DETECTION OF ATTACKS USING RFA

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree in Computer Science and Engineering

By

RAMANA CHANDRAKIRAN REDDY (Reg. No. 39110835)

RAMAKURTHI PRAVEEN KUMAR (Reg. No. 39110834)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600 119

APRIL - 2023

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **RAMANA CHANDRAKIRAN REDDY (Reg.No. 39110835), RAMAKURTHI PRAVEEN KUMAR (Reg. No. 39110834)** who carried out the project entitled **"DETECTION OF ATTACKS USING RFA"** under our supervision from January 2023 to April 2023.

Internal Guide

Dr. E. MURALI., B.E., M.TECH., Ph.D.,

Head of the Department

Dr. L. LAKSHMANAN M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner **External Examiner**

DECLARATION

I **RAMANA CHANDRAKIRAN REDDY (Reg.No.39110835), RAMAKURTHI PRAVEEN KUMAR (Reg.No.39110834)** hereby declare that the Project Report entitled **"DETECTION OF ATTACKS USING RFA"** done by me under the guidance of **Dr.E.Murali., B.E., M.Tech., Ph.D** is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering / Technology degree in **Computer Science and Engineering**.

DATE:

PLACE: **SIGNATURE OF THE CANDIDATE**

ACKNOWLEDGEMENT

We are pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA, M.E., Ph.D., Dean, School of Computing,** and **Dr. L. LAKSHMANAN M.E., Ph.D.,** Head of the Department, Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. E. Murali., B.E., M.Tech., Ph.D.,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the department of **COMPUTER SCIENCE AND ENGINEERING** who were helpful in many ways for the completion of the project.

ABSTRACT

An Intrusion Detection System (IDS) is a software application or device that monitors the system or activities of network for policy violations or malicious activities and generates reports to the management system. A number of systems may try to prevent an intrusion attempt but this is neither required nor expected of a monitoring system. The main focus of Intrusion detection and prevention systems (IDPS) is to identify the possible incidents, logging information about them and in report attempts. In addition, organizations use IDPS for other purposes, like identifying problems with security policies, deterring individuals and documenting existing threats from infringing security policies. IDPS have become an essential addition to the security infrastructure of nearly every organization. Various methods can be used to detect intrusions but each one is specific to a specific method. The main goal of an intrusion detection system is to detect the attacks efficiently. Furthermore, it is equally important to detect attacks at a beginning stage in order to reduce their impacts. This research work proposed a new approach called outlier detection where, the anomaly dataset is measured by the Neighborhood Outlier Factor (NOF). Here, trained model consists of big datasets with distributed storage environment for improving the performance of Intrusion Detection system. The experimental results proved that the proposed approach identifies the anomalies very effectively than any other approaches.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Approaches for intrusion detection can be broadly divided into two types: misuse detection and anomaly detection. In misuse detection system, all known types of attacks (intrusions) can be detected by looking into the predefined intrusion patterns in system audit traffic. In case of anomaly detection, the system first learns a normal activity profile and then flags all system events that do not match with the already established profile. The main advantage of the misuse detection is its capability for high detection rate with a difficulty in finding the new or unforeseen attacks.

1.1 BACKGROUND

With the world trending towards being steadily reliant on computers and automation, it is a challenge to build networks and systems secure for everyday use. The number of security threats to organizations is increasing exponentially with the growth of online markets and services. They are numerous solutions to combat network security threats. Intrusion detection systems are placed alongside firewalls in networks to combat security threats. They scan the network for all the incoming and outgoing traffics and analyze the signature of the packets to detect whether they are malicious or normal. Machine learning is used to help the system learn the signature of known attacks and profile normal network packets.

1.2 MOTIVATION

It is difficult or almost impossible to develop an intrusion detection system with 100 percent success rate. Most systems today have a lot of security flaws. Not all kinds of intrusions are known. Also, hackers are figuring out new ways into the networks using machine learning techniques. Quick detection of these attacks will help to identify possible intruders and limit damage effected. So, developing an efficient and accurate intrusion detection system will help to reduce network security threats.

1.3 OBJECTIVE

There are several intrusion detection systems in use today. Researchers are trying to develop systems which use machine learning techniques to identify the signature of attackers. This proposed system aims to find a suitable novel technique to be used as a backend to such a system.

1.4 COLLECTING DATA SET

The KDD data set is a well-known benchmark in the research of Intrusion Detection techniques. A lot of work is going on for the improvement of intrusion detection strategies while the research on the data used for training and testing the detection model is equally of prime concern because better data quality can improve offline intrusion detection. This paper presents the analysis of KDD data set with respect to four classes which are Basic, Content, Traffic and Host in which all data attributes can be categorized.

1.5 EVALUATING THE RESULT

Evaluation will be the final part of this project. For each scientific project, the final result should be teste d and evaluated if that is acceptable. The result will be automatically shown in the end of the program execution. For every machine learning algorithm, exceptions will always exist. In order to find the best result, result analyzing is necessary.

CHAPTER 2

AIM AND SCOPE OF THE PRESENT INVESTIGATION

2.1 MACHINE LEARNING SCOPE

Machine learning as a very likely approach to achieve human-computer integration and can be applied in many computer fields. Machine learning is not a typical method as it contains many different computer algorithms. Yu Yang algorithms aim to solve different machine learning tasks. At last, all the algorithms can help the computer to act more like a human. Machine learning is already applied in many fields, for instance, pattern recognition, Artificial Intelligence, computer vision, data mining, text categorization and so on. Machine learning gives a new way to develop the intelligence of the machines. It also becomes an easier way to help people to analyse data from huge data sets. A learning method is a complicated topic which has many different kinds of forms. Everyone has different methods to study, so does the machine. We can categorize various machine learning systems by different conditions. In general, we can separate learning problems in two main categories: supervised learning and unsupervised learning.

What is Machine Learning?

Machine Learning is a system of computer algorithms that can learn from example through self-improvement without being explicitly coded by a programmer. Machine learning is a part of artificial Intelligence which combines data with statistical tools to predict an output which can be used to make actionable insights.

The breakthrough comes with the idea that a machine can singularly learn from the data (i.e., example) to produce accurate results. Machine learning is closely related to data mining and Bayesian predictive modeling. The machine receives data as input and uses an algorithm to formulate answers.

A typical machine learning tasks are to provide a recommendation. For those who have a Netflix account, all recommendations of movies or series are based on the user's historical data. Tech companies are using unsupervised learning to improve the user experience with personalizing recommendation.

Machine learning is also used for a variety of tasks like fraud detection, predictive maintenance, portfolio optimization, automatize task and so on.

Machine learning is supposed to overcome this issue. The machine learns how the

input and output data are correlated and it writes a rule. The programmers do not need to write new rules each time there is new data. The algorithms adapt in response to new data and experiences to improve efficacy over time.

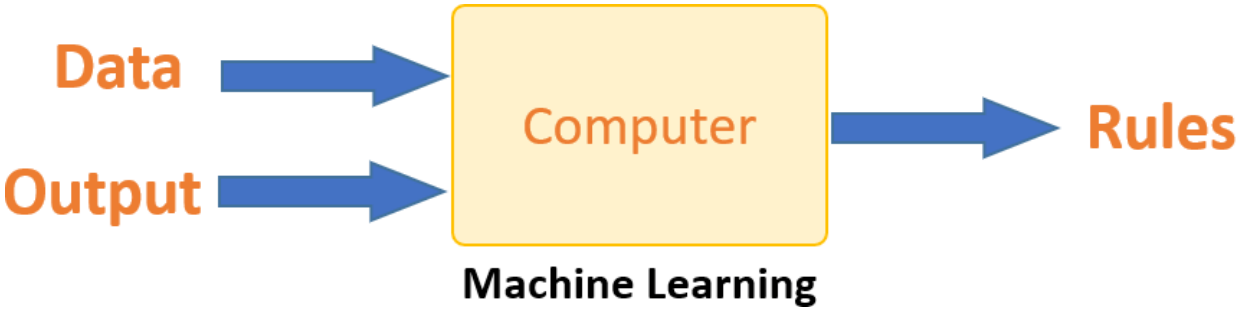


Fig. 2.1 Machine Learning

How does Machine Learning Work?

