
Block that IP address for a while
NO

Fig 2.2- Flow Chat of Proposed System

Dataset Collection and analysis

For our model training and testing to detect DDoS attacks we select a benchmark dataset which is called NSL-KDD dataset. The NSL-KDD dataset is a variant of the KDDCup'99 dataset that has been configured. In the KDDCup'99 dataset, there are many unnecessary and duplicate records. For this issue, the NSL-KDD dataset is a new version of the KDDCup'99 dataset that is unnecessary and duplicate-free, allowing the classifiers to produce better results. This dataset consists of four files. In the NSL-KDD dataset, the attack types are grouped into four categories: DoS, Probe, U2R and R2L. There are 41 features in the NSD-KDD dataset. The dataset can be split up into four categories based on these features.

● 4 categorical (Features:2,3,4,42),

● 6 Binary (Features:7,12,14,20,21,22),

● 23 discrete (Features:8,9,15,23-41,43),

● 10 continuous (Features:1,5,6,10,11,13,16,17,18,19)

41 features can also be split into four different classes, which are Basic, Content, Traffic and Host. The training set of NSL-KDD has a total of 125973 occurrence and in the test set, it has a total of 22544 instances. NSL-KDD has a version of the dataset with 20% of the training data identified as KDDTrain+_20Percent with a total number of 25192

instances. For the model, NSLKDDTrain+ is used to train the dataset and NSLKDDTest+ is used to test the dataset.

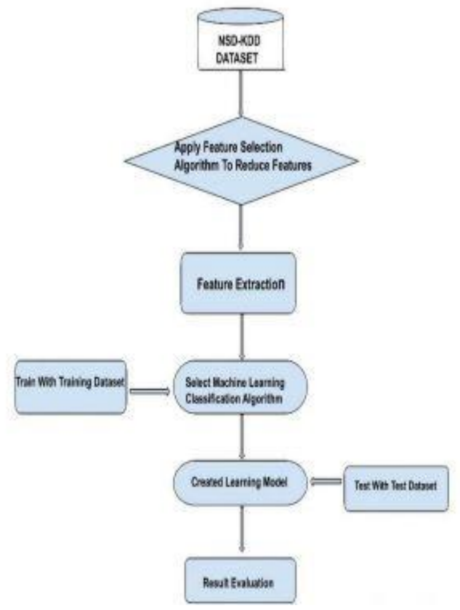
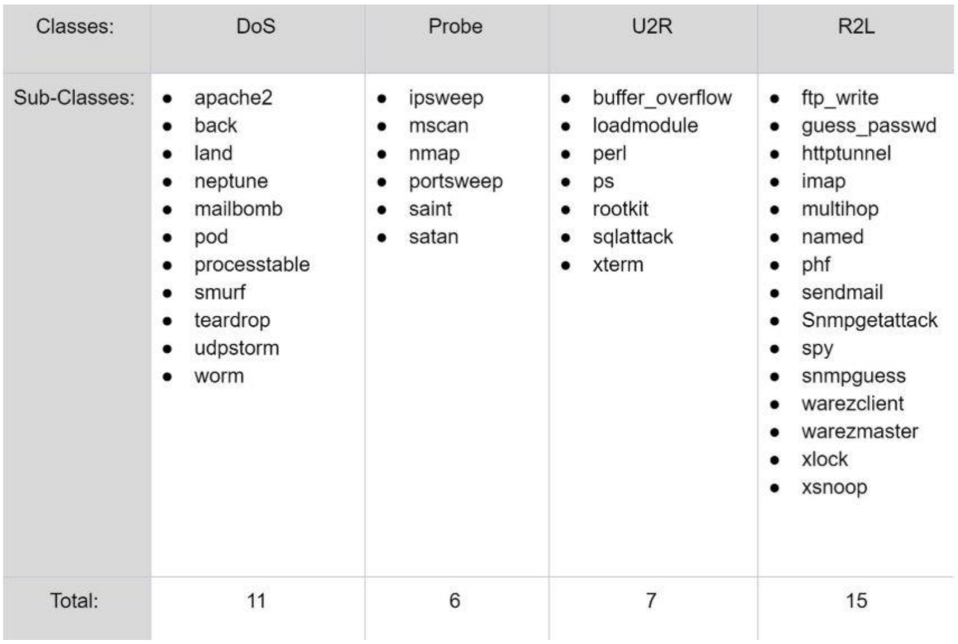

NSD-KDD DATASET
Apply Feature Selection Algorithm To Reduce Features
Feature Extraction
Train With Training Dataset
Select Machine Learning Classification Algorithm
Created Learning Model
Test With Test Dataset
Result Evaluation

Fig 2.3-Flow chart of system Model using NSL-KDD Dataset

To evaluate DDoS attacks, the NSL-KDD dataset is used to train and test the model. Three classifier algorithms are used. The algorithm are Random Forest, KNN and Decision Tree. For testing and training, two different files have been used from the NSL-KDD dataset. First, the dataset is normalized and extracted by implementing various selective attribute methods like Infogain methods. Then the model will be trained using the dataset. In this figure, we can see how the model will be trained and tested to be ready for detecting DDoS attacks. [12]



Classes:	DoS	Probe	U2R	R2L
Sub-Classes:	• apache2 • back • land • neptune • mailbomb • pod • processtable • smurf • teardrop • udpstorm • worm	• ipsweep • mscan • nmap • portsweep • saint • satan	• buffer_overflow • loadmodule • perl • ps • rootkit • sqlattack • xterm	• ftp_write • guess_passwd • httptunnel • imap • multihop • named • phf • sendmail • Snmpgetattack • spy • snmpguess • warezclient • warezmaster • xlock • xsnoop
Total:	11	6	7	15

Fig 2.4-Attack Classification

In the NSL-KDD Dataset, there are 41 features. Still, to avoid overfitting a machine learning algorithm, some features from the dataset must be reduced. It is essential to apply feature reduction techniques to find which features are most important. For selecting the quality WEKA TOOL is used and performed many feature selection techniques. From all of them, InfoGainAttributeEval (also called entropy) gives the best result for feature reduction. So InfoGainAttributeEval with Ranker Search Method is used for each attribute. The output variable ranges from 0 to 1. After the evaluation, attributes with higher information gain values are selected. The arbitrary cutoff is 0.270. All the attributes whose info gain value is higher than this cutoff value is selected. [12]

Result

In this section, we present the DDoS experimental results derived from different machine learning models with the help of the NSL-KDD dataset. At first, several selective attribute methods are being implemented in the WEKA TOOL. Then those algorithms are performed on JUPYTER NOTEBOOK to build the model to detect DDoS attacks. The accuracy of three machine learning classification algorithms is compared to see which one gives the best results. The accuracy of the attributes is chosen by performing info-gain methods. Cross-validation was done with the training dataset and also a test dataset is used to test the trained model. For our experiment, features are selected. Two categorical features are converted into numeric values for training and testing. Those nominal features are "service","flag", “Dst_Bytes”,” Num_Failed_Logins”, “Num_Compromised” etc. As their performances are evaluated using various variables, including accuracy, recall and F1-score, therefore, their given parameters can be calculated using True positive (TP), False Negative (FN), False Positive (FP) and True Negative (TN). Accuracy and recall give a measure of the relevant data points. It gives a result that demonstrates how good the model is at finding true positives of all possible values. For getting a good percentage of accuracy, a trade-off between precision and recall is needed. F1-score is the symmetric mean of accuracy and recall. It helps us to understand which parameters (accuracy, recall) are more important for our model. A good F1 score indicates a good precision and recall value.

Precision = TP / (TP + FP)

Recall = TP/(TP+FN)

Accuracy = (TP + TN)/(TP+TN+FP+FN)

F1-score = 2*((precision*recall)/ (precision+recall))

TP refers to a positive sample that is divine to be positive, and in this context is normal data divine to be normal behaviour.

TN refers to a negative sample that is divine to be negative, and in this paper is the attack data divine to be aggressive

FP refers to a negative sample that is divine to be positive, and in this paper is attack data divine to be normal behaviour

FN refers to a positive sample that is divine to be negative, and in this paper is normal data divine to be aggressive.

CHAPTER 3
METHODOLOGY

3.1 DDoS Detection Using Various supervised machine learning techniques on the highly optimized NSL-KDD dataset to create an efficient and accurate predictor of possible intrusions on a network

3.1.1 Classification algorithms

1. K-Nearest Neighbour (k-NN):

The k-nearest neighbour is an instance-based classifier. When the k-NN is used, instances within a dataset are contained in a dimensional space, where a new instance is labelled based on its similarity with other instances. These instances are referred to as neighbours. A new instance is labelled x if x is the most similar class for the neighbouring observations. A distance function is applied to determine the similarity between instances. For this study, the distance function employed is Euclidean. The Euclidean function is a relatively common method as it reflects the human perception of distance. [13]

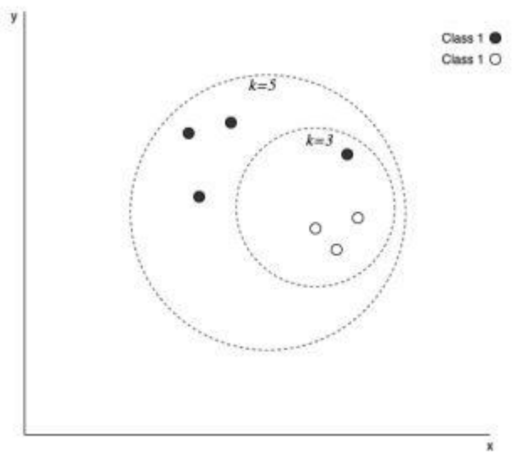


Fig 3.1- An example of a k-NN classification. When k=3, the new instance is labelled as 0. However, when the parameter is increased to k=5, the same instance is labelled as 1.

