

Intrusion Detection System with the help of Machine Learning

by

Shirish Lavania

This thesis has been submitted in partial fulfillment for the
degree of Master of Science in Cybersecurity

in the
Faculty of Engineering and Science
Department of Computer Science

15th August 2020

Declaration of Authorship

This report, Intrusion Detection System with the help of Machine Learning, is submitted in partial fulfillment of the requirements of Master of Science in Cybersecurity at Cork Institute of Technology. I, Shirish Lavania , declare that this Intrusion Detection System with the help of Machine Learning, and the work represents substantially the result of my own work except where explicitly indicated in the text. This report may be freely copied and distributed provided the source is explicitly acknowledged. I confirm that:

- This work was done wholly or mainly while in candidature Master of Science in Cybersecurity at Cork Institute of Technology.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at Cork Institute of Technology or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this project report is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Shirish Lavania

Date: 15th August 2020

CORK INSTITUTE OF TECHNOLOGY

Abstract

Faculty of Engineering and Science
Department of Computer Science

Master of Science

by Shirish Lavania

As the world is emerging with new technologies and the world wide web is growing nowadays, there are more chances to get hacked by networks or there are more chances of the network attacks[1]. The rapid development of computer networks already created many security problems related to the intrusion in the computer network[2]. These network attacks nowadays become a headache for all the bigger security organizations as well as for the national securities and also for the users which are on the internet using the internet. Many surveys revealed that the internet users are not secured nowadays and the intruders theft the privacy of the users by intruding in a network. In today's day to day life, intrusion detection has registered many different cases across the world and is continuously increasing which leads to the privacy theft for any user and has emerged as the biggest platform for attack in the past few years. Network Intrusion can be defined as any unauthorized access on a system or activity in a network. So, in order to get rid of this sort of problem, hence there evolve a technique by which intrusions can be prevented and the technique is to develop an intrusion detection system which detects if there any abnormality in the network traffic with the help of classification[3]. As the threat attacks are at escalation nowadays, the feature filtering based Intrusion Detection System also has some drawbacks due to which it is difficult for this to discover new attacks. Nowadays Anomaly detection is one of the most recognized ways of researching the areas related to the intrusion detection system. Basically it refers to finding out untypical or abnormal behavioral events in the network flow. As it is already pre-assumed that anything which is untypical in the network stream, could be harmful and might be a breach to the security[3]. So, security systems started using a bunch of Algorithms in order to detect these anomalies. However, according to recent research it is proved that the adversarial events might be miss classified by the Machine Learning trained models as there were fake samples generated and applied intentionally to the data set[3]. Due to this, false-negatives might get increased which could be the biggest vulnerability in the Intrusion Detection System[3]. The goal behind this was to check the ability of the Intrusion Detection System to resist this type of Attack.

In this proposal we have taken a NSL-KDD data set which is a benchmark data set, analysis the problems which exists in the era of Intrusion Detection System and NSL-KDD data set, existence of attack vector and proposes Machine Learning approach, which will examine each of the Hyper parameter, optimized Machine Learning algorithms that also reveal which Machine Learning Algorithm will perform better Machine Learning system is mainly developed to classify the network traffic whether it is malicious or benign.

Acknowledgements

First of all, i would love to thanks **Mr.Vincent Ryan**. Sir. Without you, this would only be a dream for me to study in CIT. You are the one who made it possible for me. Really thankful for giving admission in CIT. Also really thanks to the way you always guided me for everything which I have shared with you. My last words for you are: "YOU ARE A BEST TEACHER and Person."

Then, I would like to specially thanks to my Supervisor Dr John Creagh Sir who really worked hard for me and helped me a lot in choosing the right path for this project, right from the beginning till end. Without your efforts, i would not be able to write this project the way i wrote it. Really glad to have such a nice supervisor. Sorry that I always took your extra time in the meetings.

Also, i would love to thanks my second supervisor Dr David Stynes who also taught us Cryptography in our 2nd Semester. To be honest, you are a great teacher, and the best part is you only wants to the point answers.

Contents

Declaration of Authorship	i
Abstract	ii
Acknowledgements	iv
List of Figures	vii
List of Tables	xi
Abbreviations	xii
1 Introduction	1
1.1 Motivation	1
1.2 Contribution	2
1.2.1 Project Goals	2
1.2.2 Research Questions	2
1.2.3 Deliverables	2
1.3 Structure of This Document	2
2 Literature Review	4
2.1 What is an Intrusion	4
2.2 Types of Network Attacks or Intrusions	4
2.3 What are Anomalies?	7
2.4 Introduction to Intrusion Detection/ Prevention System:	8
2.4.1 IDS Detection Methodologies	10
2.4.2 Types of IDS Technology	13
2.4.3 Various Techniques in Anomaly Detection	14
2.5 Introduction to Machine Learning	15
2.6 Methods of Anomaly Detection:	18
2.7 Datasets	18
2.8 State Of The Art	19
3 Design	21
3.1 Problem Definition	21

3.2	Requirements	21
3.2.1	Why do I select NSL-KDD Dataset?	21
3.3	Design	24
3.3.1	Data Pre-processing and Exploratory Data Analysis	24
3.3.2	Correlation Matrix	25
3.3.3	One Hot Encoding	28
3.3.4	Principal Complementary Analysis	28
3.3.5	t-SNE	29
3.4	Machine Learning Algorithms	30
3.4.1	K Nearest Neighbor	30
3.4.2	Naive-Bayes	33
3.4.3	Logistic Regression	35
3.4.4	Decision Trees	38
3.4.5	Random Forest Classifier	41
3.4.6	XG Boost:	42
3.5	Deep Learning: Artificial Neural Networks (ANN)	44
4	Implementation	47
4.1	Main Difficulty: Imbalanced Dataset with five (5) different classes	47
4.2	Confusion Matrix and Scores:	50
4.2.1	Grid Search	51
4.3	Platforms Used for Generating Results	51
4.4	Implementing KNN	52
4.5	Implementing Naive-Bayes	53
4.6	Implementing Logistic Regression	55
4.7	Implementing Support Vector Machines	56
4.8	Implementing Decision Trees	57
4.9	Implementing Random Forest	59
4.10	Implementing XG Boost	60
4.11	Implementing CAT Boost	61
4.12	Implementing Neural Networks	62
5	Testing and Evaluation	64
6	Discussion and Conclusions	69
6.1	Discussion	69
6.2	Conclusion	70
6.3	Future Work	71
	Bibliography	72
A	Code Snippets	76
B	Wireframe Models	84

List of Figures

2.1	<i>Intruder knowledge vs Attack</i>	5
2.2	<i>Scanning Activity</i> [4]	6
2.3	<i>Anomalies in a 2-Dimension</i> [5]	7
2.4	<i>Intrusion Detection System</i> [6]	9
2.5	<i>Signature Based Detection</i> [2]	11
2.6	<i>Anomaly Based Detection</i> [2]	12
2.7	<i>Anomaly Detection Techniques</i> [7]	14
2.8	<i>Classification of Machine Learning Techniques</i> [2]	17
3.1	<i>Sub classes of Attacks in the Data Set</i>	24
3.2	<i>Sub classes of Attacks in the Data Set</i>	25
3.3	<i>Different 5 class of attacks.</i>	25
3.4	<i>Positive and Negative spearman plot between X and Y</i>	26
3.5	<i>Correlation plot between features for NSL-KDD Data set</i>	26
3.6	<i>2 Dimensional correlation plot</i>	27
3.7	<i>3 Dimensional correlation plot</i>	27
3.8	<i>Correlation plot between features</i>	28
3.9	<i>2 Dimensional PCA Plot</i>	29
3.10	<i>3 Dimensional PCA Plot</i>	29
3.11	<i>Correlation plot between features</i>	30

3.12	<i>Correlation plot between features</i>	31
3.13	<i>Correlation plot between features</i>	31
3.14	2 Dimensional PCA Plot	32
3.15	3 Dimensional PCA Plot	32
3.16	<i>Gaussian Curve</i>	35
3.17	<i>Best Separable Line</i>	37
3.18	<i>Sigmoid Curve</i>	37
3.19	<i>Decision Trees</i>	38
3.20	<i>Entropy Graph</i>	39
3.21	<i>Gini Entropy vs Entropy Graph</i>	40
3.22	<i>Random Forest Decision Criteria</i>	41
3.23	<i>XG Boost</i>	43
3.24	<i>Decision Boundary of a neural network</i>	44
3.25	<i>Dropout rate</i>	45
3.26	<i>Flow</i>	46
4.1	<i>Types of Categorical features</i>	48
4.2	<i>Showing Categorical and Numerical features in NSL-KDD Dataset</i>	49
4.3	<i>Category Wise Attack Counts</i>	49
4.4	<i>Google Console Machine Configurations</i>	51
4.5	<i>KNN Parameters</i>	52
4.6	<i>KNN best parameters</i>	52
4.7	<i>KNN Confusion Matrix</i>	53
4.8	<i>Naive-Bayes parameters</i>	54
4.9	<i>Naive-Bayes optimal parameter</i>	54
4.10	<i>Logistic Regression optimal parameters</i>	55

4.11 Logistic Regression Confusion Matrix	55
4.12 SVM parameters	56
4.13 SVM best parameters	56
4.14 Support Vector Machines Confusion Matrix	57
4.15 Decision Trees parameters	57
4.16 Decision Trees best parameters	58
4.17 Decision Trees Confusion Matrix	58
4.18 Random Forest actual parameters	59
4.19 Random Forest best parameters	59
4.20 Random Forest Confusion Matrix	60
4.21 XG Boost parameters	61
4.22 CAT Boost Confusion Matrix	62
4.23 ANN Best Results	63
5.1 <i>Artificial Neural Networks layer Wise F1-Score</i>	66
5.2 <i>Performance of Machine Learning Algorithms</i>	67
5.3 <i>Performance of Deep Learning Algorithms and BAT-MC</i>	68
A.1 <i>Pre-Processing</i>	77
A.2 <i>KNN Classifier</i>	77
A.3 <i>NB Classifier</i>	78
A.4 <i>LR Classifier</i>	78
A.5 <i>SVM Classifier</i>	79
A.6 <i>DT Classifier</i>	79
A.7 <i>RF Classifier</i>	80
A.8 <i>XGB Classifier</i>	80
A.9 <i>CATB Classifier</i>	81

A.10 <i>ANN Classifier</i>	81
A.11 <i>Loading the data</i>	81
A.12 <i>ONe-Hot</i>	82
A.13 <i>Objective Function</i>	82
A.14 <i>Fitting and running the model</i>	83

List of Tables

4.1	Total Number of Classified and Misclassified Attacks	62
5.1	Performance of different ML Algorithms	65
5.2	Precision and Recall	65

Abbreviations

IP	I nternet P rotocol
IDS	I ntrusion D etection S ystem
IPS	I ntrusion P revention S ystem
IDPS	I ntrusion D etection P revention S ystem
NSM	N etwork S ecurity M onitor
NIDES	n ext G eneration I ntrusion D etection S ystem
NIDS	N etwork B ased I ntrusion D etection S ystem
HIDS	H ost B ased I ntrusion D etection S ystem
HTTP	H ypertext T ransfer P rotocol
FTP	F ile T ransfer P rotocol
SSH	S ecure S hell
SYN	S ynchronize
ACK	A cknowledge
DOS	D enial O f S ervice
R2L	R emote T o L ocal
U2R	U ser T o R oot
KDD	K nowledge D iscovery D ata M ining
ML	M achine L earning
KNN	K Nearest neighbour
NB	N aive B ayes
LR	L ogistic R egression
SVM	S upport V ector M achines
DT	D ecision T rees
RF	R andom F orest
XGB	E xtreme G radient B oosting

ANN	A rtificial N eural N etworks
RNN	R ecurrent N eural N etworks
BLSTM	B i D irectional L ong S hort T erm M emory
TN	T ru e N egatives
TP	T ru e P ositives
FN	F alse N egatives
FP	F alse P ositives
MCA	M iss C lassified A ttacks
PCA	P rincipal C omplementary A nalysis

*I would love to dedicate this thesis to my Parents. Specially to my
beloved Dad who has given me such strength and blessings.
Without your blessings this would not be possible.*

Chapter 1

Introduction

1.1 Motivation

Anomalies are one of the significant problems in the expanding world of Cyber security that needs to be detected. Anomaly Detection is the approach of classifying the abnormal data (could be anomalies) which differs from the normal data. Anomaly detection can be applied to Intrusion detection, fraud detection, hospitalization.

Automated detection of anomalies in the network is one of the exciting topics for many years[8]. Additionally, in this thesis, I have used the Machine Learning Approach for the automation of Anomaly Detection and have trained various Machine Learning Models on Intrusion Detection Data set (NSL-KDD), and then applied those classifiers to find out the anomalies, false positives, false negatives. I have chosen this project which tends towards Machine Learning because I wanted to acquire something new and Machine Learning is precisely something which is new as well as unique and which can be implemented on different data sets to find out the anomalies, and also we can make our data sets by capturing the live data, and we can apply algorithms to find anomalies.

At the time of choosing this project, I was a bit worried because everything was new for me as the project is more in implementing Machine Learning Algorithms. With the time, I learned a lot in Machine Learning and now can implement algorithms for detecting anomalies in the field of Cyber Security which is the thing of great interest. Moreover, knowledge of Machine Learning will help in the near future. If I talk in terms of career or cyber jobs, if someone has knowledge of Machine Learning and if he knows how to implement them in order to detect the anomalies, then there would be more chances to get hired in Cyber companies especially in Malware. And we all know in near future Machine Learning, and the Cyber Security are two fields which can go hand in hand.

1.2 Contribution

1.2.1 Project Goals

- Identify false positives and negatives with the actual intrusion attacks.
- detection of the unusual traffic or anomalies in the network and then predict them with the help of Machine Learning Algorithms.

1.2.2 Research Questions

- Which Machine Learning Algorithm works best on Intrusion Detection System data set?
- How to optimally tune the Machine Learning classifiers such as ANN which has many many different hyperparameters?
- How to efficiently visualize the high dimensional data set such as NSL-KD with 42 features?
- How does Intrusion Detection System Work and what are its classifications?
- How Anomaly detection works and its different techniques?

1.2.3 Deliverables

- Intrusion Detection System and its Methodologies.
- Deployment of different Machine Learning algorithms and compare them in order to make sure which algorithm is giving the best accuracy and f1.
- We will deploy F1-Score, Accuracy and Cross entropy loss
- Deliver a supervised ensemble Stacking Classifier with the help of Machine Learning algorithms.

1.3 Structure of This Document

This part briefly describes each chapter in the thesis.

- Chapter 1, which is an Introductory part, gives us an overview of why we would have taken this project along with the project goals and deliverable.

- Chapter 2 is the Literature Review which tells us about the past researches made by different scholars on the topic.
- Chapter 3 is the Design which describes the problem definition that what is the main problem or topic of the thesis. Moreover, the requirements, which can be fulfilled as a prerequisite for the Design phase in order to start the Designing and the Implementation phase. And then in the Design phase, we have discussed the data visualization and an overview of each Machine Learning Algorithm.
- Chapter 4 is the Implementation part. In this part, we have talked about what we have done in our project i.e., different types of Machine Learning Algorithms used to evaluate the results for the thesis.
- Chapter 5 comprises of Testing and Evaluation. This chapter gives the testing part like all the results which have been generated from the algorithms like accuracy, f1 matrices, confusion matrix all lies in this chapter.
- Chapter 6 is the Discussion and Conclusion, where we are going to present what we have concluded so far in this thesis and also we have discussed and compared the results of few research papers with our thesis.

Chapter 2

Literature Review

2.1 What is an Intrusion

As the world is emerging with new technologies and the world wide web is expanding nowadays, this expanding dependence on the Internet becomes the global dilemma as it magnified the chances of illegal access into the systems. Due to these invasions, there are more numerous chances to get hacked by networks, or there are more chances of the network attacks[1].

An intrusion is unlawfully tried access or illegal activity in a network or information system which are induced by the intruders penetrating the systems from the Internet. Intrusion detection is the method of inspecting or monitoring the events or logs which appears in the system or the network and with the help of IDS investigating them whether there is any indication for an attack or incidents which can be intimidation or threats for the computer security policies, acceptable use policies, standard security practices[9]. More often the occurrences lead to the unlawful gaining of the system which can also lead to the additional information loss of the users; events can also lead to the injection of malware or trojans due to which attackers can also create the backdoors to the system. Nowadays, organizations are using Intrusion Detection systems for analyzing intrusions.

2.2 Types of Network Attacks or Intrusions

The conventional way of safeguard the computer systems is to design the secure network architecture for any organization such as keeping firewalls, Virtual Private Networks (VPNs), authentication factors that would allow the user to protect the environment.

Such security mechanisms have vulnerabilities present, i.e. they lack in affording the full security standards to an organization and also to dodge the attacks that are continuously being adapted by the attackers to breach the system security which is done due to the inadequacy of the implementation of the security measures[4]. The primary purpose of implementing network security is to preserve one's digital information by sustaining data confidentiality, dignity and integrity, and also securing the availability of resources. An attack is something which makes the system compromise, and this caused due to the lack of design of a network, users are somehow unaware and misconfiguration of its software or hardware that can be vulnerable to attacks. Attacks then also further classified into main four categories, namely Denial of Service (DoS) Attacks, Probing, R2L and U2R. Here, with the help of an image, we can see the that with the time, the Intruder Technical knowledge is increasing year by year.

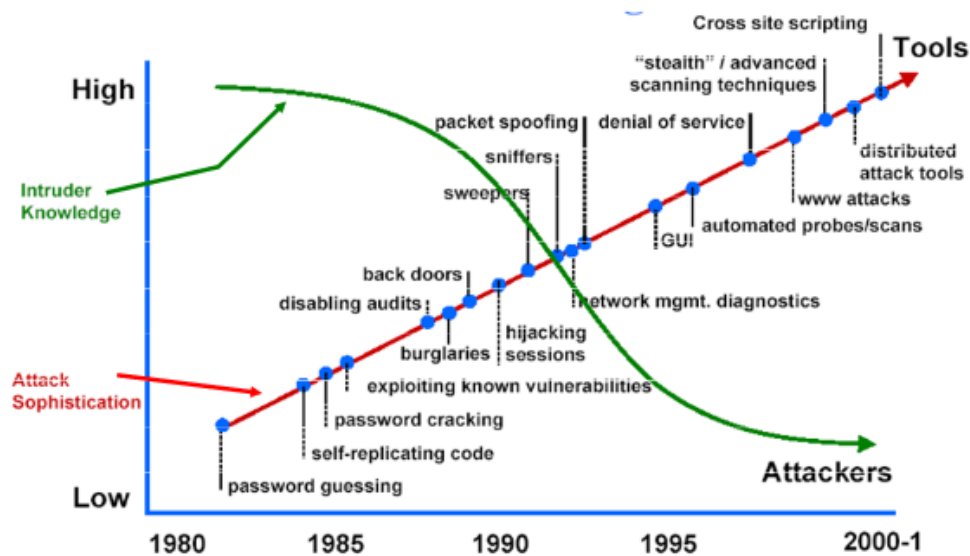


FIGURE 2.1: *Intruder knowledge vs Attack*

[4]

In this Thesis, we also categorized the number of attacks in these four categories, and we have given one category for normal incidents. Here, we are going to study the classification of attacks, so, computer crimes and invasions in history can be classified in accordance with the attack type.

1) **Denial of Service (DoS) Attacks** Denial of Service attacks operates on the SYN Flooding where are n numbers of syn packets are sent by the attackers from a different location to the victim until the victim's system went down which causes the "shut down of a computer process and also deny the use of resources or services to authorized users[4]. There are two types of DOS Attacks; the first one is System attacks, and the second one is network attacks. System attacks target the flaws in the operating systems, and the

intruders target the protocols in the network attack. The most typical example for the network attacks is the SYN packet flooding attack. In Syn flooding attack, the attacker takes advantage of the three-way handshaking process by establishing the half-open connections with the victim. An attacker uses the fake (spoofed) IP address in order to secure the connections. The attacker first sends the syn packet, receives an SYN/ACK from the victim but do not send the last acknowledgement from his side to get the connection complete as the spoofed IP address is not able to respond to the SYN/ACK received from the victim. Due to these half-open connections, the segmentation fault might occur, and the system went down. There are some other examples, such as to hinder the services between the two systems, anticipating any individual from accessing any service etc[4].

2) **Probing:** The attackers primarily practice these sorts of attacks in order to investigate the networks to recognize what services currently activated on the network and what IP addresses, information related to IP address such as the system used, what services they have. This attack is beneficial for the Reconnaissance. As with the help of this attack, the attacker can be able to gather the vulnerabilities and use this gathered information for the attacks[4]. The most common examples are IP sweep, port sweep, nmap. In the port sweep, attackers scan through many ports to determine the service related to each port. In IP sweep, the attackers scan the whole network to determine the service on the specific port. Here we can see the figure, in which the scanning is taking place.

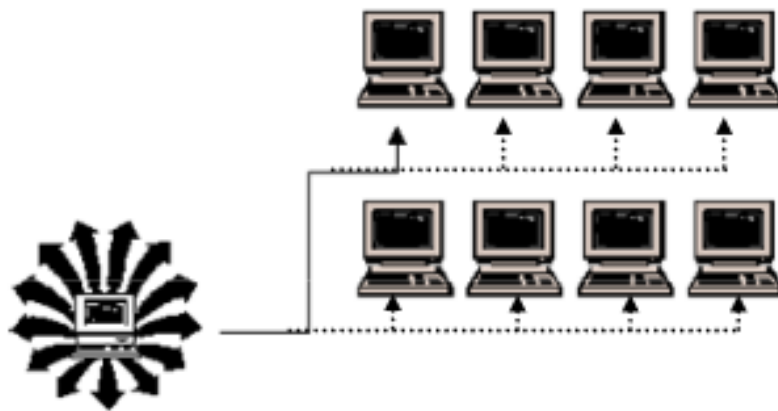


FIGURE 2.2: *Scanning Activity*[4]

There are existing techniques which can be used to detect the IP address that is making a number of connections in the t time frame. However, these tools simply detect the fast scans but are incapable of identifying the slow scans such as nmap, a stealthy scan which is quite slow. Stealthy scans are the scans that would typically do not trigger standard

scan alert technology. So, attackers nowadays reduce the frequencies and adjust the scans in order to, avoid detection.

3) **R2L (Remote to Local) Attacks:** Remote to Local attacks are the attacks in which the attacker can send the packets to the system over the network and then gained access to that system as an administrator or the typical user. Basically, the source of these attacks is only the Internet. The most common examples are the Brute force attacks where an attacker can do password guessing, and by breaching the software, they also gain the user's system access[4].

4) **U2R (User to Root) Attacks:** In User to Root Attacks, the attacker has an account on the system and can easily misuse the privileges by breaching the vulnerability in the application which already installed in the system. In R2L attacks, we know that the attacker is present outside of the network and send packets from the outer. However, in the User to Root attacks, the attacker is present inside the network and can quickly gain the root privileges. The most common example of the U2R attacks is the buffer overflow in which the attacker exploits the error in the program and then load the excess data into a buffer that is located on the stack.