Now in this Machine learning basics for beginner's tutorial, we will learn how Machine Learning (ML) works:

Machine learning is the brain where all the learning takes place. The way the machine learns is similar to the human being. Humans learn from experience. The more we know, the more easily we can predict. By analogy, when we face an unknown situation, the likelihood of success is lower than the known situation. Machines are trained the same. To make an accurate prediction, the machine sees an example. When we give the machine a similar example, it can figure out the outcome. However, like a human, if it's feed a previously unseen example, the machine has difficulties to predict.

The core objective of machine learning is the **learning** and **inference**. First of all, the machine learns through the discovery of patterns. This discovery is made thanks to the **data**. One crucial part of the data scientist is to choose carefully which data to provide to the machine. The list of attributes used to solve a problem is called a **feature vector.** You can think of a feature vector as a subset of data that is used to tackle a problem.

The machine uses some fancy algorithms to simplify the reality and transform this discovery into a **model**. Therefore, the learning stage is used to describe the data and summarize it into a model.

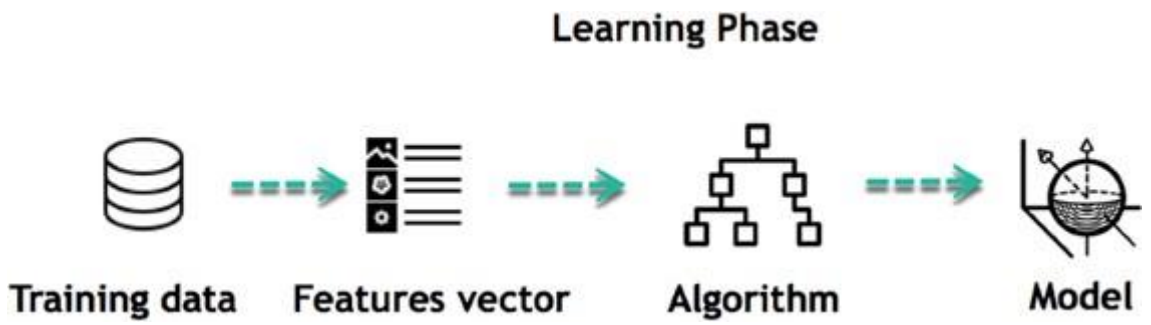


Fig. 2.1 Learning Phase

For instance, the machine is trying to understand the relationship between the wage of an individual and the likelihood to go to a fancy restaurant. It turns out the machine finds a positive relationship between wage and going to a high-end restaurant: This is the model

Inferring

When the model is built, it is possible to test how powerful it is on never-seen-before data. The new data are transformed into a features vector, go through the model an

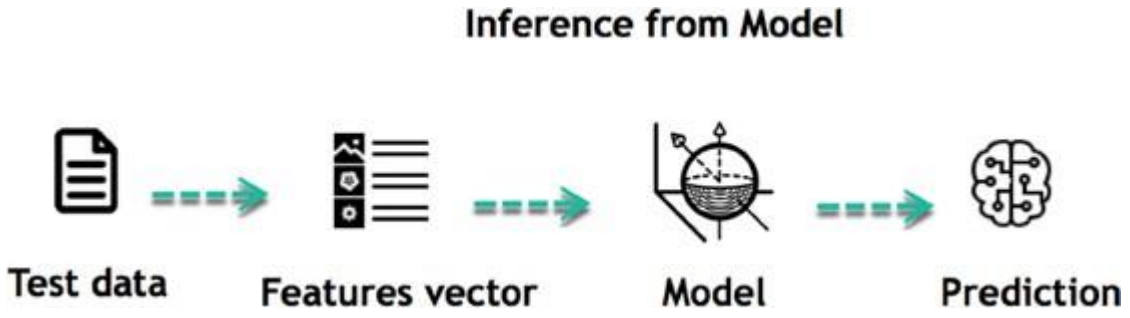


Fig. 2.1 Inference from Model

give a prediction. This is all the beautiful part of machine learning. There is no need to update the rules or train again the model. You can use the model previously trained to make inference on new data.

The life of Machine Learning programs is straightforward and can be summarized in the following points:

- Define a question

- Collect data
- Visualize data
- Train algorithm
- Test the Algorithm
- Collect feedback
- Refine the algorithm
- Loop 4-7 until the results are satisfying
- Use the model to make a prediction

Once the algorithm gets good at drawing the right conclusions, it applies that knowledge to new sets of data.

2.1.1 CLASSIFICATION OF MACHINE LEARNING

Now in this Machine learning tutorial for beginners, we will learn where Machine Learning (ML) algorithms are used:
Machine learning Algorithms Machine learning can be grouped
into two broad learning tasks:

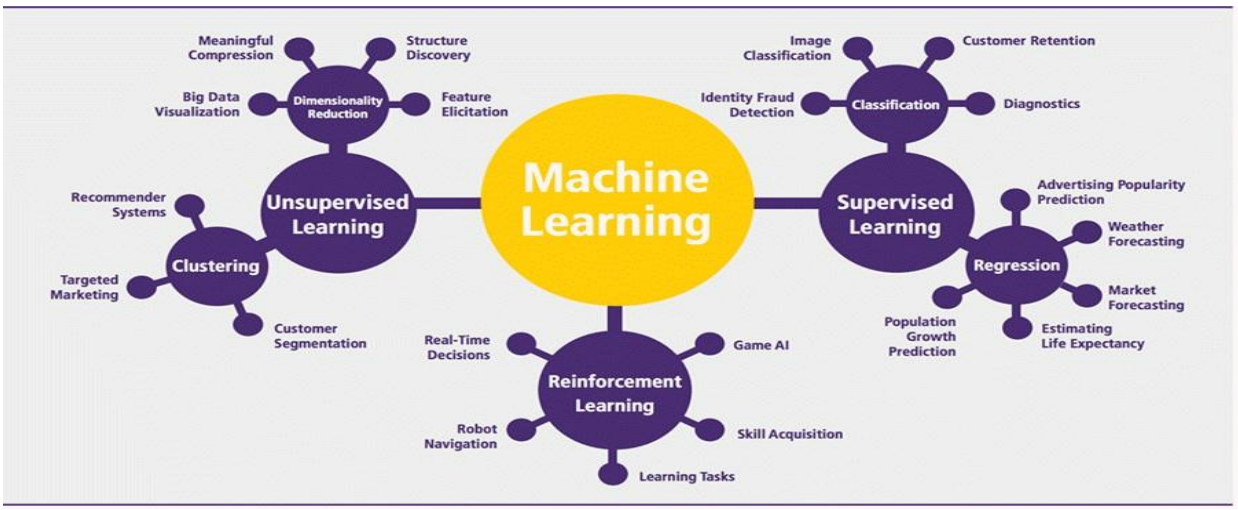


Fig. 2.1.1 Classification Of Machine Learning

- Supervised Learning
- Unsupervised learning

2.1.2 SUPERVISED MACHINE LEARNING

An algorithm uses training data and feedback from humans to learn the relationship of given inputs to a given output. For instance, a practitioner can use marketing expense and weather forecast as input data to predict the sales of cans.

You can use supervised learning when the output data is known. The algorithm will predict new data.

There are two categories of supervised learning:

- Classification task
- Regression task

Classification

Imagine you want to predict the gender of a customer for a commercial. You will start gathering data on the height, weight, job, salary, purchasing basket, etc. from your customer database. You know the gender of each of your customer, it can only be male or female. The objective of the classifier will be to assign a probability of being a male or a female (i.e., the label) based on the information (i.e., features you have collected). When the model learned how to recognize male or female, you can use new data to make a prediction. For instance, you just got new information from an unknown customer, and you want to know if it is a male or female. If the classifier predicts male = 70%, it means the algorithm is sure at 70% that this customer is a male, and 30% it is a female.

The label can be of two or more classes. The above Machine learning example has only two classes, but if a classifier needs to predict object, it has dozens of classes (e.g., glass, table, shoes, etc. each object represents a class)

Regression

When the output is a continuous value, the task is a regression. For instance, a financial analyst may need to forecast the value of a stock based on a range of feature like equity, previous stock performances, macro-economic index. The system will be trained to estimate the price of the stocks with the lowest possible error.

