PROJECT REPORT

on

INTRUSION DETECTION SYSTEM USING MACHINE LEARNING ALGORITHMS

BY

Sonam Singh (1809010119)

Utpal Anand (1809010127)

Submitted to the Department of Computer Science and Engineering in

in partial fulfilment of the requirements of the degree of

Bachelor of Technology

in

Computer Science and Engineering

Under the Supervision of

Prof. Vipin Kr. Kushwaha

Department of Computer Science and Engineering

IEC College of Engineering and Technology, Greater Noida, U.P.-201310

Dr. A.P.J. Abdul Kalam Technical University, Lucknow (UP) India

May-2022

DECLARATION

I hereby declare that this submission is my work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has made in the text.

Signature:

Name:

Roll No. :

Date:

CERTIFICATE

Certified that Sonam Singh (1809010119), Utpal Anand (1809010127) has carried out the research work presented in this thesis entitled “Intrusion Detection System using Machine Learning Algorithm” for the award of Bachelor of Technology from Dr APJ Abdul Kalam Technical University, Lucknow under my/our (print only that is applicable) supervision. The thesis embodies results of original work, and studies are carried out by the student himself/herself and the contents of the thesis do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Signature

(Name of Supervisor)

(Designation)

(Address)

Date:

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B. Tech. Final Year. We owe special gratitude to Professor Vipin Kumar, Department of Computer Science & Engineering, College of Engineering, Lucknow, for his constant support and guidance throughout our work. His sincerity, thoroughness, and perseverance have been a continuous source of inspiration for us. It is only his conscious efforts that our endeavours' have seen the light of the day.

We also do not like to miss the opportunity to acknowledge the contribution of all department faculty members for their kind assistance and cooperation during the development of our project. Last but not least, we acknowledge our friends for their contribution to the completion of the project.

Signature:

Name :

Roll No.:

Date:

TABLE OF CONTENTS

Page

ABSTRACT

The increase in internet usage brings security problems with it. Malicious software can affect the operation of the systems and disrupt data confidentiality due to the security gaps in the systems. Network Intrusion detection systems are essential for the protection of advanced communication networks. Originally, these systems were hard-coded to identify specific signatures, patterns and rule violations; now, artificial intelligence and machine learning algorithms provide promising alternatives. We propose the approach of automated vulnerability detection by using Misuse and Anomaly intrusion detection techniques. However, the plethora of publicly available and potentially outdated datasets, the differences between those datasets, the variety of evaluation methods and the occasional unclear reporting of proposed techniques significantly complicate making a fair comparison.

This project aims to solve this issue and sets out a workflow used to run existing solutions on relevant datasets and made open source so that it can easily be applied to future solutions.

We aim to train the A.I. with the help of an improved dataset to automatically defend the network from known vulnerabilities instead of just generating a system administrator alert.

LIST OF FIGURES

LIST OF TABLES

Page

CHAPTER 1 :

INTRODUCTION, PROBLEM STATEMENT

Intrusion Detection System is a software application to detect network intrusion using various machine learning algorithms. IDS monitors a network or system for malicious activity and protects a computer network from unauthorized access from users, including perhaps insider. The intrusion detector learning task is to build a predictive model (i.e., a classifier) capable of distinguishing between ‘bad connections’ (intrusion/attacks) and a ‘good (normal) connections.

1.1 Types of IDS

Intrusion detection systems are designed to be deployed in different environments. And like many cybersecurity solutions, an IDS can either be host-based or network-based.

- **Host-Based IDS (HIDS):** A host-based IDS is deployed on a particular endpoint and designed to protect it against internal and external threats. Such an IDS may have the ability to monitor network traffic to and from the machine, observe running processes, and inspect the system’s logs. A host-based IDS’s visibility is limited to its host machine, decreasing the available context for decision-making, but has deep visibility into the host computer’s internals.
- **Network-Based IDS (NIDS):** A network-based IDS solution is designed to monitor an entire protected network. It has visibility into all traffic flowing through the network and makes determinations based upon packet metadata and contents. This wider viewpoint provides more context and the ability to detect widespread threats; however, these systems lack visibility into the internals of the endpoints that they protect.

Due to the different levels of visibility, deploying a HIDS or NIDS in isolation provides incomplete protection to an organization’s system. A unified threat management solution, which integrates multiple technologies in one system, can provide more comprehensive security.

IDS vs Firewalls

Intrusion Detection Systems and firewalls are both cybersecurity solutions that can be deployed to protect an endpoint or network. However, they differ significantly in their purposes.

An IDS is a passive monitoring device that detects potential threats and generates alerts, enabling security operations centre (SOC) analysts or incident responders to investigate and respond to the potential incident. An IDS provides no actual protection to the endpoint or network. A firewall, on the other hand, is designed to act as a protective system. It performs analysis of the metadata of network packets and allows or blocks traffic based upon predefined rules. This creates a boundary over which certain types of traffic or protocols cannot pass.

Since a firewall is an active protective device, it is more like an Intrusion Prevention System (IPS) than an IDS. An IPS is like an IDS but actively blocks identified threats instead of simply raising an alert. This complements the functionality of a firewall, and many next-generation firewalls (NGFWs) have integrated IDS/IPS functionality. This enables them to both enforce the predefined filtering rules (firewalls) and detect and respond to more sophisticated cyber threats (IDS/IPS). Learn more about the IPS vs IDS debate here

1.2 Detection Method of IDS Deployment

Beyond their deployment location, IDS solutions also differ in how they identify potential intrusions:

- **Signature Detection:** Signature-based IDS solutions use fingerprints of known threats to identify them. Once malware or other malicious content has been identified, a signature is generated and added to the list used by the IDS solution to test incoming content. This enables an IDS to achieve a high threat detection rate with no false positives because all alerts are generated based upon detection of known-malicious content. However, a signature-based IDS is limited to detecting known threats and is blind to zero-day vulnerabilities.
- **Anomaly Detection:** Anomaly-based IDS solutions build a model of the "normal" behavior of the protected system. All future behavior is compared to this model, and any anomalies are labeled as potential threats and generate alerts. While this approach can detect novel or zero-day threats, the difficulty of building an accurate model of "normal" behavior means that these systems must balance false positives (incorrect alerts) with false negatives (missed detections).
- **Hybrid Detection:** A hybrid IDS uses both signature-based and anomaly-based detection. This enables it to detect more potential attacks with a lower error rate than using either system in isolation..

Problem Statement: The task is to build a network intrusion detector, a predictive model capable of distinguishing between bad connections, called intrusions or attacks, and good normal connections.

Our work is on Network based intrusion detection system and uses Anomaly detection method of IDS development.

CHAPTER 2

CATEGORIES OF ATTACK

Attacks fall into four main categories:

- **DOS:** denial-of-service.

  e.g., syn flood;
- **R2L:** unauthorized access from a remote machine.

  e.g., guessing password;
- **U2R:** unauthorized access to local superuser (root) privileges.

  e.g., various "buffer overflow" attacks;
- **Probing:** surveillance and another probing.

  e.g., port scanning.

2.1 Denial-of-Service Attack

A denial-of-service (DoS) attack is a type of cyber attack in which a malicious actor aims to render a computer or other device unavailable to its intended users by interrupting the device's normal functioning. DoS attacks typically function by overwhelming or flooding a targeted machine with requests until normal traffic is unable to be processed, resulting in denial-of-service to addition users. A DoS attack is characterized by using a single computer to launch the attack.

A distributed denial-of-service (DDoS) attack is a type of DoS attack that comes from many distributed sources, such as a botnet DDoS attack.

How does a DoS attack work?

The primary focus of a DoS attack is to oversaturate the capacity of a targeted machine, resulting in denial-of-service to additional requests. The multiple attack vectors of DoS attacks can be grouped by their similarities.

DoS attacks typically fall in 2 categories:

1.Buffer overflow attacks

An attack type in which a memory buffer overflow can cause a machine to consume all available hard disk space, memory, or CPU time. This form of exploit often results in sluggish behavior, system crashes, or other deleterious server behaviours, resulting in denial-of-service.

2.Flood attacks

By saturating a targeted server with an overwhelming amount of packets, a malicious actor is able to oversaturate server capacity, resulting in denial-of-service. In order for most DoS flood attacks to be successful, the malicious actor must have more available bandwidth than the target.

What are some historically significant DoS attacks?

Historically, DoS attacks typically exploited security vulnerabilities present in network, software and hardware design. These attacks have become less prevalent as DDoS attacks have a greater disruptive capability and are relatively easy to create given the available tools. In reality, most DoS attacks can also be turned into DDoS attacks.

A few common historic DoS attacks include:

- Smurf attack - a previously exploited DoS attack in which a malicious actor utilizes the broadcast address of vulnerable network by sending spoofed packets, resulting in the flooding of a targeted IP address.
- Ping flood - this simple denial-of-service attack is based on overwhelming a target with ICMP (ping) packets. By inundating a target with more pings than it is able to respond to efficiently, denial-of-service can occur. This attack can also be used as a DDoS attack.
- Ping of Death - often conflated with a ping flood attack, a ping of death attack involves sending a malformed packet to a targeted machine, resulting in deleterious behaviour such as system crashes.

How can you tell if a computer is experiencing a DoS attack?

While it can be difficult to separate an attack from other network connectivity errors or heavy bandwidth consumption, some characteristics may indicate an attack is underway.