2.3 What are Anomalies?

Anomalies are the patterns present in data that are unnatural in nature means anonymous. Here, the figure below, represents the anomalies present in the 2-D data set. Here, we can see the points O_1 , O_2 and O_3 are anomalies and the points N_1 and N_2 are the normal data. Attackers include the anomalies in the network data for the malicious activities such as cyber crimes, credit card frauds, breakdown of servers[5].

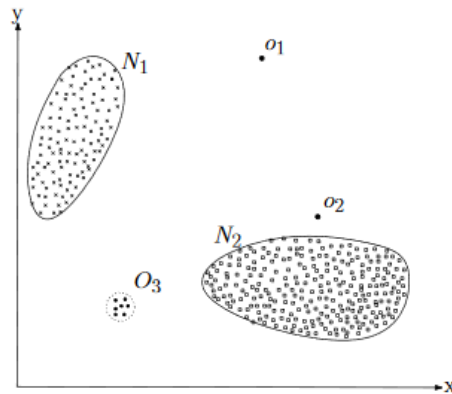


FIGURE 2.3: *Anomalies in a 2-Dimension*[5]

Anomaly detection is related to the noise accommodation which are unwanted in data. So, in order to make the data clean, the noise removal formula we have to apply. Noise removal means to remove the unwanted noise or it simply means that unwanted data from the data set. Noise accommodation refers to immunizing a statistical model estimation against anomalous observations[5]

2.4 Introduction to Intrusion Detection/ Prevention System:

An IDS is a physical device or a software application which is used to detect the packets that are to and fro the network, or it is a device designed to monitor traffic in the network for the malicious activities within the network[10]. IPS is a system having all abilities and skills similar to IDS as well as which is proficient of defending the network by endeavouring to hinder the incidents or events. IDPS predominantly centred on recognizing the potential incidents that occur, log the information related to those incidents, then it attempts to stop the incident and then report the incidents to the administrators. Many IDPS can also respond to the incidents which occurred and stop them from further succeeding. IDPS also do some critical purposes for organizations like identifying and classifying the predicaments with security policies, also documenting the intimidations or threats which already existed.[9] Organizations also used IDPS for identifying the difficulties with security policies, and also they document the threats which exist in today's scenario and hindering people from violating security policies. IDPS has become an indispensable extension for the organizations concerning security infrastructure.

So, here, we will find the study which has already been done in the past about the overview of IDS and IPS along with the uses and the detection methods which were opted by the IDS and IPS[9]. IDPS has various response techniques by which they can be able to stop the intrusions or attacks, also be able to change the security of the system and also by changing the attack content.

The role of IDS is to monitor the traffic in promiscuous mode in a particular network, which is more likely act as a network sniffer. The packets in the network is collected and then analyzed by the means of an algorithm called as 'pattern recognition' which checks whether is there any violation of rules occur. If yes, then an alarm will be generated by an IDS which gives an alert for the intrusion to the administrator. Security of a network or host can be evaluated by IDS by deploying vulnerability evaluation process. Most IDS Systems works on a speculation that there must be a difference in the normal and intrusion activities in a network, so they can be detected.[11]

So, one of the generalized models for intrusion detection system was proposed by Axelsson [6] In figure 2.1 shown below, indicates a system where the solid arrows tell us about the data flow in the IDS and the dotted arrows tells us about the response to the intrusion.

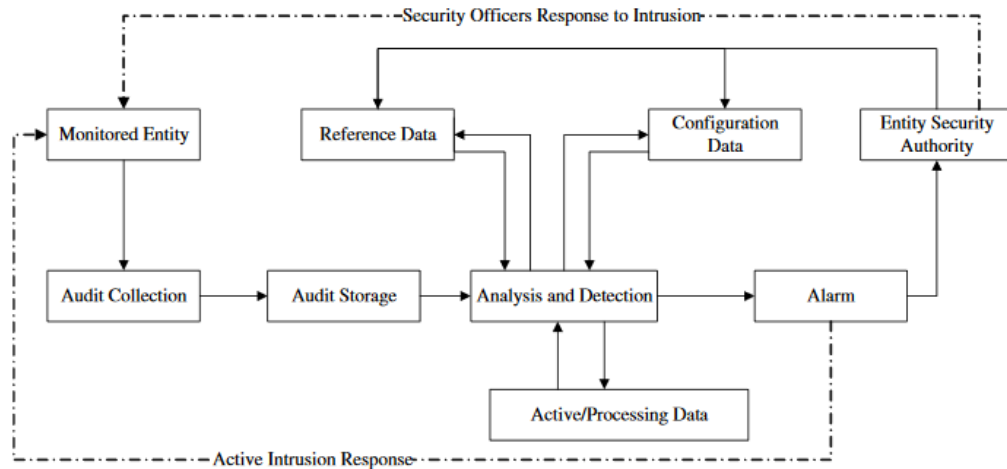


FIGURE 2.4: *Intrusion Detection System*[6]

- Audit Collection:-** Audit collection is a phase in which data is collected and the data collected under this phase undergone through an algorithm which is helpful in making the intrusion detection decisions in finding the traces of the suspicious network activities.[6] Network activity logs, command based logs, application logs etc. are the sources of the data.
- Audit Data Storage:-** Audit data must be store somewhere for the later referencing or for the temporary period. Data which is stored has the large volume. Thus, the critical issue in the field of IDS is an audit data reduction.[12]
- Analysis and Detection:-** Analysis and detection is the main part of any Intrusion Detection System, as various algorithms can be applied to data for finding any suspicious activity inside the audit data. This block also is known to be the core of IDS. For analysis and detection, algorithms can be categorized into 3 sections, namely: anomaly detection, signature-based detection, and hybrid detection.
- Active and Processing Data:-** Information about the intrusion signatures can be stored as a form of results by the processing unit.
- Alarm:-** Alarms handle all the output that is generated by the IDS. If there is any unusual activity in a network then alert is generated by the alarm to the security officer or maybe an automated response to the intrusion is generated.[12]

- **Reference Data:-** This module stores the information about the different types of known intrusion signatures which is helpful in the signature-based detection and profiles of regular behaviour in case of anomaly detection. If there is any new signature detected, then the module must be updated by the Security Officer.
- **Configuration Data:-** The configuration part is likely to be the most delicate part of an IDS as it consists of information related to the working of an IDS. The related information is when to collect the audit logs and how to respond to the alarms and when to generate the alerts etc. If this part of the IDS is breached then attacker can easily detect the type of attacks which are likely to go undetected by the IDS[6].

Although the IDS detects network traffic but also tends to give fake alerts. In order to get rid of these fake alerts, the companies need to do the complete setup for these software applications in order to make the device clear which is an intrusion and which is a false alarm and what looks like a normal traffic in the network. IDS mainly classify a data set whether the data is normal or abnormal. Normal data set means there are no intrusions or the balanced data set having no anomalies, the packets are transferring in the normal manner as they transferred before. Abnormal data set is having anomalies or imbalanced data set. The key point of IDS is basically the construction of the detector, which can also be known as the Analysis and Detection (or the data model) which includes the part of Machine Learning. The IDS which has been proposed earlier can easily recognize the known and unknown attacks and it is composed of Clustering Manager, Decision Maker and Update manager[10]. The data sets which IDS deals with are full of noises and errors, so in order to make it non-redundant we need to use the feature selection based on Machine Learning technique which make the data set free of noises. There are number of Intrusion Detection Systems that are in use by some organizations nowadays are Snort, Suricata, Security Onion and AIDE etc. The other Intrusion Detection Systems which are already surveyed such as Haystack System, IDes-real-time intrusion-detection expert system, NSM- Network Security Monitor, NIDES-next-generation IDS, BRO etc.

2.4.1 IDS Detection Methodologies

IDS technology uses multiple techniques to find out the intrusion from the network. Most IDS or ID-PS uses the Signature-based intrusion detection, anomaly-based intrusion detection and stateful protocol-based intrusion detection to find out the intrusions or incidents in the network. Most commonly and widely used Methodology is Anomaly-based. There are multiple detection methodologies which can be used separately or

simultaneously by the most IDS or IDPS, which helps them to provide further precise detection.[13]

1) Signature Based Detection

Signature is basically a pattern which co-relate with a known threat. Intrusion Detection Systems have their database stored with full of signatures which are already known as all those patterns used earlier in the attack techniques. So, signature-based detection is the methodology to detect possible incidents by relating their known signatures against the observed events. The significant advantage of using the signature-based methodology is that it is quite efficient while identifying the known threats but at the same point it lacks when it comes to unknown threats that already exist, also to the threats which are disguised by the attackers with the help of evasion techniques. One of the examples for the alteration is if suppose an attacker changes the filename which was freespace.exe of the previously injected malware to freespace1.exe, so the signature for both the files would be mismatched, then signature looking for "freespace.exe" would not match it[9].

Signature-based detection is the easiest way of identifying the intrusions because it only deals with comparing the current activities inside the network for example, from the list of signatures they compare an entry of a packet or any log entry with the help of string comparison operations. The drawback of signature based detection is that it can not be able to understand the state of complex communications as they have very low level of understanding for the network protocols[9]. Due to lack of these, the signature-based detection method aren't suitable for identifying attacks that include various events.

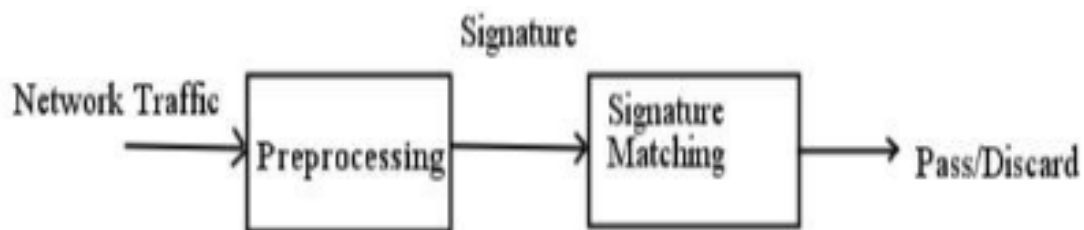


FIGURE 2.5: *Signature Based Detection* [2]

Also, Signature-based IDS always depends upon the human beings to create and impose the signatures and it is easy to deploy as there is no need to learn any sort of environment[2] and it might take a long to create a signature for an attack, which might be very long in the rapid and random attacks, so anomaly-based detection system is the solution for this human dependent signature based detection system

2) Anomaly Based Detection

Anomaly-based detection is the method of comparing the definitions considering which activity is normal or regular against the observed events in the network to identify the abnormal behaviour if present. There are profiles in IDS that represents the activities or normal behaviour of such things as users, hosts, network connections, or applications[9]. So for this, the activity inside the network can be viewed for the long period. After than, the Intrusion Detection Systems compares the currently viewed activity with the profiles. Profiles can be developed for different behaviour, for example a user visited number of web pages, how many number of failed logins for a host etc.

Anomaly detection basically focused on finding patterns in data having an abnormal behavior. The importance of anomaly detection is due to the fact that anomalies in data translate to significant actionable information in wide variety of application domains[11]. The most advantage of the anomaly based detection is that for previously unknown attacks they are quite efficient in detecting them. For example, suppose a new type of malware or virus infects the system. By doing this, malware can easily take over to the system's resources such as kernels, system calls, processing resources and can easily communicate with the different network connections i.e. can easily send emails and can easily communicate with different services on the network and can can conduct other activities that would vary substantially from the profiles that are established for the computer [9]

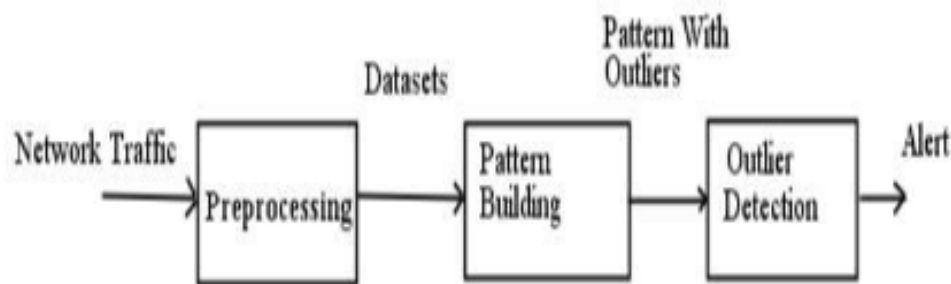


FIGURE 2.6: *Anomaly Based Detection*[2]

In Anomaly-based Detection, an initial profile is made by viewing over a long period of time. This profile can be static or dynamic. Static profile remain unchanged and this can be changed if an IDS or IDPS directed to make a new profile. Dynamic profiles can be changed constantly as there are some additional events are observed. Here the static profile which remains unchanged become non useful or inaccurate as the network and system keeps changing time to time. So, the static profile needs to be generated timely. Dynamic profile suspects the evasion techniques used by the attackers.

One of the major problem with the profile is the accuracy. To make them accurate is a very challenging issue. Anomaly-based Intrusion Detection Systems typically generates different false alarms due to benign behaviour that greatly deviates from the profiles[9].

Anomaly-Based detection techniques sometimes generate results with no cause. So, it is really difficult to define the alarm generation which is one of the major problem for any industry. Another main problem is then to validate that alert whether the alert is accurate or it is a false positive. The alert might be generated because of the complexity and number of events caused[9].

2.4.2 Types of IDS Technology

Intrusion Detection System can be classified into 2 categories:

1. **Network-Based Intrusion Detection System:** NIDS mainly uses either statistical estimation or measured limits on feature sets such as packet length, inter-arrival time, flow size and other parameters of network traffic to model them within a fixed time span accurately. The main drawback of Network-Based IDS is they undergo a very high rate of false negatives and positives. The high rate of false negatives shows the inability of NIDS to detect the attacks more frequently[14] and the high positive rate means there is no attack but still NIDS generating unnecessary alarms for the attacks. So, we can say that these types of Intrusion detection system are not capable of detecting the attacks more frequently.

So, to overcome this problem, they gave a new approach which is a self learning system. This is the system which uses the Machine Learning Algorithms both supervised and unsupervised approaches and is one of the effective approach to deal with the day to day attacks. Machine Learning helps in pattern recognition for both the malicious and normal activities[14].

2. **Host-Based Network Detection System:** To examine and gather the information from the target, or to find out the vulnerabilities in the target system, there are lots of different typch as Metasploit, Nmap etc[14]. So, with the help of this information gathering, an attacker can attack with the help of HTTP server, FTP servees of software tools and frameworks are present sur, ssh server etc. The new technologies such as the cryptography solutions, firewalls, the authentication process can help the hosts against these application attacks, but still, the attacker can quickly gain unauthorized access to the host system with the help of advanced malicious attacks. To control these attacks on system-level, a Host-based Intrusion Detection System functions at host-level. This Host-based IDS monitors all the

traffic activities on the system files and on the system[14]. System call interacts between the kernel functions and the low-level system applications. So, a trace is generated with the help of length and behaviour of an application which is unique for each as the application communicates with an operating system with the help of system calls. With the help of these unique traces, it is distinguished the known and unknown applications[14]. The system calls of an intrusive application is quite different from the normal one.

The three main components of Host-based IDS are data source, the sensor, and the decision engine that plays a significant role in the detection of the intrusions. The change in the source of data can be monitored by the component called a sensor and also the machine learning-based approach is implemented to the decision engine for the implementation of IDS[14].

2.4.3 Various Techniques in Anomaly Detection

Anomaly detection is the process of finding out the anomalies that are present in the network. Anomaly Detection mainly consists two phases: one is training phase and second is the testing phase. The various techniques that are used previously are given below in the figure. In this thesis, we only use Machine Learning for anomaly detection[10].

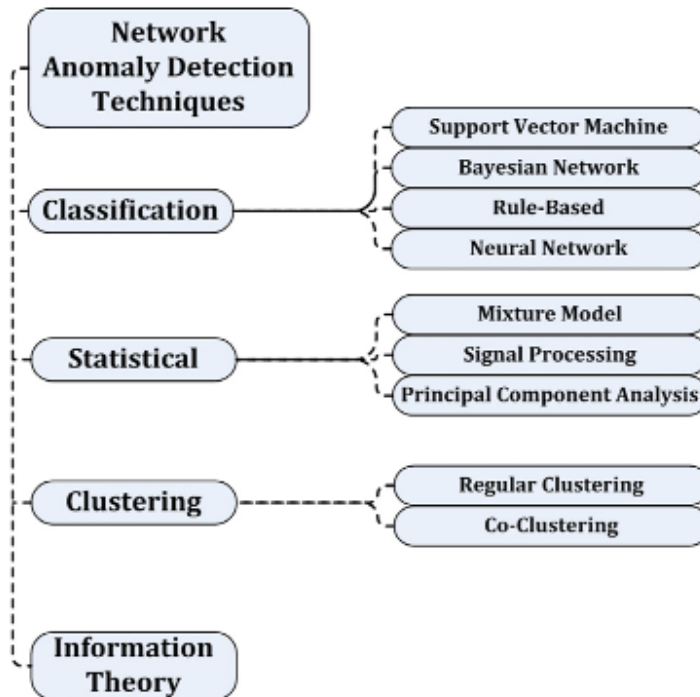


FIGURE 2.7: *Anomaly Detection Techniques*[7]

1) Statistical Approach: Here, the system creates the profiles after examining the subject's activity and produces the result to interpret the behaviour. The profile tells us about the categorical measures and measures such as CPU usage which comes inside ordinal measure[12]. So what system does, it generates two profiles for the objects, the stored and the current profile. So, IDS updates the current profile when the network activity increases like incoming packets and then calculates the anomalies. IDS calculate the anomalies on behalf of comparing the profiles like it compares the stored and the current profile[12].

2) Machine Learning-Based Approach: The role of IDS is to enhance the security by recognition of Malicious and Suspicious events in the network traffic. In order to reduce the fake alarms and improve the detection rate, the accuracy of IDS must improve, and the performance can be achieved by using different techniques of Machine Learning[12]

Machine learning has used for detection of intrusions or anomalies in the intrusion detection system from decades and now because of new algorithms and deep learning, there is a need for survey and a through taxonomy. Many scholar and researchers did study on the largely available benchmark datasets such as KDDcup99, Darpa and NSL-KDD in order to improve the accuracy of IDS. We are also studying on NSL-KDD dataset and will impose the Machine Learning Algorithms to achieve better accuracy and F1-Score in order to decrease the number of false-positive ratio in Intrusion Detection System.

2.5 Introduction to Machine Learning

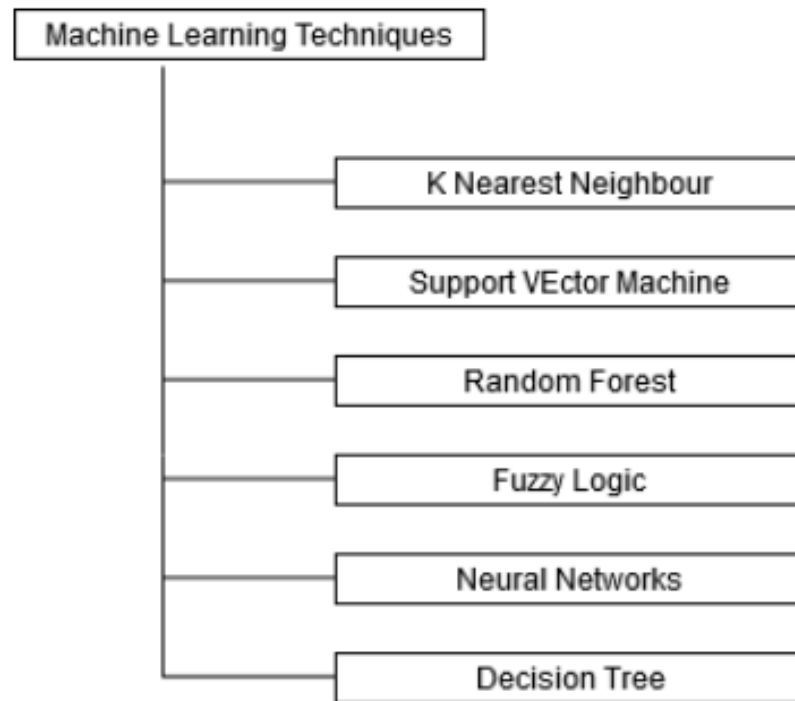
Machine learning is the sequence of algorithms which is specially used for pattern recognition. At the heart of computer programming, there is a concept of an Algorithm lies[15]. Basically, it is an array of instructions which help to alter the input to an output. There are many tasks humans can do without any use of Algorithms, but for some of them, the algorithms are required. In our day to day life, we still do many things like driving a car, recognizing people from the pictures etc[16]. In machine learning, the approach is to learn these sorts of things with the help of Algorithms. For this, we need to deploy a model, and then we need to pass some parameters, and with the help of these parameters, the model can perform all types of tasks. Now, for the best results or to match accurately, we need to manipulate the parameters. Now, after training of a model, the particular instantiation of the general template, is the algorithm for the task. Let's take the example of a supermarket which is selling thousands of products to millions of people. The details related to each sale are stored, which includes the amount, the date, the customer id, goods bought and the total transaction[16]. This

means they have got a lot of data on a daily basis. They want to determine the mood of the customers by analyzing the data in order to predict what customers want to buy in order to maximize their profits by getting sales. On the other hand, each customer also wants the best product by viewing the variety of products on different range basis. So for this, we can't say exactly what sort of people wish to buy this particular thing. So, for this, we can say there are some patterns in the data such as people wanting soup in the winters and the cold coffee in the summer. This is what we can say the pattern in the data. So, on behalf of this particular data, we can say that we can construct a good model which can detect a certain amount of patterns, and this is the Niche of Machine Learning[16]. We can make predictions with the help of Machine Learning on the data sets. Machine learning nowadays has become much more popular and is dealing with many fields nowadays in order to detect anomalies. Basically, it is applied in the banking sectors, fraud detection, stock market, network detection etc[16]. Machine learning is the main part of Artificial Intelligence, as the model has the ability to learn new environments. Basically, it helps in the optimizing the solution with the help of Machine Learning approach.

As we all know that there are so many Machine Learning approaches such as supervised, semi-supervised and unsupervised. But, out of them, we are going to use the Supervised Machine Learning Approach for this project, and the Algorithms which we are going to implement are:

- 1) K Nearest Neighbor**
- 2) Naive Bayes**
- 3) Logistic Regression**
- 4) Support Vector Machines**
- 5) Decision Trees**
- 6) Random Forest Ensemble Classifier**
- 7) XG Boost**
- 8) CAt Boost**
- 9) Artificial Neural Networks**

In the recent past years, many people research upon the datasets with the help of Machine Learning Algorithms. So, in order to implement anomaly-based IDS, a model can be trained with the help of Machine Learning algorithms that can easily learn the data and can provide the decision for that particular data[2]. Figure 4 represents the frequently used Machine Learning Techniques by which we can distinguish normal and abnormal data in a network.