- Linear regression
- Logistic regression
- Decision tree
- Naive Bayes
- Support vector machine

- Random forest

2.1 UNSUPERVISED LEARNING

In unsupervised learning, an algorithm explores input data without being given an explicit output variable (e.g., explores customer demographic data to identify patterns) You can use it when you do not know how to classify the data, and you want the algorithm to find patterns and classify the data for you

- K-means clustering
- Gaussian mixture model
- Hierarchical clustering
- Recommender system
- PCA/T-SNE

2.2 DECISION TREE SCOPE

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy). Leaf node (e.g., Play) represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data. The core algorithm for building decision trees called ID3 by J. R. Quinlan which employs a top-down, greedy search through the space of possible branches with no backtracking. ID3 uses *Entropy* and *Information Gain* to construct a decision tree. In Zero R model there is no predictor, in One R model we try to find the single best predictor, naive Bayesian includes all predictors using Bayes' rule and the independence assumptions between predictors but decision tree includes all predictors with the dependence assumptions between predictors. The core algorithm for building decision trees called ID3 by J. R. Quinlan which employs a top-down, greedy search through the space of possible branches with no backtracking. ID3 uses Entropy and Information Gain to construct a decision tree. In Zero R model there is no predictor, in One R model we try to find the single best predictor, naive Bayesian

includes all predictors using Bayes' rule and the independence assumptions between predictors but decision tree includes all predictors with the dependence assumptions between predictor.

2.2 GENETIC ALGORITHM SCOPE

Genetic Algorithms (GAs) are adaptive heuristic search algorithms that belong to the larger a part of evolutionary algorithms. Genetic algorithms are supported the ideas of survival and genetics. These are intelligent exploitation of random search given historical data to direct the search into the region of higher performance in solution space. It is not technically feasible to create a system which has no vulnerabilities. So, intrusion detection has become a crucial area of research. If an intrusion slightly deviates from the already defined pattern then it will consider as normal and if normal behavior slightly changes it may be treated as intrusion. Intrusion detection system offer many techniques which recognize and differentiate between normal and intrusion data. Genetic algorithm are often wont to tune the membership function of IDS. Genetic Algorithm may be a family of computational model supported principles of evolution and survival. GA converts the matter into a model by using chromosomes like arrangement and evolves the chromosomes using selection, recombination and mutation operator.

2.3 INTRUSION DETECTION SYSTEM SCOPE

IDS are generally deployed with the purpose to monitor and analyze user and system activity, audit system configurations and vulnerabilities, assess the integrity of any critical system and data files, perform statistical analysis of activity patterns based on the matching to known attacks, detect abnormal activity and audit operating systems.

2.4 EXISTING SYSTEM

Today network has become an essential part of public infrastructures with the inception of public and private cloud computing. The traditional networking approach has become too complex. This complexity has resulted in a barrier for creating new and innovative services within a single data center, difficulties in interconnecting data centers, interconnection within enterprises, and bigger barrier in the continued growth of the Internet in general.

2.5 PROBLEM STATEMENT

To distinguish the activities of the network traffic that the intrusion and normal is very difficult and to need much time consuming. An analyst must review all the data that large and wide to find the sequence of intrusion on the network connection. It needs a way that can detect network intrusion to reflect the current network traffics. Combination of IDS and firewall so-called the IPS, so that besides detecting the existence of intrusion also can execute by doing deny of intrusion as prevention**.**

CHAPTER 3

EXPERIMENTAL OR METHODS AND ALGORITHMS USED

3.1 DECISION TREE MODELLING

In computational complexity the decision tree model is the model of computation in which an algorithm is basically a decision tree, i.e., a sequence of branching operations based on comparisons of some quantities, the comparisons being assigned unit computational cost. The branching operations are called "tests" or "queries". In this setting the algorithm in question may be viewed as a computation of a Boolean function {\display style f:\{0,1\}^{n}\right arrow \{0,1\}} f:\{0,1\}^{n}\rightarrow \{0,1\} where the input is a series of queries and the output is the final decision. Each query may be dependent on previous queries. Several variants of decision tree models have been introduced, depending on the complexity of the operations allowed in the computation of a single comparison and the way of branching. Decision trees models are instrumental in establishing lower bounds for complexity theory for certain classes of computational problems and algorithms. The computational complexity of a problem or an algorithm expressed in terms of the decision tree model is called its decision tree complexity or query complexity. A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements. Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal but are also a popular tool in machine learning.

A decision tree consists of three types of nodes:

Decision nodes – typically represented by squares

Chance nodes – typically represented by circles

End nodes – typically represented by triangles

Decision trees are commonly used in operations research and operations management. If, in practice, decisions must be taken online with no recall under incomplete knowledge, a decision tree should be paralleled by a probability model as a best choice model or online selection model algorithm. Another use of decision tree is as a descriptive means for calculating conditional probabilities. Decision trees, influence diagrams, utility functions, and other decision analysis tools and methods are taught to undergraduate students in schools of business, health economics, and public

health, and are examples of operations research or management science methods.

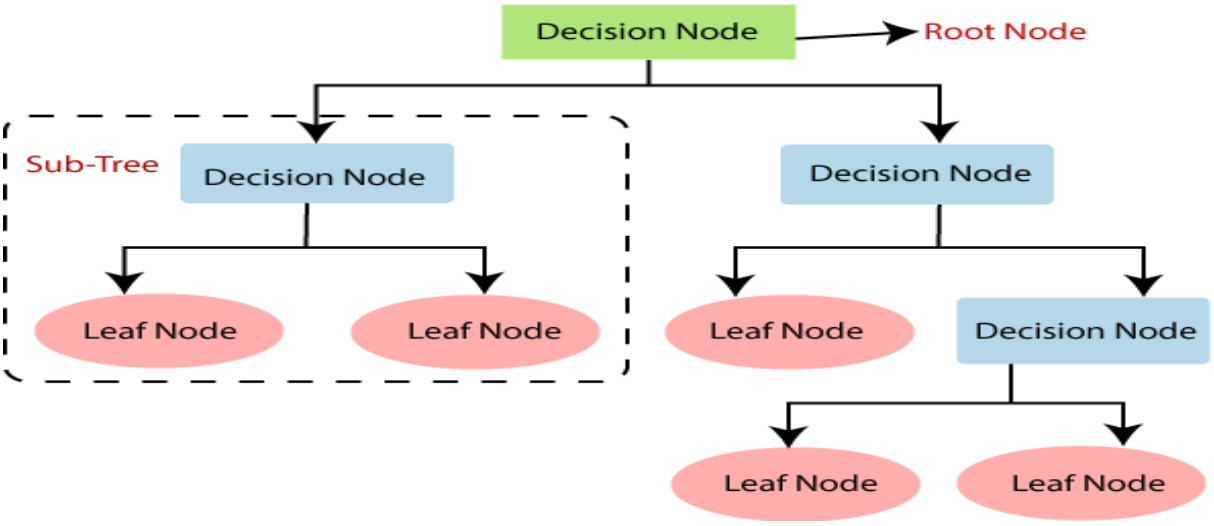


Fig. 3.1 Decision Tree

3.2 GENETIC ALGORITHM

Genetic Algorithm (GA) is a search-based optimization technique based on the principles of Genetics and Natural Selection. It is frequently used to find optimal or near-optimal solutions to difficult problems which otherwise would take a lifetime to solve. It is frequently used to solve optimization problems, in research, and in machine learning. In GAs, we have a **pool or a population of possible solutions** to the given problem. These solutions then undergo recombination and mutation (like in natural genetics), producing new children, and the process is repeated over various generations. Each individual (or candidate solution) is assigned a fitness value (based on its objective function value) and the fitter individuals are given a higher chance to mate and yield more “fitter” individuals. This is in line with the Darwinian Theory of “Survival of the Fittest”. In this way we keep “evolving” better individuals or solutions over generations, till we reach a stopping criterion. Genetic Algorithms are sufficiently randomized in nature, but they perform much better than random local search (in which we just try various random solutions, keeping track of the best so far), as they exploit historical information as well.