Algorithm 1: k-Nearest Neighbour

```
start
Let S = {a1, a2, …, an}, where S represents the training set and a
represents article documents
k ← the desired number of nearest neighbours
Compute d(x,y) between new instance i and all a ∈ S
Select the k closest training samples to i
Classic ← best voted class
End
```

1. Support Vector Machine (SVM):

The learning process in SVM is carried out in two steps: firstly, the inputs are plotted in an n-dimensional space, where n is based on the number of attributes; the coordinates of individual attributes are referred to as support vectors. Secondly, a hyperplane separates the instances. A hyperplane is a line that linearly separates a set of data points into two distinct classes. The SVM is the hyperplane which best splits the data set. When dealing with the mapping of complex nonlinear functions, computation issues are highly probable. The larger the dimensional space, the bigger the separation problem. Kernel tricks are used to mitigate this problem. A kernel can transform extremely complex functions into infinitely higher dimensional spaces and then uses predefined labels to split the inputs. [13][14]

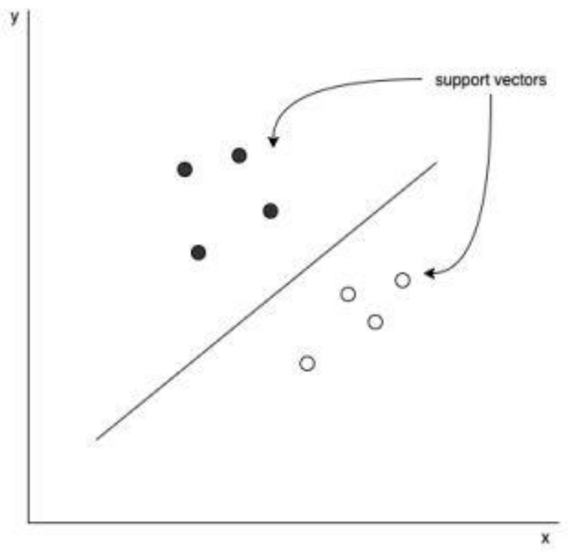


Fig 3.2-Illustration of SVM classification

Algorithm- 2 Support Vector Machine

```
start
∀ document ∈ training set S:
Create SVM classification objects
Objects → Higher Dimensional Space
Apply a kernel trick to transform f(x) into a linear separable one
A hyperplane is computed ⇒ binary classification
end
```

2. Decision Tree:

Decision tree classification starts at the root node and classifies observations on the basis of the values of the respective attributes. Every node represents a single feature, while they represent the values that the node can assume. Starting from the root node, the algorithm works its way down by iteratively computing the information gained for each feature in the training set. Information gain is used to determine the level of discrimination imposed by the features towards the target classes. The higher the information gain, the higher the importance of the attribute in the classification of each observation. The root node is replaced by the attribute that possesses the highest information gain, and the algorithm continues splitting the data set by the selected feature to produce subsets. [13] [14]

Algorithm- 3 Decision Tree

```
start
∀ attributes a1, a2, …, an
Find the attribute that best divides the training data using information gain
a_best ← the attribute with highest information gain
Create a decision node that splits on a_best
Recurse on the sub-lists obtained by splitting on a_best and add
those nodes as children of node
end
```

3. MLP:

MLP Classifier 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, MLP Classifier relies on an underlying Neural Network to perform the task of classification.

4. Random Forest:

Random forest is a collection of decision trees trained on different dataset subsets and then averaged to increase predictive accuracy. It is created randomly with a collection of decision trees. Here each node selects a set of features at random to calculate the outcome. The output of individual decision trees is combined in the random forest to produce the outcome.

3.1.2 Data pre-processing

Data pre-processing is a crucial step in machine learning that involves raw data in a suitable format for building predictive models. The main goal of data pre-processing is to ensure that the data is clean, consistent, and in the right format to be used by machine learning algorithms.

The following has been performed in order

- checking for Null Values
- checking for Duplicate Rows
- Identify Categorical Features

3.1.3 Feature Scaling

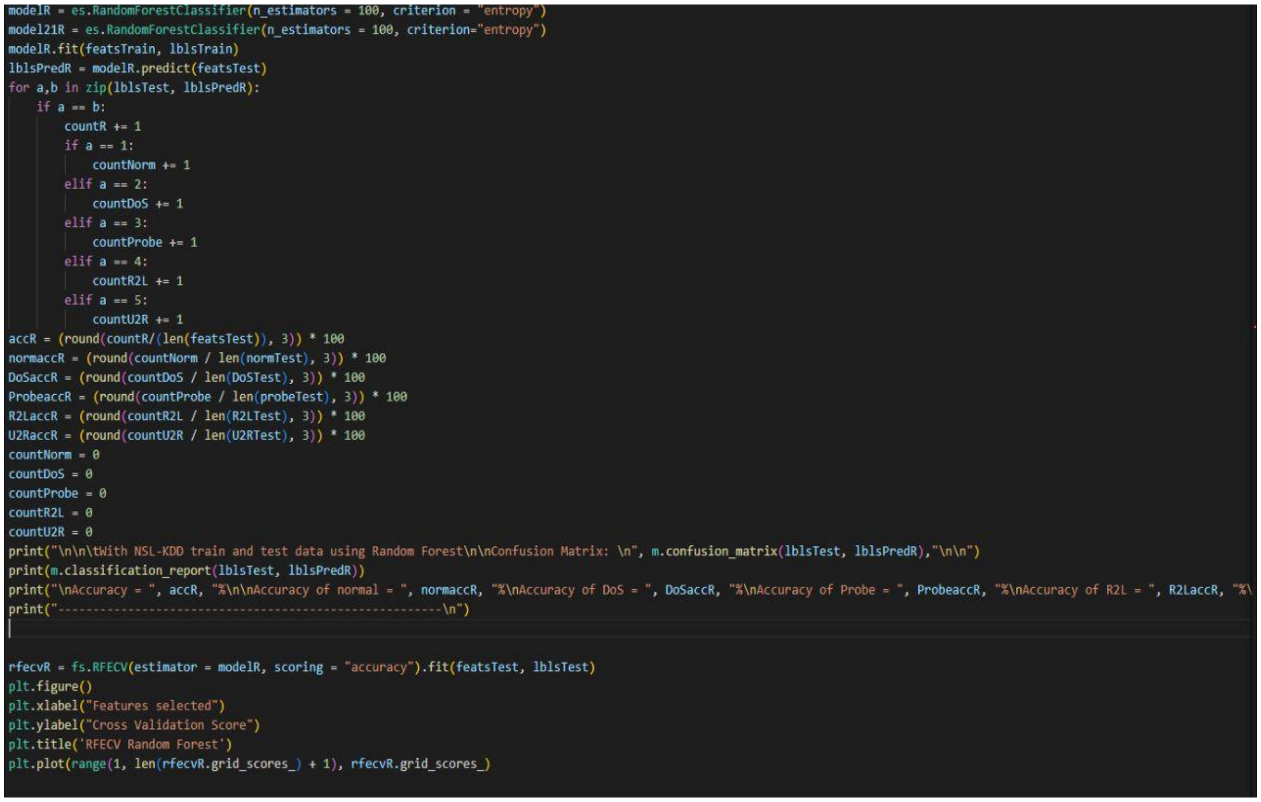
Feature Scaling is a method used to normalize the range of independent variables or features of data.in data processing we have used StandardScaler which standardizes a feature by subtracting the mean and then scaling to unit variance.

3.1.4 Feature selection

Feature selection also known as variable selection or attribute selection is the process of selection a subset of relevant features for use in model construction. Here the features selected for Dos are protocol_type, serror_rate, dst_host_srv_serror_rate etc.