**Figure:4**FIGURE 2.8: *Classification of Machine Learning Techniques*[\[2\]](#)

In order to defend the network systems from malicious activities, anomaly-based Intrusion System is a worth technology. There are lots of methods for the anomaly-based intrusion detection that are already discussed and described in the recent past years, security is just beginning, and there are problems that are still to be solved[\[17\]](#). There has been the usage of many Machine Learning algorithms by which anomalies can be detected such as Linear Regression, Support Vector Machines, K-Nearest Neighbor algorithm, Naive Bayes classifier, Decision Tree, Random Forest Classifier etc. But among them, Support Vector Machines is the most used learning and implemented algorithm on the anomaly detection[\[17\]](#). One of the issues with the Anomaly-based detection is the false alarm rate which is high, though these techniques detect the attacks but still lack false alarms. The most challenging issue is the insufficient or small data sets which lead to the down inaccuracy of the performance of these IDS[\[17\]](#). Most research on anomaly-based techniques we have seen in the recent years was evaluated using KDD data set.[\[17\]](#)

2.6 Methods of Anomaly Detection:

Anomalies can be detected with the help of Machine Learning Algorithms, which is used by past researchers as the detection method. The first approach is based on statistics where they used it to figure out a distribution of attributes and then apply a statistical interpretation based test to know whether the instance which has been observed is an anomaly or not. In the last many years, Machine Learning has been used to sort out the various sorts of problems like for pattern recognition. Machine learning can be classified into three methods which are based on the training set are[3]:

- **Supervised Learning:**In the supervised mode, the task is to match a new observation with exactly 1 class. In this, the training data sets contain the labels due to which prediction can be done easily[3]. Supervised mode usually learns and improve performance after every iteration.
- **Unsupervised Learning:**In this unsupervised mode, the datasets do not contain any labels and useful information, or we can say they are unlabelled[3]. The algorithms basically assign the observations to the group of a dataset and then determine the similarity level.

Basically, the algorithms that are based on supervised and unsupervised machine learning are usually used or have been applied in the anomalies detection in the network as there is an assumption set that the model which has been trained to detect the anomalies can detect the attack vectors as they are very uncommon that not likely to occur on a daily basis so that can be classified by the model easily[3].

2.7 Datasets

There are some benchmark data sets which were mainly used by the researchers, and that can also be used to evaluate the methods of anomaly detection methods and their systems.

1) Synthetic Datasets: These are the datasets which have generated and then used in the past to meet out the needs that the real data satisfy. These datasets can be useful at the time of theoretical analysis so that design can be refined[11]. Basically, this dataset is used in the testing, which is helpful for creating different types of test scenarios. Basically, it sets a platform for the users by building realistic profiles based on the evolved dataset to test the proposed system.

2) Benchmark Datasets: Benchmark Datasets we are going to present the four different types of datasets which are available publically which are generated using the stimulating environments that include the number of attack scenarios to different users.

a) KDDcup99 Dataset: This is the most popular dataset used for the evaluation of network-based anomaly detection methods which has been prepared by Stolfo et al. [18]. This dataset is mainly used for studying and competition purpose. The dataset has been prepared on the data captured in the DARPA98. The KDD dataset (Training part) consists of 4,900,000 single connection vectors, and each contains 41 features and is labelled as either normal or attack with defined attack type [19]. The KDD dataset (Testing part) contains about 300,000 samples. This dataset has been criticized by many scholars as they found the irregularities in the dataset such as the attack packet have a TTL of 126 and the normal packet have a TTL of 127 [3]. As KDDcup99 is the reproduction of traffic present in a network, there are very much duplicacy in the training and the test set. So, in order to solve this problem, a new dataset NSL-KDD was published.

b) NSL-KDD Dataset: Due to the drawbacks in the KDDcup99 dataset which has been revealed by the NSL-KDD dataset is the refined version for the KDDcup 99 Dataset. In order to develop an Intrusion Detection System, researchers in the past carried away many sorts of analysis with the help of different techniques. NSL-KDD is one of the first research data sets in the field of anomaly detection that was used for the Knowledge Discovery and the Data Mining tools Competition. It was created by Lincoln Lab in Massachusetts Institute of technology called DARPA 99 which one year later changed to KDDcup99 and then in 2007 changed it to NSL-KDD. This data set has 41 features and almost 494.021 instances [20]. There are total 40 types of unique categories which are later categorized in 4 attack groups, namely: 1) DOS, 2) Probe, 3) R2L and 4) U2R. We have found this data set from [21] and the data set is divided into the train and test data set. In train part, it contains 125973 rows and in test part it contains 22544 rows.

2.8 State Of The Art

The Deep Learning approach can be a perfect future scope as compared to the Machine Learning Algorithms as these models got a very low accuracy and they are got easily biased as compared to the Deep Learning Models.

In [22] they proposed the traffic anomaly detection model BAT, which is the combination of Bi-directional Long-Short term memory (BLSTM) and the Attention mechanism in order to solve the problems related to the low accuracy and the F1 Score for multi-class classification in Intrusion Detection. In [22], to capture the features of the traffic in the

network, they have adopted multiple convolutional layers as they are used to process data samples, so they refer the BAT model as BAT-MC Model. They have crossed the accuracy which has obtained so far with any of the models, and they are now State of the Art. **The accuracy they achieved on Multi-class classification with BAT-MC model is 84.25% on KDD test.**

In [14], they worked on various different benchmark data sets such as KDDcup99, NSL-KDD, UNSW-NB15, WSN-DS and CICIDS-2017 and performed various Machine Learning approach to the data sets. They have also developed a model with Deep Neural Networks based Intrusion Detection System to discover and classify abrupt and unpredictable cyberattack. They have worked for both binary classifications as well as for Multi-class classification problem while collecting the real-time network and host-based features and then deployed them to their DNN model to detect the intrusions in the network. Their DNN model learns the abstract and the model represents the high-dimensional feature of the network traffic captured by an IDS by passing the data into the hidden layers[14]. **They have got the best F1 Score on the 5th layer of Artificial Neural Networks which is 76.5%**

In [23], they have worked with Recurrent Neural Networks (RNN) and proposed a deep learning approach using Recurrent Neural Networks for the detection of Intrusions in the Network. They have studied the performance of the model for Multi-class classification and the binary classification as well. They have compared the RNN model with the Machine Learning approaches used in the past, with the Deep Neural Network and also with J48. They have researched that the accuracy is totally based on the number of neurons and different learning rate as it impacted the accuracy much.[23] **They have got the 81.29% of accuracy on KDD test data set.**

Chapter 3

Design

Design Section consists of Data Pre-processing part for this thesis and also summarize some of the Machine Learning Algorithms used for this project.

3.1 Problem Definition

Anomaly detection has been researched since quite long within the diverse application domains as it is an enormous problem and which has to be researched comprehensively and in a structured manner. Anomaly detection has done to find out the patterns in the data which are different from the expected behaviour. These different or non-conform patters are called as anomalies, exceptions, unknown behaviour, outliers etc. in different application domains[5]

The two most common terms used in the context of anomaly detection are anomalies and the outliers. The most important use of anomaly detection is in the intrusion detection in cybersecurity, fraud detection with respect to credit cards and also in the military surveillance system to find out the enemy activities.[5]

3.2 Requirements

3.2.1 Why do I select NSL-KDD Dataset?

NSL-KDD data set is the advanced or upgraded version of the KDDcup99 data set. There were many drawbacks of KDDcup data set, so researchers generated the NSL-KDD. This data set has the main feature or records of the KDDcupo data set.[24]. The main reasons due to which i selected this data set over KDDcup99 are:

- 1) Redundant records are removed from the train set so that classifiers might not get biased.
- 2) No duplicate results present in the test sets due to which the classifiers are not biased.
- 3) The records which have selected from the difficulty groups are inversely proportional to the percentage of records present in the KDD original data set[25].
- 4) Here, train and test data sets contains an adequate amount of records (rows and columns) which allowed us to execute the results on the data set.

The NSL-KDD data set consists of 41 features and 5 attack classes. Here, we can see that there are 41 attributes and from the below table we can find the description of each of the attributes.

TABLE 3.1: Attributes with their descriptions[8]

Begin of Table	
Features	Description
Duration	Length of the time duration off the connection.
Protocol type	This indicates the protocol used.
Service	Network Service which is used such as Https, Ftp etc.
Flag	Status of the connection whether it is Normal or Error
Src.Bytes	Number of data bytes transferred from source to dest
Dest.Bytes	Number of data bytes transferred from dest to source
Land	If src and dest ip are equal then takes value 1 else 0.
Wrong_frag	Number of wrong fragments used in a connection.
Urgent	Number of urgent bits activated packets ion the connection.
Hot	Hot indicates the valuable thing in the content.
Num_failed login	Indicates the failed login attempts.
Logged in	Tells logging status as 1 or 0. for successful login 1 else 0.
Num_compromised	Number of compromised conditions.
Root.Shell	If root shell mis obtained then 1 else 0.
SU_Attempted	If su attempted then 1 else 0.
Num_Root	These are number of operations performed as a root.
Num file relations	Numbert of files created or operation takes place
Num_shells	Number of shell prompted.
Num access files	Number of operations on access control file.
Continuation of Table 3.1	
Feature	Description

Num_Outbound_cmds	Number of cmds in an ftp session.
Is_Hot_Login	1 if login as a root else 0.
Is_Guest_Login	1 if login as a guest else 0.
Count	Number of connections to the same dest in past 2 seconds.
Srv_Count	Number of connections to the same dest in past 2 seconds.
Error_rate	Percentage of connections activated flags among the connections aggregated in count
Srv_Error_rate	Percentage of connections activated flags among the connections aggregated in srv_count.
Error_rate	Percentage of connections activated flag REJ among the connections aggregated in count
Srv_Error_rate	Percentage of connections activated flag REJ among the connections aggregated in Srv_count
Same_srv_rate	Percentage of connections which belongs to the same service.
Diff_srv_rate	Percentage of connections which belongs to the diff service.
Srv_diff_host_rate	% of connections were to different destination machines among the connections in srv count.
Dst_host_count	Number of connections have the same destination IP address
Dst_host_srv_count	Number of connections having same port number.
Dst_host_same_srv_rate	% of connections were to same service among the connections in dst_host_count
Dst_host_diff_srv_rate	% of connections were to diff service among the connections in dst_host_count
Dst_host_same_src_port_rate	% of connections were to same port among the connections aggregated in dst_host_srv_count
Dst_host_srv_diff_host_rate	Percentage of connections were to diff dest among the connections aggregated in dst_host_srv_count
Dst_host_error_rate	% of connections that activated the flags among the connections. aggregated in dst_host_count
Dst_host_srv_error_rate	% of connections that activated the flags among the connections aggregated in dst_host_srv_count
Dst_host_error_rate	% of connections activated the flag REJ among the connections aggregated in dst_host_count
Dst_host_srv_error_rate	% of connections activated the flag REJ among

Out of these 42 features there are 9 categorical features and rest of the features are numerical ones.

So, here we can see that the 41 features tell us about the different types of activities and also the label assigned to them represents whether its an attack or a normal activity. We just see the attributes described in the above table. Now, the 42nd attributes contains the different forms of attack classes. There are total 4 attack classes which

are DOS, U2R, R2L and Probe and one of the class is Normal, which represents the normal activity. We already have discussed all the attack classes in the previous Chapter "Literature review." Here, in the below figure, we find out the different attacks mapped to their attack classes.

3.3 Design

3.3.1 Data Pre-processing and Exploratory Data Analysis

Data Pre-processing is a technique by which we can transform the raw data into an understanding version. We more likely take that the data in the real world is not consistent, sometimes data sets contain null values in rows, or it might be having some errors, and also the raw data is sometimes not complete. So, we can not implement the data into any classifier without doing any pre-processing part. So, we can say that with the help of Data Pre-processing, we can quickly solve the problems related to the data sets. Data processing method includes:

- 1) Data Normalization
- 2) Remove the null values and fill in the missing values if any.
- 3) Changing the categorical features to the numerical features with the help of Label encoding as classifier only works on the numerical values, which is the part of pre-processing.
- 4) Pre-processing reduced the volume of the data but also produce effective results

Here, for the pre-processing, we first split our data set into two main sets, splitting it with terms of rows. First set we call the training set, and the second set is testing set. The training set consists of 125973 rows and 43 columns whereas testing data consists of 22544 rows and 43 columns.

Then, we checked for the null values, whether there are any null values present or not. Furthermore, we have a total of 40 subclasses in the data set. The 40 subclasses have shown in the snippet below:

```
[ 'normal' 'neptune' 'warezclient' 'ipsweep' 'portsweep' 'teardrop' 'nmap'
  'satan' 'smurf' 'pod' 'back' 'guess_passwd' 'ftp_write' 'multihop'
  'rootkit' 'buffer_overflow' 'imap' 'warezmaster' 'phf' 'land'
  'loadmodule' 'spy' 'perl' 'saint' 'mscan' 'apache2' 'snmpgetattack'
  'processtable' 'httptunnel' 'ps' 'snmpguess' 'mailbomb' 'named'
  'sendmail' 'xterm' 'worm' 'xlock' 'xsnoop' 'sqlattack' 'udpstorm']
```

FIGURE 3.1: *Sub classes of Attacks in the Data Set*

then the Spearman correlation coefficient is positive, and if the Y tends to decrease with the increase of X-axis then the Spearman coefficient is negative. Also, when X and Y remain unchanged or becomes monotonically related, then the spearman correlation becomes 1. The + indicates the positive relationship that the features are strongly correlated with each other and the value -a represents the negative correlation between the variables.

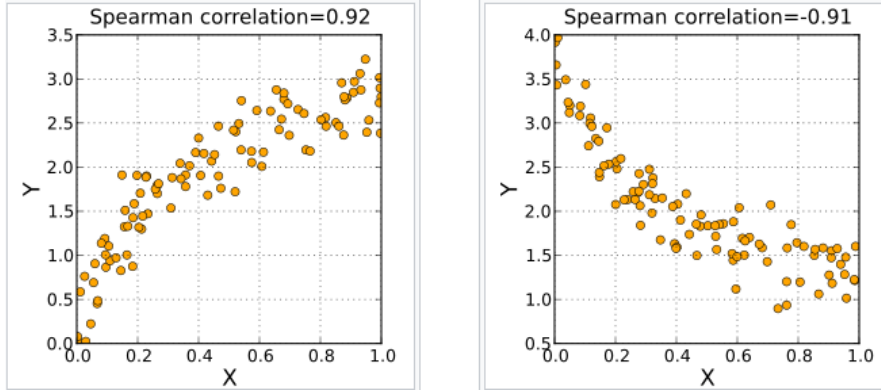


FIGURE 3.4: *Positive and Negative spearman plot between X and Y*

A correlation matrix can be measured by several different methods that we can implement such as Pearson rank, Spearman rank and Kendall rank correlation etc.

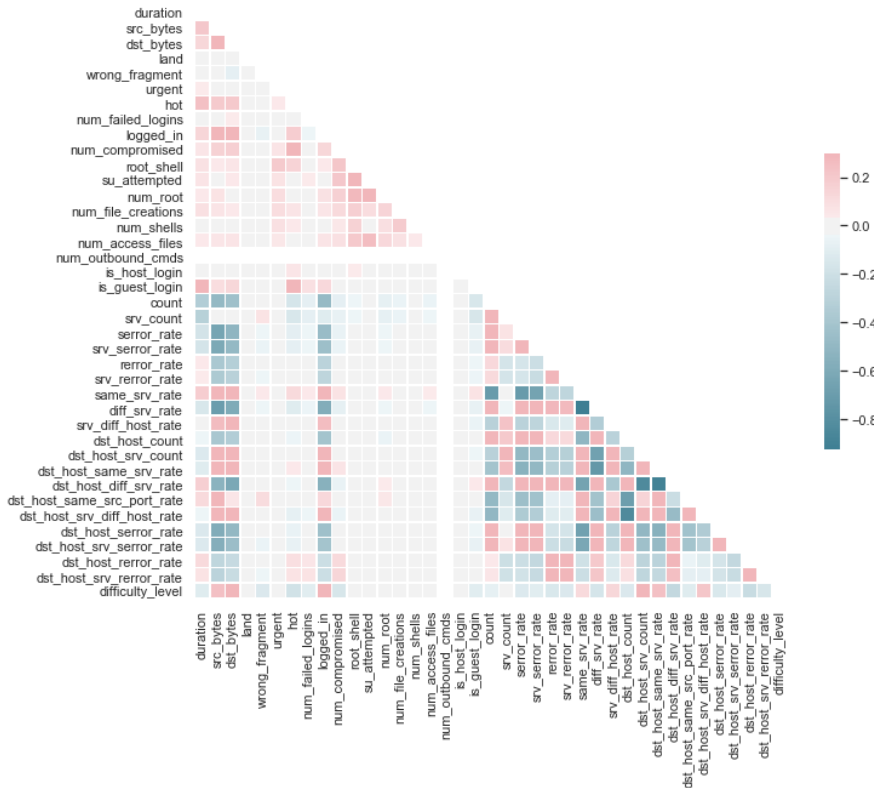


FIGURE 3.5: *Correlation plot between features for NSL-KDD Data set*

n this thesis, out of all these methods, we have used the Spearman rank correlation coefficient. Instead of using assumptions Spearman correlation uses the ranks for distribution of 2 variables which gave us the relationship between the variables.

In the above figure, we can see the Spearman coefficient plot, which tells the correlation between all 41 features. Also, there is a range bar on the right side which tells us about the positive or negative relationship between the features. From white colour to Bluish it represents the continuous negativity, and from white colour to red, it shows us the positive relationship.

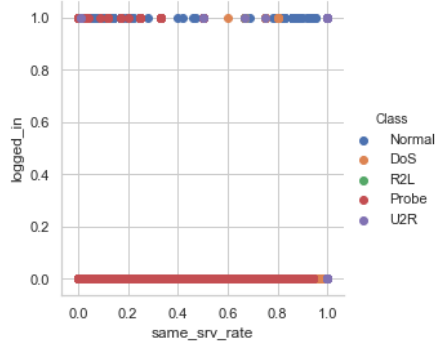


FIGURE 3.6: 2 Dimensional correlation plot

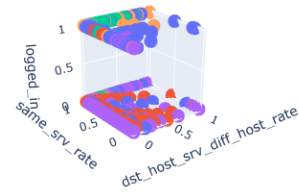
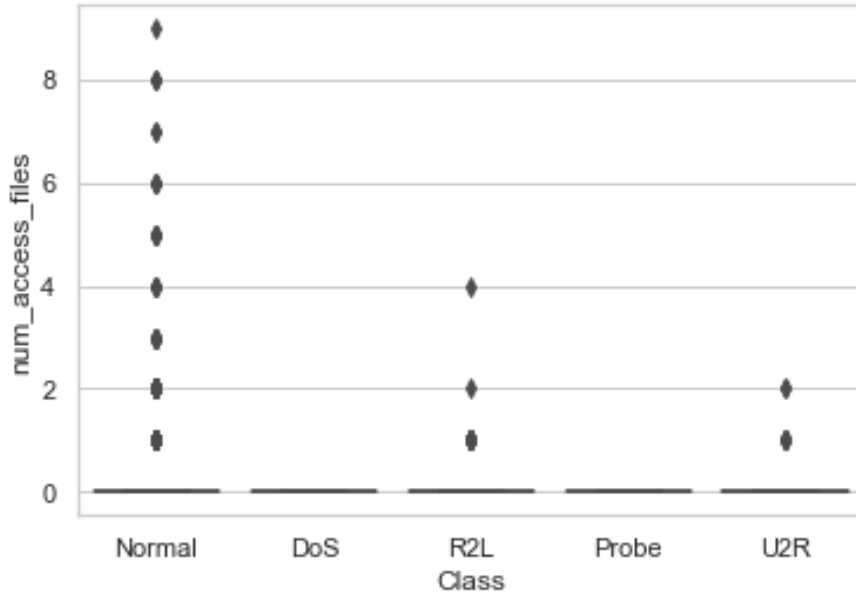


FIGURE 3.7: 3 Dimensional correlation plot

Above figures resembles the 2D and 3d plots. First figure is a 2D correlation plot with the help of spearman coefficient in between the 2 features which are highly correlated to each other. With the above figure we can easily say that the features "same_srv_rate", "logged_in" [21] are much correlated with each other for 2D and second figure is a 3D plot in between 3 features "same_srv_rate", "logged_in and dst_host_srv_diff_host_rate" that are much correlated for 3D plot. So, here we can see the overlapping in the plots. With the help of these plots, we can say that the features are related to each other, but there exists no clear visually distinguishable decision boundary that can separate each type of attack.

Now, further, we applied a Box plot which tells us about the outliers in the given data. Here we can see that there are 5 attack types and the anomalies and they are not well distributed as some of the points lie above the ideal distribution of the boxes.

FIGURE 3.8: *Correlation plot between features*

3.3.3 One Hot Encoding

One hot encoding is a method of transforming the categorical features which are nominal (Nominal means which can not) in nature to the numerical or binary features. The Machine Learning Classifiers works on the numerical values, so it is essential to one hot them in order to get the results from the classifiers.

There are a few disadvantages of doing one-hot encoding as well. The first disadvantage is that if we implement one-hot encoding, then the number of features increases from the previous features due to which the curse of dimensionality occurs, and another disadvantage is that in order to make the numerical values, there will be more number of 0s in the matrix, which is called as sparse matrix and generally classifiers will not perform well on both the points. However, our data NSL-KDD is one of the typical data sets which have multiple classes of attacks and also have nominal categorical features as stated above which makes the problem more typical and compel us to implement one-hot encoding on the nominal categorical features. The nominal features are protocol_type, flags and session on which we implemented the one-hot encoding.

3.3.4 Principal Complementary Analysis

PCA is a computational method used for identifying trends from a data set. It is also a method that is used for dimensional reduction[26]. The data sets for intrusion detection are so large and multidimensional[12]. Moreover, with high speed and distributed

networks, data visualization, processing, and understanding is more of the typical thing. PCA transforms the data set, which can be used to define similarities, differences, relationships between the data, which is non-visible so that we can easily make changes such as data compression, dimensional reduction. Principal component analysis is mostly used for the data reduction, i.e. dimension reduction. PCA takes a data set which might be in n number of dimensions and then it flattens the data set into 2 or 3 dimensions which can be easily plotted and visualized.

Also, we agreed that the dimensional reduction might affect the data as we might lose some information but the huge data sets which have 1000's of features to apply on the Machine Learning classifiers. So, in that case, with this huge amount of features, the classifiers will not work well as it is quite tough to process. Thus, PCA makes the new features from the data and then use the new features instead of using the original features of the data set. PCA calculates the covariance matrix[27] and then choose the principal features and then with the help of these principal features, it constructs the new data set. Co-variance matrix tells us about the variance of 2 features[26]. PCA is used for the Exploratory Data Analysis. It helps to visualize the data which have more than three dimensions. Moreover, it helps in finding the patterns in the data.