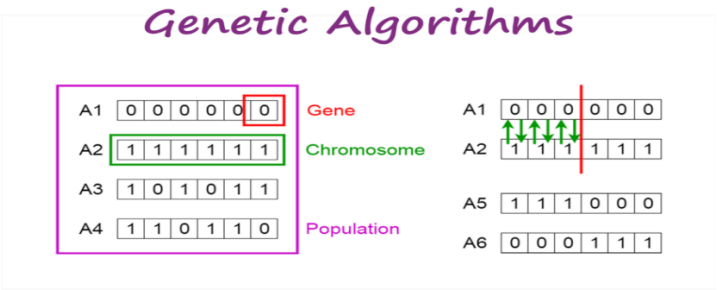


Fig. 3.2 Genetic Algorithms

Notion of Natural Selection

The process of natural selection starts with the selection of fittest individuals from a population. They produce offspring which inherit the characteristics of the parents and will be added to the next generation. If parents have better fitness, their offspring will be better than parents and have a better chance at surviving. This process keeps on iterating and at the end, a generation with the fittest individuals will be found.
This notion can be applied for a search problem. We consider a set of solutions for a problem and select the set of best ones out of them.
Five phases are considered in a genetic algorithm.

1. Initial population
2. Fitness function
3. Selection
4. Crossover
5. Mutation

Initial Population

The process begins with a set of individuals which is called a **Population**. Each individual is a solution to the problem you want to solve. An individual is characterized by a set of parameters (variables) known as **Genes**. Genes are joined into a string to form a **Chromosome** (solution). In a genetic algorithm, the set of genes of an individual is represented using a string, in terms of an alphabet. Usually, binary values are used (string of 1s and 0s). We say that we encode the genes in a chromosome.

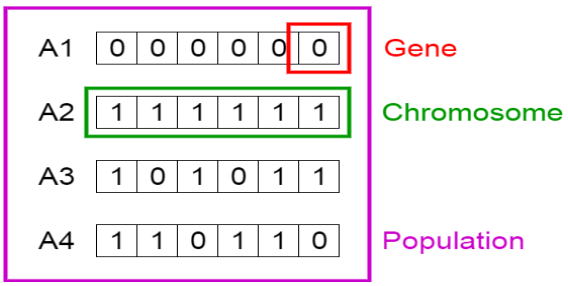


Fig. 3.2 Initial Population

Fitness Function

The fitness function determines how fit an individual is (the ability of an individual to compete with other individuals). It gives a fitness score to each individual. The probability that an individual will be selected for reproduction is based on its fitness score.

Selection

The idea of selection phase is to select the fittest individuals and let them pass their genes to the next generation. Two pairs of individuals (parents) are selected based on their fitness scores. Individuals with high fitness have more chance to be selected for reproduction.

Crossover

Crossover is the most significant phase in a genetic algorithm. For each pair of parents to be mated, a crossover point is chosen at random from within the genes.

For example, consider the crossover point to be 3 as shown below.

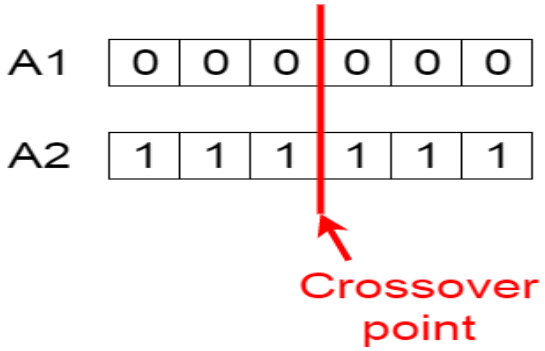


Fig. 3.2 Crossover

Crossover point

Offspring are created by exchanging the genes of parents among themselves until the crossover point is reached.

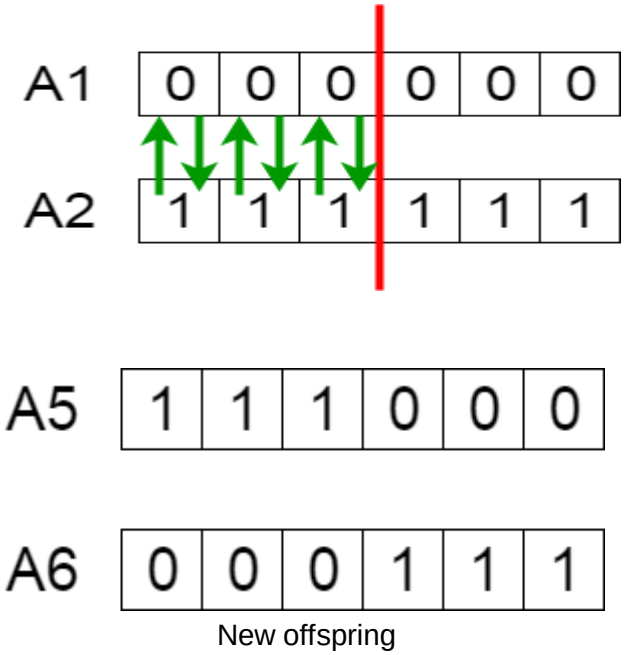
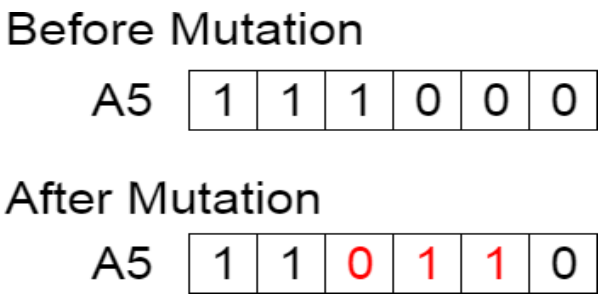


Fig. 3.2 Crossover Point

Mutation

In certain new offspring formed, some of their genes can be subjected to a **mutation** with a low random probability. This implies that some of the bits in the bit string can be flipped.

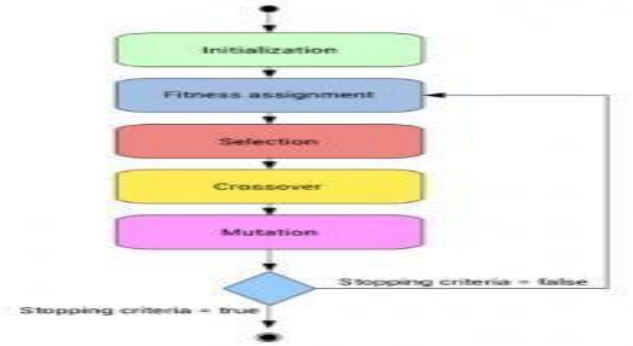


Mutation: Before and After

Fig. 3.2 Mutation

Mutation occurs to maintain diversity within the population and prevent premature convergence.

Fig. 3.2 Workflow



3.3 INTRUSION DETECTION SYSTEM

An intrusion detection system (IDS) is a device or software application that monitors a network for malicious activity or policy violations. Any malicious activity or violation is typically reported or collected centrally using a security information and event management system. Some IDS's are capable of responding to detected intrusion upon discovery. These are classified as intrusion prevention systems (IPS).

3.3.1 IDS Detection Types

There is a wide array of IDS, ranging from antivirus software to tiered monitoring systems that follow the traffic of an entire network. The most common classifications are:

- Network intrusion detection systems (NIDS): A system that analyzes incoming network traffic.
- Host-based intrusion detection systems (HIDS): A system that monitors

important operating system files.

There is also subset of IDS types. The most common variants are based on signature detection and anomaly detection.

- Signature-based: Signature-based IDS detects possible threats by looking for specific patterns, such as byte sequences in network traffic, or known malicious instruction sequences used by malware. This terminology originates from antivirus software, which refers to these detected patterns as signatures.

  Although signature-based IDS can easily detect known attacks, it is impossible to detect new attacks, for which no pattern is available.
- Anomaly-based: a newer technology designed to detect and adapt to unknown attacks, primarily due to the explosion of malware. This detection method uses machine learning to create a defined model of trustworthy activity, and then compare new behavior against this trust model. While this approach enables the detection of previously unknown attacks, it can suffer from false positives: previously unknown legitimate activity can accidentally be classified as malicious.

IDS Usage in Networks

When placed at a strategic point or points within a network to monitor traffic to and from all devices on the network, an IDS will perform an analysis of passing traffic, and match the traffic that is passed on the subnets to the library of known attacks. Once an attack is identified, or abnormal behavior is sensed, the alert can be sent to the administrator.

3.3.2 Evasion Techniques

Being aware of the techniques available to cyber criminals who are trying to breach a secure network can help IT departments understand how IDS systems can be tricked into not missing actionable threats:

- Fragmentation: Sending fragmented packets allow the attacker to stay under the radar, bypassing the detection system's ability to detect the attack

signature.