Modelling Source Code

Random Forest Model:

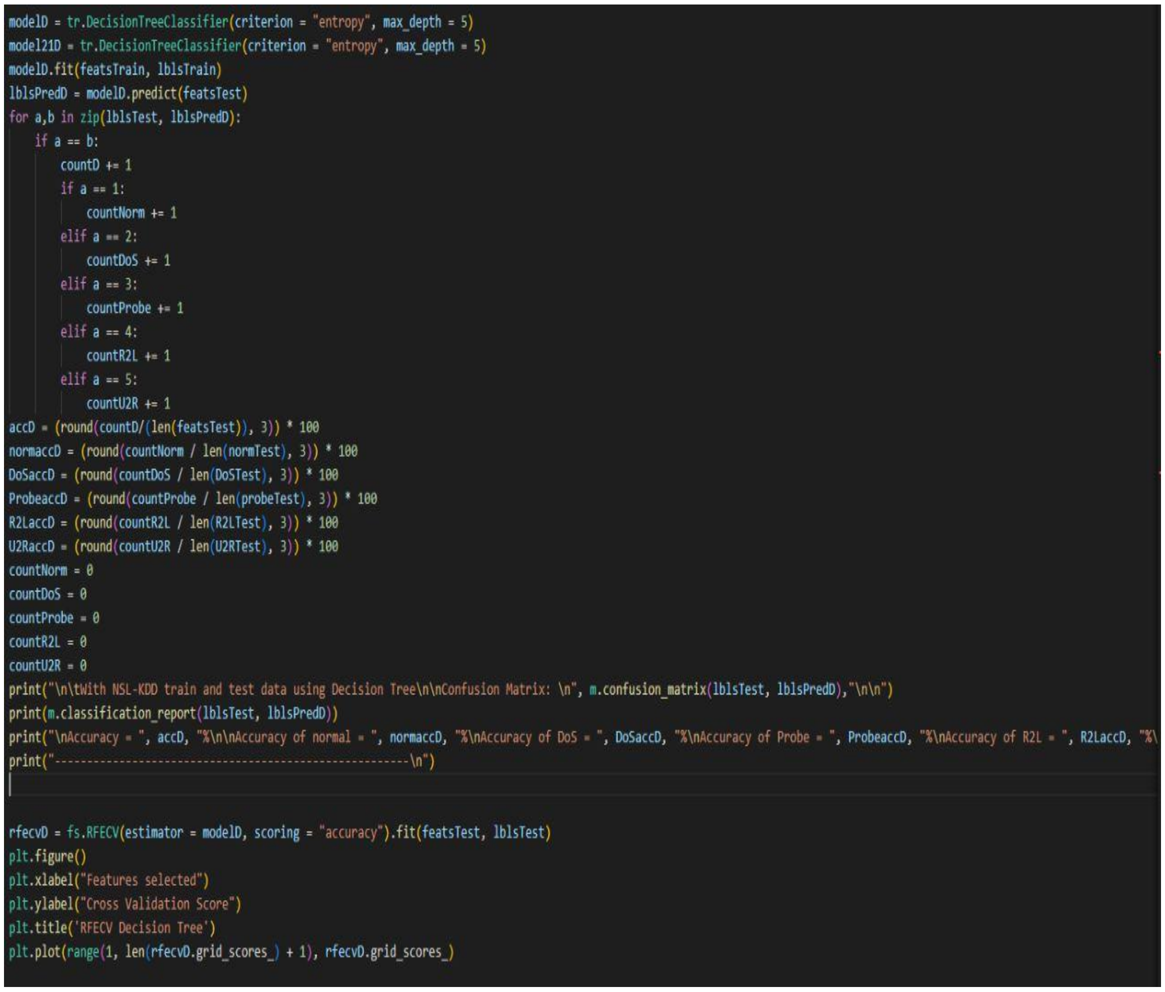


```
modelR = es.RandomForestClassifier(n_estimators = 100, criterion = "entropy")
model21R = es.RandomForestClassifier(n_estimators = 100, criterion="entropy")
modelR.fit(featsTrain, lblsTrain)
lblsPredR = modelR.predict(featsTest)
for a,b in zip(lblsTest, lblsPredR):
    if a == b:
        countR += 1
        if a == 1:
            countNorm += 1
        elif a == 2:
            countDoS += 1
        elif a == 3:
            countProbe += 1
        elif a == 4:
            countR2L += 1
        elif a == 5:
            countU2R += 1
accR = (round(countR/(len(featsTest)), 3)) * 100
normaccR = (round(countNorm / len(normTest), 3)) * 100
DoSaccR = (round(countDoS / len(DoSTest), 3)) * 100
ProbeaccR = (round(countProbe / len(probeTest), 3)) * 100
R2LaccR = (round(countR2L / len(R2LTest), 3)) * 100
U2RaccR = (round(countU2R / len(U2RTest), 3)) * 100
countNorm = 0
countDoS = 0
countProbe = 0
countR2L = 0
countU2R = 0
print("\n\n\tWith NSL-KDD train and test data using Random Forest\n\nConfusion Matrix: \n", m.confusion_matrix(lblsTest, lblsPredR),"\n\n")
print(m.classification_report(lblsTest, lblsPredR))
print("\nAccuracy = ", accR, "%\n\nAccuracy of normal = ", normaccR, "%\nAccuracy of DoS = ", DoSaccR, "%\nAccuracy of Probe = ", ProbeaccR, "%\nAccuracy of R2L = ", R2LaccR, "%\
print("------------------------------------------------------\n")


rfecvR = fs.RFECV(estimator = modelR, scoring = "accuracy").fit(featsTest, lblsTest)
plt.figure()
plt.xlabel("Features selected")
plt.ylabel("Cross Validation Score")
plt.title('RFECV Random Forest')
plt.plot(range(1, len(rfecvR.grid_scores_) + 1), rfecvR.grid_scores_)
```

Fig 3.3-Modelling Source Code of Random Forest Model

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

Decision Tree Model:

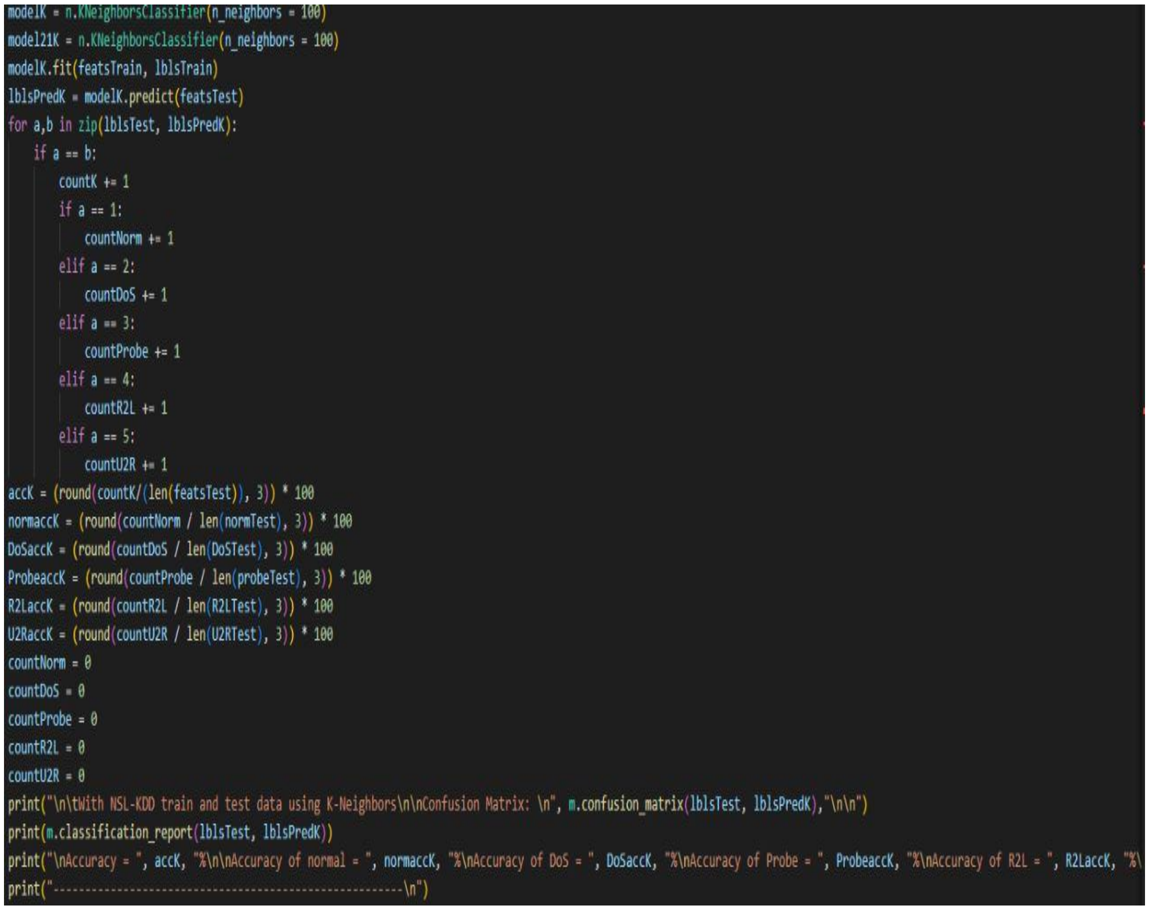


```
modelD = tr.DecisionTreeClassifier(criterion = "entropy", max_depth = 5)
model21D = tr.DecisionTreeClassifier(criterion = "entropy", max_depth = 5)
modelD.fit(featsTrain, lblsTrain)
lblsPredD = modelD.predict(featsTest)
for a,b in zip(lblsTest, lblsPredD):
    if a == b:
        countD += 1
        if a == 1:
            countNorm += 1
        elif a == 2:
            countDoS += 1
        elif a == 3:
            countProbe += 1
        elif a == 4:
            countR2L += 1
        elif a == 5:
            countU2R += 1
accD = (round(countD/(len(featsTest)), 3)) * 100
normaccD = (round(countNorm / len(normTest), 3)) * 100
DoSaccD = (round(countDoS / len(DoSTest), 3)) * 100
ProbeaccD = (round(countProbe / len(probeTest), 3)) * 100
R2LaccD = (round(countR2L / len(R2LTest), 3)) * 100
U2RaccD = (round(countU2R / len(U2RTest), 3)) * 100
countNorm = 0
countDoS = 0
countProbe = 0
countR2L = 0
countU2R = 0
print("\n\tWith NSL-KDD train and test data using Decision Tree\n\nConfusion Matrix: \n", m.confusion_matrix(lblsTest, lblsPredD),"\n\n")
print(m.classification_report(lblsTest, lblsPredD))
print("\nAccuracy = ", accD, "%\n\nAccuracy of normal = ", normaccD, "%\nAccuracy of DoS = ", DoSaccD, "%\nAccuracy of Probe = ", ProbeaccD, "%\nAccuracy of R2L = ", R2LaccD, "%\
print("-----------------------------------------------------\n")


rfecvD = fs.RFECV(estimator = modelD, scoring = "accuracy").fit(featsTest, lblsTest)
plt.figure()
plt.xlabel("Features selected")
plt.ylabel("Cross Validation Score")
plt.title('RFECV Decision Tree')
plt.plot(range(1, len(rfecvD.grid_scores_) + 1), rfecvD.grid_scores_)
```