Here, we have generated 2-D and 3-D plots after implementing PCA on NSL-KDD Data set. Figures are shown above. There are five attack categories, namely Normal, DOS, Probe, U2R and R2L, which are denoted from colour coding and numbers as seen above in the image. 0 denotes to Normal, 1 denotes to DOS, 2 denotes to R2L, 3 denotes to Probe and 4 denotes to U2R attacks. ('Normal':0, 'DoS':1, 'R2L':2, 'Probe':3, 'U2R':4)

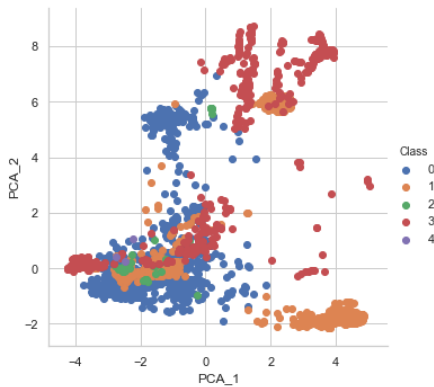


FIGURE 3.9: 2 Dimensional PCA Plot

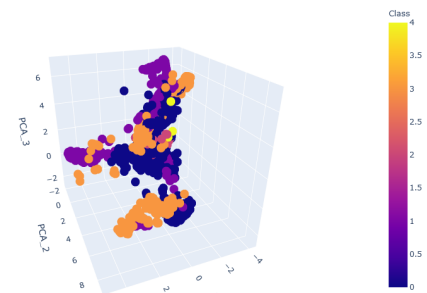


FIGURE 3.10: 3 Dimensional PCA Plot

3.3.5 t-SNE

t-SNE is a popular method for high-dimensional data exploration and has extensively used in the era of machine learning. We used t-SNE in our project for more 2D and 3D

evaluation[28]. Here we can see the 2D and 3D image below:

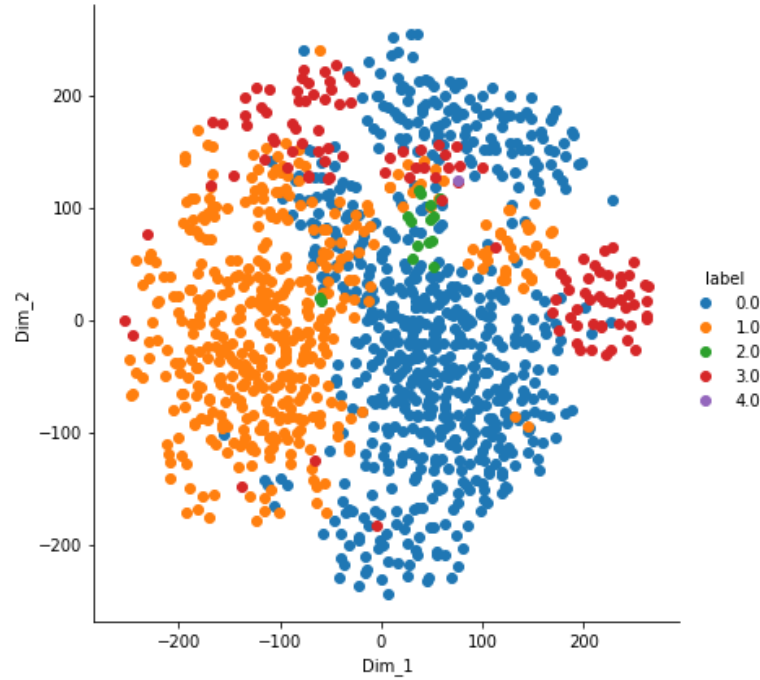


FIGURE 3.11: *Correlation plot between features*

3.4 Machine Learning Algorithms

For finding Anomaly detection, we are using the Machine Learning techniques. Therefore, we first start from the K-Nearest Neighbor; then we will use Naive-Bayse, then we will use Logistic Regression, then Support Vector Machines, then Decision trees and its variants such as random Forest, XG Boost, CAT Boost etc.

3.4.1 K Nearest Neighbor

KNN is a non-parametric, supervised machine learning algorithm. It works on the principle that similar patterns or things always lie in close distance or space. KNN usually works well with the linear data and on the smaller data sets. KNN is very simple to understand and implement. KNN works for classification as well as for the regression but usually works well for the classification. All data used for training can be used in the testing phase as well due to which it takes time in testing phase.[29]. For the prediction of falling of the new point, it uses the distance and then the voting from objects. Here, the value of k defines the total number of nearest neighbours.

KNN works in order of "ND" which is the time and the dimensions. So, it takes lots of time to generate the results.

Working of KNN:

As we already discussed that K represents the number of neighbours. Here, K nearest neighbours decide where the new query point will fall. Query data points are classified based on the how neighbours are classified[30].

Let us take an example of how KNN works. Here is a data set which has some labels as stars and triangles and Supposes Q is the query point which is falling in the data.

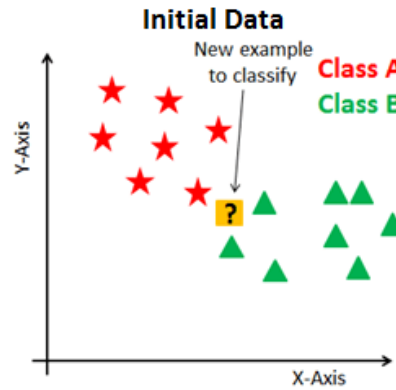


FIGURE 3.12: Correlation plot between features

[31]

Now, the question arises how to take the value of K . As the value of K is the hyperparameter and which depends on the requirement of each dataset. As the larger the value of K , the higher would be the computational power. Past researches also prove that the lower value of K results in lower biasing and the excellent fitting, but in case of large K value, the biasing is high, but the decision boundary is quite smooth. Generally, the value of K can be chosen in odd numbers for better voting results.

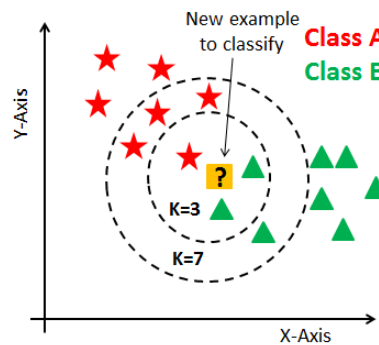


FIGURE 3.13: Correlation plot between features

[31]

Suppose the value of K is 5 (Value of K should be an odd value for successful voting), then first we have to calculate the distance of that query point from the first 5 neighbours in order to find out the closest similar points. There are many different distances that we can use, such as Euclidean distance, Manhattan, Minkowski or cosine etc. After that, we have to find out the closest neighbours and then vote for the labels. Each point will vote for their class and the class with the majority votes considered as a prediction[31]. In below two figures, we can figured out that first KNN calculates the distance using different distance metrics and then on the basis of distance, it first find out the neighbours and then with the help of weights, it vote for the labels.

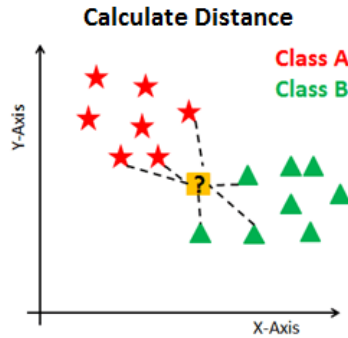


FIGURE 3.14: 2 Dimensional PCA Plot

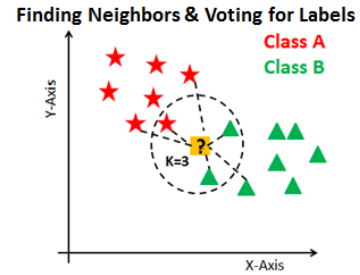


FIGURE 3.15: 3 Dimensional PCA Plot

[31]

KNN works on different parameters but mostly uses three parameters, i.e. weights, metric and value of nearest neighbours. So, weight is a function that used to predict query point[29]. The parameter weight is used for the prediction purpose. The value weight is either uniform or distance. With uniform weight, a value which is assigned to the query point is computed through the voting of the neighbours. Sometimes a case can occur, where there are two types of points like a star and a circle and a query point falls near to the circle. However, it does not mean that the query point should be the only circle; it might be a start as well. So according to the prediction KNN must predict the query point as a circle, so under these circumstances, the weight should be uniform. And voting must take place.

NOTE: Now, in this thesis, we have taken an NSL-KDD Data set which has 41 features and having a problem of 5 different classes. The data set contains some numerical features as well as the categorical features due to which we implemented one-hot encoding. One hot encoding resulted in the increment of features from 41 to 121, which makes it more typical to find out the best results, especially for KNN. Increment in dimension results in over-fitting problem. KNN would not perform well in the curse of dimensionality. As there are 121 features so there are 121 dimensions and we know that KNN

works in order of $O(nd)$ due to which it took lots of computational power and time. For this problem, we have not made the feature selection as our motive is to find out the exact attack type not only the attack.

Drawbacks of KNN:

- 1) **Space Complexity:** It works in order of $O(nd)$, which makes it difficult in the computational power as it gives the RAM issues or it requires the large memory to access it.
- 2) **Time Complexity:** It works in order of nd , which makes it difficult in the computational time. It takes lots of time if a data set has more features.
- 3) **Overfitting Problem:** Having larger features, the problem of overfitting occurs.
- 4) **Curse of Dimensionality:** If there are more number of Dimensions, then KNN will not work well. So, the problem of having more dimensions is the curse of dimensionality.

3.4.2 Naive-Bayes

Naive-Bayes is another algorithm of Machine Learning which works well on the large data, and also it is very very fast in computation. Naive-Bayes is dependent on the Bayes theorem, which is classification based. Naive-Bayes works with the probability like the probability of something in a given sample.

Bayes Theorem:

$$P\left(\frac{A}{E_1}\right) = \frac{P\left(\frac{E_1}{A}\right)P(A)}{P\left(\frac{E_1}{A}\right)P(A) + P\left(\frac{E_2}{A}\right)P(A) + P\left(\frac{E_3}{A}\right)P(A) + \dots}$$

So, here we can assume

$$P\left(\frac{E_1}{A}\right)P(A) + P\left(\frac{E_2}{A}\right)P(A) + P\left(\frac{E_3}{A}\right)P(A) + \dots as P(A)$$

So, the Bayes theorem is:

$$P\left(\frac{A}{E_1}\right) = \frac{P\left(\frac{E_1}{A}\right)P(A)}{P(A)} [27][29]$$

where,

- $P(A/E_1)$ is the probability of getting A given the data E_1 .

- $P(E1/A)$ is the probability of

Why its called as Naive Bayes?

The theorem is called Naive Bayes because they made a naive assumption of conditional independence between features. It means that all the features are independent with respect to each other.

Working of Naive-Bayes:

Naive-Bayes first calculates the probability for the different class labels which are already given. Then, for each class, Naive-Bayes calculates the likelihood probability with each of the feature present in the particular class. After calculating the probability and the likelihood probability, calculate the posterior probability from the Bayes formula and then gives the output in terms of higher probability[29].

$$\hat{y} = \underset{y}{argmax} P(y) \prod_{i=1}^n P(x_i | y) [29]$$

So, here we can use the posterior estimation to estimate the values of $P(y)$ and $P(x_i|y)$

The main important part of Naive-Bayes classifiers is that they only require a very small amount of data for training to learn and estimate the required parameters. So, there are different classifiers of Naive-Bayes that works on the different types of data[29].

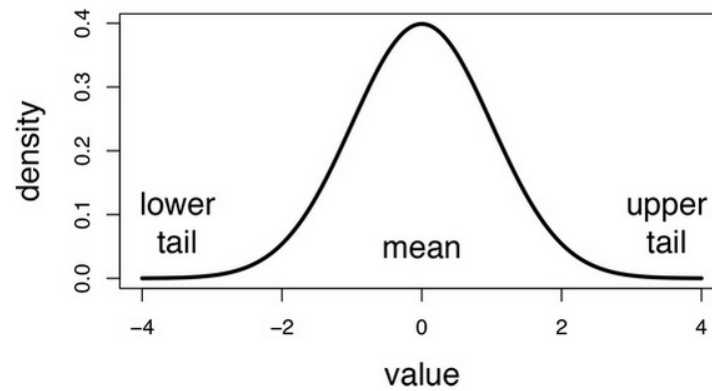
1) **Gaussian Naive-Bayes:** This is one of the types of Naive-Bayes when there are some peaks in the data set. The formula for the Gaussian Naive-Bayes is:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) [29]$$

2) **Multinomial Naive-Bayes:** This is one of the types of Naive-Bayes when the data is polynomially distributed. The formula for the Laplace Smoothing naive-Bayes is:

$$\hat{\theta}_{y_i} = \frac{N_{y_i} + \alpha}{N_y + \alpha n} [29]$$

3) **Bernoulli Naive-Bayes:** This is another type of the Naive-Bayes classifier in which the multiple features are present in the data and each of the feature presumed to be a

FIGURE 3.16: *Gaussian Curve*

binary variable[29]. The formula for the Bernoulli Naive-Bayes is:

$$P(x_i|y) = P(i|y)x_i + (1 - P(i|y))(1 - x_i)[29]$$

Advantages of Naive-Bayes:

- 1) Naive-Bayes is super fast in terms of computation as it works in order of d only and efficiently work on the larger data sets.
- 2) It has different Naive-Bayes Variances that can perform on different sorts of data sets.

Drawbacks of Naive-Bayes:

- 1) The major drawback of Naive-Bayes is that it is heavily affected by the class imbalance, and it becomes biased towards the majority class.
- 2) Also, there is no possibility of getting the independent features as in real-world scenario. The features are always dependent on each other.
- 3) One of the major problems is if any feature has no labels, then Naive-Bayes consider the whole class as 0 and due to this, the probability of that class becomes 0[27].

3.4.3 Logistic Regression

Logistic Regression is one of the Machine Learning Algorithms which work on the classification problems, not on regression which predicts the probability of the variables which are categorical dependent. Logistic Regression draws the line in order to separate the data. Logistic regression deals with the concept that the variable is only a binary variable, i.e. it is used in predicting the binary classes. It only works on the two possible

classes, i.e. 0 or 1. We know that the equation of line in 2-Dimensional is:

$$Y = mx + c$$

or

$$Y = Wx + b$$

Equation of line in 3-Dimensions is:

$$ax + by + cz + d = 0$$

So, for the multi-dimensions, the equation of line is:

$$Y = W^T X + b$$

, where W^T is the W transpose.

We know that the Logistic Regression separates the two classes of data with the help of a linear line. The data above the line is considered as positive data and represents with 1 and the data which comes under the line consider as negative data and represents with 0. The equation for finding the data, whether it is above the line or under the line is:

$$y(W^T X + b > 1) [29]$$

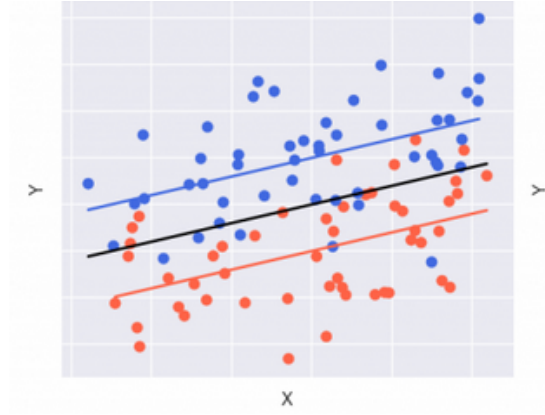
If the equation is greater than 1, then the data lies above the line

$$y(W^T X + b < 1)$$

If the equation is lesser than 1, then the data lies under the line.

From these equations, we can easily distinguish the data and can assume the class of data whether it is 0 or 1, So, there may be many lines which can separate the data but for the best possible line, the equation is:

$$W^* = \operatorname{argmax} \sum_{i=1}^n y_i W^T X [27]$$

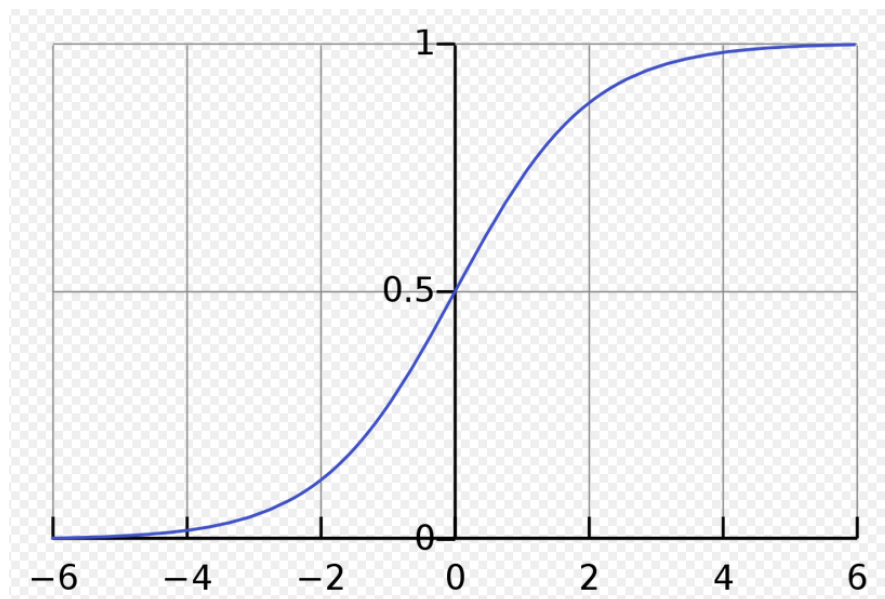
FIGURE 3.17: *Best Separable Line*

As we want our output to lie in the category between 0 and 1 inferring the probability where 0 corresponds to negative, and 1 corresponds to positive. Now, the value in between 0 and 1, say 0.5 denotes the point which is equally possible to lie in either of the side[29].

Now, we need another optimization function such as sigmoid which outputs the value between 0 and 1. So, we need to use the sigmoid function. The equation for the sigmoid function is:

$$f(x) = \frac{1}{1 + e^{-x}} [29]$$

The sigmoid curve can be represented as:

FIGURE 3.18: *Sigmoid Curve*

Now, by considering the class labels, my optimization equation thereby becomes

$$W^* = \operatorname{argmax} \sum_i^n \frac{1}{1 + e^{-(y_i W^T X)}}$$

Now, as we know that Logistic Regression classifier also will not perform well if there any outlier (anomaly) comes. So, if we do not want outliers to impact the classifier, we need to apply monotonically increasing function such as $\log(x)$ [27]. Monotonically increasing means if x is increasing then the $\log(x)$ should also be increasing. So, With the help of $\log(x)$, the problem of outliers are resolved. Here, we need to minimize the arg function.

The equation of using the log term in order to reduce the impact from the anomalies is:

$$W^* = \operatorname{argmin} \sum_i^n \log 1 + e^{-y W^T X} + \frac{\lambda}{2} \|W\|^2 [29]$$

Where,

$$C = \frac{1}{\lambda}$$

3.4.4 Decision Trees

Decision Trees are also a form of supervised machine learning which can be used for both classification and regression. DT is also a non-parametric algorithm. A Decision Tree is a tree-like structured diagram where always the first node is the root node of the tree. Then, decision node represents the features. Trees always are split into two parts like an if-else condition either yes or no and will grow further like that only. So, the branch represents the if/else condition. And the leaf node represents the output of the if/else conditions.

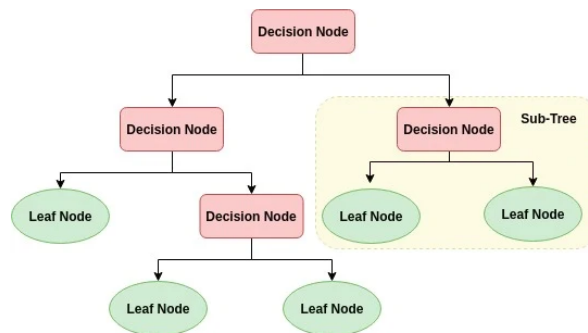


FIGURE 3.19: *Decision Trees*

[32]

To avoid over fitting problem, the size of any Decision Trees should not more than 10 leafs. This means any tree should not contain more than 10 decision-making conditions at one side[32]. Let us suppose if an anomaly occurred in the 8th or 9th leaf by any human mistake (like any recorded mistake, record 0 instead of 1), so if we further divide the tree into if/else conditions then because of that anomaly our full tree would go for a stake or simply we can not toss the whole tree for that anomaly. So, if our tree is in 8th or 10th leaf, then we generally consider some losses at one point, and we should not grow tree further. We should take all the points in 1 criterion and stop growing tree further.

Basically which feature to get selected and split into the parts depends upon the entropy of that feature or on the Information Gain of the feature. There is 1 condition on each and every node which needs to be satisfied in order to make the if/else condition work. Entropy decides which feature needs to be split first. **Higher the entropy of the feature, the more would be the chances of getting that feature split for decision making, and that results in the highest Information gain (IG).**

The entropy of any feature can be calculated by the formula:

$$H(X) = - \sum_{i=1}^k P(y_i) \log(P(y_i)) [33]$$

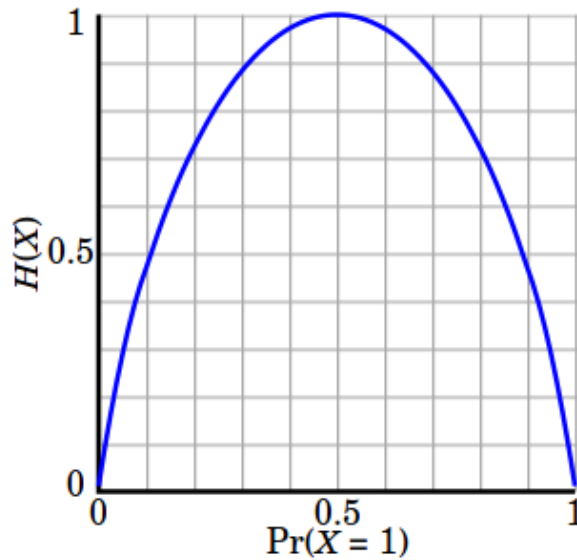


FIGURE 3.20: Entropy Graph

Entropy of any feature directly depends upon the selection of that feature for the splitting to further decision making.

Entropy can be calculated up to n number of classes. For classes with good balance, the entropy is always high. We can find the curve for entropy having a well balanced class. The Entropy of any feature can lies in between the range 0 to 1.

$$I_G(Y, X) = PreviousNode - WeightedEntropy$$

Information Gain or Gini Impurity shows similar behaviour as Entropy. But there is no log-term hence the computation is much faster as compared to an Entropy. So, we generally prefer Gini Entropy over Entropy. Information gain computes the difference between entropy before split and average entropy after split of the dataset based on given attribute values[32].

$$I_G(Y) = 1 - \sum_{i=1}^k P(y_i)^2 [29][32]$$

In the given below the figure, we can see the curves plotted in between an Entropy and Gini Entropy. Highest Entropy of any feature can onlky be 1, and the highest Gini Entropy for any feature can be 0.5 which is just half of the Entropy.