Indicators of a DoS attack include:

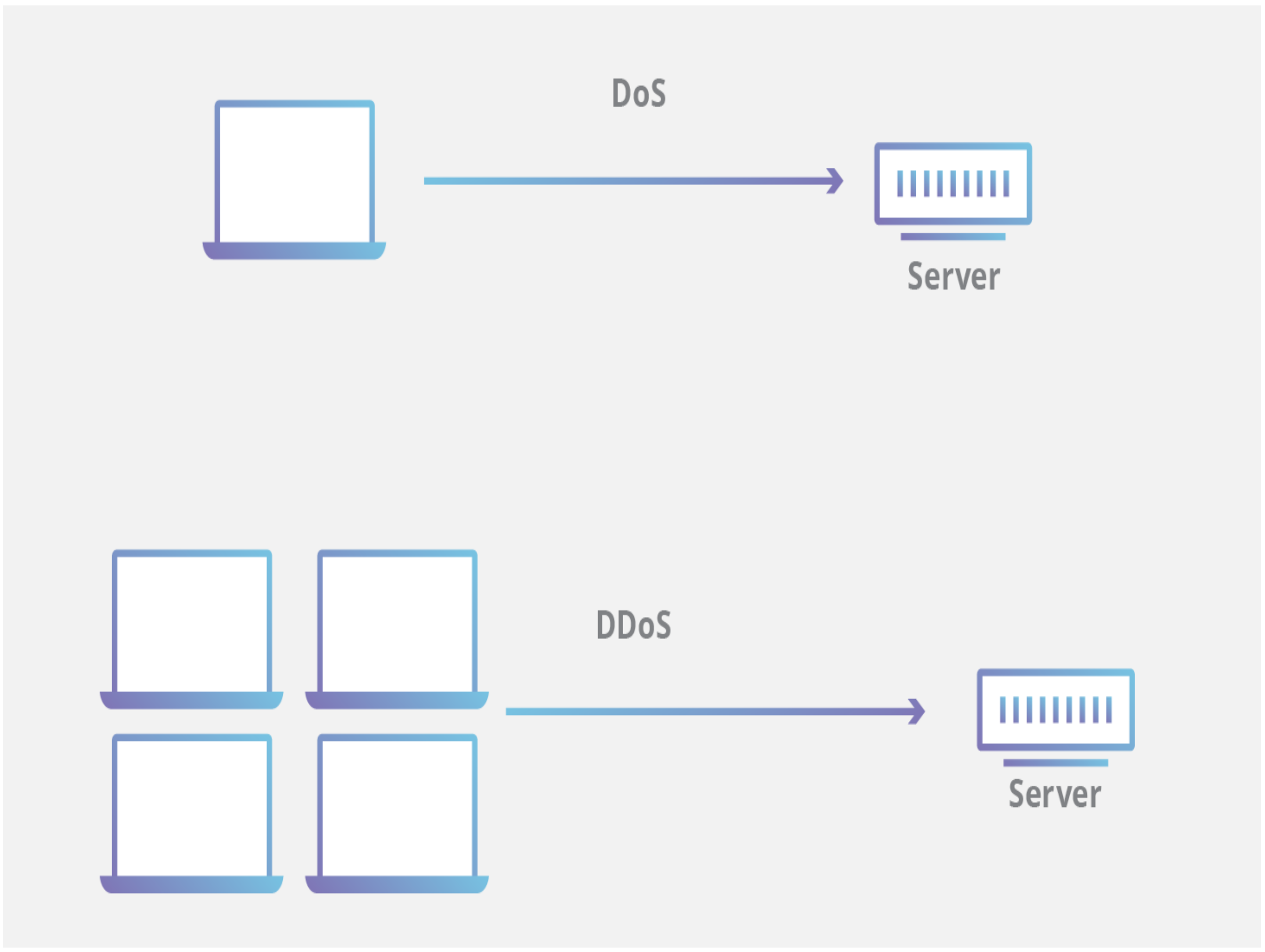
- Atypically slow network performance such as long load times for files or websites

- The inability to load a particular website such as your web property
- A sudden loss of connectivity across devices on the same network

What is the difference between a DDoS attack and a DOS attack?

The distinguishing difference between DDoS and DoS is the number of connections utilized in the attack. Some DoS attacks, such as “low and slow” attacks like Slowloris, derive their power in the simplicity and m

nimal requirements needed to them be effective.



(Figure – 1)

DoS utilizes a single connection, while a DDoS attack utilizes many sources of attack traffic, often in the form of a botnet. Generally speaking, many of the attacks are fundamentally similar and can be attempted using one more many sources of malicious traffic. Learn how Cloudflare's DDoS protection stops denial-of-service attacks.

2.2 Remote to Local Attack

This class of attacks sends packets to the network with an intention of prying on their vulnerabilities to gain illegal local access to resources that exist on that network. They include Ftp-Write, Xsnoop, Guest and the Dictionary that target misconfigured or weak system securities. Xlock attack is another that uses social engineering to gain access.

Remote to local attack (r2l) has been widely known to be launched by an attacker to gain unauthorized access to a victim machine in the entire network. Similarly user to root attack (u2r) is usually launched for illegally obtaining the root's privileges when legally accessing a local machine. One approach for detecting both attacks is to formulate both problems as a binary classification problem by deciding whether to accept or reject access requests from remote sites to local user machine or by accepting or rejecting access as root attempts. However, the cost caused by incorrect decision due to accepting illegitimate access request in a form of the damage that it might lead to is more expensive than the opposite case resulting from rejecting a valid access request. Due to this, in this paper we handle both problems in cost sensitive learning framework. We investigate how various cost-sensitive machine learning methods can be used to produce various cost sensitive detection models for detecting illegitimate remote access and access as a root requests. Those models are optimized for a user-defined cost matrix. Empirical experiment shows that the produced cost sensitive detection models are effective in reducing the overall cost of illegal remote access and access as root detection.

2.3 User to Root Attack

These attacks are exploitations in which the hacker starts off on the system with a normal user account and attempts to abuse vulnerabilities in the system in order to gain super user privileges e.g., perl, xterm.

2.4 Probing

Probing is an attack in which the hacker scans a machine or a networking device in order to determine weaknesses or vulnerabilities that may later be exploited so as to compromise the system. This technique is commonly used in data mining e.g. saint, portsweep, mscan, nmap etc.

Probing attacks are an invasive method for bypassing security measures by observing the physical silicon implementation of a chip. As an invasive attack, one directly accesses the internal wires and connections of a targeted device and extracts sensitive information.

The objective in the Probing Attacks mission is to find a weakness your opponent's line and break through into the rear. Recon and Scout troops are excellent at sneaking through the lines and harassing enemy supply chains and lines of communication and they are more effective if they can penetrate the line undetected.

CHAPTER 3

(KDD CUP 1999 DATASET)

3.1 Introduction

The original KDD Cup 1999 dataset from UCI machine learning repository contains 41 attributes (34 continuous, and 7 categorical), however, they are reduced to 4 attributes (service, duration, src_bytes, dst_bytes) as these attributes are regarded as the most basic attributes (see kddcup.names), where only 'service' is categorical. Using the 'service' attribute, the data is divided into {http, smtp, ftp, ftp_data, others} subsets. Here, only 'http' service data is used. Since the continuous attribute values are concentrated around '0', we transformed each value into a value far from '0', by $y = \log(x + 0.1)$

NSL-KDD is a data set suggested to solve some of the inherent problems of the KDD'99 data set which are mentioned in [1]. Although, this new version of the KDD data set still suffers from some of the problems discussed by McHugh and may not be a perfect representative of existing real networks, because of the lack of public data sets for network-based IDSs, we believe it still can be applied as an effective benchmark data set to help researchers compare different intrusion detection methods.

Furthermore, the number of records in the NSL-KDD train and test sets are reasonable. This advantage makes it affordable to run the experiments on the complete set without the need to randomly select a small portion. Consequently, evaluation results of different research work will be consistent and comparable.During the last decade, anomaly detection has attracted the attention of many researchers to overcome the weakness of signature-based IDSs in detecting novel attacks, and KDDCUP'99 is the mostly widely used data set for the evaluation of these systems. Having conducted a statistical analysis on this data set, we found two important issues which highly affects the performance of evaluated systems, and results in a very poor evaluation of anomaly detection approaches. To solve these issues, we have proposed a new data set, NSL-KDD, which consists of selected records of the complete KDD data set and does not suffer from any of mentioned shortcomings.

This is the data set used for The Second International Knowledge Discovery and Data Mining Tools Competition, which was held in conjunction with KDD-98 The Fourth International Conference on Knowledge Discovery and Data Mining. The competition task is a regression problem where the goal is to estimate the return from a direct mailing in order to maximize donation profits.

<u>3.2 Dataset Description:</u>

Data files:

- kddcup.names : A list of features.
- kddcup.data.gz : The full data set
- kddcup.data_10_percent.gz : A 10% subset.
- kddcup.newtestdata_10_percent_unlabeled.gz
- kddcup.testdata.unlabeled.gz
- kddcup.testdata.unlabeled_10_percent.gz
- corrected.gz : Test data with corrected labels.
- training_attack_types : A list of intrusion types.
- typo-correction.txt : A brief note on a typo in the data set that has been corrected

3.3 <u>Features:</u>

feature name	description	type
duration	Length (number of seconds) of the connection	Continuous
protocol_type	Type of the protocol, e.g. tcp, udp, etc.	Discrete
service	Network service on the destination, e.g., http, telnet, etc.	Discrete
src_bytes	Number of data bytes from source to destination	Continuous
dst_bytes	Number of data bytes from destination to source	Continuous
flag	Normal or error status of the connection	Discrete
land	1 if connection is from/to the same host/port; 0 otherwise1	Discrete
wrong_fragment	Number of “wrong” fragments	Continuous

urgent	Number of urgent packets	Continuous

Table 1: Basic features of individual TCP connections.

feature name	description	Type
hot	number of “hot” indicators	Continuous
num_failed_logins	number of failed login attempts	Continuous
logged_in	1 if successfully logged in; 0 otherwise	Discrete
num_compromised	number of “compromised” conditions	Continuous
root_shell	1 if root shell is obtained; 0 otherwise	Discrete
su_attempted	1 if “su root” command attempted; 0 otherwise	Discrete
num_root	number of “root” accesses	Continuous
num_file_creations	number of file creation operations	Continuous
num_shells	number of shell prompts	Continuous
num_access_files	number of operations on access control files	Continuous
num_outbound_cmds	number of outbound commands in an ftp session	Continuous
is_hot_login	1 if the login belongs to the “hot” list; 0 otherwise	Discrete

Table 2: Content features within a connection suggested by domain knowledge.

feature name	description	type
count	number of connections to the same host as the current connection in the past two seconds	Continuous
serror_rate	% of connections that have "SYN" errors	Continuous
rerror_rate	% of connections that have "REJ" errors	Continuous
same_srv_rate	% of connections to the same service	Continuous
diff_srv_rate	% of connections to different services	Continuous
srv_count	number of connections to the same service as the current connection in the past two seconds	Continuous
srv_serror_rate	% of connections that have "SYN" errors	Continuous
srv_rerror_rate	% of connections that have "REJ" errors	Continuous
srv_diff_host_rate	% of connections to different hosts	Continuous

Table 3: Traffic features computed using a two-second time window.