Fig 3.4- Modelling Source Code of Decision Tree Model

Decision trees are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

KNN Model:

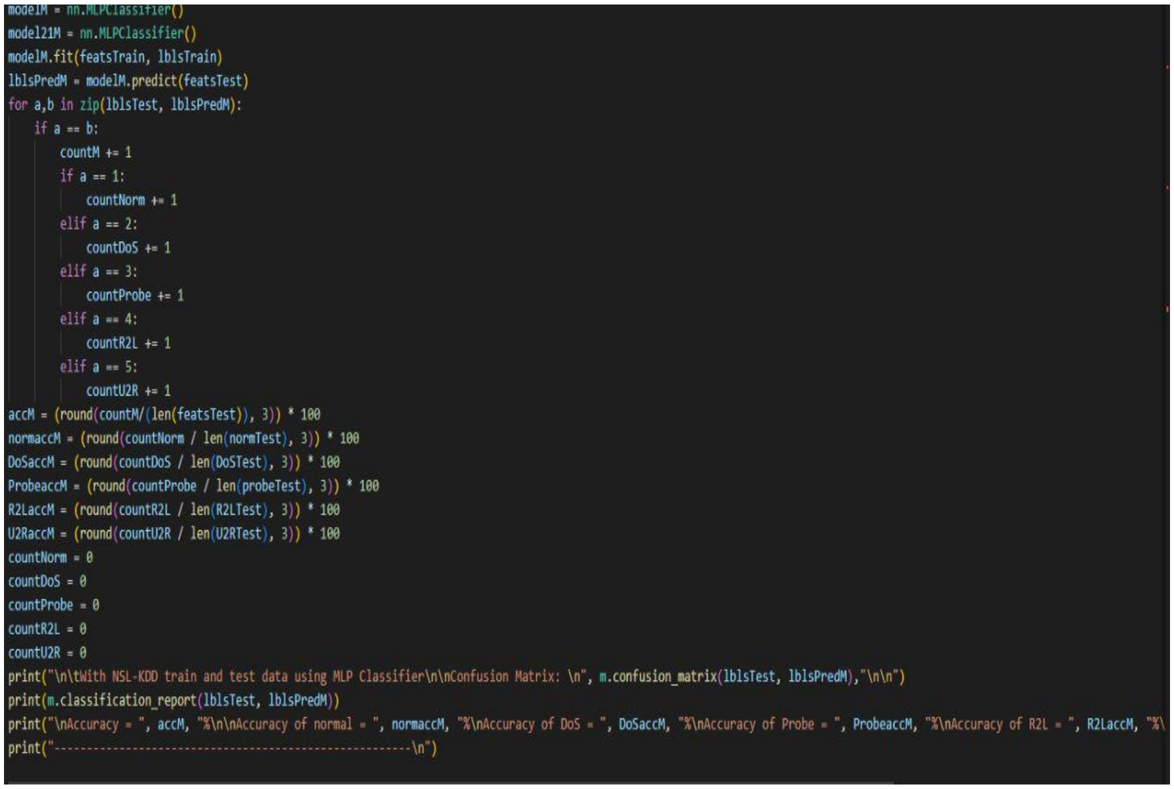


```
modelK = n.KNeighborsClassifier(n_neighbors = 100)
model21K = n.KNeighborsClassifier(n_neighbors = 100)
modelK.fit(featsTrain, lblsTrain)
lblsPredK = modelK.predict(featsTest)
for a,b in zip(lblsTest, lblsPredK):
    if a == b:
        countK += 1
        if a == 1:
            countNorm += 1
        elif a == 2:
            countDoS += 1
        elif a == 3:
            countProbe += 1
        elif a == 4:
            countR2L += 1
        elif a == 5:
            countU2R += 1
accK = (round(countK/(len(featsTest)), 3)) * 100
normaccK = (round(countNorm / len(normTest), 3)) * 100
DoSaccK = (round(countDoS / len(DoSTest), 3)) * 100
ProbeaccK = (round(countProbe / len(probeTest), 3)) * 100
R2LaccK = (round(countR2L / len(R2LTest), 3)) * 100
U2RaccK = (round(countU2R / len(U2RTest), 3)) * 100
countNorm = 0
countDoS = 0
countProbe = 0
countR2L = 0
countU2R = 0
print("\n\tWith NSL-KDD train and test data using K-Neighbors\n\nConfusion Matrix: \n", m.confusion_matrix(lblsTest, lblsPredK),"\n\n")
print(m.classification_report(lblsTest, lblsPredK))
print("\nAccuracy = ", accK, "%\n\nAccuracy of normal = ", normaccK, "%\nAccuracy of DoS = ", DoSaccK, "%\nAccuracy of Probe = ", ProbeaccK, "%\nAccuracy of R2L = ", R2LaccK, "%\
print("------------------------------------------------------\n")
```

Fig 3.5- Modelling Source Code of KNN Model

K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

MLP Mode:

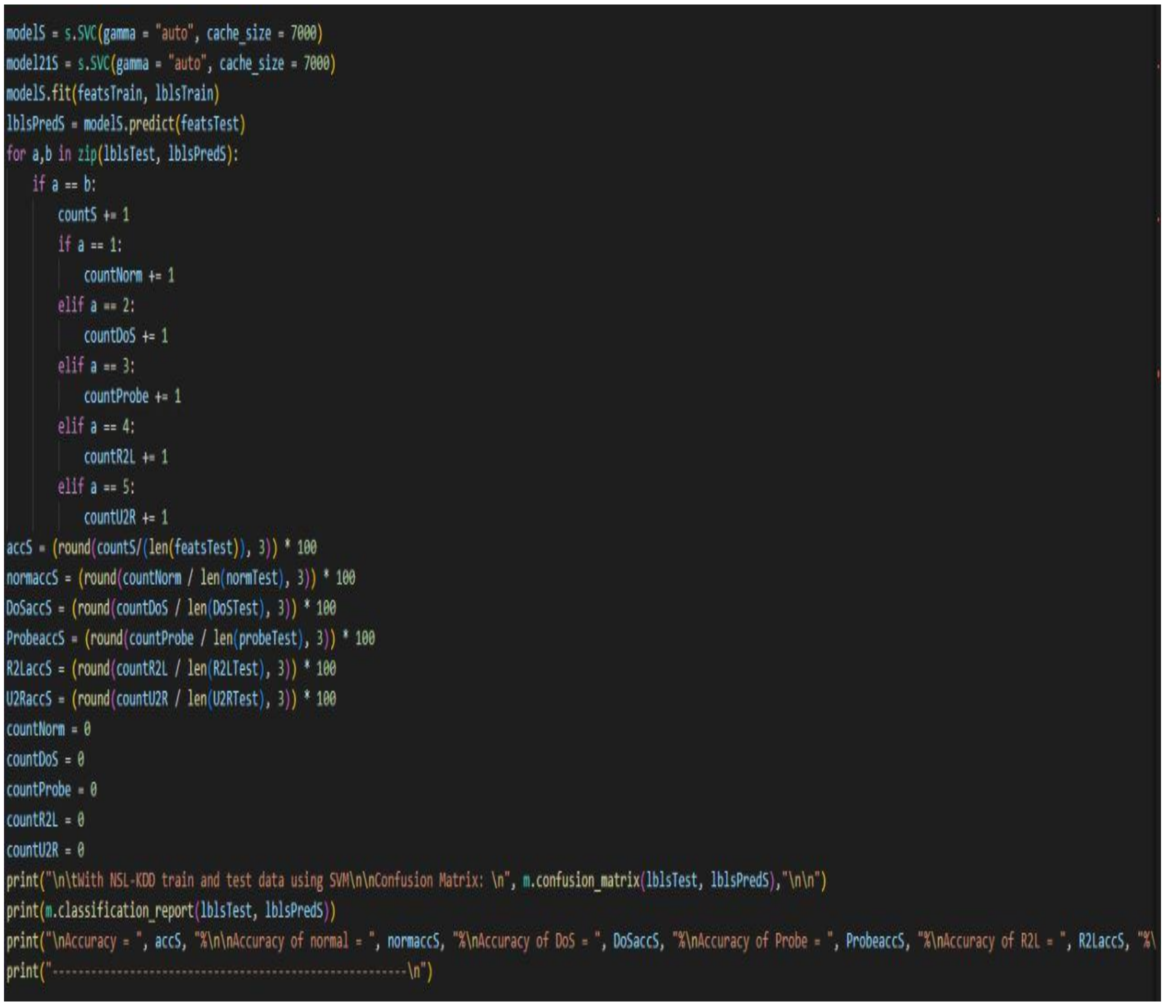


```
modelM = nn.MLPClassifier()
model21M = nn.MLPClassifier()
modelM.fit(featsTrain, lblsTrain)
lblsPredM = modelM.predict(featsTest)
for a,b in zip(lblsTest, lblsPredM):
    if a == b:
        countM += 1
        if a == 1:
            countNorm += 1
        elif a == 2:
            countDoS += 1
        elif a == 3:
            countProbe += 1
        elif a == 4:
            countR2L += 1
        elif a == 5:
            countU2R += 1
accM = (round(countM/(len(featsTest)), 3)) * 100
normaccM = (round(countNorm / len(normTest), 3)) * 100
DoSaccM = (round(countDoS / len(DoSTest), 3)) * 100
ProbeaccM = (round(countProbe / len(probeTest), 3)) * 100
R2LaccM = (round(countR2L / len(R2LTest), 3)) * 100
U2RaccM = (round(countU2R / len(U2RTest), 3)) * 100
countNorm = 0
countDoS = 0
countProbe = 0
countR2L = 0
countU2R = 0
print("\n\tWith NSL-KDD train and test data using MLP Classifier\n\nConfusion Matrix: \n", m.confusion_matrix(lblsTest, lblsPredM),"\n\n")
print(m.classification_report(lblsTest, lblsPredM))
print("\nAccuracy = ", accM, "%\n\nAccuracy of normal = ", normaccM, "%\nAccuracy of DoS = ", DoSaccM, "%\nAccuracy of Probe = ", ProbeaccM, "%\nAccuracy of R2L = ", R2LaccM, "%\
print("-----------------------------------------------------\n")
```