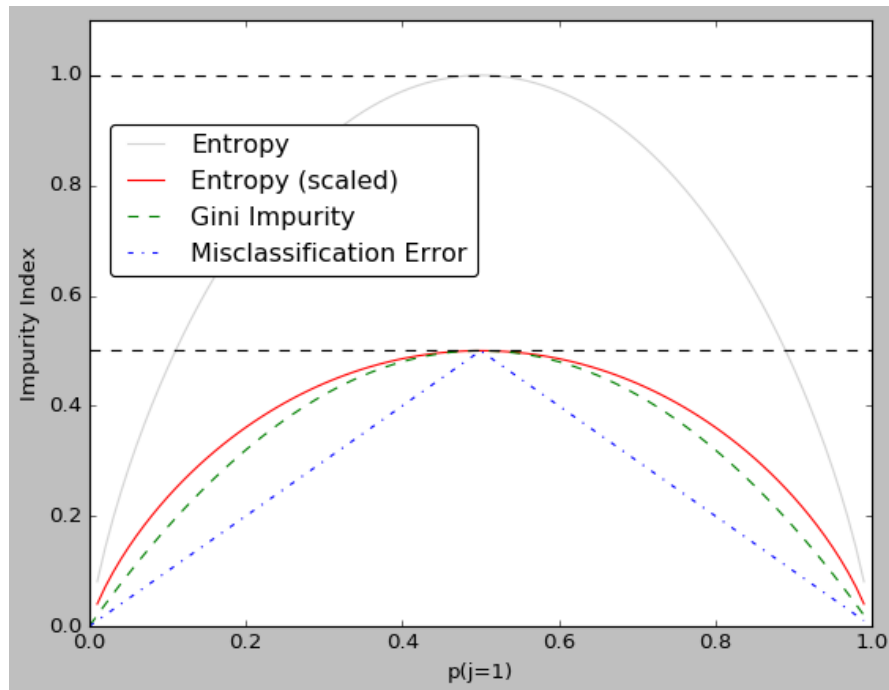


FIGURE 3.21: *Gini Entropy vs Entropy Graph*

3.4.5 Random Forest Classifier

Random Forest is a supervised machine learning algorithm that can be used for both regression as well as for classification. Random Forest is the variance of Decision Trees. Forest has consisted of many trees. Eventually, the Random Forest has consisted of many Decision Trees. It can be 1000s as well depending upon the problem. Random Forest works upon the principle of doing sampling in the data and on behalf of sampling make n numbers of Decision trees, and at last, each tree gives its prediction, and after all the prediction, the best optimal solution can be generated by the majority of voting[27]. Basically, its a technique used for the behaviour analysis depending upon the decision trees.

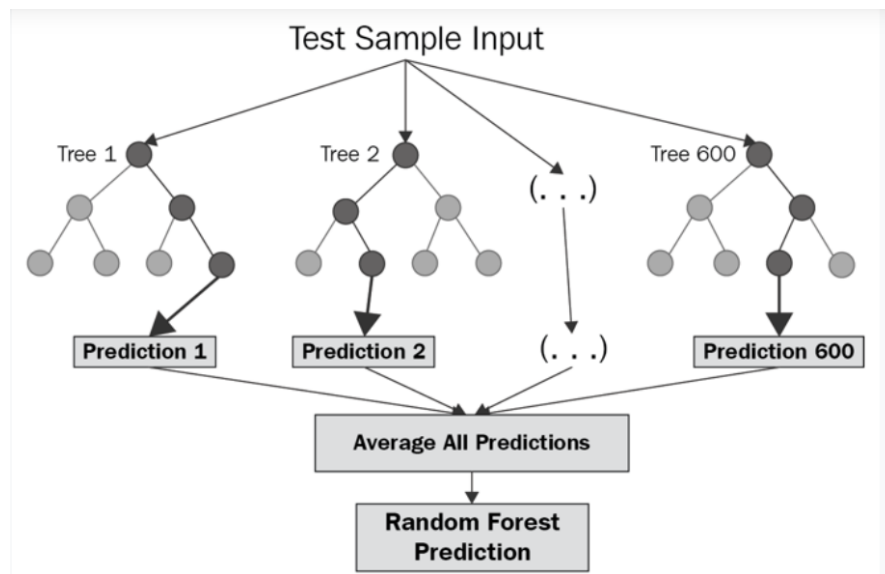


FIGURE 3.22: *Random Forest Decision Criteria*

The size of the tree is controlled by the set of the parameters with default values, The default values for the parameters. The default parameters are leaf, sampling, max depth, best_estimator etc which results in fully grown and unpruned trees which can potentially be very large on some data sets[29].

These parameters can be set in order to tune the classifier for best results and also to make computation low and to reduce the memory issues, also to control the size of the decision trees.

Working of Random Forest:

There are mainly 5 types of classifiers which are used in the Machine Learning such as Bagging, Boosting, Stacking, Cascading and Voting classifiers. And the Random Forest works on the bagging classifier. Bagging Classifier is nothing but the number of decision trees which is set as a parameter.

First of all, the data set is selected, and then the sampling is done on the full data set. Sampling is the process of sample out the features into n number of bagging classifiers (only column sampling) where any number of samples will go in any bagging classifiers. Let us suppose if there are 200 features (columns) and 2000 rows, and the n_estimator is set to 60 and the column separation is set to 20 then sampling can be done in such a way that there will be 60 decision trees having 20 random column samples in each bagging classifier (DT). Now after when the sampling is done, the prediction from each classifier takes place, it takes the average prediction, and then the voting must be done on behalf of the classifier's prediction. And at last, it considers the majority voting as the final prediction.

Here, the decision trees have samples of reduced features, and as there can be n number of trees so each and every tree learn some aspect of data, which lower down the amount of variance and also it would help in the prediction accuracy.

Advantages of Random Forest:

- 1) The main advantage of the Random Forest is that it beats the accuracy of the Decision Trees as there is n number of decision trees taking part in the process.
- 2) No need for doing the Sample reduction as bagging classifier itself does the feature reduction and selection.
- 3) Due to the average of all the predictions, random forest does not suffer from the overfitting problem[34].

Drawbacks of Random Forest: The main drawback of the random forests is that due to the number of decision trees the prediction time so long as all the decision trees have to make the prediction at the same time.

3.4.6 XG Boost:

XG (Extreme Gradient) Boost is one of the best Machine Learning algorithms, and it is another variance of Decision Trees. XG Boost works on the Boosting classifier as well as the Bagging classifier. XG Boost generally gives us a low variance and a high biased model. So, the core idea is to reduce the biasing but also keep the variance low. So, in order to remove the biasing, we have to do additive combining which generally means that we are going to train a new model and take the error or residual error and with the help of that residual error we have to train another tree and so on.[27]

Let us take a data set and then from the training data we train a tree called M_0 , while training the tree M_0 on $\{x_i, y_i\}$ will have large training error. Now, we have to train

another model called as M_1 , and now, the feature on which the model has to be trained are $\{x_i, error_i\}$ and then similarly we will have to train another model and from the error of that model we will have to train another model, maybe up to a large number of times. And then, at last, we have to take the sum:[27]

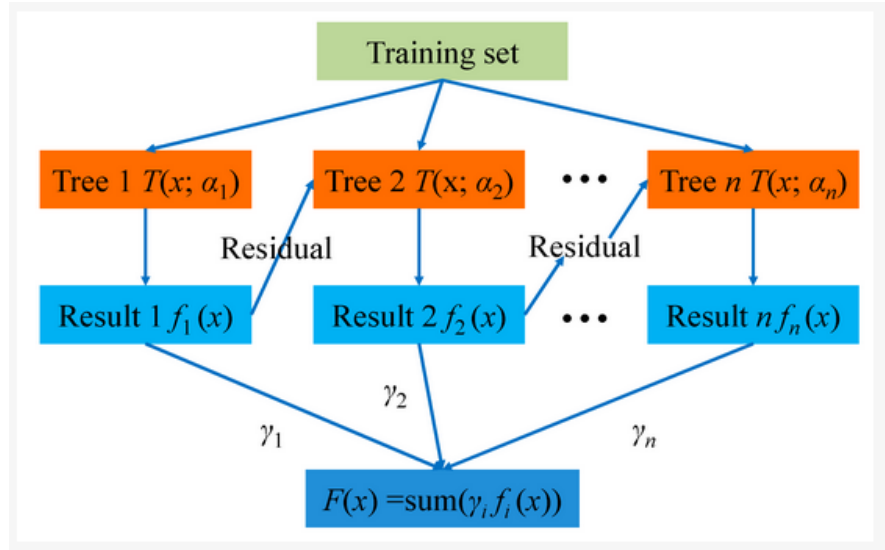


FIGURE 3.23: XG Boost

So, for the first decision tree to get trained, let us say the tree called M_0

$$M_0 = \{x_i, y_i\}$$

Then, the second tree M_1 should be trained on the x_i and $error_i$

$$M_1 = \{x_i, error_i\}$$

$$F_0(x) = \alpha_0 h_0(x)$$

Now, for the second decision tree, the model at the end of stage 1 is $\{x_i, error_i\}$.

$$F_1(x) = \alpha_1 h_1(x) [27][29]$$

So, if there are k number of stages, then after the end of stage k:

$$F_k(x) = \sum_{i=1}^K \alpha_i h_i(x) [27]$$

Here, k is the hyperparameter and $\alpha_i h_i(x)$ is the additive weighted model. Thus, each of this model has trained to fit the residual error at the end of the previous stage. Due

to this, the training error reduces and at every stage, it keeps reducing the errors as well.

XG Boost works on the bagging as well as Boosting classifier. Bagging we already discussed in Random Forest, that column-wise feature reduction comes under Bagging and now in Boosting, the row-wise feature reduction occur. Here each tree gives a prediction score which fully depends upon the data, and then the scores are summed up in order to obtain the final result. There are few hyperparameters due to which the performance of an algorithm may vary. The hyperparameters are:

- `max_depth`- which tells us about up to how much length a tree can grow.
- `n_estimators`- this tells us about the number of trees
- `colsample_bytree`- Features used by each tree. Overfitting can be occurred due to the high value of col sampling[35]
- `subsample`- Samples that are used by each tree.

Advantages of XG Boost:

- 1) The main advantage of XG Boost is that it handles out the missing values in the data. There is no need to extract out the null values, XG Boost tries the left and right-hand split and then it checks the higher loss for each node.
- 2) XG Boost is really much faster than any of the Decision Trees or as compared to the Gradient Boost. It works on the different cores of the CPU.

3.5 Deep Learning: Artificial Neural Networks (ANN)

An artificial neural network is also known as Multi-Layer Perceptron (MLP) capable of creating non-linear decision boundaries for high dimensional data.

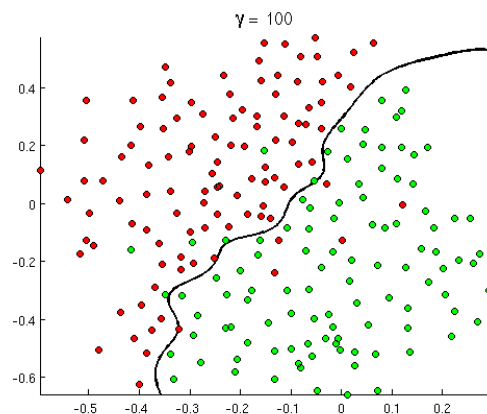


FIGURE 3.24: *Decision Boundary of a neural network*

Neural network follows a network architecture with each layer comprising of certain number of nodes also known as neurons which generally fire when an activation unit such as sigmoid produce a value greater than 0.5.

Each of the layers is fully connected with every neuron in the next and the previous layer. For the last layer, we apply a softmax function to get interpretable probability scores. The Softmax function ensures that the probabilities are interpretable in terms of class labels. Softmax also controls the output when some of the weights are assigned negative values which can often lead to negative outcomes for large network architectures. Application of Softmax layer ensures the class priors for each class label into an interpretable probability.

A Dropout is applied to ensure more robust training ensuring that the network does not essentially memorize the training data.[36]

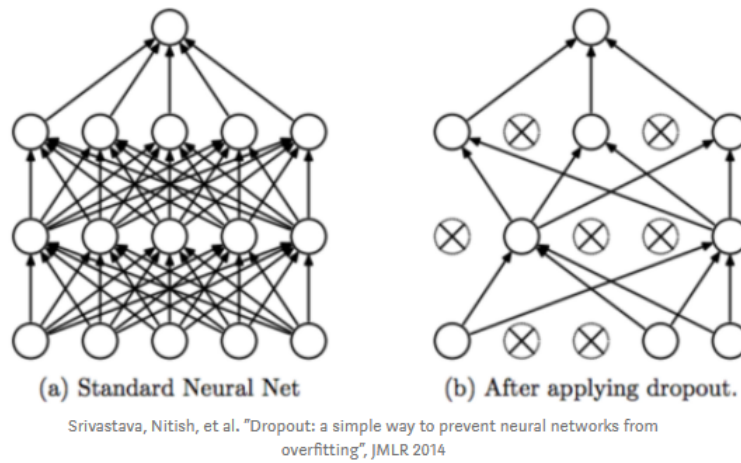


FIGURE 3.25: *Dropout rate*

[36]

ANN's are trained through a two-step process:

1) Forward propagation: weights are randomly assigned via an initialization scheme depending upon the activation function. For forward propagation, we employ a weighted sum for each layer and then push it through an activation function which decides if the neuron will fire. Essentially neurons fire in the Early layer indicate learning very low-level features while Neurons firing in the later layers indicate the presence of Higher features[37].

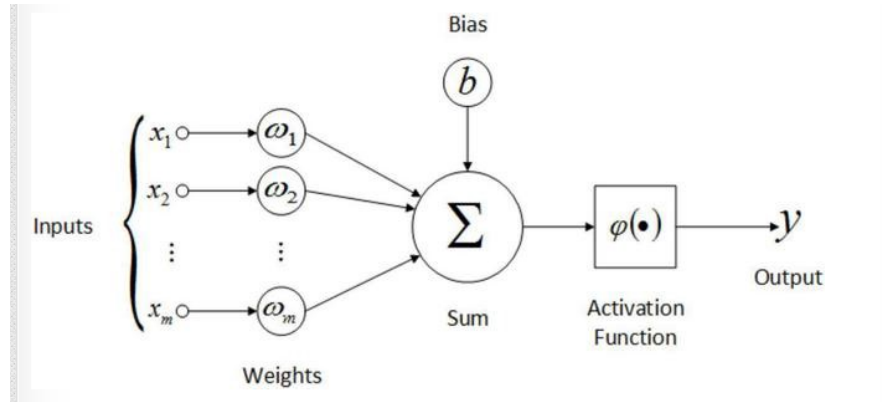


FIGURE 3.26: Flow
[38]

2) Backward Propagation: Since weights are randomly initialized, they certainly have to go through weights update mechanism to suit the data better. The weight update mechanism is determined by the flow of gradients from the loss function. It is extremely crucial to have a differential loss function and activation function. The weight upgrade usually takes place with Stochastic Gradient descent algorithm (SGD) or a variant of Gradient Descent for avoiding local minima.

Chapter 4

Implementation

4.1 Main Difficulty: Imbalanced Dataset with five (5) different classes

NSL-KDD is one of the first research data sets in the field of anomaly detection that has used for the Knowledge Discovery and the Data Mining tools Competition. We got this data set from the website [21] While using this benchmark data set for evaluation of Anomaly detection, we faced many problems in finding the good accuracy and f1 score which is mainly due to the curse of dimensionality, class imbalance in the data and multi-class classification as we were predicting whether any incoming packet is an attack and if yes then which attack type the attack belonged to but not only attack or non-attack.

Also, the labels which have given in the data set represents "Subclass" which is 42nd column in the data set. This subclass has approx 40 different unique classes (attacks) of data which means that the data sets contains approx 40 different classes

t_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	subclass	difficulty_level
0.17	0.00	0.00	0.00	0.05	0.00	normal	20
0.88	0.00	0.00	0.00	0.00	0.00	normal	15
0.00	0.00	1.00	1.00	0.00	0.00	neptune	19
0.03	0.04	0.03	0.01	0.00	0.01	normal	21
0.00	0.00	0.00	0.00	0.00	0.00	normal	21

Then in order to reduce the number of classes, we mapped all these different attack types into 5 attack categories and named this as "Class". Now, our data set consists of 5 different classes instead of having 40 unique categories.

Our data set contains 41 different features which are mainly categorical and numerical features. So, due to the 5 different classes and due to the categorical features the data set is quite imbalanced which impacted the f1 score.

Furthermore, in this data set we have categorical features, but those features are grouped into 2 categories like nominal and ordinal categorical features.

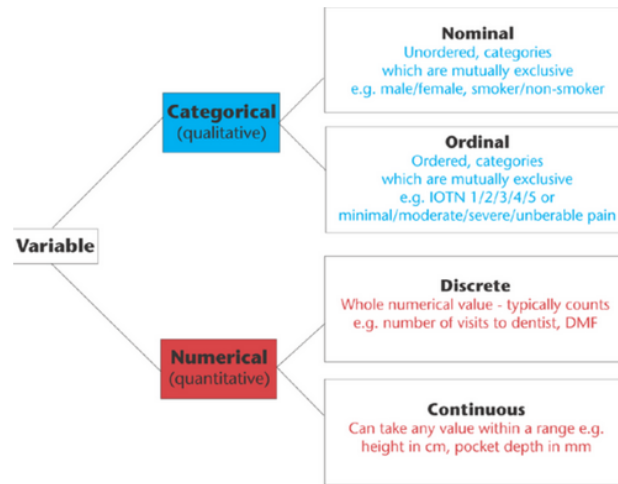


FIGURE 4.1: *Types of Categorical features*
[39]

A variable that does not have any ranking or order comes under a nominal category. Any calculation for the nominal variables such as a median or mean would not be suitable to them as because the order is arbitrary and we can not compare them by 0 or 1[39]. For example, there is a category of Gender in which there are sub-categories such as Man, Woman and others. Here, we can not say the man is bigger or woman or other, so we can not compare them with the help of numbers, and we can not say which is more significant[39].

An ordinal category is one where the order matters but not the difference between values. Like we can predict and compare them with each other through Machine Learning Algorithms on the scale of 0 and 1 through[39].

Out of 42 features, there were 27 numerical features, 12 categorical features (but they are ordinal) and 3 other categorical features which can not be compared with each other (nominal features). Now, in categorical features those who are ordinal means which can be distinguished by 0 and 1, we have not changed them. However, for those who are nominal features like protocol_type, flags and services, we have implemented one-hot encoding to them. One hot encoding assigns the values to the sub-features in 1 feature but do not compare them in order to save biasing. After implementing One Hot Encoding, the feature increases from 42 to 122.

In figure 4.2 which is shown below, we listed all the features like numerical, categorical-nominal and categorical-ordinal.

```
numerical_features_train=list(X_tr.columns[:27])
print(numerical_features_train)
categorical_features_train=list(X_tr.columns[27:27+11])
print(categorical_features_train)
one_hot_encoded_features_train=list(X_tr.columns[27+11:])
print(one_hot_encoded_features_train)

['duration', 'src_bytes', 'dst_bytes', 'hot', 'num_compromised', 'num_root', 'num_file_creations', 'count', 'srv_count', 'serror_rate', 'srv_error_rate', 'error_rate', 'srv_error_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_rate', 'dst_host_srv_diff_host_rate', 'dst_host_error_rate', 'dst_host_srv_error_rate', 'dst_host_error_rate', 'dst_host_srv_error_rate', 'difficultv_level']

['land', 'wrong_fragment', 'urgent', 'num_failed_logins', 'logged_in', 'root_shell', 'su_attempted', 'num_shells', 'num_access_files', 'is_host_login', 'is_guest_login']

['protocol_type_icmp', 'protocol_type_tcp', 'protocol_type_udp', 'service_ihl', 'service_xll', 'service_c39_50', 'service_801', 'service_auth', 'service_bgp', 'service_courier', 'service_csnet_ns', 'service_ctf', 'service_daytime', 'service_discard', 'service_domain', 'service_domain_u', 'service_echo', 'service_eco_i', 'service_ecr_i', 'service_efs', 'service_exec', 'service_fin_gm', 'service_ftp', 'service_ftp_data', 'service_gopher', 'service_harvest', 'service_hostnames', 'service_http', 'service_http_2784', 'service_http_443', 'service_http_8001', 'service_imap4', 'service_iso_tsap', 'service_klogin', 'service_kshell', 'service_ldap', 'service_link', 'service_login', 'service_mtp', 'service_name', 'service_netbios_dgm', 'service_netbios_ns', 'service_netbios_ssn', 'service_netstat', 'service_nntp', 'service_nttp', 'service_ntp_u', 'service_other', 'service_pm_dump', 'service_pop_2', 'service_pop_3', 'service_printer', 'service_private', 'service_red_i', 'service_remote_job', 'service_rje', 'service_shell', 'service_smtp', 'service_sql_net', 'service_ssh', 'service_sunrpc', 'service_supdup', 'service_systat', 'service_teln_et', 'service_tftp_u', 'service_tim_i', 'service_time', 'service_urh_i', 'service_urp_i', 'service_uucp', 'service_uucp_path', 'service_vminet', 'service_whois', 'flag_OTH', 'flag_RE3', 'flag_RST0', 'flag_RST0S0', 'flag_RST0R', 'flag_S0', 'flag_S1', 'flag_S2', 'flag_S3', 'flag_SF', 'flag_SH']
```

FIGURE 4.2: Showing Categorical and Numerical features in NSL-KDD Dataset

Two major problems that we faced during the analysis and implementation are and which affected our f1 score:

1. Multi-Class classification with imbalanced data and insufficient data in the data set which is the biggest problem as we predicted the attack and its attack type but not only whether the attack happened or not.
2. Particular class such as U2R, L2R also merged with different types of attacks, so it is quite tricky for the classifier to draw a singular decision boundary and to consider all the attacks as a whole 1 category.

```
[21]: print(combined_data["Class"].value_counts())
```

Normal	77232
DoS	53387
Probe	14077
R2L	3702
U2R	119
Name: Class, dtype: int64	

FIGURE 4.3: Category Wise Attack Counts

From above figure we can see that the data in the categories Probe, R2L and U2R is insufficient.

In figure 3.3 we can see that the attack categories U2R, Probe and R2L has not had a sufficient amount of data which is one of the biggest problems as the classifiers need data to learn. Moreover, with this non-sufficient amount of data, classifiers would not perform well and had not delivered the decent f1 score.

Implementation Part

4.2 Confusion Matrix and Scores:

Confusion Matrix is the table or array that helped us to visualize the performance of the supervised Machine Learning Algorithms. It is also known as an Error matrix. The table contains 2 rows and 2 columns that states the False Positives, False Negatives, True Positives and True Negatives.

1. True Positives are total number of incoming packets inside the network. True Positives can be calculated as:

$$\text{True positive} = \frac{TP}{P} [29]$$

2. True Negatives are the packet which are correctly rejected by the Intrusion Detection System.

3. False Positives are total number of falsely predict anomalies or false alarms generated by Intrusion Detection System. False Positives can be calculated as:

$$\text{False Positives} = \frac{FP}{N} [29]$$

4. False Negative are total number of missed attacks by the Detection System. It can be calculated as:

$$\text{False Negative} = \frac{FN}{P}$$

Precision is the estimated probability that the document which is selected is relevant. Basically, it is True Positive. Precision can be calculated as

$$\text{Precision} = \frac{TP}{TP + FP} [29]$$

Accuracy is the best match in the Matrix but it can be a mislead Matric for the imbalanced Data sets such as NSL-KDD

$$\text{Accuracy} = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + TN + FP + FN} [29]$$

F1 Score is the harmonic mean of the data which combines precision and recall. F1 score can be calculated as

$$F1Score = \frac{2TP}{2TP + FP + FN} [29]$$

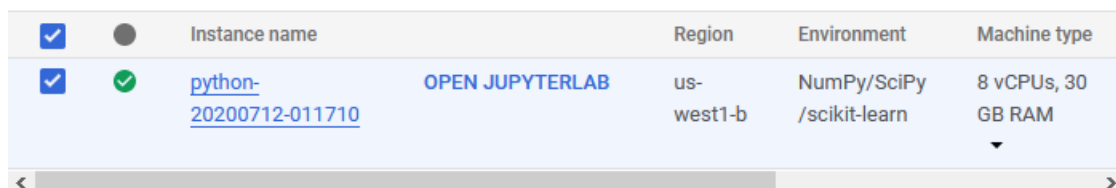
4.2.1 Grid Search

Grid search is the mechanism by which we can figure out the hyperparameters by tuning the classifiers in order to delimit the optimal values for a given classifier. The hyperparameters which we get define the overall performance of the classifier. [40]. So, to obtain the hyperparameters, Grid Search is present in the Scikit learn library to make the process automated.