<u>Improvements to the KDD'99 dataset</u>

The NSL-KDD data set has the following advantages over the original KDD data set:

- It does not include redundant records in the train set, so the classifiers will not be biased towards more frequent records.
- There is no duplicate records in the proposed test sets; therefore, the performance of the learners are not biased by the methods which have better detection rates on the frequent records.

- The number of selected records from each difficultylevel group is inversely proportional to the percentage of records in the original KDD data set. As a result, the classification rates of distinct machine learning methods vary in a wider range, which makes it more efficient to have an accurate evaluation of different learning techniques.
- The number of records in the train and test sets are reasonable, which makes it affordable to run the experiments on the complete set without the need to randomly select a small portion. Consequently, evaluation results of different research works will be consistent and comparable.

Statistical observations

One of the most important deficiencies in the KDD data set is the huge number of redundant records, which causes the learning algorithms to be biased towards the frequent records, and thus prevent them from learning unfrequent records which are usually more harmful to networks such as U2R and R2L attacks. In addition, the existence of these repeated records in the test set will cause the evaluation results to be biased by the methods which have better detection rates on the frequent records.

In addition, we analyzed the difficulty level of the records in KDD data set. Surprisingly, about 98% of the records in the train set and 86% of the records in the test set were correctly classified with all the 21 learners.

In order to perform our experiments, we randomly created three smaller subsets of the KDD train set each of which included fifty thousand records of information. Each of the learners where trained over the created train sets. We then employed the 21 learned machines (7 learners, each trained 3 times) to label the records of the entire KDD train and test sets, which provides us with 21 predicated labels for each record. Further, we annotated each record of the data set with a *#successfulPrediction* value, which was initialized to zero. Now, since the KDD data set provides the correct label for each record, we compared the predicated label of each record given by a specific learner with the actual label, where we incremented *#successfulPrediction* by one if a match was found. Through this process, we calculated the number of learners that were able to correctly label that given record. The highest value for *#successfulPrediction* is 21, which conveys the fact that all learners were able to correctly predict the label of that record.

Statistics of redundant records in the KDD train set

Original records | Distinct records | Reduction rate

- **Attacks:** 3,925,650 | 262,178 | 93.32%

- **Normal:** 972,781 | 812,814 | 16.44%
- **Total:** 4,898,431 | 1,074,992 | 78.05%

Statistics of redundant records in the KDD test set

Original records | Distinct records | Reduction rate

- **Attacks:** 250,436 | 29,378 | 88.26%
- **Normal:** 60,591 | 47,911 | 20.92%
- **Total:** 311,027 | 77,289 | 75.15%

The KDD Cup 99 dataset is one of the most widely used training sets; it is based on the DARPA 1998 dataset. This dataset contains 4 900 000 replicated attacks on record. There is one type of the normal type with the identity of normal and 22 attack types, which are divided into five major categories: DoS (Denial of Service attacks), R2L (Root to Local attacks),

U2R (User to Root attack), Probe (Probing attacks) and Normal. For each record, the KDD Cup 99 training dataset contains 41 fixed feature attributes and a class identifier. Of the 41 fixed feature attributes, seven characteristic properties are the symbolic type; the others are continuous. In addition, the features include basic features , content features , and traffic features as shown in Table 2. The testing set has specific attack types that disappear in the training set, which allows it to provide a more realistic theoretical basis for intrusion detection. To date, the KDD Cup ’99 dataset remains the most

thoroughly observed and freely available dataset, with fully labeled connection records spanning several weeks of network traffic and a large number of different attacks [23]. Each connection record contains 41 input features grouped into basic features and higher-level features. The basic features are directly extracted or derived from the header information of IP packets and TCP/UDP segments in the tcpdump files of each session.

The listfiles for tcpdump from the DARPA training data were used to label the connection records. The so-called content-based higher-level features use domain knowledge to look specifically for attacks in the actual data of the segments recorded in the tcpdump files. These address ‘r2l’ and ‘u2r’ attacks, which occasionally either require only a single connection or are without any prominent sequential patterns. Typical features include the number of failed login attempts and whether root access was obtained during the session. Furthermore, there are time-based and connection-based derived features to address ‘DoS’ and ‘probe’ attacks. Time- based features examine connections within a time window of two seconds and provide statistics about these. To provide statistical information about attacks exceeding a two-second time-window, such as slow probing attacks, connection-based features use a connection window of 100 connections.

Both are further split into same-host features, which provide statistics about connections with the same destination host, and same-service features, which examine only connections with the same service . The KDD Cup ’99 competition provides the training and testing datasets in a full set and also provides a so-called ‘10%’ subset version. The ‘10%’ subset was created due to the huge amount of connection records present in the full set; some ‘DoS’ attacks have millions of records.

Therefore, not all of these connection records were selected. Furthermore, only connections within a time-window of five minutes before and after the entire duration of an attack were added into the ‘10%’ datasets . To achieve approximately the same distribution of intrusions and normal traffic as the original DARPA dataset, a selected set of sequences with ‘nor-mal’ connections were also left in the ‘10%’ dataset. Training and test sets have different probability distributions.

The full training dataset contains nearly five million records. The full training dataset and the corresponding ‘10%’ both contain 22 different attack types in the order that they were used in the 1998 DARPA experiments. The full test set, with nearly three million records, is only available unlabeled; however, a ‘10%’ subset is provided both as unlabeled and labeled test data. It is specified as the ‘corrected’ subset, with a different distribution and additional attacks not part of the training set. For the KDD Cup ’99 competition, the ‘10%’ subset was intended for training.

The ‘corrected’ subset can be used for performance testing; it has over 300,000 records containing 37 different attacks.

CHAPTER 4

MACHINE LEARNING ALGORITHM

Various Algorithms Applied : Gaussian Naive Bayes, Decision Tree, Random Forest, Support Vector Machine, Logistic Regression.

4.1 Gaussian Naive Bayes:

Naïve Bayes is a probabilistic machine learning algorithm used for many classification functions and is based on the Bayes theorem. Gaussian Naïve Bayes is the extension of naïve Bayes. While other functions are used to estimate data distribution, Gaussian or normal distribution is the simplest to implement as you will need to calculate the mean and standard deviation for the training data.

What is the Naive Bayes Algorithm?

Naive Bayes is a probabilistic machine learning algorithm that can be used in several classification tasks. Typical applications of Naive Bayes are classification of documents, filtering spam, prediction and so on. This algorithm is based on the discoveries of Thomas Bayes and hence its name.

The name “Naïve” is used because the algorithm incorporates features in its model that are independent of each other. Any modifications in the value of one feature do not directly impact the value of any other feature of the algorithm. The main advantage of the Naïve Bayes algorithm is that it is a simple yet powerful algorithm.

It is based on the probabilistic model where the algorithm can be coded easily, and predictions did quickly in real-time. Hence this algorithm is the typical choice to solve real-world problems as it can be tuned to respond to user requests instantly. But before we dive deep into Naïve Bayes and Gaussian Naïve Bayes, we must know what is meant by conditional probability.

Conditional Probability Explained

We can understand conditional probability better with an example. When you toss a coin, the probability of getting ahead or a tail is 50%. Similarly, the probability of getting a 4 when you roll dice with faces is 1/6 or 0.16.

If we take a pack of cards, what is the probability of getting a queen given the condition that it is a spade? Since the condition is already set that it must be a spade, the denominator or the selection set becomes 13. There is only one queen in spades, hence the probability of picking a queen of spade becomes 1/13= 0.07.

The conditional probability of event A given event B means the probability of event A occurring given that event B has already occurred. Mathematically, the conditional probability of A given B can be denoted as P[A|B] = P[A AND B] / P[B].

Let us consider a little complex example. Take a school with a total of 100 students. This population can be demarcated into 4 categories- Students, Teachers, Males and Females. Consider the tabulation given below:

	Female	Male	Total
Teacher	8	12	20
Student	32	48	80
Total	40	50	100

(Table- 4)

Here, what is the conditional probability that a certain resident of the school is a Teacher given the condition that he is a Man.

To calculate this, you will have to filter the sub-population of 60 men and drill down to the 12 male teachers.