- Avoiding defaults: A port utilized by a protocol does not always provide an indication to the protocol that's being transported. If an attacker had reconfigured it to use a different port, the IDS may not be able to detect the presence of a trojan.
- Coordinated, low-bandwidth attacks: coordinating a scan among numerous attackers, or even allocating various ports or hosts to different attackers. This makes it difficult for the IDS to correlate the captured packets and deduce that a network scan is in progress.
- Address spoofing/proxying: attackers can obscure the source of the attack by using poorly secured or incorrectly configured proxy servers to bounce an attack. If the source is spoofed and bounced by a server, it makes it very difficult to detect.
- Pattern change evasion: IDS rely on pattern matching to detect attacks. By making slight adjust to the attack architecture, detection can be avoided.

3.3 SYSTEM ARCHITECTURE

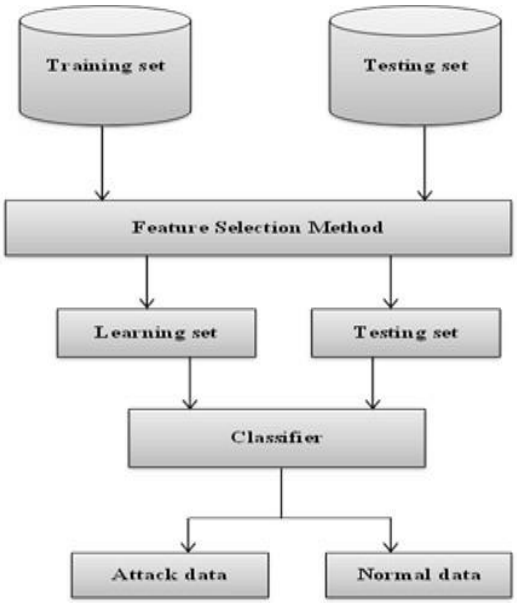


Fig. 3.3 System Architecture

Software:

- Front End : Python
- Back End : SQL (5.0)

Hardware:

- I3 Processor
- 4 Gb Ram
- Windows 10 Operating System
- Internet Connection.

CHAPTER 4

IMPLEMENTATION DETAILS

4.1 Machine Learning Models

The machine learning model is nothing but a piece of code; an engineer or data scientist makes it smart through training with data. So, if you give garbage to the model, you will get garbage in return, i.e. the trained model will provide false or wrong prediction.

We can define the machine learning workflow in below.

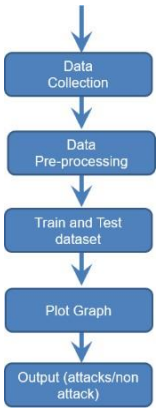


Fig. 4.1 Machine Learning Models

1. Data Collection
2. Data pre-processing
3. Future Extraction
4. Model Training
5. Testing Model
6. Evaluation
7. Prediction

4.1.1. Data Collection

Collecting data allows you to capture a record of past events so that we can use data analysis to find recurring patterns.

KDD datasets:

The KDD data set is a well-known benchmark in the research of Intrusion Detection techniques. A lot of work is going on for the improvement of intrusion detection strategies while the research on the data used for training and testing the detection model is equally of prime concern because better data quality can improve offline intrusion detection. This paper presents the analysis of KDD data set with respect to four classes which are Basic, Content, Traffic and Host in which all data attributes can be categorized.

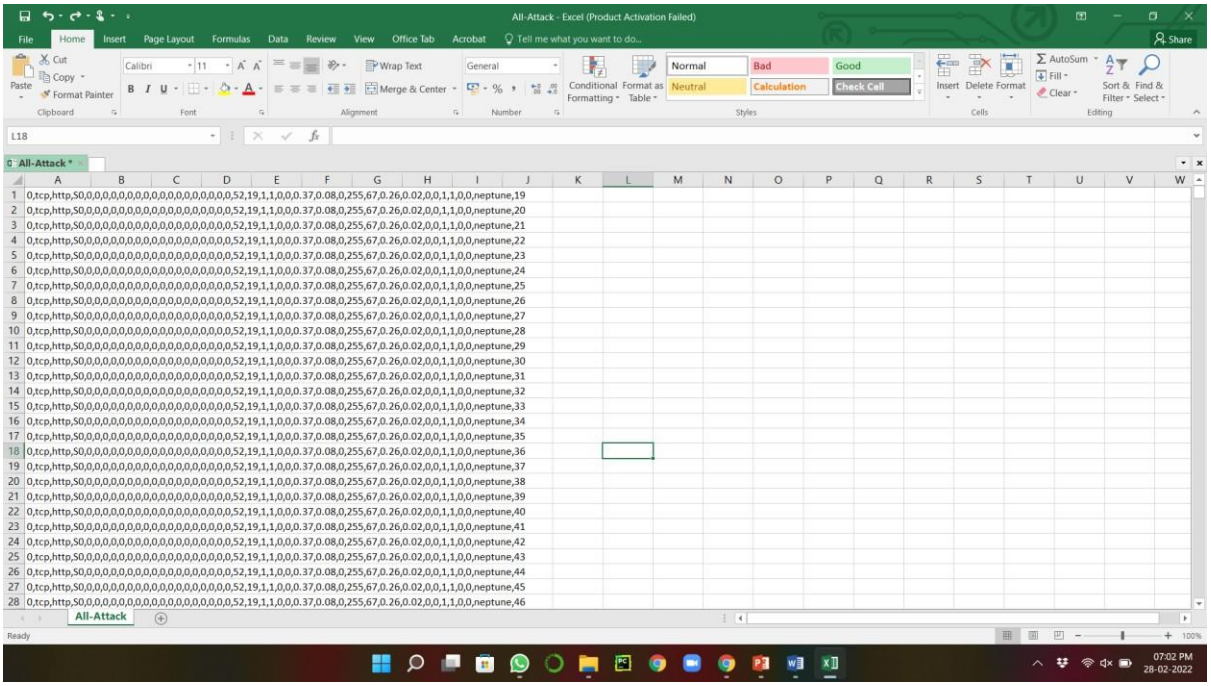


Fig. 4.1.1 Data Collection

4.1.2 Data Pre-Processing

Data pre-processing is a process of cleaning the raw data i.e. the data is collected in the real world and is converted to a clean data set. In other words, whenever the data is gathered from different sources it is collected in a raw format and this data isn't feasible for the analysis. Therefore, certain steps are executed to convert the data into a small clean data set, this part of the process is called as data pre-processing.

4.1.3 Feature Extraction:

This is done to reduce the number of attributes in the dataset hence providing advantages like speeding up the training and accuracy improvements.

4.1.4 Model training:

A training model is a dataset that is used to train an ML algorithm. It consists of the sample output data and the corresponding sets of input data that have an influence on the output.

4.1.5 Testing model:

In this module we test the trained machine learning model using the test dataset. Quality assurance is required to make sure that the software system works according to the requirements. Were all the features implemented as agreed? Does the program behave as expected? All the parameters that you test the program against should be stated in the technical specification document.

4.1.6 Performance Evaluation:

In this module, we evaluate the performance of trained machine learning model using performance evaluation criteria such as F1 score, accuracy and classification error. Performance Evaluation is defined as a formal and productive procedure to measure an employee's work and results based on their job responsibilities. It is used to gauge the amount of value added by an employee in terms of increased business revenue, in comparison to industry standards and overall employee return on investment (ROI).

4.1.7 Prediction:

The algorithm will generate probable values for an unknown variable for each record in the new data, allowing the model builder to identify what that value will most likely be. The word "prediction" can be misleading. In some cases, it really does mean that you are predicting a future outcome, such as when you're using machine learning to determine the next best action in a marketing campaign.