Grid Search CV applies a “fit” and a “score” method. It also applies “predict”, “predict_proba”, “transform”, “decision_function”, and “inverse_transform” if they are implemented in the estimator used[29].

4.3 Platforms Used for Generating Results

We all know that Machine Learning Algorithms take lots of computation power and time. Algorithms like KNN, which works in the order of $O(nd)$, SVM, Logistic Regression etc. take lots of computation power and time. As I do not have sufficient amount of processors and RAM and also in order save my computation time and to boost my computation power, I have used the Google Cloud platform and used the Google console in order to run the Algorithms. There i have found the 8 CPUs and 30 GB RAM.



<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Instance name	Region	Environment	Machine type
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	python-20200712-011710 OPEN JUPYTERLAB	us-west1-b	NumPy/SciPy /scikit-learn	8 vCPUs, 30 GB RAM

FIGURE 4.4: *Google Console Machine Configurations*

Also, for implementing the Artificial Neural Networks, i have used the tensor flow framework. As tensor flow does not work on CPU, so I have used Google Colab which have provided the platform to run Artificial Neural Networks on GPUs.

4.4 Implementing KNN

We applied GridSearchCV function from the sci-kit learn library in order to get the tuned parameters. So, we can see in figure 4.5, which is shown below, that Grid Search first finds out the series of parameters. The parameters are metric, n_neighbors and weight. There are 3 metrics for which grid search will find out the optimal and the metrics are: 'cosine', 'euclidean', 'manhattan', weights as uniform or distance and here we choose their value of k=50.

```
from joblib import load
model_1=load("KNN.joblib")
print(model_1.get_params())

<bound method BaseEstimator.get_params of GridSearchCV(cv=10, estimator=KNeighborsClassifier(algorithm='brute'),
n_jobs=-1,
param_grid={'metric': ['cosine', 'euclidean', 'manhattan'],
'n_neighbors': [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21,
23, 25, 27, 29, 31, 33, 35, 37, 39, 41,
43, 45, 47, 49],
'weights': ['uniform', 'distance']}},
scoring=make_scorer(f1_score, average=macro, verbose=10)>
```

FIGURE 4.5: *KNN Parameters*

Then in figure 4.6 we can see that the best_parameters that are selected by the Grid Search are: 'metric': 'manhattan', 'n_neighbors': 3, 'weights': 'uniform' and these are the optimal parameters on which KNN Classifier has worked up to the mark.

```
print("="*20)
print("The F1 score is the harmonic mean between the precision and recall hence we really don't need to calculate both of them")
print("my f1 score on test: ", f1_score(y_test, y_pred, average="macro"))
print("my accuracy score on test: ", accuracy_score(y_test, y_pred))
print("="*20)

(125973, 122) (125973,) (22544, 122) (22544,)

this is my models accuracy :0.7432576295244855
KNN confusion matrix using brute:

tn:9627
fp:97
fn:1501
tp:5657

=====
The F1 score is the harmonic mean between the precision and recall hence we really don't need to calculate both of them
my f1 score on test: 0.5128323735408885
my accuracy score on test: 0.7432576295244855
=====
```

FIGURE 4.6: *KNN best parameters*

Now, with the help of these best parameters, we can view the Confusion Matrix, which is for 5 different classes. It tells us about the number of false positives, false negatives, true positives and true negatives.

Here, TN=9627, TP=7129, FN=4771, FP=262, MCA=755

The precision and recall calculated as given below:

Precision=96.4%

Recall=59.90%

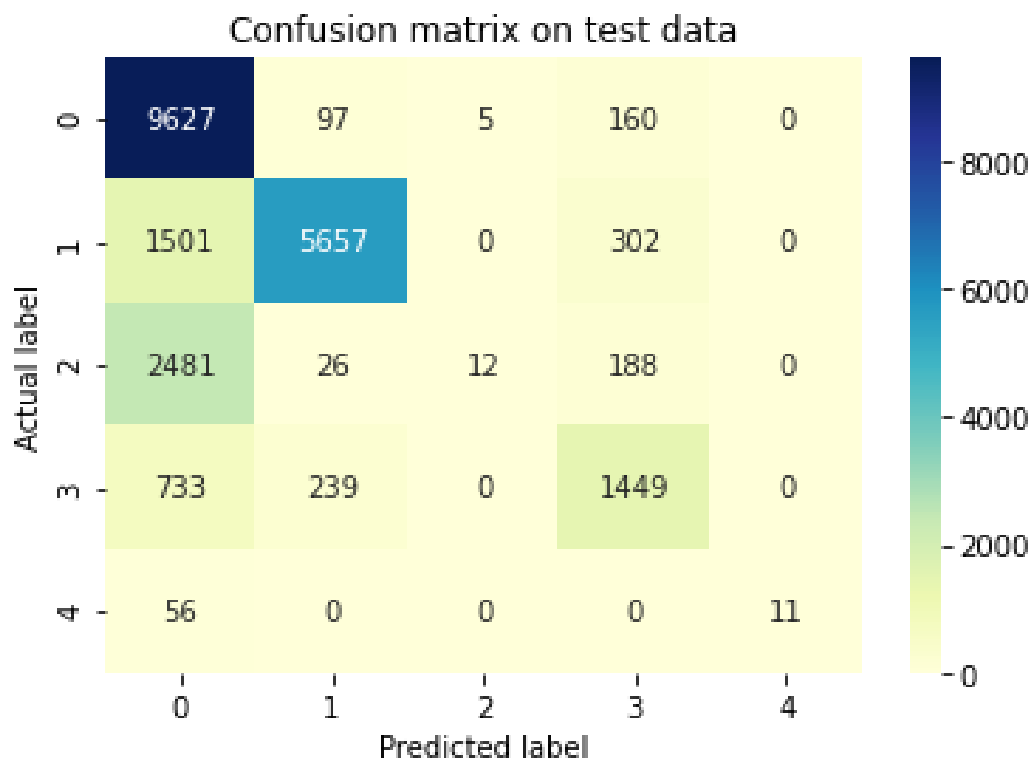


FIGURE 4.7: *KNN Confusion Matrix*

. Obtained "Accuracy is: 0.7432576295244855"

Obtained "F1 Score is: 0.5128323735408885"

4.5 Implementing Naive-Bayes

Again, we have applied GridSearchCV function from the sci-kit learn library in order to get the tuned parameters. So, we can see in figure 4.5, which is shown below, that Grid Search first finds out the series of parameters. The parameter here is only an α (alpha), which is Laplace smoothing and the value of α should be in between 1 to 5.

In the figure given below, we can see that the Grid search is applying various different values of Laplace Smoothing in order to find out the optimal value for the Naive-Bayes classifier.

```

: from joblib import load
  model_1=load("NB.joblib")
  print(model_1.get_params())

<bound method BaseEstimator.get_params of GridSearchCV(cv=10, estimator=GaussianNB(), n_jobs=-1,
  param_grid={'var_smoothing': [0.1, 0.2, 0.30000000000000004, 0.4,
                                0.5, 0.6000000000000001,
                                0.7000000000000001, 0.8, 0.9, 1.0,
                                1.1, 1.2000000000000002, 1.3,
                                1.4000000000000001, 1.5, 1.6,
                                1.7000000000000002, 1.8,
                                1.9000000000000001, 2.0, 2.1, 2.2,
                                2.3000000000000003,
                                2.4000000000000004, 2.5, 2.6, 2.7,
                                2.8000000000000003,
                                2.9000000000000004, 3.0, ...]},
  scoring=make_scorer(f1_score, average=macro), verbose=10)>

```

FIGURE 4.8: Naive-Bayes parameters

In the figure 4.9, which is given below, the best or optimal parameter which has been tuned by the GridSearchCV is: var_smoothing': 1.0.

```

from joblib import load
model_1=load("NB.joblib")
print(model_1.best_params_)

{'var_smoothing': 1.0}

```

FIGURE 4.9: Naive-Bayes optimal parameter

Now, with the help of these best parameters, we can view the Confusion Matrix, which is for 5 different classes. It tells us about the number of false positives, false negatives, true positives and true negatives.

Here, TN=9153

TP=7004

FN=4683

FP=736

MCA=968

The precision and recall calculated as given below:

Precision=90.49%

Recall=59.92% Here,

Obtained accuracy score: 0.7166873669268985 Obtained f1 score on test: 0.5202248467014893

4.6 Implementing Logistic Regression

Here, we have applied GridSearchCV function from the sci-kit learn library in order to get the tuned parameters. So, we can see in figure 4.5 which is shown below, that Grid Search first finds out the series of parameters. The parameters here are the value of C, class weight, l1_ratio, and solver.

```
model_1=load("logistic_regression_final.joblib")
print(model_1.get_params())
<bound method BaseEstimator.get_params of LogisticRegression(C=12.742749857031322, class_weight='balanced', l1_ratio=0.1,
random_state=0, solver='saga')>
```

FIGURE 4.10: Logistic Regression optimal parameters

In the figure above figure, we can see that the Grid search is applying several values of C and checking for the various solvers, but the optimal parameters tuned by the GridSearchCV are:

C=12.742749857031322, class_weight='balanced', l1_ratio=0.1, random_state=0, solver='saga'

Now, with the help of these best parameters, we can view the Confusion Matrix, which is for 5 different classes. It tells us about the number of false positives, false negatives, true positives and true negatives.

Here, TN=8987, TP=8270, FN=3601, FP=902, MCA=784. The precision and recall calculated as given below:

Precision=90.16% and Recall=69.66%

```
The F1 score is the harmonic mean between the precision and recall hence we
my f1 score on test: 0.5746303705582378
my accuracy score on test: 0.7654808374733854
*****
```

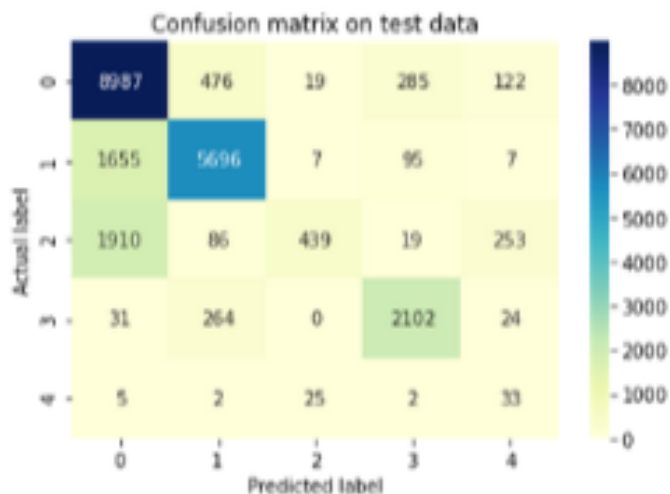


FIGURE 4.11: Logistic Regression Confusion Matrix

Obtained accuracy is: 0.7654808374733854

Obtained F1 Score is: 0.5746303705582378

4.7 Implementing Support Vector Machines

From the library sci-kit learn, we have applied GridSearchCV function in order to obtain the tuned parameters. So, we can see in figure 4.13, which is shown below, that Grid Search first finds out the series of parameters. The parameters here are the value of C, class weight, and kernel. Here we set the class weights = balanced as our data is quite Imbalanced.

```
from joblib import load
model_1=load("svm.joblib")
print(model_1.get_params())

<bound method BaseEstimator.get_params of GridSearchCV(cv=10, estimator=SVC(class_weight='balanced'), n_jobs=-1,
param_grid={'C': array([1.00000000e-02, 2.78255940e-02, 7.74263683e-02, 2.15443469e-01,
5.99484250e-01, 1.66810054e+00, 4.64158883e+00, 1.29154967e+01,
3.59381366e+01, 1.00000000e+02]),
'kernel': ['linear', 'poly', 'rbf']),
scoring=make_scorer(f1_score, average=macro, verbose=10)>
```

FIGURE 4.12: SVM parameters

The best optimal parameters that are found by the Grid SearchCV are shown below in the figure:

'C': 100.0, 'kernel': 'rbf'

Generally, the value of C varies from 10^{-2} to 10^2

```
model_1=load("svm.joblib")
print(model_1.best_params_)

{'C': 100.0, 'kernel': 'rbf'}
```

FIGURE 4.13: SVM best parameters

Now, with the help of these best parameters, we can view the Confusion Matrix, which is for 5 different classes. It tells us about the number of false positives, false negatives, true positives and true negatives.

Here, TN=9551, TP=4754, FN=4542, FP=338, MCA=3359

The precision and recall calculated as given below:

Precision=93.36%

Recall=51.14%

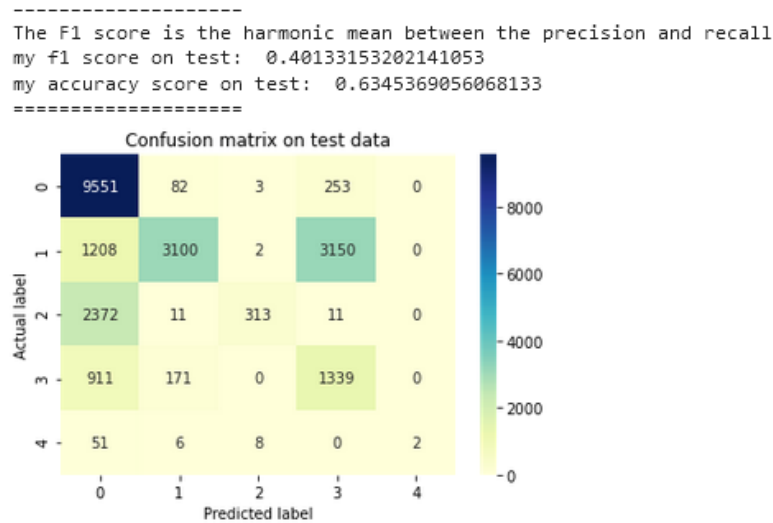


FIGURE 4.14: Support Vector Machines Confusion Matrix

Obtained Accuracy is: 0.6345369056068133

Obtained F1 Score is: 0.40133153202141053

4.8 Implementing Decision Trees

From the library sci-kit learn, we have applied GridSearchCV function in order to obtain the tuned parameters. So, we can see in figure 4.16, which is shown below, that Grid Search first finds out the series of parameters. The parameter here is the value of max_depth. The maximum depth should not exceed from 10. Here we set the class weights = balanced as our data is quite Imabalanced.

```

: from joblib import load
  model_1=load("DT.joblib")
  print(model_1.get_params)

<bound method BaseEstimator.get_params of GridSearchCV(cv=10,
  estimator=DecisionTreeClassifier(class_weight='balanced',
                                   max_features=122,
                                   random_state=1),
  n_jobs=-1, param_grid={'max_depth': [2, 3]},
  scoring=make_scorer(f1_score, average=macro), verbose=10)>

```

FIGURE 4.15: Decision Trees parameters

The best parameters that are tuned by the GridSearchCV is the max_depth, and it should be only 3, as shown in the figure below.


```

: model_1=load("DT.joblib")
print(model_1.best_params_)

{'max_depth': 3}

```

FIGURE 4.16: Decision Trees best parameters

Now, with the help of these best parameters, we can view the Confusion Matrix, which is for 5 different classes. It tells us about the number of false positives, false negatives, true positives and true negatives.

Here, TN=9065

TP=4352

FN=1091

FP=824

MCA=7212

The precision and recall calculated as given below:

Precision=84.08%

Recall=79.65%

The F1 score is the harmonic mean between the precision and recall
 my f1 score on test: 0.41846877090005774
 my accuracy score on test: 0.5951472675656494
 =====

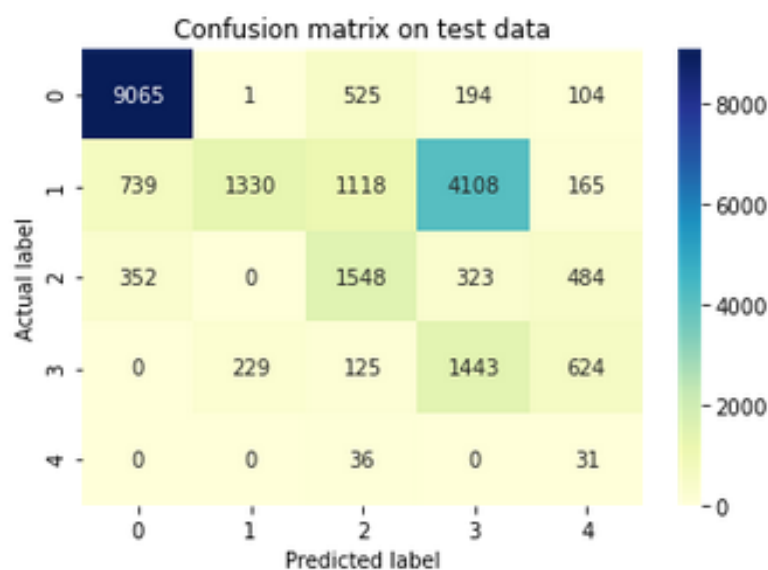


FIGURE 4.17: Decision Trees Confusion Matrix

Obtained Accuracy is: 0.5951472675656494

Obtained F1 Score is: 0.41846877090005774

4.9 Implementing Random Forest

From the library sci-kit learn, we have applied GridSearchCV function in order to obtain the tuned parameters. So, we can see in figure 4.19, which is shown below, that Grid Search first finds out the series of parameters. The parameters here are the value of class_weight and the n_estimators. GridSearchCV can take any value for the n_estimators.

```
from joblib import load
model_1=load("randomforest.joblib")
print(model_1.get_params)

<bound method BaseEstimator.get_params of GridSearchCV(cv=10, estimator=RandomForestClassifier(random_state=1), n_jobs=-1,
param_grid={'class_weight': ['balanced', 'balanced_subsample'],
'n_estimators': [150, 160, 170, 180, 190, 200, 210,
220, 230, 240, 250, 260, 270, 280,
290, 300, 310, 320, 330, 340, 350,
360, 370, 380, 390, 400, 410, 420,
430, 440, ...]}},
scoring=make_scorer(f1_score, average=macro), verbose=10)>
```

FIGURE 4.18: Random Forest actual parameters

Here, the optimal parameters searched by the Grid Search are:

'class_weight = balanced_subsample, n_estimators = 460

```
: model_1=load("randomforest.joblib")
print(model_1.best_params_)

{'class_weight': 'balanced_subsample', 'n_estimators': 460}
```

FIGURE 4.19: Random Forest best parameters

Now, with the help of these best parameters, we can view the Confusion Matrix, which is for 5 different classes. It tells us about the number of false positives, false negatives, true positives and true negatives.

Here, TN=9621, TP=7286, FN=5127, FP=268, MCA=242

The precision and recall calculated as given below:

Precision=96.45%

Recall=58.69%

```

=====
The F1 score is the harmonic mean between the precision and recall
my f1 score on test: 0.536765426300927
my accuracy score on test: 0.7499556422995032
=====

```

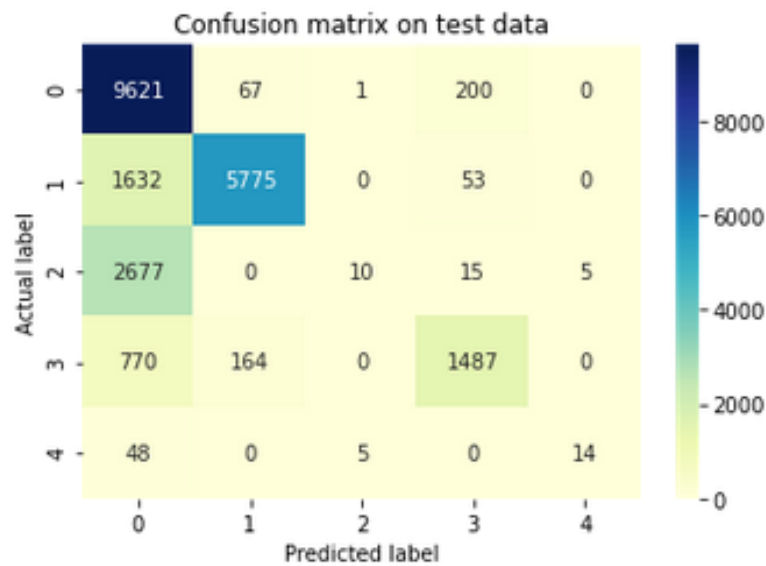


FIGURE 4.20: Random Forest Confusion Matrix

Obtained Accuracy is: 0.7499556422995032

Obatined F1 Score is: 0.536765426300927

4.10 Implementing XG Boost

XG Boost does not require parameter search as the default parameter is decision tree-based base learners with bagging and boosting classifiers. Here, with the help of the Confusion Matrix, the given classification and misclassification attacks are:

Here, TN=9446, TP=8486, FN=3742, FP=443, MCA=427

The precision and recall calculated as given below:

Precision=95.03%

Recall=69.39%

The F1 score is the harmonic mean between the precision and recall
 my f1 score on test: 0.6454282820070695
 my accuracy score on test: 0.7954222853087296
 =====

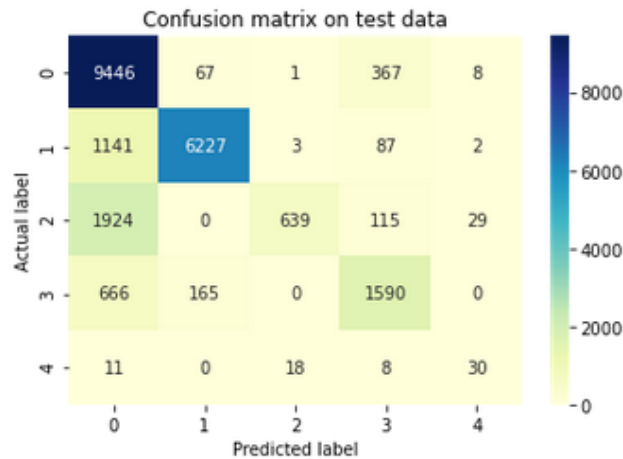


FIGURE 4.21: XG Boost parameters

Obatined Accuracy is: 0.7954222853087296

Obatined F1 Score is: 0.6454282820070695

4.11 Implementing CAT Boost

CAt Boost is one of the new Machine Learning Algorithms and is the state of the Art which beats the XG Boost in terms of time complexity for both the run time and train time and is basically used for the Categorical Features.