So, the expected conditional probability P[Teacher | Male] = 12/60 = 0.2

P (Teacher | Male) = P (Teacher ∩ Male) / P(Male) = 12/60 = 0.2

This can be represented as a Teacher(A) and Male(B) divided by Male(B). Similarly, the conditional probability of B given A can also be calculated. The rule that we use for Naïve Bayes can be concluded from the following notations:

$P(A \mid B) = P(A \cap B) / P(B)$

$P(B \mid A) = P(A \cap B) / P(A)$

The Bayes Rule

In the Bayes rule, we go from P (X | Y) that can be found from the training dataset to find P (Y | X). To achieve this, all you need to do is replace A and B with X and Y in the above formulae. For observations, X would be the known variable and Y would be the unknown variable. For each row of the dataset, you must calculate the probability of Y given that X has already occurred.

But what happens where there are more than 2 categories in Y? We must compute the probability of each Y class to find out the winning one.

Through Bayes rule, we go from P (X | Y) to find P (Y | X)

Known from training data: $P(X \mid Y) = P(X \cap Y) / P(Y)$

P (Evidence | Outcome)

Unknown – to be predicted for test data: $P(Y \mid X) = P(X \cap Y) / P(X)$

P (Outcome | Evidence)

Bayes Rule = $P(Y \mid X) = P(X \mid Y) * P(Y) / P(X)$

The Naïve Bayes

The Bayes rule provides the formula for the probability of Y given condition X. But in the real world, there may be multiple X variables. When you have independent features, the Bayes rule can be extended to the Naïve Bayes rule. The X's are independent of each other. The Naïve Bayes formula is more powerful than the Bayes formula

Gaussian Naïve Bayes

So far, we have seen that the X's are in categories but how to compute probabilities when X is a continuous variable? If we assume that X follows a particular distribution, you can use the probability density function of that distribution to calculate the probability of likelihoods.

If we assume that X's follow a Gaussian or normal distribution, we must substitute the probability density of the normal distribution and name it Gaussian Naïve Bayes. To compute this formula, you need the mean and variance of X.

$$P(X|Y=c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{\frac{-(x-\mu_c)^2}{2\sigma_c^2}}$$

In the above formulae, sigma and mu is the variance and mean of the continuous variable X computed for a given class c of Y.

Representation for Gaussian Naïve Bayes

The above formula calculated the probabilities for input values for each class through a frequency. We can calculate the mean and standard deviation of x's for each class for the entire distribution.

This means that along with the probabilities for each class, we must also store the mean and the standard deviation for every input variable for the class.

mean(x) = 1/n * sum(x)

where n represents the number of instances and x is the value of the input variable in the data.

standard deviation(x) = sqrt(1/n * sum(xi-mean(x)^2))

Here square root of the average of differences of each x and the mean of x is calculated where n is the number of instances, sum() is the sum function, sqrt() is the square root function, and xi is a specific x value.

Predictions with the Gaussian Naïve Bayes Model

The Gaussian probability density function can be used to make predictions by substituting the parameters with the new input value of the variable and as a result, the Gaussian function will give an estimate for the new input value's probability.

Naïve Bayes Classifier

The Naïve Bayes classifier assumes that the value of one feature is independent of the value of any other feature. Naïve Bayes classifiers need training data to estimate the parameters required for classification. Due to simple design and application, Naïve Bayes classifiers can be suitable in many real-life scenarios.

Conclusion

The Gaussian Naïve Bayes classifier is a quick and simple classifier technique that works very well without too much effort and a good level of accuracy.

4.2 <u>Decision Tree:</u>

Decision trees give structure to a set of rules, derived from and using the input features for, among other things, classification . Random forests (RF) then combine many different decision trees for the same problem and take an ensemble of their results to achieve a more robust classifier . In , the RF approach is used to construct a misuse- based classifier, an anomaly-detector and a hybrid approach evaluated on KDD99.

For the misuse-based approach, a RF is used as a classifier to differentiate between normal traffic and intrusion, resulting in a reported error rate of 7.07%. By using the proximity of samples of traffic in the RF, outliers can be detected for anomaly detection with a reported DR of 65% and FPR of 1%. Finally, by running all traffic unclear to the misuse-classifier through the anomaly-detector, the authors propose a hybrid IDS approach with an overall DR of 94.7%. As NIDS datasets suffer from class imbalance, Wei Zong et al. propose a two-stage approach for network intrusion detection.

After over-sampling the minority classes and down-sampling the majority classes, they first classify input data into a minority class or other. This other class is then classified into the different majority classes in the second step. Both classifiers are RF, but can be exchanged for other techniques. After testing on UNSW-NB15, they achieve a result comparable to the best result in the analysis of that dataset .

Combining ensemble methods and DL, the authors inaim to achieve high detection performance with a low training time. They apply both Multi-Grained Traversing as well as Cascade Forest in an effort to surpass the work they compare against, which succeeds.

4.3 <u>Random Forest:</u>

Random forests (RF) then combine many different decision trees for the same problem and take an ensemble of their results to achieve a more robust classifier [90]. In [54], the RF approach is used to construct a misuse- based classifier, an anomaly-detector and a hybrid approach evaluated on KDD99. For the misuse-based approach, a RF is used as a classifier to differentiate between normal traffic and intrusion, resulting in a reported error rate of 7.07%.

By using the proximity of samples of traffic in the RF, outliers can be detected for anomaly detection with a reported DR of 65% and FPR of 1%. Finally, by running all traffic unclear to the misuse-classifier through the anomaly-detector, the authors propose a hybrid IDS approach with an overall DR of 94.7%.

As NIDS datasets suffer from class imbalance, Wei Zong et al. propose a two-stage approach for network intrusion detection. After over-sampling the minority classes and down-sampling the majority classes, they first classify input data into a minority class or other. This other class is then classified into the different majority classes in the second step. Both classifiers are RF, but can be exchanged for other techniques. After testing on UNSW-NB15, they achieve a result comparable to the best result in the analysis of that dataset .

Combining ensemble methods and DL, the authors inaim to achieve high detection performance with a low training time. They apply both Multi-Grained Traversing as well as Cascade Forest in an effort to surpass the work they compare against, which succeeds.

4.4 <u>Support Vector Machine:</u>

Even to date, one of the most prevalent ML-based intrusion detection techniques is the Support Vector Machine (SVM).

SVMs are classifiers that work by finding the hyperplane separating classes with a maximal margin . In 2003, researchers explored the use of Robust SVMs to classify process and session data from the DARPA1998 dataset . Starting from the observation that datasets supposedly clean usually contain some noise consisting of malicious traffic, they strove to design a robust classifier trainable on noisy data. By modifying the constraints and objective function of the SVM, they obtained a recall of 100% with a FPR of 8% where the reference SVM was unable to even reach a recall of100% without having the FPR reach 100% as well. The authors in describe another way to deal with dataset issues, proposing a modified K-means approach to generate a high-quality dataset from KDD99.

By using a distance threshold, they add a cluster centroid to the set whenever a sample exceeds this threshold for every other centroid. Furthermore, they construct a multi-level classifier, detecting a traffic class at each level. Notably, they use an Extreme Learning Machine (ELM) rather than an SVM to detect Probe attacks, as they argue an ELM is better suited for that task. With every other classifier being an SVM, they achieved state-of-the-art results.

In , the authors use SVM and Multilayer Perceptron classifiers on UNSW-NB15, while employing the feature dimensionality reduction approaches Principal Component Analysis (PCA) and a chi-squares test. They achieve both a high accuracy and F-measure of above 90%, surpassing other work

on the same dataset. In , the authors strive to implement a stable and accurate incremental SVM learning scheme.
This allows for manageable retraining in real-time to counteract increasing training set and training time. By using reserved sets with weighted non-support vectors, they can retrain the classifier without having to use the entire training set. Moreover, they also modify the RBF kernel to further reduce training and testing speed and increase reliability.

Support Vector Machine (SVM) is one of the most robust and accurate methods in all machine-learning algorithms. It primarily includes Support Vector Classification (SVC) and Support Vector Regression (SVR). The SVC is based on the concept of decision boundaries. A decision boundary separates a set of instances having different class values between two groups. The SVC supports both binary and multi- class classifications.
The support vector is the closest point to the separation hyperplane, which determines the optimal separation hyperplane. In the classification process, the mapping input vectors located on the separation hyperplane side of the feature space fall into one class, and the positions fall into the other class on the other side of the plane. In the case of data points that are not linearly separable, the SVM uses appropriate kernel functions to map them into higher dimensional spaces so that they become separable in those spaces .
Kotpalliwar et al. choose two representative datasets "Mixed" and "10% KDD Cup 99" datasets. The RBF is used as a kernel function for SVM to classify DoS, Probe, U2R, and R2L datasets. The study calculates parameter values related to intrusion-detector performance evaluation. The validation accuracy of the "mixed" dataset and the classification accuracy of the "10% KDD" dataset were estimated to be 89.85% and 99.9%, respectively.
Unfortunately, the study did not assess accuracy or recall except for accuracy. Saxena et al. [23] proposed a Hybrid PSO-SVM approach for building IDS. The study used two feature reduction techniques: Information Gain and BPSO. The 41 attributes reduced to 18 attributes. The classification performance was reported as 99.4% on the DoS, 99.3% on Probe or Scan, 98.7% on R2L, and 98.5% on the U2R.
The method provides a good detection rate in the case of a Denial of Service (DoS) attack and achieves a good detection rate in the case of U2R and R2L attacks. However, the precision of Probe, U2R and R2L is 84.2%, 25.0% and 89.4%, respectively. In other words, the method provided by the essay leads to a higher false alarm rate.

Pervez et al. proposes a filtering algorithm based on a Support Vector Machine (SVM) classifier to select multiple intrusion classification tasks on the NSL-KDD intrusion detection dataset. The method achieves 91% classification accuracy using only three input features and 99% classification accuracy using 36 input features, whereas all 41 input features achieve 99% classification accuracy. The method performed well on the training set with an F1-score of 0.99.