CHAPTER 5

RESULTS AND DISCUSSION

S. No	Types of Attacks	Count
1	All Attacks	4019
2	Normal Attacks	1589

5.1 Attacks

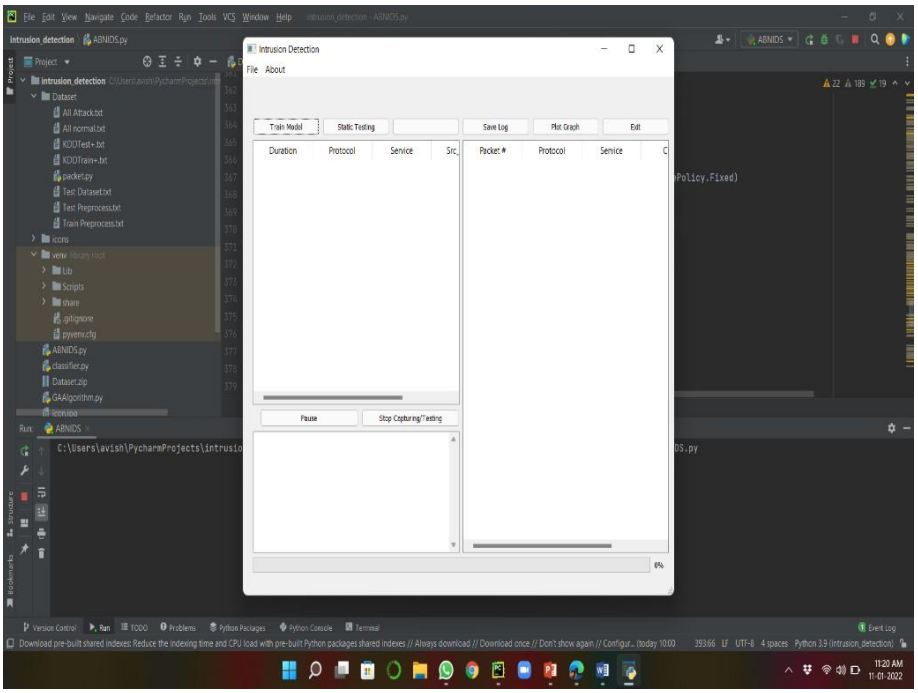


Fig. 5.2 Training the Dataset

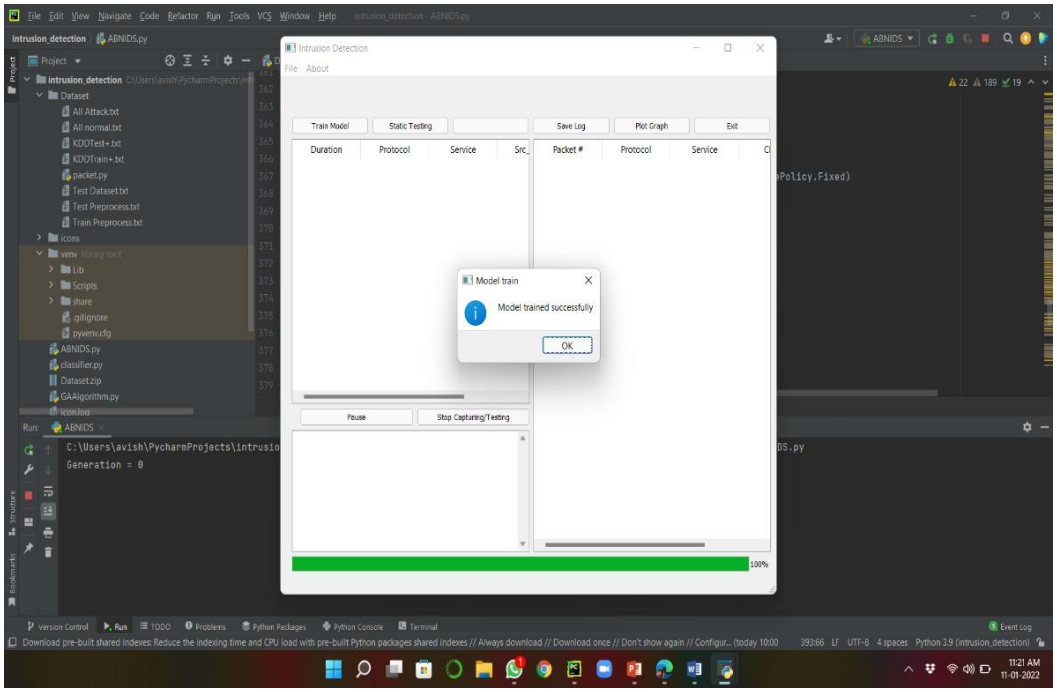


Fig. 5.3 Trained successfully

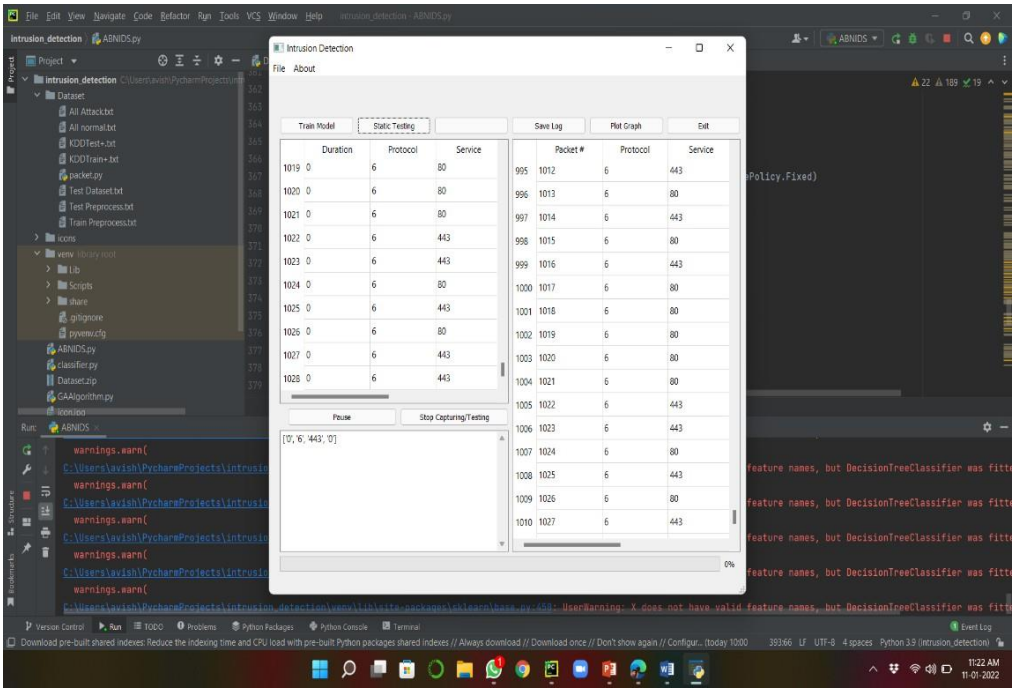


Fig. 5.4 Testing All Attacks

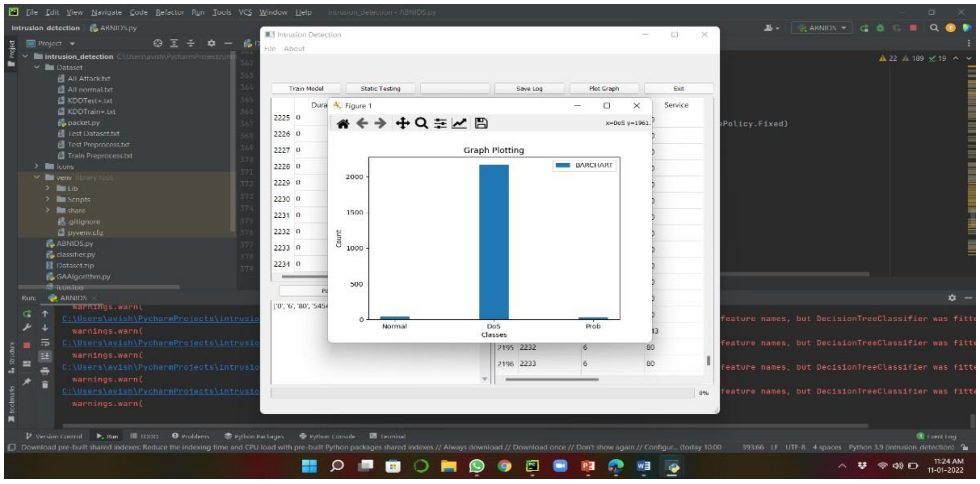


Fig. 5.5.All Attacks Plot graph

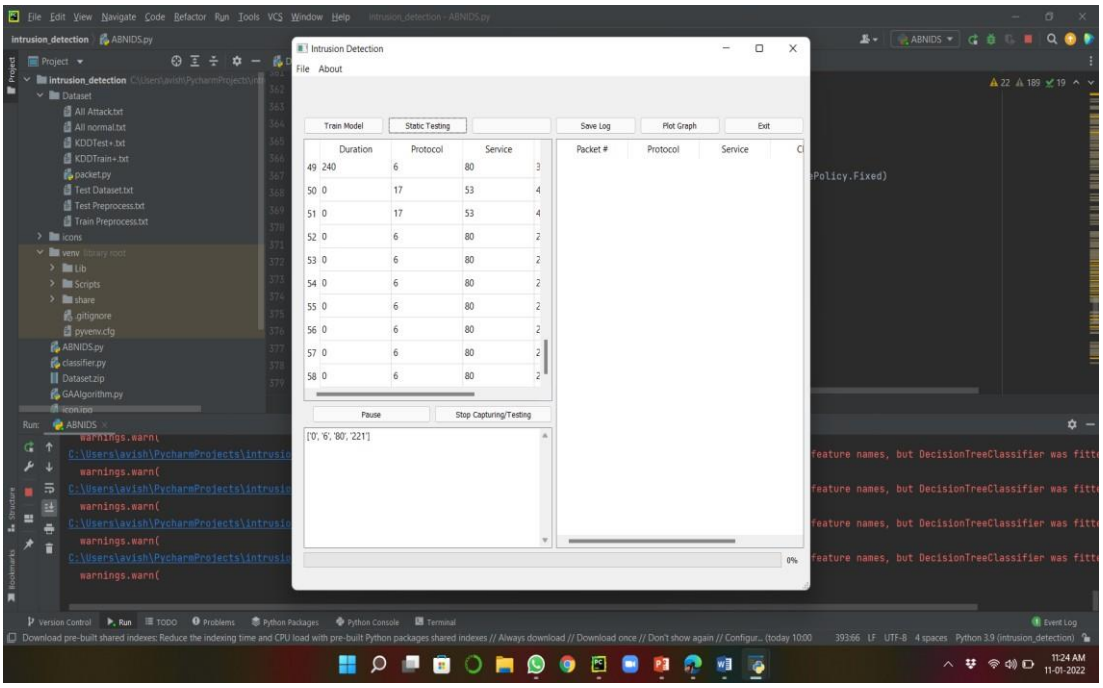


Fig. 5.6 Normal Attacks Testing

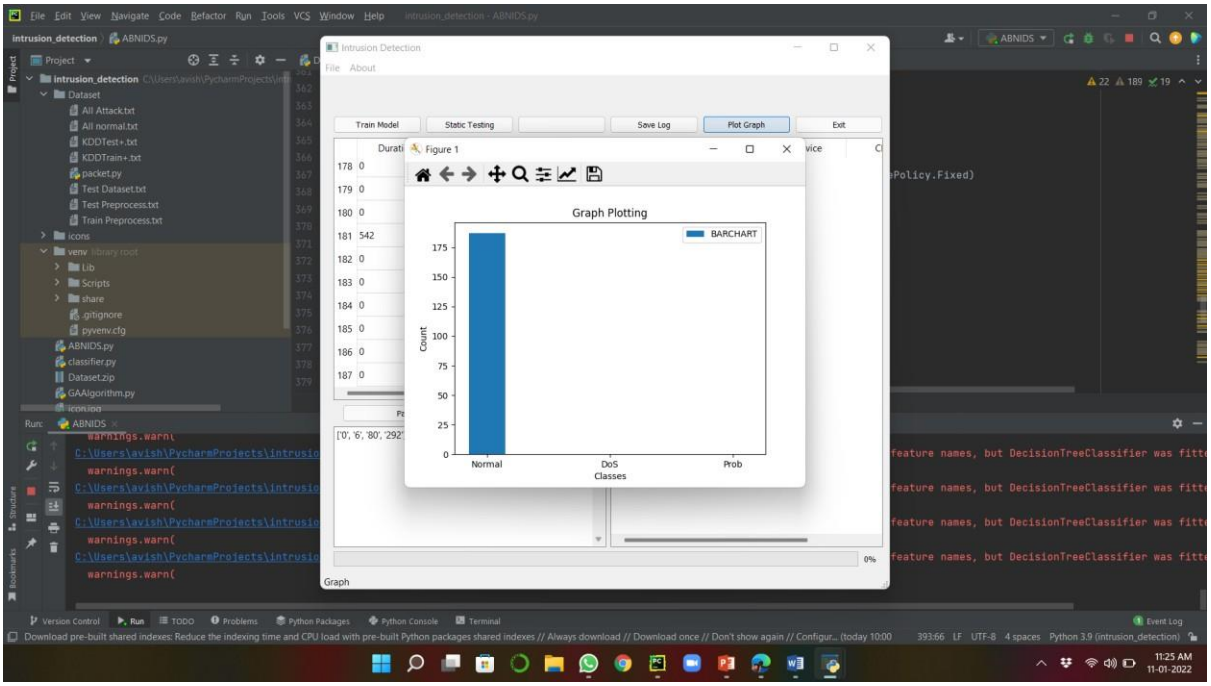


Fig. 5.7 Normal Attacks Plot Graph

CHAPTER 6

CONCLUSION

6.1 Conclusion

Network traffic logs to describe patterns of behavior in network traffic accident with intrusive or normal activity. Decision tree technique is good for the intrusion characteristic of the network traffic logs for security attack and implemented in the genetic algorithm as prevention. The other hand, this technique is also good efficiency and optimize rule for the security attack rules such as avoid redundancy. An intrusion detection system is one of the most effective tools available for detecting potential security threats and minimizing the damage caused by an attack.

6.2 Future Work

The main objective of this paper is to provide an overview of the necessity and utility of intrusion detection system. This paper gives complete study about types of IDS, life cycle, various domains, types of attacks and tools. IDS are becoming essential for day today security in corporate world and for network users. IPS defines about the preventing measures for the security. In the lifecycle