Fig 3.6- Modelling Source Code of MLP Model

MLP Classifier 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, MLP Classifier relies on an underlying Neural Network to perform the task of classification.

SVM Model:



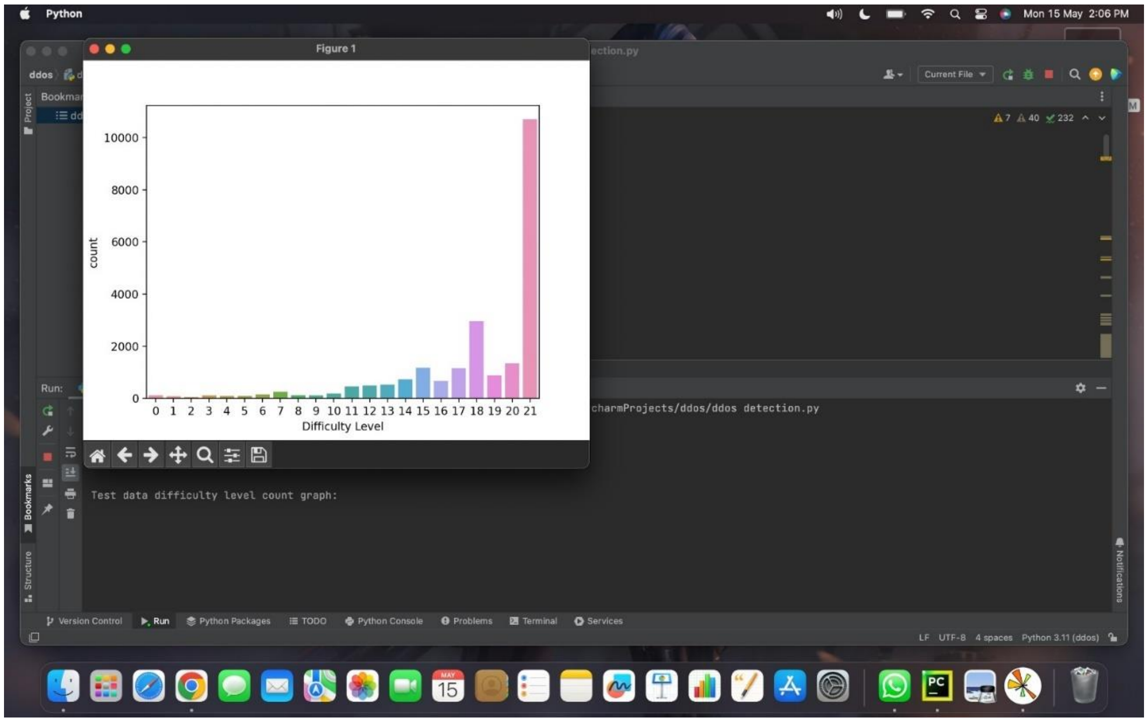
```
modelS = s.SVC(gamma = "auto", cache_size = 7000)
model21S = s.SVC(gamma = "auto", cache_size = 7000)
modelS.fit(featsTrain, lblsTrain)
lblsPredS = modelS.predict(featsTest)
for a,b in zip(lblsTest, lblsPredS):
    if a == b:
        countS += 1
        if a == 1:
            countNorm += 1
        elif a == 2:
            countDoS += 1
        elif a == 3:
            countProbe += 1
        elif a == 4:
            countR2L += 1
        elif a == 5:
            countU2R += 1
accS = (round(countS/(len(featsTest)), 3)) * 100
normaccS = (round(countNorm / len(normTest), 3)) * 100
DoSaccS = (round(countDoS / len(DoSTest), 3)) * 100
ProbeaccS = (round(countProbe / len(probeTest), 3)) * 100
R2LaccS = (round(countR2L / len(R2LTest), 3)) * 100
U2RaccS = (round(countU2R / len(U2RTest), 3)) * 100
countNorm = 0
countDoS = 0
countProbe = 0
countR2L = 0
countU2R = 0
print("\n\tWith NSL-KDD train and test data using SVM\n\nConfusion Matrix: \n", m.confusion_matrix(lblsTest, lblsPredS),"\n\n")
print(m.classification_report(lblsTest, lblsPredS))
print("\nAccuracy = ", accS, "%\n\nAccuracy of normal = ", normaccS, "%\nAccuracy of DoS = ", DoSaccS, "%\nAccuracy of Probe = ", ProbeaccS, "%\nAccuracy of R2L = ", R2LaccS, "%\
print("----------------------------------------------------------\n")
```

Fig 3.7- Modelling Source Code of SVM Model

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers' detection.

CHAPTER 4
OUTPUT

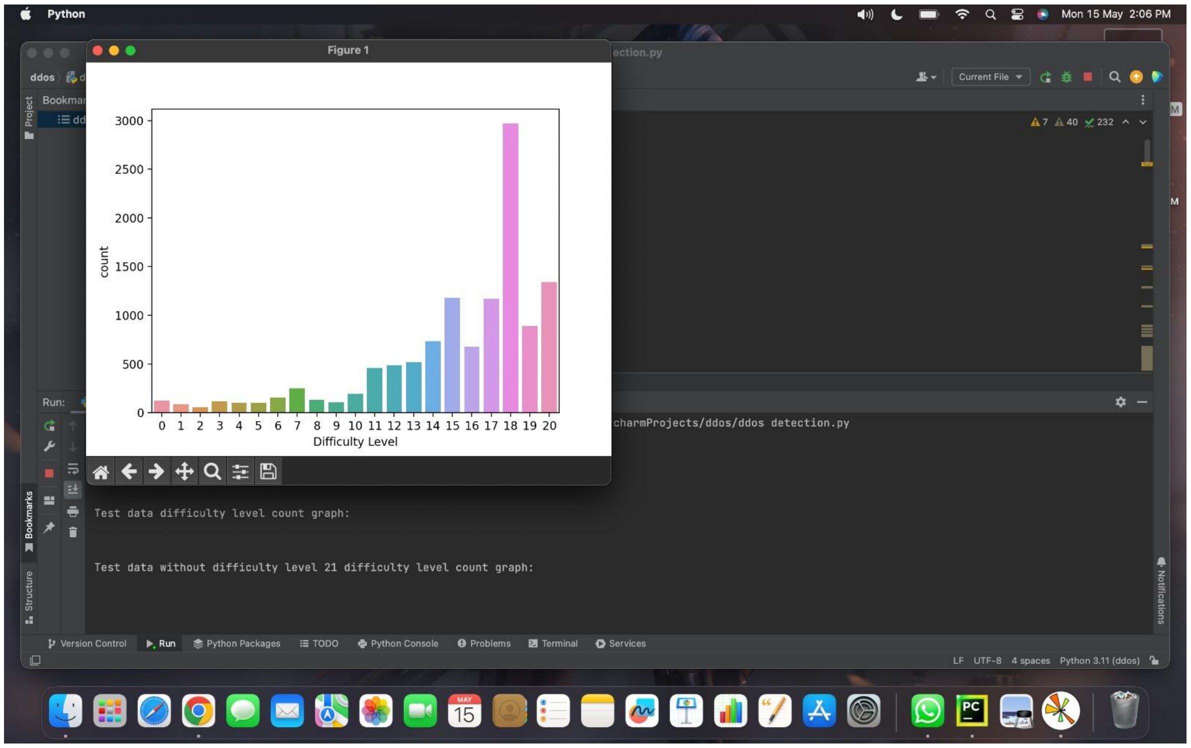
Train data difficulty level count graph:



```
charmProjects/ddos/ddos detection.py



Test data difficulty level count graph:
```