However, in the test set, the performance is worse; the F1-score is only 0.77. With poor generalization, it cannot effectively detect unknown network intrusions. The work by Chandrasekhar et al. [32] integrates fuzzy C-means clustering, an artificial neural network and support vector machine-intrusion detection technology.

With the help of the fuzzy C-means clustering technique, the heterogeneous training data are collected into homogeneous subsets, reducing the complexity of each subset, which helps to improve detection accuracy. After the initial clustering, ANNs are trained on the corresponding homogeneous subsets and use the linear SVM classifier to perform the final classification. The experimental results obtained with the calibrated KDD CUP 1999 dataset show the effectiveness of this method. In the same work, the KDD Cup 99 dataset is divided into 4 subsets according to different intrusion types and trained separately; DoS and PROBE attacks have a higher frequency and can be effortlessly separated from normal activity.

In contrast, U2R and R2L attacks are embedded in the data portion of the packet, making it difficult to achieve detection accuracy on both types of attacks. The technique has attained a consistent peak scores for all types of intrusions. Overall accuracy of the DoS, Probe, R2L and U2R categories was 99.66%, 98.55%, 98.99% and 98.81%, respectively. Compared with other reported intrusion detection approaches, this method is better in classification effect, but the trained classifier cannot effectively detect the abnormal in the actual network.

4.5 Logistic Regression:

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes.

In simple words, the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no).

Mathematically, a logistic regression model predicts P(Y=1) as a function of X. It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection etc.

Types of Logistic Regression

Generally, logistic regression means binary logistic regression having binary target variables, but there can be two more categories of target variables that can be predicted by it. Based on those number of categories, Logistic regression can be divided into following types −

Binary or Binomial

In such a kind of classification, a dependent variable will have only two possible types either 1 and 0. For example, these variables may represent success or failure, yes or no, win or loss etc.

Multinomial

In such a kind of classification, dependent variable can have 3 or more possible unordered types or the types having no quantitative significance. For example, these variables may represent "Type A" or "Type B" or "Type C".

Ordinal

In such a kind of classification, dependent variable can have 3 or more possible ordered types or the types having a quantitative significance. For example, these variables may represent "poor" or "good", "very good", "Excellent" and each category can have the scores like 0,1,2,3.

Logistic Regression Assumptions

Before diving into the implementation of logistic regression, we must be aware of the following assumptions about the same −

In case of binary logistic regression, the target variables must be binary always and the desired outcome is represented by the factor level 1.

There should not be any multi-collinearity in the model, which means the independent variables must be independent of each other .

We must include meaningful variables in our model.

We should choose a large sample size for logistic regression.

Binary Logistic Regression model

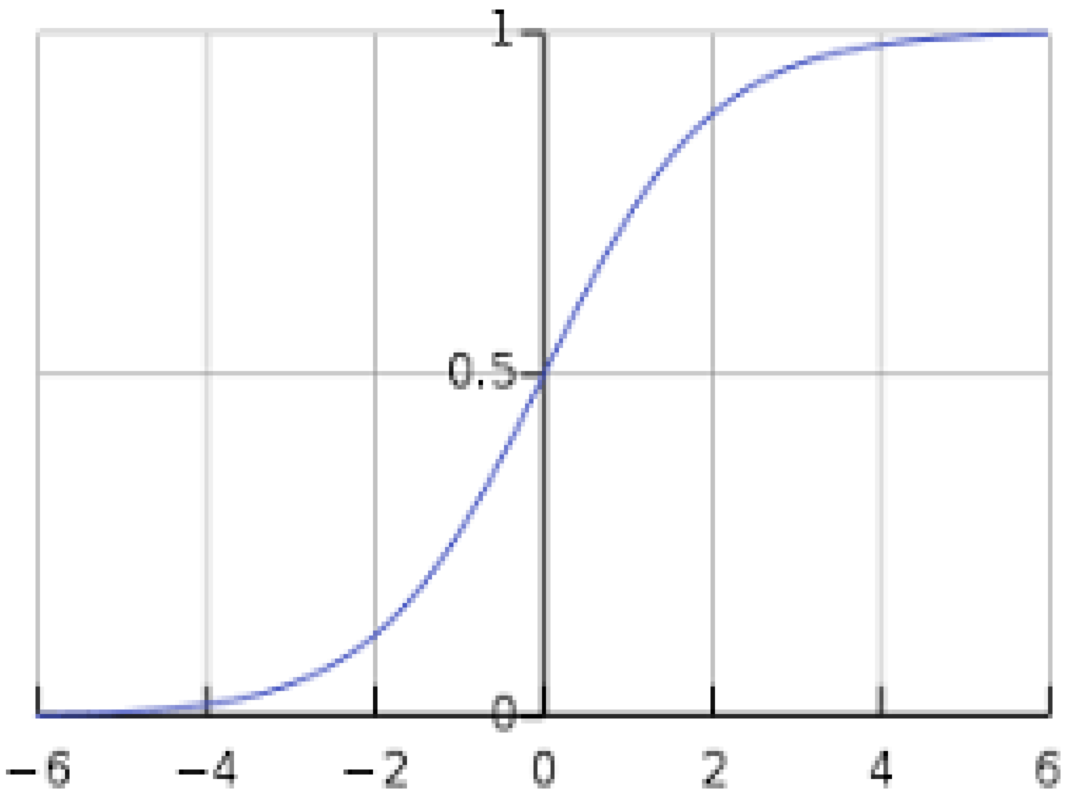
The simplest form of logistic regression is binary or binomial logistic regression in which the target or dependent variable can have only 2 possible types either 1 or 0. It allows us to model a relationship between multiple predictor variables and a binary/binomial target variable. In case of logistic regression, the linear function is basically used as an input to another function such as g in the following relation −

$$h\theta(x)=g(\theta Tx)\text{where}0\leq h\theta\leq 1$$

Here, g is the logistic or sigmoid function which can be given as follows −

$$g(z)=11+e-z\text{where}z=\theta Tx$$

To sigmoid curve can be represented with the help of following graph. We can see the values of y-axis lie between 0 and 1 and crosses the axis at 0.5.



(Figure -2)

The classes can be divided into positive or negative. The output comes under the probability of positive class if it lies between 0 and 1. For our implementation, we are interpreting the output of hypothesis function as positive if it is ≥ 0.5, otherwise negative.

We also need to define a loss function to measure how well the algorithm performs using the weights on functions, represented by theta as follows −

$$h=g(X\theta)$$

$$J(\theta)=1m.(-yTlog(h)-(1-y)Tlog(1-h))$$

Now, after defining the loss function our prime goal is to minimize the loss function. It can be done with the help of fitting the weights which means by increasing or decreasing the weights. With the help of derivatives of the loss function w.r.t each weight, we would be able to know what parameters should have high weight and what should have smaller weight.

The following gradient descent equation tells us how loss would change if we modified the parameters −

$$\delta J(\theta)\delta\theta j=1mXT(g(X\theta)-y)$$

CHAPTER 5

DATA PREPROCESSING

Code: Importing libraries and reading features list from ‘kddcup.names’ file.

```
import os

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import time


# reading features list

with open("..\\kddcup.names", 'r') as f:

    print(f.read())
```

Code: Appending columns to the dataset and adding a new column name target’ to the dataset.

```
cols ="""duration,

protocol_type,

service,

flag,

src_bytes,

dst_bytes,

land,
wrong_fragment,

urgent,

hot,

num_failed_logins,

logged_in,

num_compromised,

root_shell,

su_attempted,

num_root,

num_file_creations,

num_shells,

num_access_files,

num_outbound_cmds,

is_host_login,

is_guest_login,

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_same_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"""


columns =[]

for c in cols.split(', '):

    if(c.strip()):

       columns.append(c.strip())


columns.append('target')

print(len(columns))
```

Output:

```
42
```

Code: Reading the ‘attack_types’ file.

```
with open("..\\training_attack_types", 'r') as f:

    print(f.read())
```

Output:

back dos

buffer_overflow u2r

ftp_write r2l

guess_passwd r2l

imap r2l

ipsweep probe

land dos

loadmodule u2r

multihop r2l

neptune dos

nmap probe

perl u2r

phf r2l

pod dos

portsweep probe

rootkit u2r

satan probe

smurf dos

spy r2l

teardrop dos

warezclient r2l

warezmaster r2l

Code: Creating a dictionary of attack_types

```
attacks_types = {

    'normal': 'normal',
'back': 'dos',

'buffer_overflow': 'u2r',

'ftp_write': 'r2l',

'guess_passwd': 'r2l',

'imap': 'r2l',

'ipsweep': 'probe',

'land': 'dos',

'loadmodule': 'u2r',

'multihop': 'r2l',

'neptune': 'dos',

'nmap': 'probe',

'perl': 'u2r',

'phf': 'r2l',

'pod': 'dos',

'portsweep': 'probe',

'rootkit': 'u2r',

'satan': 'probe',

'smurf': 'dos',

'spy': 'r2l',

'teardrop': 'dos',

'warezclient': 'r2l',

'warezmaster': 'r2l',

}
```

Code: Reading the dataset('kddcup.data_10_percent.gz') and adding Attack Type feature in the training dataset where attack type feature has 5 distinct values i.e., dos, normal, probe, r2l, u2r.

```
path = "..\\kddcup.data_10_percent.gz"

df = pd.read_csv(path, names = columns)


# Adding Attack Type column

df['Attack Type'] = df.target.apply(lambda r:attacks_types[r[:-1]])

df.head()
```

Code: Shape of dataframe and getting data type of each feature

```
df.shape
```

Output:

```
 (494021, 43)
```

Code: Finding missing values of all features.

```
df.isnull().sum()
```

Output:

```
duration              0
protocol_type         0
service               0
flag                  0
src_bytes             0
dst_bytes            0
```

land	0
wrong_fragment	0
urgent	0
hot	0
num_failed_logins	0
logged_in	0
num_compromised	0
root_shell	0
su_attempted	0
num_root	0
num_file_creations	0
num_shells	0
num_access_files	0
num_outbound_cmds	0
is_host_login	0
is_guest_login	0
count	0
srv_count	0
serror_rate	0
srv_serror_rate	0
rerror_rate	0
srv_rerror_rate	0
same_srv_rate	0
diff_srv_rate	0
srv_diff_host_rate	0
dst_host_count	0
dst_host_srv_count	0
dst_host_same_srv_rate	0

```
dst_host_diff_srv_rate         0
dst_host_same_src_port_rate    0
dst_host_srv_diff_host_rate    0
dst_host_serror_rate           0
dst_host_srv_serror_rate       0
dst_host_rerror_rate           0
dst_host_srv_rerror_rate       0
target                         0
Attack Type                    0
dtype: int64
```

No missing value found, s

can further proceed to our next step.