the phases developed and the stages are illustrated. Still, there are more challenges to overcome. The techniques of anomaly detection and misuse detection are specifically illustrated and more techniques can be used. Further Work will be done on comparative analysis of some popular data mining algorithms applied to IDS and enhancing a classification based IDS using selective feedback methods.

REFERENCES

- [1] R. M. A. Ujjan, Z. Pervez, K. Dahal, A. K. Bashir, R. Mumtaz, and J. González, "Towards sFlow and adaptive polling sampling for deep learning based DDoS detection in SDN," Future Generation Computer Systems, vol. 111, pp. 763-779, 2020, doi: 10.1016/j.future.2019.10.015.
- [2]“Software Defined Networking Definition” https://www.opennetworking.org/sdn-definition/ (accessed March, 2, 2020).
- [3] S. Garg, K. Kaur, N. Kumar, and J. J. P. C. Rodrigues, "Hybrid Deep-Learning-Based Anomaly Detection Scheme for Suspicious Flow Detection in SDN: A Social Multimedia Perspective," IEEE Transactions on Multimedia, vol. 21, no. 3, pp. 566-578, 2019, doi: 10.1109/tmm.2019.2893549.
- [4] M. Nobakht, V. Sivaraman, and R. Boreli, "A Host-Based Intrusion Detection and Mitigation Framework for Smart Home IoT Using OpenFlow," presented at the 2016 11th International Conference on Availability, Reliability and Security (ARES), 2016.
- [5] M. S. Elsayed, N. Le-Khac, S. Dev, and A. D. Jurcut, "MachineLearning Techniques for Detecting Attacks in SDN," in 2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT), 19-20 Oct. 2019 2019, pp. 277-281, doi: 10.1109/ICCSNT47585.2019.8962519