Fig 4.1-Train data difficulty level count graph

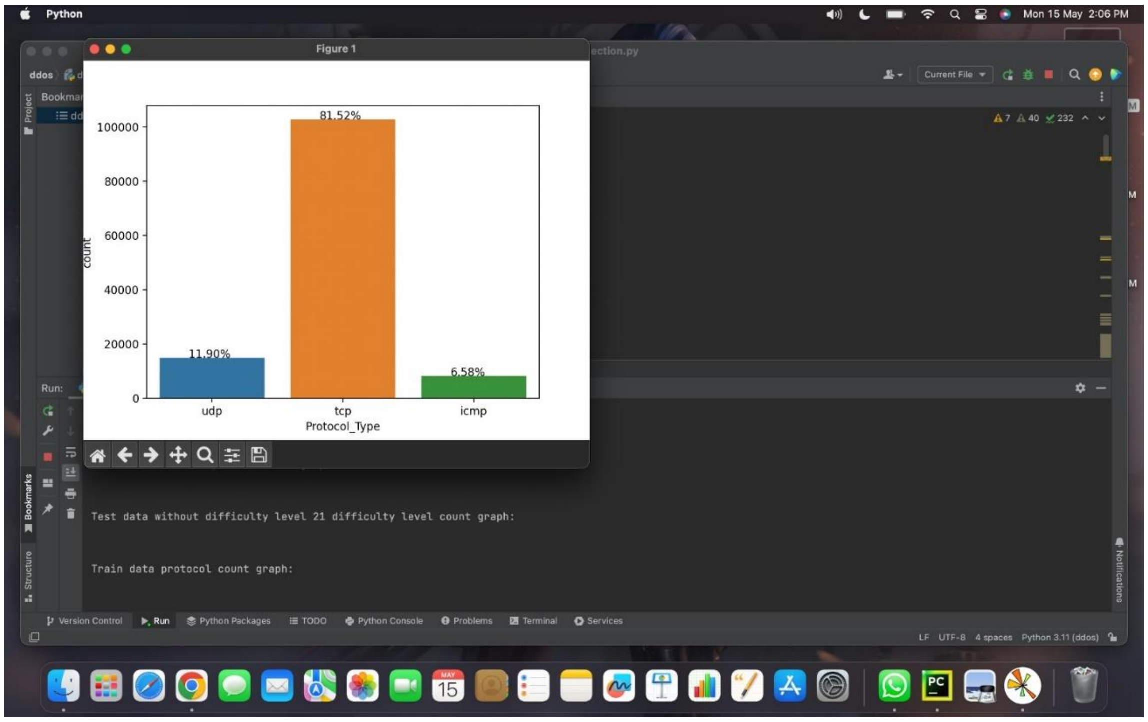
Test data difficulty level count graph:



```
charmProjects/ddos/ddos detection.py

Test data difficulty level count graph:

Test data without difficulty level 21 difficulty level count graph:
```

Fig 4.2-Test data difficulty level count graph

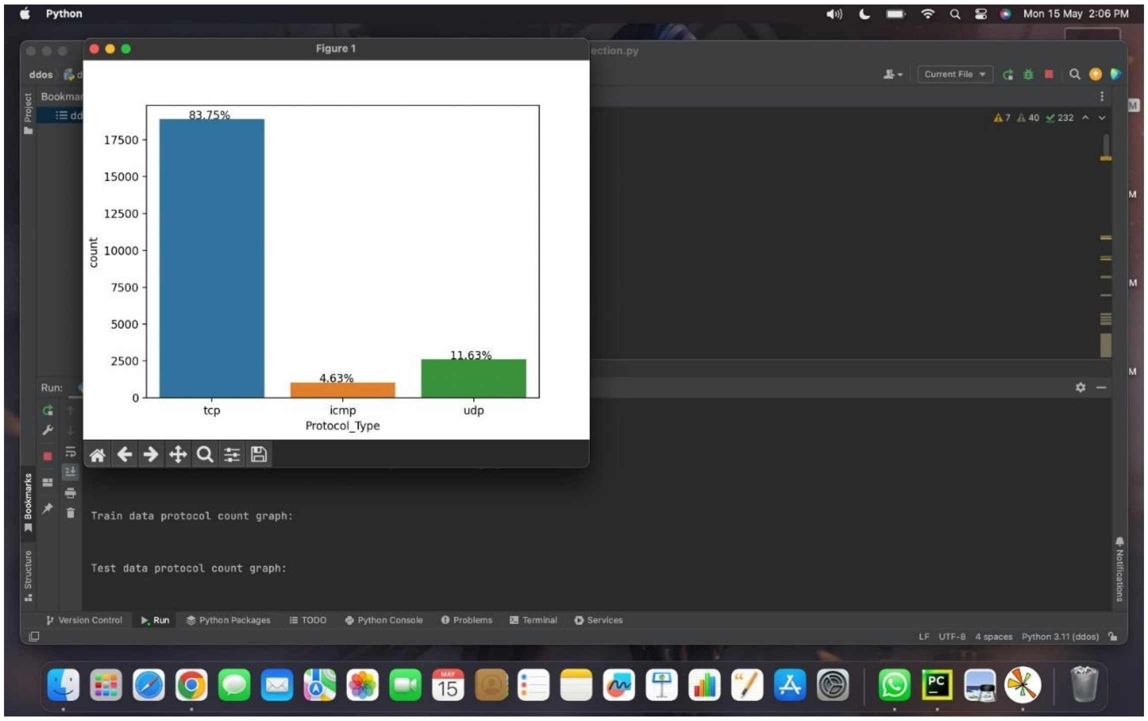
Test data without difficulty level 21 difficulty level count graph:



```
Test data without difficulty level 21 difficulty level count graph:

Train data protocol count graph:
```

Fig 4.3-Test data without difficulty level 21 difficulty level count graph

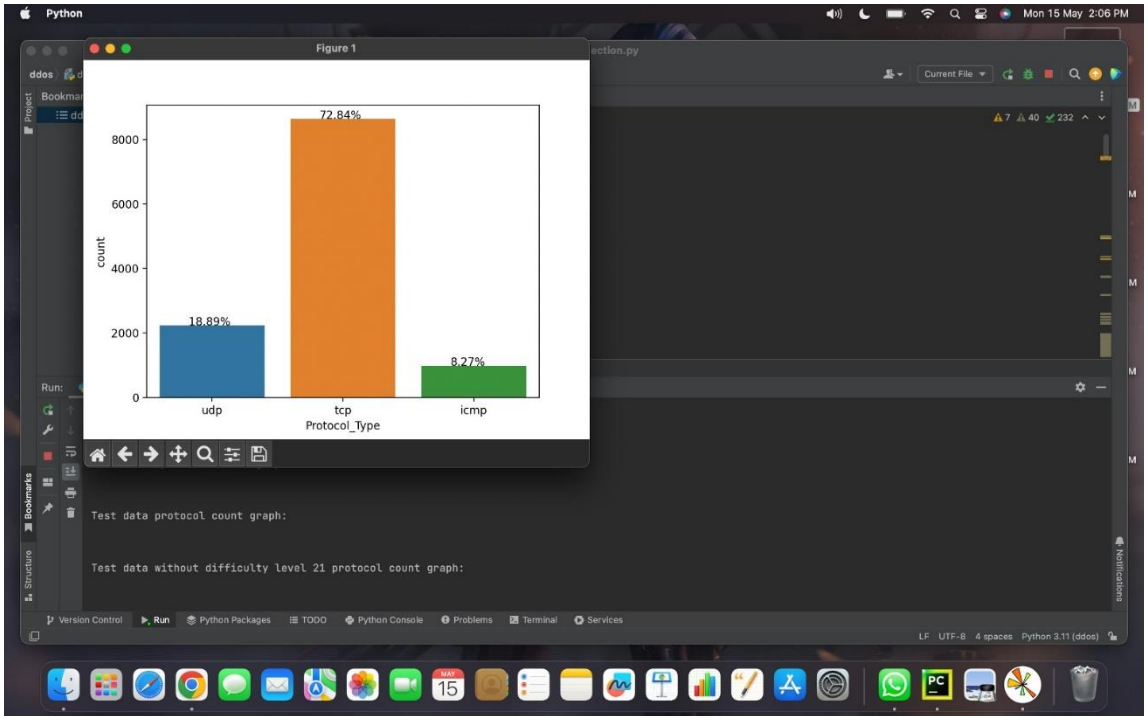
Train data protocol count graph:



```
Train data protocol count graph:

Test data protocol count graph:
```