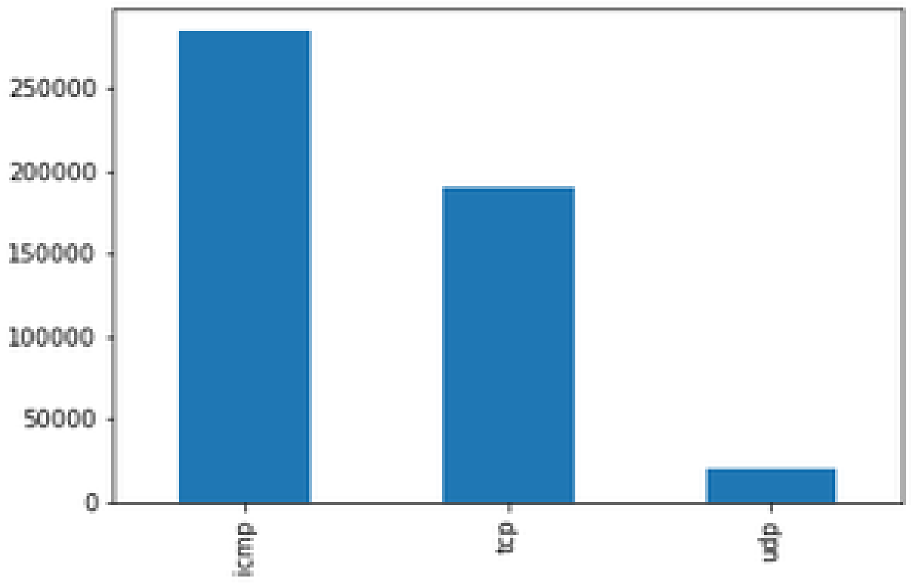
Code: Finding Categorical Features

```
# Finding categorical features
num_cols = df._get_numeric_data().columns


cate_cols = list(set(df.columns)-set(num_cols))
cate_cols.remove('target')
cate_cols.remove('Attack Type')


cate_cols
```

Output:

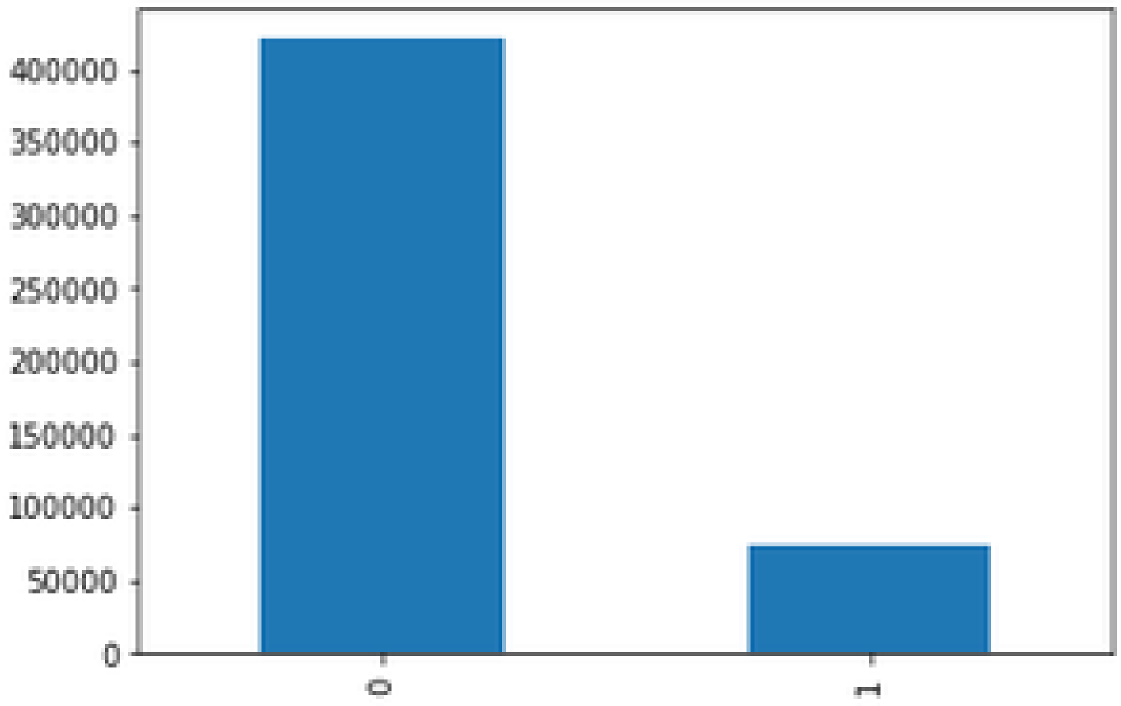
```
 ['service', 'flag', 'protocol_type']
```

Visualizing Categorical Features using bar graph



(Figure - 3)

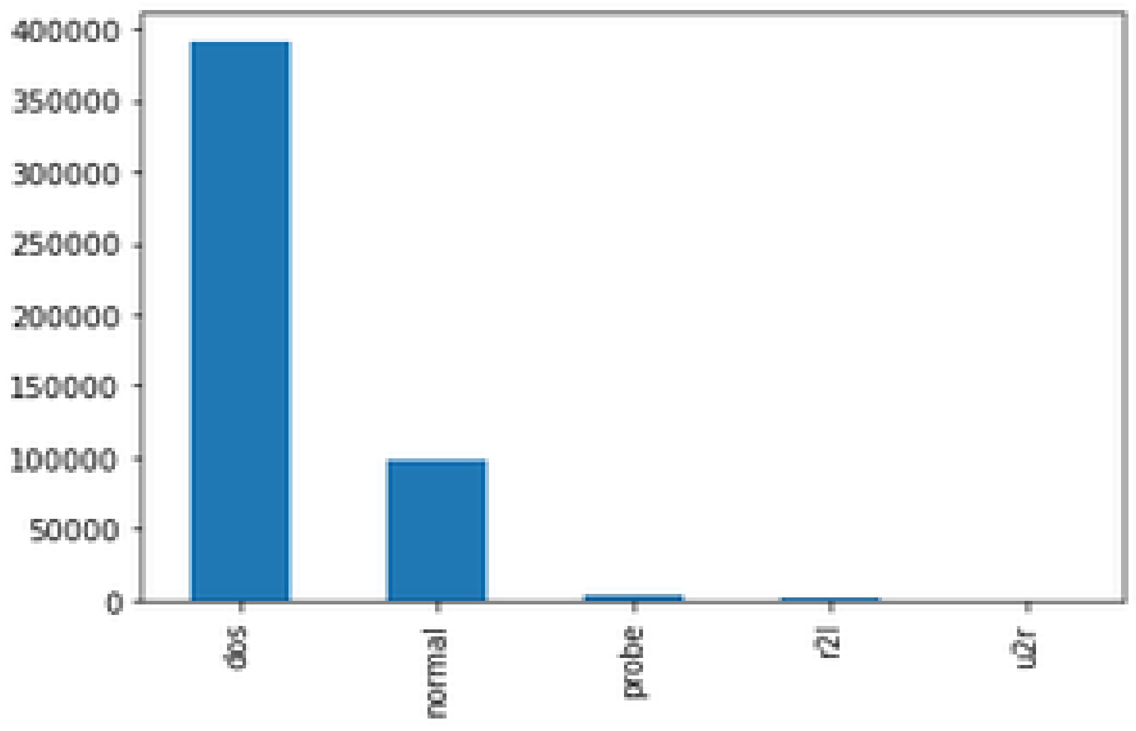
Protocol type: We notice that ICMP is the most present in the used data, then TCP and almost 20000 packets of UDP type



(Figure - 4)

*l*ogged_in (1 if successfully logged in; 0 otherwise): We notice that just 70000 packets are successfully logged in.