Here, the figure describes the Confusion Matrix for the CAT Boost classifier.

The total number of classified and misclassified attacks are: Here, TN=9457, TP=8432, FN=3777, FP=432, MCA=446

The precision and recall calculated as given below:

Precision=95.12%

Recall=69.06%

Obatined Accuracy is: 0.7935149041873669

Obatined F1 Score is: 0.649946338432737

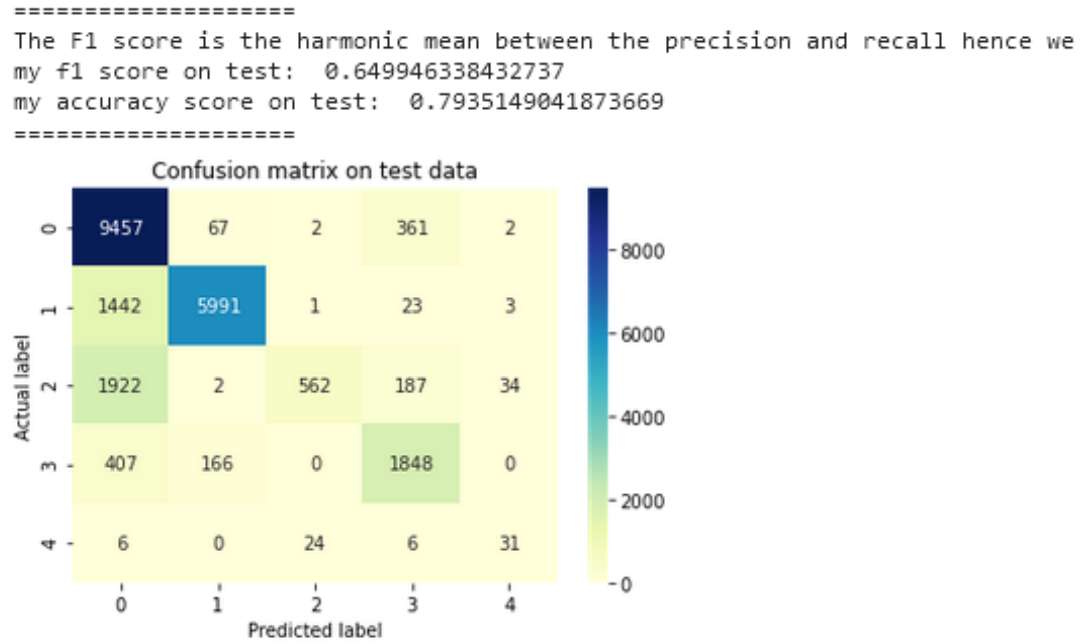


FIGURE 4.22: CAT Boost Confusion Matrix

Thus, after the implementation of all the Algorithms, we have calculated the TP, TN, FP, FN and also the Misclassified Errors for all the Algorithms And we have shown in the table given below:

TABLE 4.1: Total Number of Classified and Misclassified Attacks

Algorithms	TN	TP	FN	FP	Misclassified Attacks
KNN	9627	7129	4771	262	755
NB	9153	7004	4683	736	968
LR	8987	8270	3601	902	784
SVM	9551	4754	4542	338	3359
DT	9065	4352	1091	824	7212
RF	9621	7286	5127	268	242
XGB	9446	8486	3742	443	427
CATB	9457	8432	3777	432	446

4.12 Implementing Neural Networks

For training our neural network model, we employ a search space of hidden layers ranging from 2 to 40 with each layer having 96 neurons maximum. For easy gradient flow and weight updates we set our neural network to have a fixed activation function for

each of the layer, The activation function employed in the final network is again a hyperparameter which along with the architecture schema determines the loss of the network[38]. In our thesis, we have searched upon Swish, Sigmoid, Relu, tanh activation functions. A dropout is applied after every layer to reduce the amount of overfitting in the network. The dropout rate is constant and another hyperparameter determined by the optimization algorithm. Since our network can take a large number of layers that can result in the co-variance shift for some of the deeper architectures, we have also employed Batch normalization hard coding it after every 2 hidden layers in the network. Hard coding the batch normalization ensures a systematic, and easy to remember neural network architecture. Each of the network architecture is then tested upon the final categorical cross-entropy loss achieved after training for 15 epochs straight. The final model and lowest loss network architecture is then trained upon for an even larger number of Epochs to explore any further potential decrease in the loss function.

```
# Again Loading the object from the nn_net file and updating it again.
study = optuna.create_study(direction='maximize',study_name="NN_tuning",storage="sqlite:///nn_net.db",load_if_exists=True)
import pandas as pd
print("best_params",study.best_params)
print("f1--",study.best_value)
print("best trial--",study.best_trial)
```

[I 2020-08-03 03:45:42,163] Using an existing study with name 'NN_tuning' instead of creating a new one.
best_params {'activation_type': 'tanh', 'dr_rate': 0.03587723263680438, 'num_hidden_units_1': 29, 'num_hidden_units_2': 49, 'num_hidden_units_3': 62, 'num_hidden_units_4': 77, 'num_hidden_units_5': 20, 'num_hidden_units_6': 70, 'number_of_hidden_layers': 6}
f1-- 0.7686746120452881

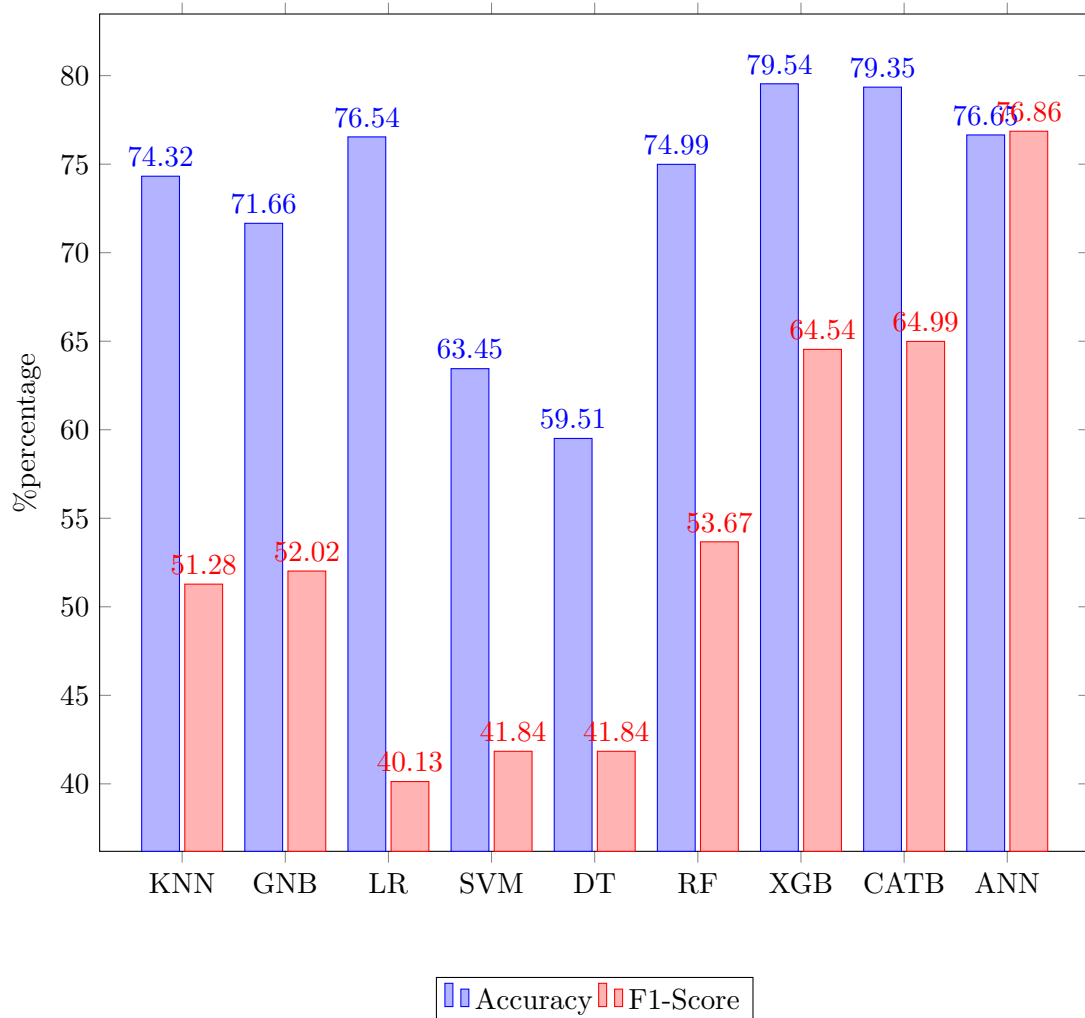
FIGURE 4.23: ANN Best Results

Obtained F1-Score is: 76.867461%

Chapter 5

Testing and Evaluation

After Implementation part, here we are evaluating all the machine learning algorithms and Neural Networks. Evaluation is based on the confusion Matrix which are generated for total number of false negatives, false positives, true positives and true negatives and also based on the accuracy and F1-Scores produced by them for NSL-KDD data set.



So, the data set NSL-KDD, which is a benchmark data set usually used for the evaluation of the performance for Machine Learning and Deep Learning Algorithms. This data set has been separated into a train and test data set. On train data sets, machine learning models have been trained, and these machine learning models then been evaluated with the use of test data set.

With the help of the above-generated bar graph for all the Machine Learning Algorithms, we can view and analyze the accuracy and F1-Score for each of the Algorithms.

As reported in table 5.1, we can see and observe that the Deep Neural Network is performing very well as compared to other Machine Learning Algorithms in terms of F1-Score, which can reach 76.86 % the NSL-KDD data set.

TABLE 5.1: Performance of different ML Algorithms

Algorithms	KNN	GNB	LR	SVM	DT	RF	XGB	CATB	ANN
Accuracy %	74.32	71.66	76.54	63.45	59.51	74.99	79.54	79.35	76.67
F1-Score %	51.28	52.02	57.46	40.13	41.84	53.67	64.54	64.99	76.86

With the below-generated table, we can find the precision and recall for the Machine Learning Algorithms which we have implemented so far. Also, the table tells us about the Misclassification Error rate means the number of attacks misclassified by the Algorithms.

TABLE 5.2: Precision and Recall

Algorithms	KNN	GNB	LR	SVM	DT	RF	XGB	CATB
Precision %	96.4	90.49	90.16	93.36	84.08	96.45	95.03	95.12
Recall %	59.90	59.92	69.66	51.14	79.95	58.69	69.39	69.06
Misclassification Error %	3.34	4.29	3.47	14.89	31.99	1.07	1.89	1.97

As we are dealing with the multi-class classification problem, so to achieve a high accuracy is the biggest problem and issue., As we were about to predict not only the attack but also the type of attack. Here, from the above table 5.1, we can see that the classifiers KNN, Naive-Bayes, Logistic regression, Random Forest, XG Boost, CAT Boost and Artificial Neural Networks gave us the decent accuracy which lies in between 70 to 80 %. Here we can see that the Algorithms Support Vector MACHines did not work well in terms of both Accuracy and F1-Score because SVM and DT are not directly applicable for the multi-class problems as in multi-class we have to deal with identifying each attack.

Now, in order to objectively evaluate and determine critically the accuracy and the amount of work we have done in this project with the help of Machine Learning Algorithms and the Deep Neural Network, I have compared my phenomenal approach with some of the papers with related and similar works proposed by [22][23][14][24]. Some of the researchers and scholars solve the problem of NSL-KDD data set with very good accuracy, and F1-Score **but have converted the multi-class classification problem to the binary class classification problem.** Author [24] classified the whole data set and normalized it in 0 and 1 for different attack classes in order to achieve better accuracy and also with feature reduction, they have reduced the features from 41 to 6 only. But in my opinion, the approach of binary classification is meaningless as any classifier will work and give high accuracy. and also not a valid approach for the multi-class classification problem as each of the classifiers would only suggest if there would be an attack or not.

In our case, we have not converted the problem to the binary class and as we have to predict the attack and its type. Though it would be very tough to deal multi-class classification problem with Imbalanced data, still somehow managed to get a decent accuracy and a good F1-Score with the implementation of Artificial Neural Networks.

Author [14] accomplished both the approaches; binary classification as well as the multi-class classification and achieved a decent F1-Score on the multi-class classification problem as we have discussed in Chapter-2 in the "State of the Art". They have also reported a very low score for the Naive-Bayes and Support Vector machines, And they have reported 76.5% F1-Score with the help of ANN. We can have a look at the figure below comprises the performance results of different classifiers they used.

Architecture	Accuracy	Precision	Recall	F-score
Multi-class classification - NSL-KDD				
DNN 1 layer	0.778	0.780	0.778	0.760
DNN 2 layers	0.777	0.777	0.777	0.757
DNN 3 layers	0.781	0.785	0.781	0.764
DNN 4 layers	0.780	0.816	0.780	0.763
DNN 5 layers	0.785	0.810	0.785	0.765

FIGURE 5.1: *Artificial Neural Networks layer Wise F1-Score*

[14]

In our case, we have implemented Artificial Neural Networks on the NSL-KDD data set

and as compared to the paper [14], we have achieved better performance with the help of Artificial Neural Networks which. As shown in the figure above, they obtained 76.5% F1-Score with ANN, and as we mentioned in Table 5.1, we achieved better F1-Score, which is 76.86 by implementing ANN.

Authors [22] also follows the multi-class classification problem and achieved the best result with their model BAT-MC as we discussed in State of the Art and also shown in Figure 5.3. Here they achieve the decent scores on the Machine Learning Algorithms as shown in figure 5.2, because of the multi-class problem and also due to Imbalanced data set. In the figure below, we can analyze that all the Machine Learning Classifiers gave them a decent Accuracy score in between 70 top 80%.

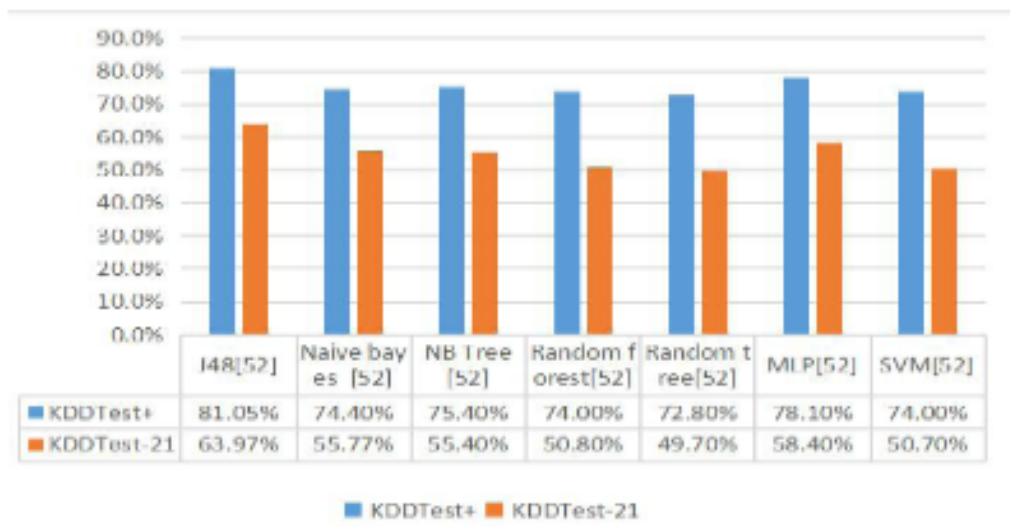
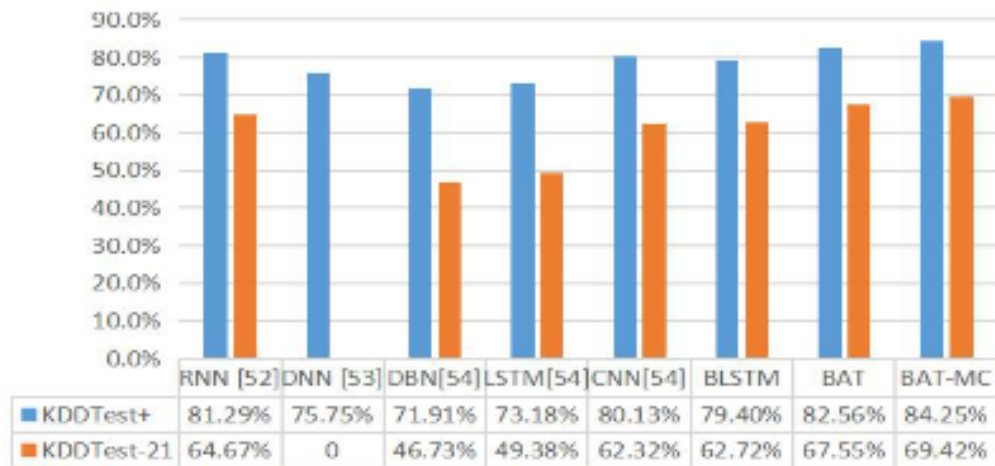


FIGURE 5.2: *Performance of Machine Learning Algorithms*
[22]

Here, they have also accomplished the good F1-Score for different Deep Learning classifiers such as DNN, RNN etc as shown the figure 5.3 below and imposed the new model which is BAT-MC and which gave the highest accuracy of 84.25 % on NSL-KDD data set.

FIGURE 5.3: *Performance of Deep Learning Algorithms and BAT-MC*


[22]

In comparison, we also accomplished the decent accuracy in between 70 to 80% from most of the classifiers except SVM and Decision Trees. Also, we have achieved a high accuracy with ANNs as compared to [22] and [14] which is 76.86 %.

Chapter 6

Discussion and Conclusions

6.1 Discussion

According to our evaluation, We have tried most of the supervised Machine Learning Algorithms on NSL-KDD data set, but we were not successful in achieving the good results in terms of F1-Score as there are few problems related to NSL-KDD data set. One of the problems is that the data set has two types of categorical features, namely; ordinal and nominal features. Nominal features are Protocol_Types, Flags and Services. So, for these three nominal features, we have to implement the One hot encoding due to which our number of features become 123. Due to this high number of features, we have faced the problem curse of dimensionality. Another problem is that the data set is highly imbalanced, which we can conclude after implementation of Machine Learning Algorithms on NSL-KDD data set. Moreover, the last problem which we have faced was Multiple classes which are not easy for any Machine Learning Algorithms to give a good score. In terms of Accuracy we achieved a decent accuracy rate in between 70% to 80% from the Algorithms. So for more better results in terms of F1-Score, we have implemented the Artificial Neural Networks and as we can see them from the Bar graph , the optimum best results we have achieved from Artificial Neural Networks. Some of the Algorithms like XG Boost and CAT Boost also provided us with much better results in terms of accuracy.

Machine Learning Algorithms did not give us good results because of the problems that we have discussed in the last paragraph. We know that Artificial Neural Networks are made to solve the Multi-class classification problem and perform well with number of features, so it is free from the curse of dimensionality, and also it does not matter if the data is Imbalanced. In Chapter 5, we have evaluated, compared and discussed our

results with some of the research papers, which are State of the Arts. We can see in figure 5.3 which has taken from the current State of the Art[22]. Here, they have achieved 75.75% F1, and if we can see 5.1, we have got 76.86%. We have achieved this with the help of Optuna, which is one of the optimization Algorithm. Optuna helped us to find out the perfect parameters like Dropout rate, Number of neurons in each layer, number of outbound layers.

Our problem is quite big, and I have addressed my problem in Chapter 4 where I have mentioned that we are working with the Multi-class classification. I also have discussed in Chapter 4 that some of the researchers made this problem quite simple by transforming the problem into the Binary classification and also with the help of Feature reduction, but that is not the correct method to solve the Multi-Class classification.

Despite of the fact that this is difficult to achieve good accuracy and F1, we still focused on the Multi-class without feature reduction, which is the only correct method to solve the problem. We have faced difficulties as Machine Learning algorithms were not performing well for us with this data set problem but with the continuous reading of different algorithms made me achieve the better Accuracy and F1-Score with the help of XG Boost, CAT Boost and ANN. In my opinion, there is no concept of transforming the problem into the Binary classification and also the State of the Art[22] proves it as they also worked on the Multi-class classification without being converted the problem into the Binary Class. Because of heavy variation in the data, we can observe that the State of the Art solution proposed has difficulty in achieving good accuracy with Machine Learning Algorithms as evident in figure 5.2. But they really perform well with Deep Learning Algorithms as we see it clearly in the figure5.3

If I would have an opportunity to do this project again, then I would have started the implementation part with capturing the live data. And on that live data, then i would have started implementing the Algorithms. I would definitely go beyond ANN and definitely go till the State of the Art which is BLSTM. After this thesis, I will carry my work and will learn more about Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Long-Short Term Memory (LSTM) and then Bi-directional Long-Short term memory (BLSTM). As i found this field quite interesting, i might carry forward my thesis work and will implement the further Algorithms.

6.2 Conclusion

Here, we can conclude that after implementation of Machine Learning Algorithms on NSL-KDD data set that it has a multi class-classification problem which is much difficult to be solved with the help of classical Machine Learning Algorithms. We can look Table5.1 where we can see that Machine-Learning Algorithms struggled to perform well

on the data set. If we talk in terms of Accuracy then we only achieve a decent accuracy rate which is in between 70% to 80%. We have concluded that Algorithms like Support Vector Machines, can not be used for Multiple classification problem as it gives us the minimum F1 Score and can only be worked efficient on smaller data sets and can be practically well suited for a Binary class classification problem. We have also concluded that scholar [24] also transformed the Multi-class classification problem to Binary class classification by comparing all the attacks with Normal category which is not possibly the good solution because any classifier can work for Binary problem and can give good accuracy rate. [24] also reduced the number of features from 41 to 6. So, we know that less the number of features, high will be the accuracy. The problem can only be solved with the help of Deep Learning as we have got some shreds of evidence from State of the Arts where only Deep Learning Algorithms worked so efficiently for the Multi-Class Classification problem. Similarly, in our case, as we have seen that Artificial Neural Networks gave us the best F1 Score which 76.86%. And the best F1-Score is achieved by BAT-MC Model which is 84.25%.

From the confusion matrix table 4.1, we can identify the critical false positives and negatives that vastly measure the performance of each of the classifiers built, critically analyzing the trade-offs of each of the algorithms such as class bias due to imbalance in the data. A general observation is that the classifier fails to generalize well for minority samples such as U2R with the lowest precision and recall score due to lack of samples to update the weights of each of the classifier built. And with the help of Confusion Matrix table 4.1, we can identify the attacks and their categories, directly corresponding to the F1-Score of the classifiers built.

6.3 Future Work

In this thesis, we have implemented Artificial Neural Network and also various Machine Learning Algorithms to the benchmark data set known as NSL-KDD data set. However, the deployment of Artificial Neural Network is the first model of deep learning, and still, there are lots of other models which can be deployed to get better performance. Performance in terms of better accuracy to detect the false negatives more precisely and reduce false positives.

As future work, The performance on NSL-KDD data set can be further enhanced by deploying the Recurrent Neural Networks (RNN) and also with the help of Bi-directional Long-Short term memory (BLSTM). We have already discussed the BAT-MC model proposed by [22] in the State of the Art Section, which gave the best accuracy and F1-Score and also we have discussed Recurrent Neural Networks (RNN) which will also be the future scope for this thesis work.