Fig 4.4-Train data protocol count graph

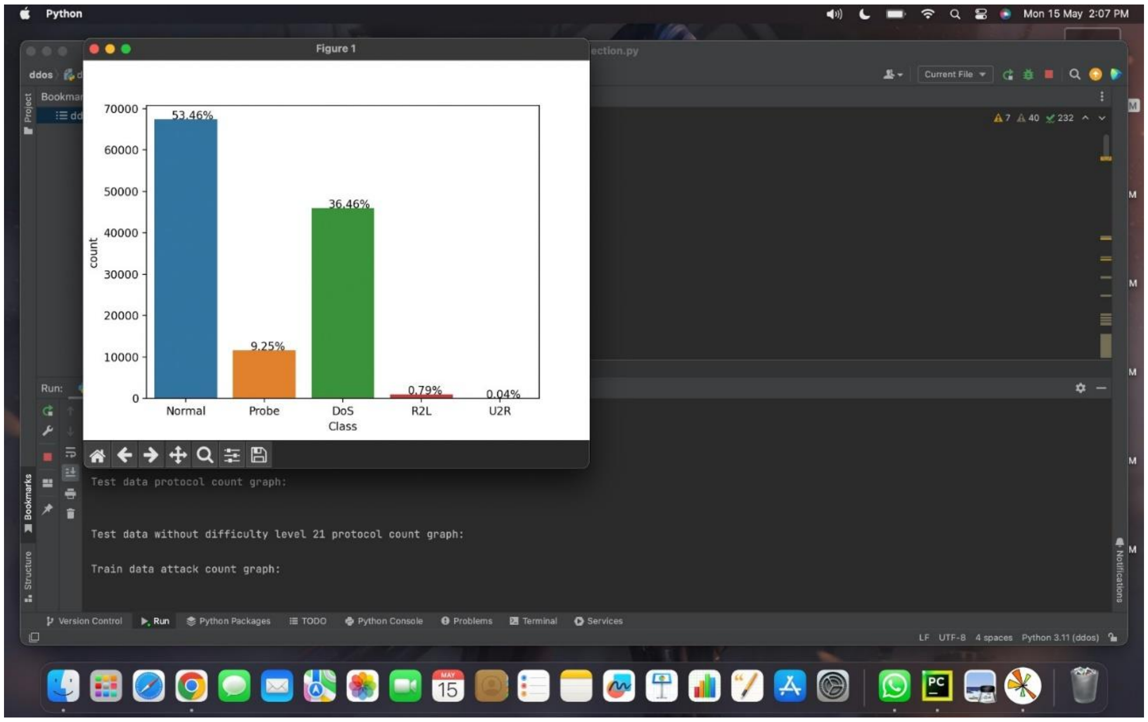
Test data protocol count graph:



```
Test data protocol count graph:


Test data without difficulty level 21 protocol count graph:
```

Fig 4.5-Test data protocol count graph

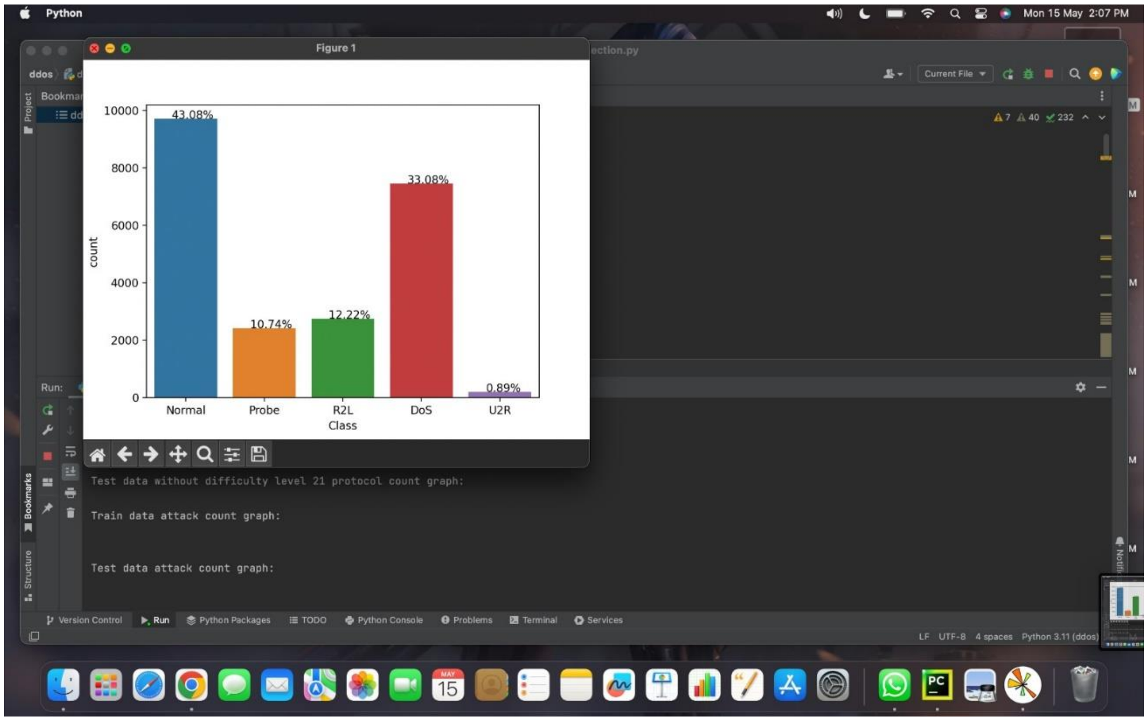
Test data without difficulty level 21 protocol count graph:



```
Test data protocol count graph:


Test data without difficulty level 21 protocol count graph:

Train data attack count graph:
```

Fig 4.6-Test data without difficulty level 21 protocol count graph

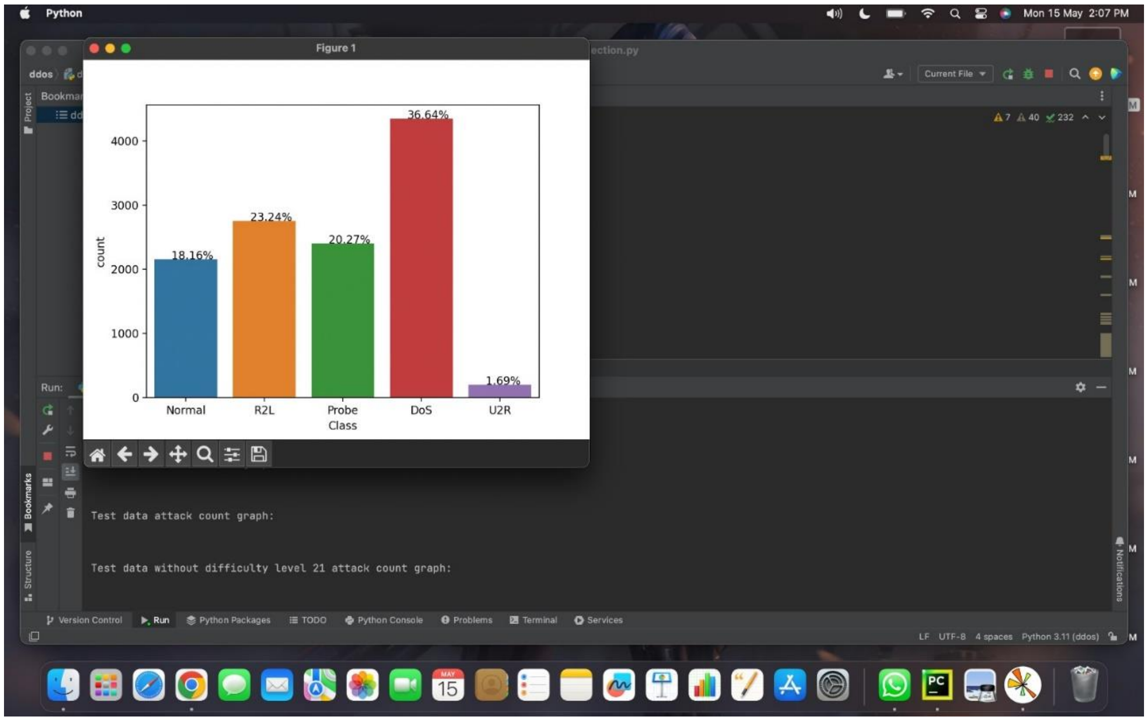
Train data attack count graph:



```
Test data without difficulty level 21 protocol count graph:

Train data attack count graph:


Test data attack count graph:
```

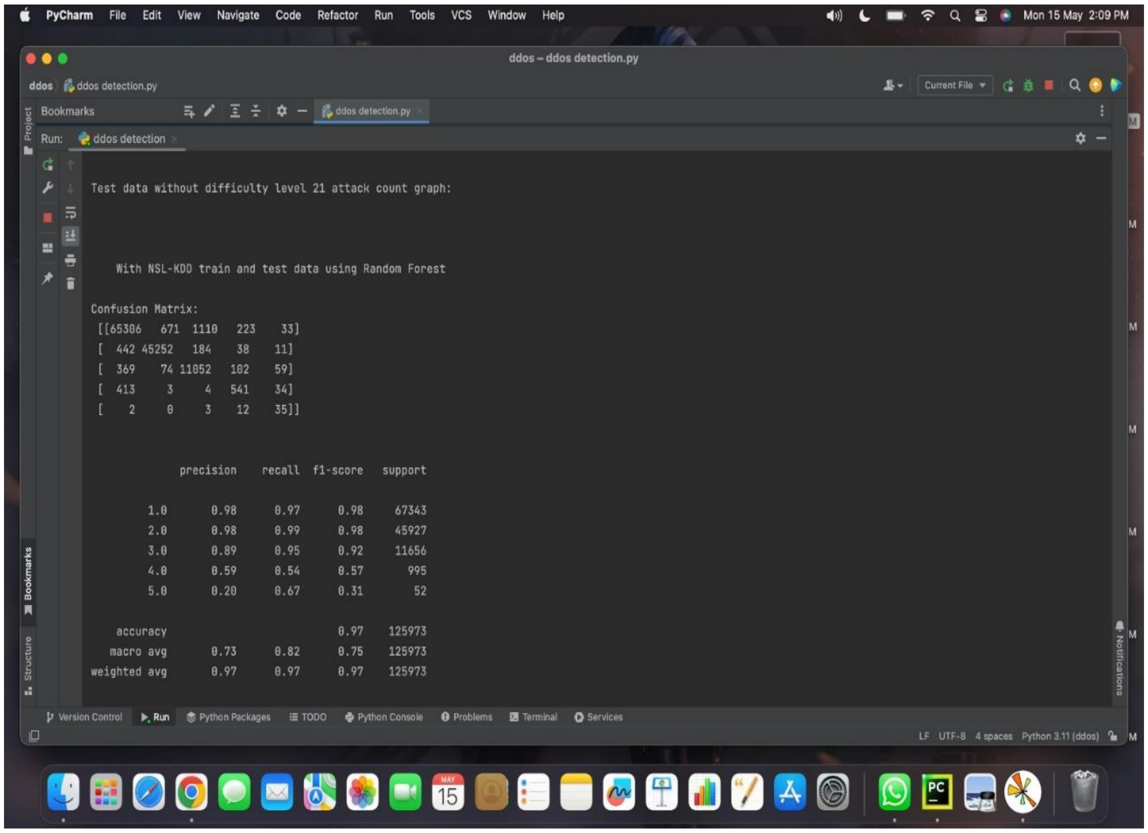
Fig 4.7-Train data attack count graph

Test data attack count graph:



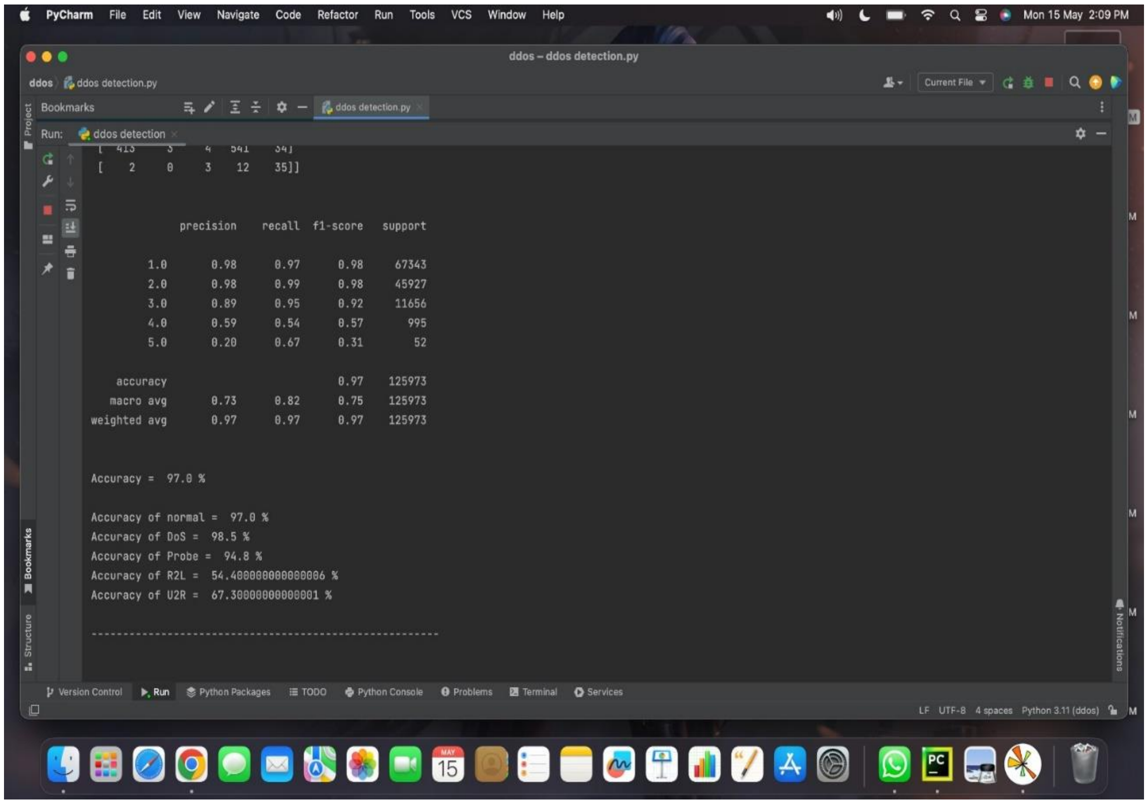
```
Test data attack count graph:

Test data without difficulty level 21 attack count graph:
```

Fig 4.8-Test data attack count graph



```
Test data without difficulty level 21 attack count graph:



   With NSL-KDD train and test data using Random Forest

Confusion Matrix:
 [[65306   671  1110   223    33]
 [  442 45252   184    38    11]
 [  369    74 11052   102    59]
 [  413     3     4   541    34]
 [    2     0     3    12    35]]


              precision    recall  f1-score   support

         1.0       0.98      0.97      0.98     67343
         2.0       0.98      0.99      0.98     45927
         3.0       0.89      0.95      0.92     11656
         4.0       0.59      0.54      0.57       995
         5.0       0.20      0.67      0.31        52

    accuracy                           0.97    125973
   macro avg       0.73      0.82      0.75    125973
weighted avg       0.97      0.97      0.97    125973
```

Fig 4.9- With NSL-KDD train and test data using Random Forest



```
 [  413     3     4   541    34]
 [    2     0     3    12    35]]


              precision    recall  f1-score   support

         1.0       0.98      0.97      0.98     67343
         2.0       0.98      0.99      0.98     45927
         3.0       0.89      0.95      0.92     11656
         4.0       0.59      0.54      0.57       995
         5.0       0.20      0.67      0.31        52

    accuracy                           0.97    125973
   macro avg       0.73      0.82      0.75    125973
weighted avg       0.97      0.97      0.97    125973


Accuracy =  97.0 %

Accuracy of normal =  97.0 %
Accuracy of DoS =  98.5 %
Accuracy of Probe =  94.8 %
Accuracy of R2L =  54.400000000000006 %
Accuracy of U2R =  67.30000000000001 %

------------------------------------------------------
```

Fig 4.10- Accuracy Percentages

CONCLUSION

Ddos attack will be imposing a huge threat to many big and small organizations, as it causes many different damages to the online users. The areas that may depend on human operator, massive computing time and lack of free data and resources availability. Still, we are lacking in lots of areas which need to be focus to detect DDoS attack. Algorithm such as Linear Regression, Random Forest, Support Vector Machine, Gaussian and Naïve Bayes classifier are used for identification of DDoS attack. The deep learning also plays major role for identification of DDoS attack which uses Convolution neural network. So, we have review on multiple technique to detect DDoS attack.

FUTURE WORK

A concept of SQL Injection attack detection model will be considered in conjunction with this model in the future. Where both attacks (SQL Injection and DDoS) can be prevented by the system. As a result, tests will be run on the SQL datasets consisting of both malicious and non-malicious queries. Where tokenization method will be used to split queries into tokens of features, which is the process of breaking the queries into meaningful elements called tokens. Multiple Machine learning algorithms will be used to train the model with these datasets for comparing the results to find the best possible algorithm to combine with role-based access control for classification for the most accuracy and precision while detecting SQL injection attacks.

REFERENCE

[1] Tuan, N.N. et al. (2020) ‘A ddos attack mitigation scheme in ISP Networks using machine learning based on SDN’, Electronics, 9(3), p. 413. doi:10.3390/electronics9030413.

[2] T. X. Y. Mahjabin and G. J. Sun, "A survey of distributed denial-of-service attack, prevention, and mitigation techniques," vol. 13, no. 12, 2017.

[3] ASSIGNMENT ON DDoS Attack Submitted by Shahin John JS.

[4] Lakshminarasimman, S., Ruswin, S. and Sundarakantham, K. (2017) ‘Detecting ddos attacks using decision tree algorithm’, 2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN) [Preprint]. doi:10.1109/icscn.2017.8085703.

[5] P. T. H. a. K. T. Khaing, "Detection Model for Danielof-Service Attacks using Random Forest and k-Nearest Neighbors," 2013.

[6] T. A. D. K.R.W.V.Bandara, "Preventing DDoS attack using Data mining Algorithms," 2016.

[7] Pande, S. et al. (2020) ‘DDOS detection using machine learning technique’, Recent Studies on Computational Intelligence, pp. 59–68. doi:10.1007/978-981-15-8469-5_5.

[8] Taher, K.A., Mohammed Yasin Jisan, B. and Rahman, Md.M. (2019) ‘Network intrusion detection using supervised machine learning technique with feature selection’, 2019 International Conference on Robotics,Electrical and Signal Processing Techniques (ICREST) [Preprint]. doi:10.1109/icrest.2019.8644161.

[9] Mishra, S. SQL injection detection using machine learning [Preprint]. doi:10.31979/etd.j5dj-ngvb.

[10] Barki, L. et al. (2016) ‘Detection of distributed denial of service attacks in software defined networks’, 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI) [Preprint]. doi:10.1109/icacci.2016.7732445.

[11] Tuan, N.N. et al. (2020) ‘A ddos attack mitigation scheme in ISP Networks using machine learning based on SDN’, Electronics, 9(3), p. 413. doi:10.3390/electronics9030413.

[12] G. Saporito. [Online]. Available: https://towardsdatascience.com/a-deeper-dive-into-thensl-kdd-data-set-15c753364657.

[13] Chen, L.-P. (2019) ‘Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar: Foundations of Machine Learning, Second edition’, Statistical Papers, 60(5), pp. 1793–1795. doi:10.1007/s00362-019-01124-9.

[14] T. Mitchell, Machine Learning. Burr Ridge, IL: McGraw Hill, 1997.