Target Feature Distribution:



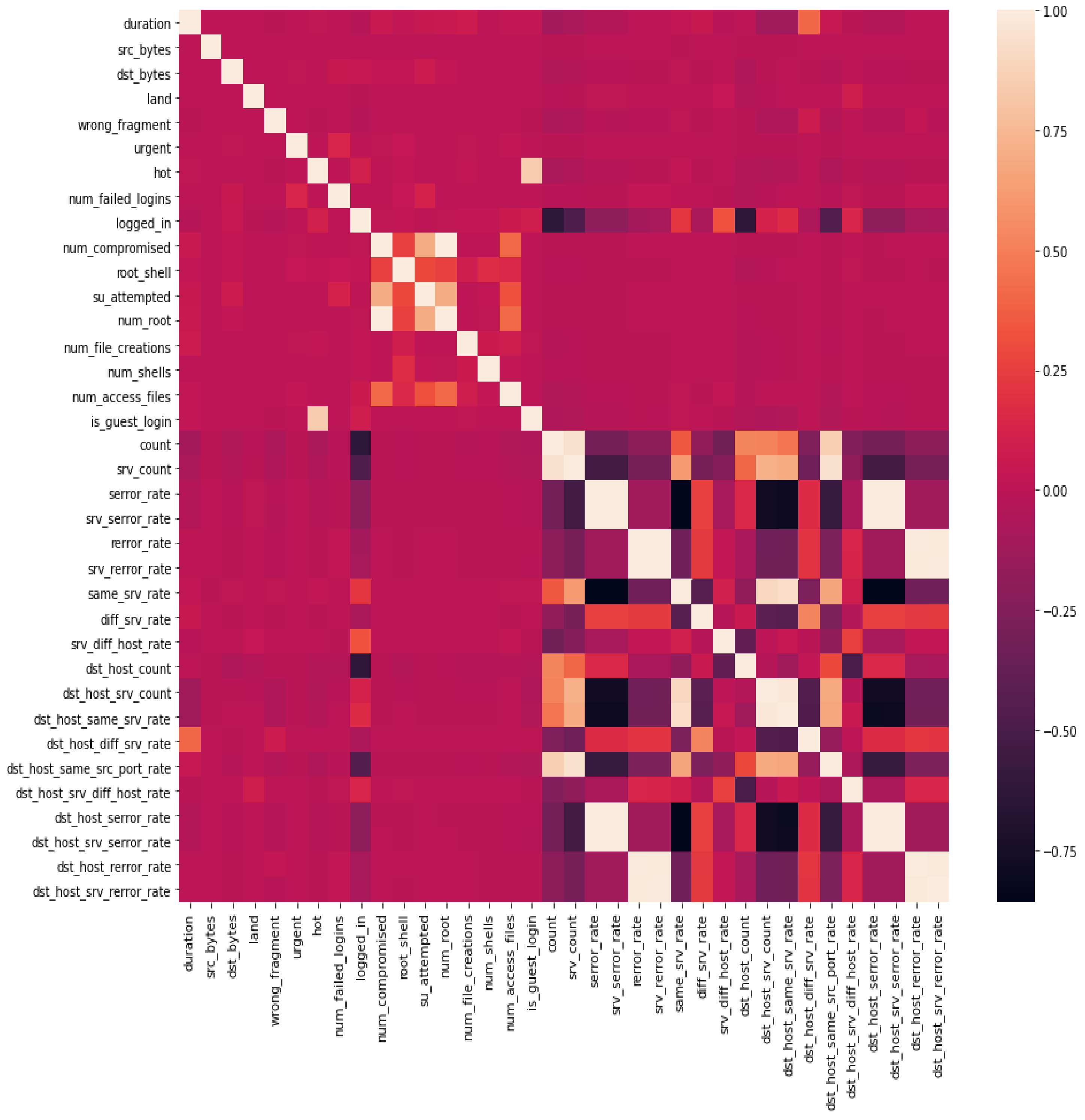
(Figure - 5)

Attack Type(The attack types grouped by attack, it's what we will predict)

Code: Data Correlation – Find the highly correlated variables using heatmap and ignore them for analysis.

```
df = df.dropna('columns')# drop columns with NaN

df = df[[col for col in df if df[col].nunique() > 1]]# keep columns where there are more than 1 unique values

corr = df.corr()

plt.figure(figsize =(15, 12))

sns.heatmap(corr)

plt.show()
```

Output:



(Figure - 6)

HEATMAP

A Heatmap (or heat map) is a type of data visualization that displays aggregated information in a visually appealing way. User interaction on a website such as clicks/taps, scrolls, mouse movements, etc. create heatmaps. To get the most useful insight the activity is then scaled (least to most).

To display the data, heatmaps use a warm-to-cool schema for their visual representation. As a result, warmer colours (reds) show higher activity while cooler colours (blues) represent lower activity.

Correlation is a statistical measure that expresses the strength of the relationship between two variables. The two main types of correlation are positive and negative. Positive correlation occurs when two variables move in the same direction; as one increases, so do the other. For example, there is a positive correlation between hours of study and grades on a test. A negative correlation occurs when two variables move in opposite directions; as one increases, the other decreases. For example, there is a negative correlation between smoking and life expectancy. Correlation can be used to test hypotheses about cause-and-effect relationships between variables. Correlation is often used in the real world to predict trends. For example, if there is a strong positive correlation between the number of hours spent studying and grades on a test, we can predict that if someone spends more hours studying, they will get a higher grade on the test.

What is Correlation Heatmap?

Correlation heatmaps are a type of plot that visualize the strength of relationships between numerical variables. Correlation plots are used to understand which variables are related to each other and the strength of this relationship. A correlation plot typically contains a number of numerical variables, with each variable represented by a column. The rows represent the relationship between each pair of variables. The values in the cells indicate the strength of the relationship, with positive values indicating a positive relationship and negative values indicating a negative relationship. Correlation heatmaps can be used to find potential relationships between variables and to understand the strength of these relationships. In addition, correlation plots can be used to identify outliers and to detect linear and nonlinear relationships. The color-coding of the cells makes it easy to identify relationships between variables at a glance. Correlation heatmaps can be used to find both linear and nonlinear relationships between variables.

- A correlation heatmap is a graphical representation of a correlation matrix representing the correlation between different variables.
- The value of correlation can take any value from -1 to 1.
- Correlation between two random variables or bivariate data does not necessarily imply a causal relationship.
- Correlation between two variables can also be determined using a scatter plot between these two variables.

Code:

```
# This variable is highly correlated with num_compromised and should be ignored for analysis.
#(Correlation = 0.9938277978738366)
df.drop('num_root', axis = 1, inplace = True)


# This variable is highly correlated with serror_rate and should be ignored for analysis.
#(Correlation = 0.9983615072725952)
df.drop('srv_serror_rate', axis = 1, inplace = True)


# This variable is highly correlated with rerror_rate and should be ignored for analysis.
#(Correlation = 0.9947309539817937)
df.drop('srv_rerror_rate', axis = 1, inplace = True)


# This variable is highly correlated with srv_serror_rate and should be ignored for analysis.
#(Correlation = 0.9993041091850098)
df.drop('dst_host_srv_serror_rate', axis = 1, inplace = True)


# This variable is highly correlated with rerror_rate and should be ignored for analysis.
#(Correlation = 0.9869947924956001)
```

```
df.drop('dst_host_serror_rate', axis = 1, inplace = True)


# This variable is highly correlated with srv_rerror_rate and should be ignored for analysis.

#(Correlation = 0.9821663427308375)

df.drop('dst_host_rerror_rate', axis = 1, inplace = True)


# This variable is highly correlated with rerror_rate and should be ignored for analysis.

#(Correlation = 0.9851995540751249)

df.drop('dst_host_srv_rerror_rate', axis = 1, inplace = True)


# This variable is highly correlated with srv_rerror_rate and should be ignored for analysis.

#(Correlation = 0.9865705438845669)

df.drop('dst_host_same_srv_rate', axis = 1, inplace = True)
```

Code: Feature Mapping – Apply feature mapping on features such as : ‘protocol_type’ & ‘flag’.

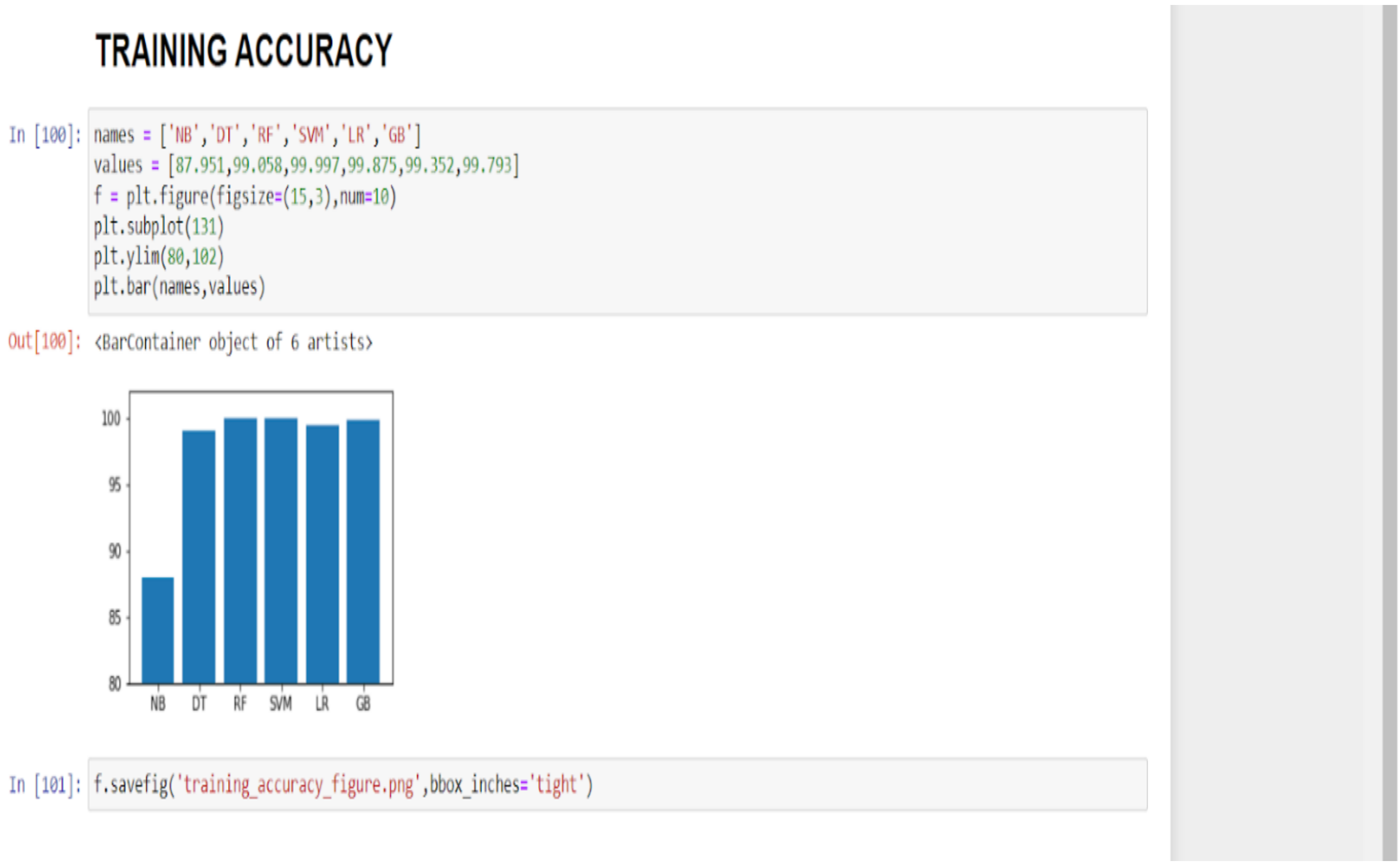
```
# protocol type feature mapping

pmap = {'icmp':0, 'tcp':1, 'udp':2}

df['protocol_type'] = df['protocol_type'].map(pmap)
```