Bibliography

- [1] G. Shang-fu and Z. Chun-lan, “Intrusion detection system based on classification,” in *2012 IEEE International Conference on Intelligent Control, Automatic Detection and High-End Equipment*, July 2012, pp. 78–83.
- [2] J. Singh and M. J. Nene, “A survey on machine learning techniques for intrusion detection systems,” *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 2, no. 11, pp. 4349–4355, 2013.
- [3] A. Warzyński and G. Kołaczek, “Intrusion detection systems vulnerability on adversarial examples,” in *2018 Innovations in Intelligent Systems and Applications (INISTA)*. IEEE, 2018, pp. 1–4.
- [4] A. Lazarevic, V. Kumar, and J. Srivastava, “Intrusion detection: A survey,” in *Managing Cyber Threats*. Springer, 2005, pp. 19–78.
- [5] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.
- [6] S. Axelsson, “Research in intrusion-detection systems: A survey,” Citeseer, Tech. Rep., 1998.
- [7] M. Ahmed, A. N. Mahmood, and J. Hu, “A survey of network anomaly detection techniques,” *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.
- [8] H. Ringberg, M. Roughan, and J. Rexford, “The need for simulation in evaluating anomaly detectors,” *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 1, pp. 55–59, 2008.
- [9] K. Scarfone and P. Mell, “Guide to intrusion detection and prevention systems (idps),” National Institute of Standards and Technology, Tech. Rep., 2012.
- [10] A. Halimaa and K. Sundarakantham, “Machine learning based intrusion detection system,” in *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*. IEEE, 2019, pp. 916–920.

- [11] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 303–336, First 2014.
- [12] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Computer networks*, vol. 51, no. 12, pp. 3448–3470, 2007.
- [13] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.
- [14] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41 525–41 550, 2019.
- [15] O. Simeone, *A Brief Introduction to Machine Learning for Engineers*. now, 2018. [Online]. Available: <https://ieeexplore-ieee-org.cit.idm.oclc.org/document/8453245>
- [16] E. Alpaydin, *Introduction to machine learning*. MIT press, 2020.
- [17] K. A. Taher, B. M. Y. Jisan, and M. M. Rahman, "Network intrusion detection using supervised machine learning technique with feature selection," in *2019 International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*. IEEE, 2019, pp. 643–646.
- [18] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan, "Cost-based modeling for fraud and intrusion detection: Results from the jam project," in *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, vol. 2. IEEE, 2000, pp. 130–144.
- [19] D. D. Protić, "Review of kdd cup'99, nsl-kdd and kyoto 2006+ datasets," *Vojnotehnički glasnik*, vol. 66, no. 3, pp. 580–596, 2018.
- [20] A. Gül and E. Adalı, "A feature selection algorithm for ids," in *2017 International Conference on Computer Science and Engineering (UBMK)*, 2017, pp. 816–820.
- [21] L. Lab. Nsl-kdd data set. [Online]. Available: <https://www.unb.ca/cic/datasets/nsl.html>
- [22] T. Su, H. Sun, J. Zhu, S. Wang, and Y. Li, "Bat: Deep learning methods on network intrusion detection using nsl-kdd dataset," *IEEE Access*, vol. 8, pp. 29 575–29 585, 2020.

- [23] C. Yin, Y. Zhu, J. Fei, and X. He, “A deep learning approach for intrusion detection using recurrent neural networks,” *Ieee Access*, vol. 5, pp. 21 954–21 961, 2017.
- [24] L. Dhanabal and S. Shantharajah, “A study on nsl-kdd dataset for intrusion detection system based on classification algorithms,” *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 6, pp. 446–452, 2015.
- [25] D. H. Deshmukh, T. Ghorpade, and P. Padiya, “Improving classification using preprocessing and machine learning algorithms on nsl-kdd dataset,” in *2015 International Conference on Communication, Information & Computing Technology (ICCICT)*. IEEE, 2015, pp. 1–6.
- [26] K. Yasar. Pca: Principal component analysis. [Online]. Available: <https://medium.com/@kyasar.mail/pca-principal-component-analysis-729068e28ec8>
- [27] A. AI. Artificial intelligence. [Online]. Available: <https://www.appliedaicourse.com/>
- [28] T.-S. Visualization. (2014) t-sne. [Online]. Available: <https://colah.github.io/posts/2014-10-Visualizing-MNIST/>
- [29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [30] S. India. Knn machine learning algorithm. [Online]. Available: https://medium.com/@springboard_ind/knn-machine-learning-algorithm-explained-596d60336076
- [31] A. Navlani, “Knn classification using scikit-learn,” *Data Camp, August*, 2018.
- [32] —, “Decision tree classification in python,” *Data Camp*, 2018.
- [33] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.
- [34] N. Avinash, “Understanding random forests classifiers in python,” 2018. [Online]. Available: [”https://www.datacamp.com/community/tutorials”](https://www.datacamp.com/community/tutorials)
- [35] M. Pathak, “Using xg boost in python,” *Data Camp*, 2019. [Online]. Available: <https://www.datacamp.com/community/tutorials/xgboost-in-python>

- [36] A. Budhiraja. (2015) Drop out in machine learning. [Online]. Available: <https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning>
- [37] L. Bungaro. Artificial neural network- ann. [Online]. Available: <https://medium.com/@luigi.bungaro/artificial-neural-network-ann-dce6514d1501>
- [38] (2019) Activation function for multi-layer neural networks. [Online]. Available: <https://mc.ai/activation-function-for-multi-layer-neural-networks/>
- [39] G. Pad. (2019) Categorical features. [Online]. Available: <https://www.graphpad.com/support/faq/what-is-the-difference-between-ordinal-interval-and-ratio-variables-why-should-i-care/>
- [40] Krishni. An introduction to grid search. [Online]. Available: <https://medium.com/datadriveninvestor/an-introduction-to-grid-search-ff57adcc0998>

Appendix A

Code Snippets

I have used the documentation work and kernel for implementing the code. The links from where we understand the code are:

https://scikit-learn.org/stable/supervised_learning.html#supervised-learning

<https://www.tensorflow.org/tutorials/keras/classification>

<https://www.kaggle.com/echoyosh/kevin-lu-kkl5245-p3>

<https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html>

<https://colah.github.io/posts/2014-10-Visualizing-MNIST/>

<https://xgboost.readthedocs.io/en/latest/python/index.html>

https://catboost.ai/docs/concepts/python-reference_catboostclassifier.html

<https://plotly.com/python/3d-scatter-plots/>

<https://seaborn.pydata.org/api.html>

https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html

<https://www.datacamp.com/community/tutorials/decision-tree-classification-python>

<https://datascience.stackexchange.com/questions/12321/difference-between-fit-and-fit-transform-in-scikit-learn-models>

The code snippets are given below:

Here we are attaching the code snippets. The code which we have implemented for the results. We start with pre-processing results. Pre-processing includes Standard Scaling, One-Hot Encoding and the label Encoder. After this pre-processing, we attached the codes for all Machine Learning Classifiers.

After ML Algorithms, we have implemented and attached Code for Artificial Neural Networks by using Tensor flow Documentation.

```

#we make a list of columns that we want to apply one hot encoding to
print(list(combined_data.columns))

#we deal with categorical features

def one_hot(df, cols):
    """
    @param df pandas DataFrame
    @param cols a list of columns to encode
    @return a DataFrame with one-hot encoding
    """
    for each in cols:
        # makes the one hot encoding in dummies df
        dummies = pd.get_dummies(df[each], prefix=each, drop_first=False)
        #concat dummies df on vertical axis
        df = pd.concat([df, dummies], axis=1)
        #and then finally drop the original data vertically
        df = df.drop(each, 1)
    return df

# Applying one hot encoding to combined data on nominal categorical features
temp = one_hot(c_combined_data,categorical_features)

#now we apply standard scaling

from sklearn.preprocessing import StandardScaler
scale=StandardScaler() #we will later use this function to transform the testing data to the same scale

# Scaling down both train and test data set
#https://datascience.stackexchange.com/questions/12321/difference-between-fit-and-fit-transform-in-scikit-learn-models

print(scale.fit(x_tr))
X_tr=scale.fit_transform(x_tr)

#transforming the testing data so that it has the same mean and variance as we applied in train
X_test=scale.fit_transform(x_test)

#no change in shape can be seen although values have changed
print(X_tr.shape,y_tr.shape,X_test.shape,y_test.shape)

```

FIGURE A.1: *Pre-Processing*

```

from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics #for confusion matrix

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import f1_score, make_scorer, accuracy_score

#k nearest neighbour based approach
mylist=range(1,50)
neighbors = list(filter(lambda x: x % 2 != 0, mylist)) #neighbours is a list of numbers like 1,3,5....49

# Defining the initial parameters. Building KNN classifier
clf_1=KNeighborsClassifier(algorithm="brute")

# Parameters that have to be tuned for optimum results
tuned_parameters = {'n_neighbors': neighbors,"weights":["uniform","distance"],"metric":["cosine","euclidean","manhattan"]}

# Using "macro" F1 for Multi class classification
f1 = make_scorer(f1_score , average='macro')

# Defining the search space with the help of GridSearchCV where cross fold validation=10
model_1 = GridSearchCV(clf_1, tuned_parameters, scoring = f1, cv=10,n_jobs=-1, verbose=10)
model_1.fit(X_tr, y_tr)

best_k=model_1.best_params_["n_neighbors"]
best_distance_type=model_1.best_params_["weights"]
best_distance_metric=model_1.best_params_["metric"]

print(best_k,best_distance_metric,best_distance_type)
print("\nthis is the hyper parameter optimized k value: {} and this is the distance metric used: {}".format(best_k,best_disti

```

FIGURE A.2: *KNN Classifier*

```

#get the naive bayes running
print(numerical_features_train.shape,y_tr.shape,numerical_features_test.shape,y_test.shape)

from sklearn.naive_bayes import GaussianNB
from sklearn import metrics #for confusion matrix

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import f1_score, make_scorer, accuracy_score

# Defining the initial parameters. Building Gaussian Naive-Bayes classifier
clf_2=GaussianNB()
alphas=[x*0.1 for x in range(1,50)]

# Parameters that have to be tuned for optimum results
tuned_parameters = {'var_smoothing': alphas}

# F1 as Macro for Multi class
f1 = make_scorer(f1_score , average='macro')

# Defining the search space with the help of GridSearchCV where cross fold validation=10
model_1 = GridSearchCV(clf_2, tuned_parameters, scoring = f1, cv=10,n_jobs=-1, verbose=10)
model_1.fit(train_df, y_tr)

best_alpha=model_1.best_params_["var_smoothing"]
print(model_1.best_score_)
print(best_alpha)
print("\nthis is the hyper parameter optimized alpha value: {}".format(best_alpha))

```

FIGURE A.3: NB Classifier

```

#get the logistic regression running
print(numerical_features_train.shape,y_tr.shape,numerical_features_test.shape,y_test.shape)

from sklearn.linear_model import LogisticRegression
from sklearn import metrics #for confusion matrix

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import f1_score, make_scorer, accuracy_score

# Defining the initial parameters. Building Bernoulli NB classifier
clf_2=LogisticRegression(random_state=0,class_weight="balanced",solver="saga")

C_vals=np.logspace(-3,3,20) #trying log space really helps

penalties=["l1","l2","elasticnet"]

l1_ratios=[x*0.1 for x in range(1,10)]

# Paramters to get tuned for optimum results
tuned_parameters = {'C': C_vals,"penalty":penalties,"l1_ratio":l1_ratios}

# F1 as macro for Multi-Class classification
f1 = make_scorer(f1_score , average='macro')

# Defining the search space with the help of GridSearchCV where cross fold validation=10
model_1 = GridSearchCV(clf_2, tuned_parameters, scoring = f1, cv=10,n_jobs=-1, verbose=10)
model_1.fit(train_df, y_tr)

best_c=model_1.best_params_["C"]
best_penalty=model_1.best_params_["penalty"]

```

FIGURE A.4: LR Classifier

```

#svm with gridsearch
print(numerical_features_train.shape,y_tr.shape,numerical_features_test.shape,y_test.shape)

from sklearn.svm import SVC
from sklearn import metrics #for confusion matrix

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import f1_score, make_scorer, accuracy_score

# Defing the initial parameters. Building SVM classifier
clf = SVC(class_weight="balanced")

C_vals=np.logspace(-3,3,20) #trying log space really helps

kernels=["linear", "poly", "rbf", "sigmoid"]

# Paramters to get tuned for optimum results
tuned_parameters = {'C': C_vals,"kernel":kernels}

# F1 Score as Macro for Multi-Class classification
f1 = make_scorer(f1_score , average='macro')

# Defining the search space with the help of GridSearchCV where cross fold validation=10
model_1 = GridSearchCV(clf, tuned_parameters, scoring = f1, cv=10,n_jobs=-1, verbose=10)
model_1.fit(X_tr, y_tr)

best_c=model_1.best_params_["C"]
best_penalty=model_1.best_params_["kernel"]
print(model_1.best_score_)
print(best_c,best_penalty,best_ratio)

```

FIGURE A.5: SVM Classifier

```

# Decision trees does not require standard scaling
print(numerical_features_train.shape,y_tr.shape,numerical_features_test.shape,y_test.sh

from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics #for confusion matrix

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import f1_score, make_scorer, accuracy_score

# Defing the initial parameters. Building Decidsion Trees classifier
clf = DecisionTreeClassifier(max_features=122,class_weight="balanced",random_state=1)

max_depths=[x for x in range(2,4)]

#Parameter to get tuned
tuned_parameters = {'max_depth': max_depths}

# F1 as macro for multi-class
f1 = make_scorer(f1_score , average='macro')

# Defining the search space with the help of GridSearchCV where cross fold validation=:
model_1 = GridSearchCV(clf, tuned_parameters, scoring = f1, cv=10,n_jobs=-1, verbose=14)
model_1.fit(X_tr, y_tr)

best_depth=model_1.best_params_["max_depth"]
print(model_1.best_score_)
print(best_depth)

```

FIGURE A.6: DT Classifier

```

#full rowssampling and col sampling by rootn
# Defing the initial parameters. Building Random Forest classifier
clf = RandomForestClassifier(random_state=1)

n_estimators=[x for x in range(150,500,10)]

class_weigting_types=["balanced","balanced_subsample"]

# Parameters to be tuned for optimum results
tuned_parameters = {'n_estimators': n_estimators,"class_weight":class_weigting_types}

# F1-Score is macro for Multi-Class classification
f1 = make_scorer(f1_score , average='macro')

# Defining the search space with the help of GridSearchCV where cross fold validation=10
model_1 = GridSearchCV(clf, tuned_parameters, scoring = f1, cv=10,n_jobs=-1, verbose=10)
model_1.fit(X_tr, y_tr)

best_estimator_counts=model_1.best_params_["n_estimators"]
best_weighing_scheme=model_1.best_params_["class_weight"]

print(model_1.best_score_)
print(best_estimator_counts)
print("\nthis is the hyper parameter optimized number of trees value: {}".format(best_estimator_counts))

y_pred=model_1.predict(X_test)
acc=accuracy_score(y_test, y_pred)
print("\nthis is my models accuracy :{}".format(acc))

#plotting the confusion matrix from the data
confusion_matrix= metrics.confusion_matrix(y_test,y_pred)
(tn, fp, fn, tp)=(confusion_matrix[0][0],confusion_matrix[0][1],confusion_matrix[1][0],confusion_matrix[1][1])

```

FIGURE A.7: RF Classifier

```

#boosting based algotihm-GBDT, but we will use xgboost becuase it has row sampling and column sampling
#bagging and boosting

#Let's do a normal implementation of xgboost

# import XGBClassifier
from xgboost import XGBClassifier

# instantiate the classifier
xgb_clf = XGBClassifier()

# fit the classifier to the training data
xgb_clf.fit(X_tr, y_tr)

y_pred=xgb_clf.predict(X_test)
acc=accuracy_score(y_test, y_pred)
print("\nthis is my models accuracy :{}".format(acc))

#plotting the confusion matrix from the data
confusion_matrix= metrics.confusion_matrix(y_test,y_pred)
(tn, fp, fn, tp)=(confusion_matrix[0][0],confusion_matrix[0][1],confusion_matrix[1][0],confusion_matrix[1][1])
print("XG Boost confusion matrix:")
print("\ntn:%d\nfp:%d\nfn:%d\nntp:%d\n"%(tn, fp, fn, tp))

p = sns.heatmap(pd.DataFrame(confusion_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
plt.title('Confusion matrix on test data')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')

print("="*20)
print("The F1 score is the harmonic mean between the precision and recall... ")
print("my f1 score on test: ",f1_score(y_test,y_pred ,average="macro"))
print("my accuracy score on test: ",accuracy_score(y_test,y_pred))
print("="*20)

```

FIGURE A.8: XGB Classifier

```

from sklearn import metrics #for confusion matrix

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import f1_score, make_scorer, accuracy_score

from catboost import CatBoostClassifier

# Direct model implementation
model=CatBoostClassifier()
model.fit(X_tr,y_tr,cat_features=categorical_feat,verbose=10)

y_pred=model.predict(X_test)
acc=accuracy_score(y_test, y_pred)
print("\nthis is my models accuracy :{}".format(acc))

#plotting the confusion matrix from the data
confusion_matrix= metrics.confusion_matrix(y_test,y_pred)
(tn, fp, fn, tp)=(confusion_matrix[0][0],confusion_matrix[0][1],confusion_matrix[1][0],confusion_matrix[1][1])
print("catboost confusion matrix using brute:")
print("\ntn:%d\nfp:%d\nfn:%d\nntp:%d\n"%(tn, fp, fn, tp))

p = sns.heatmap(pd.DataFrame(confusion_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
plt.title('Confusion matrix on test data')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')

print("\n\n")
print("The F1 score is the harmonic mean between the precision and recall hence we really don't need to calcul

```

FIGURE A.9: CATB Classifier

Artificial Neural Networks with the help of Tensor Flow

```

[5]: from IPython.core.display import display, HTML
      display(HTML("""<a href="https://www.tensorflow.org/tutorials/keras/classification">Tensor Flow Documentation</a>"""))

Tensor Flow Documentation

[ ]: # Tensorflow ANN
      # https://www.tensorflow.org/tutorials/keras/classification

```

FIGURE A.10: ANN Classifier

```

# Loading data from test and train files from google drive
from joblib import load
X_tr=load("/content/drive/My Drive/Data set/X_tr.joblib")
X_test=load("/content/drive/My Drive/Data set/X_test.joblib")
y_tr=load("/content/drive/My Drive/Data set/y_tr.joblib")
y_test=load("/content/drive/My Drive/Data set/y_test.joblib")

print(X_tr.shape,y_tr.shape,X_test.shape,y_test.shape)

(125973, 122) (125973,) (22544, 122) (22544,)

```

FIGURE A.11: Loading the data


```

from keras.utils import np_utils

# converting class labels to one hot encoded features
y_tr=np_utils.to_categorical(y_tr)
y_test=np_utils.to_categorical(y_test)
print(y_tr.shape,y_test.shape)

# Getting the input and output from the data set.
input_dimensions=X_tr.shape[1]
output_dimension=y_tr.shape[1]

(125973, 5) (22544, 5)

```

FIGURE A.12: *One-Hot*

```

# Optuna citation - https://optuna.org/#code_examples
import keras
import tensorflow as tf
from tensorflow.keras.layers import Activation, Dropout, Dense, BatchNormalization
import optuna

# Defining an objective function which needs to be maximized.
def objective(trial):

    # Hard coded values (not changing with respect to code)
    batch_size=128
    epochs=15
    batch_normalization_after_layer=2

    # define search space; number of hidden layers, dropout rate, activation function.
    num_hidden_layers=trial.suggest_int("number_of_hidden_layers",1,10)
    drop_out_rate=trial.suggest_uniform("dr_rate",0,1)
    activation_list=["tanh","sigmoid","swish","relu"]
    activation_type=trial.suggest_categorical("activation_type",activation_list)
    print(num_hidden_layers,drop_out_rate,activation_type)

    # Initializing the model
    model=tf.keras.Sequential()

    # Here, 1 is the hidden layer. So, we are starting from the 1st hidden layer
    # Because 0th layer is the input of dimensions i.e. number of columns
    for layer in range(1,num_hidden_layers+1):
        # We are going to add number of hidden units.
        num_hidden_units=trial.suggest_int("num_hidden_units_{}".format(layer),2,96)
        if(layer==1):
            model.add(Dense(num_hidden_units,activation=activation_type,input_shape=(input_dimensions,)))
        else:
            model.add(Dense(num_hidden_units,activation=activation_type))
        # Adding dropout for randomly discarding neurons
        model.add(Dropout(drop_out_rate))

```

FIGURE A.13: *Objective Function*

```

# Batch normalization after 3rd layer
if(layer%batch_normalization_after_layer+1==0):
    model.add(BatchNormalization())

# For out put we use SOFTMAX
# Softmax by default which provides probabilities
# Softmax function inbuilt in tensorflow
model.add(Dense(output_dimension,activation="softmax"))

# Compilation of the model
# Choosing optimiser "adam" whcih is better than "Gradient Descent".
model.compile(loss='categorical_crossentropy', optimizer="adam", metrics=[f1,"accuracy"])

# Fitting the model
# Returning from F1-Score
model.fit(X_tr,y_tr,batch_size=batch_size,epochs=epochs,validation_data=(X_test,y_test))
results=model.evaluate(X_test,y_test)
print("="*20)
print("this is my loss:{}".format(results[0]))
print("this is my accuracy:{}".format(results[1]))
print("="*20)

return results[2]

# Creating a study object and optimizing the objective function.
study = optuna.create_study(direction='maximize',study_name="NN_tuning",storage="sqlite:///nn_net.db",load_if_exists=True)
# n_trials means number of trials
study.optimize(objective, n_trials=100)

[I 2020-08-03 01:26:25,715] Using an existing study with name 'NN_tuning' instead of creating a new one.
9 0.032642182568550754 tanh
Epoch 1/15
985/985 [=====] - 8s 8ms/step - loss: 0.2279 - f1: 0.9272 - accuracy: 0.9305 - val_loss: 1.4800 - val_f
1: 0.7285 - val_accuracy: 0.7278
Epoch 2/15

```

FIGURE A.14: *Fitting and running the model*

```

# Again Loading the object from the nn_net file and updating it again.
study = optuna.create_study(direction='maximize',study_name="NN_tuning",storage="sqlite:///nn_net.db",load_if_exists=True)
import pandas as pd
print("best_params",study.best_params)
print("f1--",study.best_value)
print("best trial--",study.best_trial)

[I 2020-08-03 03:45:42,163] Using an existing study with name 'NN_tuning' instead of creating a new one.
best_params {'activation_type': 'tanh', 'dr_rate': 0.03587723263680438, 'num_hidden_units_1': 29, 'num_hidden_units_2': 49, 'num_
hidden_units_3': 62, 'num_hidden_units_4': 77, 'num_hidden_units_5': 20, 'num_hidden_units_6': 70, 'number_of_hidden_layers': 6}
f1-- 0.7686746120452881

```

Appendix B

Wireframe Models