CHAPTER 6

CONCLUION

TO CHECK WHICH ALGORITHM IS BETTER :

1 Training accuracy
2 Testing accuracy
3 Training time
4 Testing time



TRAINING ACCURACY

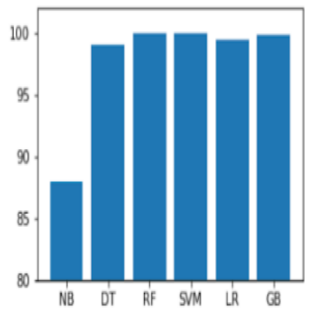
```
In [100]: names = ['NB','DT','RF','SVM','LR','GB']
          values = [87.951,99.058,99.997,99.875,99.352,99.793]
          f = plt.figure(figsize=(15,3),num=10)
          plt.subplot(131)
          plt.ylim(80,102)
          plt.bar(names,values)

Out[100]: <BarContainer object of 6 artists>
```

```
In [101]: f.savefig('training_accuracy_figure.png',bbox_inches='tight')
```

(Figure - 7)

TRAINING ACCURACY

```
In [100]: names = ['NB','DT','RF','SVM','LR','GB']
          values = [87.951,99.058,99.997,99.875,99.352,99.793]
          f = plt.figure(figsize=(15,3),num=10)
          plt.subplot(131)
          plt.ylim(80,102)
          plt.bar(names,values)

Out[100]: <BarContainer object of 6 artists>
```

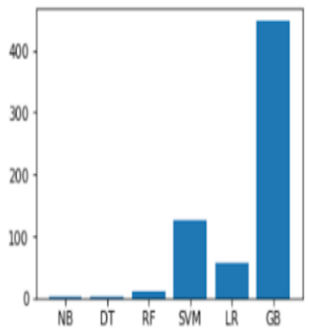


```
In [101]: f.savefig('training_accuracy_figure.png',bbox_inches='tight')
```

(Figure - 8)

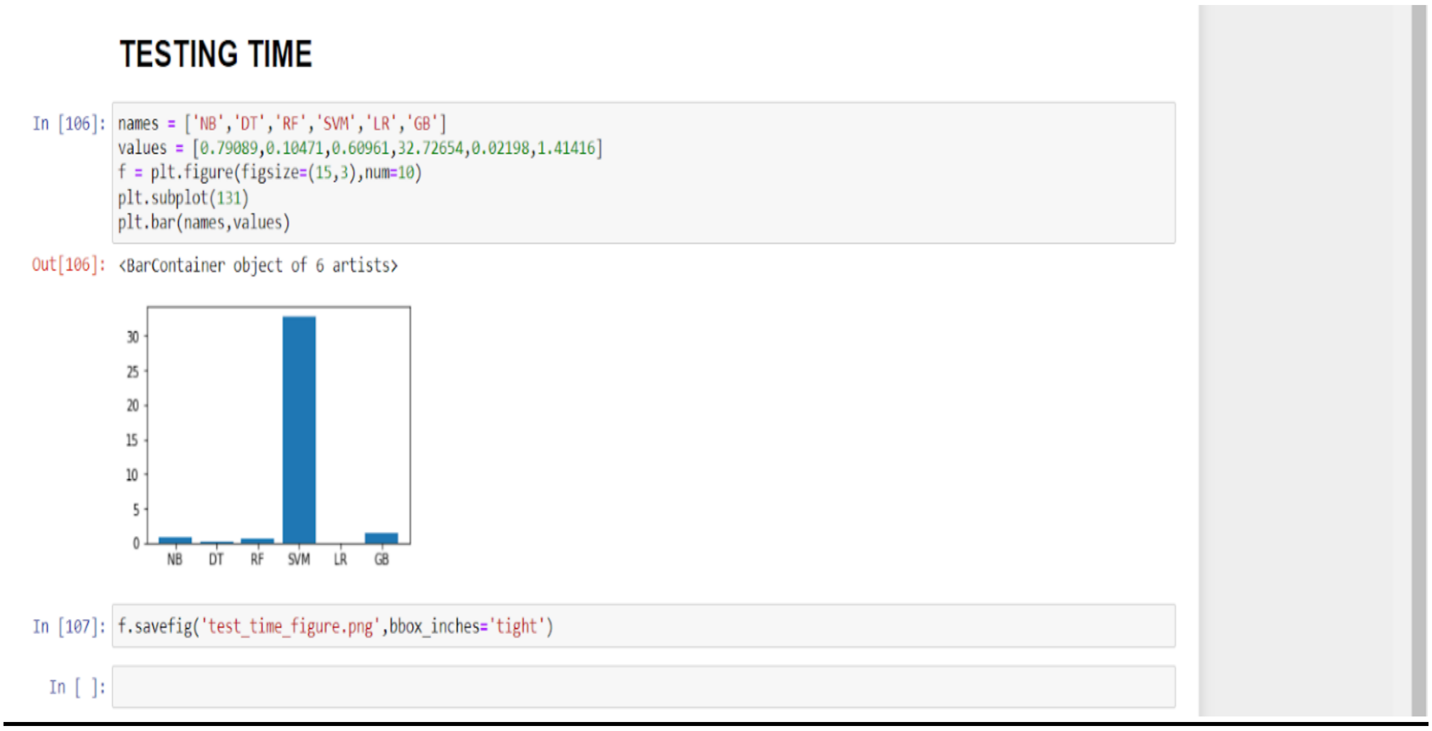
TRAINING TIME

```
In [104]: names = ['NB','DT','RF','SVM','LR','GB']
          values = [1.04721,1.50483,11.45332,126.96016,56.67286,446.69099]
          f = plt.figure(figsize=(15,3),num=10)
          plt.subplot(131)
          plt.bar(names,values)

Out[104]: <BarContainer object of 6 artists>
```



```
In [105]: f.savefig('train_time_figure.png',bbox_inches='tight')
```

(Figure - 9)



TESTING TIME

```
In [106]: names = ['NB','DT','RF','SVM','LR','GB']
          values = [0.79089,0.10471,0.60961,32.72654,0.02198,1.41416]
          f = plt.figure(figsize=(15,3),num=10)
          plt.subplot(131)
          plt.bar(names,values)

Out[106]: <BarContainer object of 6 artists>
```

```
In [107]: f.savefig('test_time_figure.png',bbox_inches='tight')

In [ ]:
```

(Figure- 10)

The above analysis of different models states that the Decision Tree model best fits our data considering both accuracy and time complexity.

CHAPTER 7

REFERENCES

- Kumar, Vipin, Jaideep Srivastava, and Aleksandar Lazarevic, “Managing cyber threats: Issues, approaches, and challenges”. Vol. 5.Springer, 2006.
- Maheshkumar Sabhnani and Gursel Serpen, “Why Machine Learning Algorithms Fail in Misuse Detection on KDD Intrusion Detection Data Set”. ACM Transactions on Intelligent Data Analysis,(pp.403-415) (2004).
- 3.M. Shyu, S. Chen, K. Sarinnapakorn, and L. Chang, “A novel anomaly detection scheme b ased on principal component classifier”.Proceedings of the IEEE Foundations and New Direct ions of Data Mining Workshop, in conjunction with the Third IEEE InternationalConference on Data Mining (ICDM03), pp. 172– 179, 2003.
- MIT Lincoln Labs. (2014, Nov.). DARPA intrusion detection evaluation [Online]. available: http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/index.html
- 6.. J. McHugh, “Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory”. ACM Transactions on Information and System Security, vol. 3, no. 4, pp. 262–294, 2000.
- Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani, “A Detailed analysis of theKDD CUP 99 Data Set”. In the Proc. Of the IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA 2009), pp. 1-6, 2009.
- S. Revathi, Dr. A. Malathi, “A detailed analysis of KDD cup99 Dataset for IDS”. International Journal of Engineering Research & Technology (IJERT) Vol. 2 Issue 12, December – 2013.
- R. P. Lippmann, D. J. Fried, and I. Graf, “Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation”. In Proceedings of the 2000 DARPA Information Survivability Conference and Exposition (DISCEX’00), (2000).
- “Nsl-kdd data set for network-based intrusion detection systems”. Available on: http://nsl.cs.unb.ca/NSL-KDD/, November 2014