APPENDIX

SOURCE CODE

```
import pyshark
import time
import random
class Packet:
    packet_list = list()
    def initiating_packets(self):
        self.packet_list.clear()
        capture = pyshark.LiveCapture(interface="Wi-Fi")
        for packet in capture.sniff_continuously(packet_count=25):
            try:
                if "<UDP Layer>" in str(packet.layers) and "<IP Layer>" in
str(packet.layers):
                    self.packet_list.append(packet)
                elif "<TCP Layer>" in str(packet.layers) and "<IP Layer>"
in str(packet.layers):
                    self.packet_list.append(packet)
            except:
                print(f"No Attribute name 'ip' {packet.layers}")
    def udp_packet_attributes(self,packet):
        attr_list = list()
        a1 = packet.ip.ttl
        a2 = packet.ip.proto
        a3 = self._get_service(packet.udp.port, packet.udp.dstport)
        a4 = packet.ip.len
        a5 = random.randrange(0,1000)
        a6 = self._get_land(packet,a2)
        a7 = 0
        a8, a10, a11 =
self._get_count_with_same_and_diff_service_rate(packet.udp.dstport, a3)
#23, 29, 30
        a9, a12 =
self.__get_srv_count_and_srv_diff_host_rate(packet.ip.dst, a3) #24, 31
```

```
        a13, a15, a16 = self._get_dst_host_count(packet.ip.dst, a3) #
32,34,35
        a14, a17, a18 = self._get_dst_host_srv_count(packet.udp.port,
packet.udp.dstport, packet.ip.dst) #33, 36, 37

attr_list.extend((a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15,a16,a1
7,a18))
        return self.get_all_float(attr_list)

    def tcp_packet_attributes(self,packet):
        attr_list = list()
        a1 = packet.ip.ttl #duration
        a2 = packet.ip.proto    #protocol
        a3 = self.__get_service(packet.tcp.port, packet.tcp.dstport) #
service
        a4 = packet.ip.len
        a5 = random.randrange(0,1000)
        a6 = self._get_land(packet,a2)
        a7 = packet.tcp.urgent_pointer
        a8, a10, a11 =
self._get_count_with_same_and_diff_service_rate(packet.tcp.dstport, a3)
#23, 29, 30
        a9, a12 =
self._get_srv_count_and_srv_diff_host_rate(packet.ip.dst, a3) #24, 31
        a13, a15, a16 = self.__get_dst_host_count(packet.ip.dst, a3) #
32,34,35
        a14, a17, a18 = self._get_dst_host_srv_count(packet.tcp.port,
packet.tcp.dstport, packet.ip.dst) #33, 36, 37

attr_list.extend((a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15,a16,a1
7,a18))
        return self.get_all_float(attr_list)

    def_get_service(self,src_port,dst_port):
        services = [80,443,53]
        if int(src_port) in services:
            return int(src_port)
        elif int(dst_port) in services:
            return int(dst_port)
        else:
            return 53

    def__get_land(self,packet, protocol):
        if int(protocol) == 6:
            if(packet.ip.dst == packet.ip.src and packet.tcp.port ==
packet.tcp.dstport):
                return 1
            else:
                return 0
        elif int(protocol) == 17:
            if(packet.ip.dst == packet.ip.src and packet.udp.port ==
packet.udp.dstport):
                return 1
            else:
                return 0


    def_get_count_with_same_and_diff_service_rate(self,dst_port,
service): #23, 29, 30
        count = 0
        packet_with_same_service = 0
```

```
        for p in self.packet_list:
                if "<UDP Layer>" in str(p.layers):
                    if (p.udp.dstport == dst_port):
                        count+=1
                        if (self.__get_service(p.udp.port, p.udp.dstport)
== service):
                            packet_with_same_service+=1
                elif "<TCP Layer>" in str(p.layers):
                    if (p.tcp.dstport == dst_port):
                        count+=1
                        if (self.__get_service(p.tcp.port, p.tcp.dstport)
== service):
                            packet_with_same_service+=1
        same_service_rate=0.0
        diff_service_rate = 1.0
        if not count==0:            # To avoid zero divison error
            same_service_rate = ((packet_with_same_service*100)/count)/100
            diff_service_rate = diff_service_rate-same_service_rate
        return (count, same_service_rate, diff_service_rate)

    def __get_srv_count_and_srv_diff_host_rate(self,dst_ip, service): #24,
31
        diff_dst_ip = 0
        service_count = 0
        for p in self.packet_list:
                if "<UDP Layer>" in str(p.layers):
                        if (self.__get_service(p.udp.port, p.udp.dstport)
== service):
                            service_count+=1
                            if not (p.ip.dst == dst_ip):            # not
added
                                diff_dst_ip+=1
                elif "<TCP Layer>" in str(p.layers):
                    if (self.__get_service(p.tcp.port, p.tcp.dstport) ==
service):
                            service_count+=1
                            if not (p.ip.dst == dst_ip):             # not
added
                                diff_dst_ip+=1
        srv_diff_host_rate = 0.0
        if not(service_count == 0):
            srv_diff_host_rate = ((diff_dst_ip*100)/service_count)/100
        return (service_count, srv_diff_host_rate)

    def __get_dst_host_count(self,dst_ip, service): #32, 34, 35
        same_dst_ip = 0
        same_service=0
        for p in self.packet_list:
            if(p.ip.dst == dst_ip):
                same_dst_ip+=1
                if "<UDP Layer>" in str(p.layers):
                    if (self.__get_service(p.udp.port, p.udp.dstport) ==
service):
                            same_service+=1
                elif "<TCP Layer>" in str(p.layers):
                    if (self.__get_service(p.tcp.port, p.tcp.dstport) ==
service):
                            same_service+=1
        dst_host_same_srv_rate = 0.0
        dst_host_diff_srv_rate = 1.0
        if not same_dst_ip==0:
```

```
            dst_host_same_srv_rate = ((same_service*100)/same_dst_ip)/100
            dst_host_diff_srv_rate = dst_host_diff_srv_rate-
dst_host_same_srv_rate
        return (same_dst_ip, dst_host_same_srv_rate,
dst_host_diff_srv_rate)

    def___get_dst_host_srv_count(self,src_port, dst_port, dst_ip): #33, 36,
37
        dst_host_srv_count = 0
        same_src_port = 0
        diff_dst_ip = 0
        for p in self.packet_list:
            if "<UDP Layer>" in str(p.layers):
                if (p.udp.dstport == dst_port):       # same destination
port
                    dst_host_srv_count+=1
                    if (p.udp.port == src_port):     # same src port
                        same_src_port+=1
                    if not (p.ip.dst == dst_ip):          # different
destination Ip
                        diff_dst_ip+=1

            elif "<TCP Layer>" in str(p.layers):
                if (p.tcp.dstport == dst_port):      # same destination port
                    dst_host_srv_count+=1
                    if (p.tcp.port == src_port):      # same src port
                        same_src_port+=1
                    if not (p.ip.dst == dst_ip):          #different
destination ip
                        diff_dst_ip+=1
        dst_host_same_src_port_rate = 0.0
        dst_host_srv_diff_host_rate = 0.0
        if not dst_host_srv_count==0:
            dst_host_same_src_port_rate =
((same_src_port*100)/dst_host_srv_count)/100
            dst_host_srv_diff_host_rate =
((diff_dst_ip*100)/dst_host_srv_count)/100
        return (dst_host_srv_count, dst_host_same_src_port_rate,
dst_host_srv_diff_host_rate)


    def get_all_float(self,l):

        all_float = list()
        for x in l:
            all_float.append(round(float(x),1))
        return all_float
```

```
import Population
import random


class GAAlgorithm():

    def_init_(self,train_dataset, test_dataset, population_size,
mutation_rate,gene_length=18):
        self.train_dataset = train_dataset
        self.test_dataset = test_dataset
        self.population_size = population_size
        self.mutation_rate = mutation_rate
```

```
        self.gene_length = int(gene_length)
        self.population = Population.Population(self.train_dataset,
self.test_dataset, self.population_size, self.gene_length)

    def initialization(self):
        self.population.initialize_population()

    def calculate_fitness(self):
        self.population.calculate_fitness()

    def selection(self):
        parents = list()
        end = int(self.population_size/2)
        no_of_parents = int(self.population_size/2)
        for x in range(no_of_parents):
            p1 = random.randint(0,end-1)
            p2 = random.randint(end,self.population_size-1)
            parents.append([p1,p2])
        return parents
    def cross_over(self,parents):
        self.population.cross_over(parents)

    def mutation(self):
        self.population.mutation(self.mutation_rate)

    def clear_population(self):
        self.population.clear_population()
```

```
import random
import string
import pandas
from classifier import DecisionTree

class Individual:

    chromosome = list()
    fitness = 0
    def __init__(self, train_dataset, test_dataset, gene_length=18):
        self.gene_length=int(gene_length)
        self.chromosome = [random.randint(0,1) for x in
range(self.gene_length)]
        self.train_dataset = train_dataset
        self.test_dataset = test_dataset
        self.gene_length = gene_length


    def calculate_fitness(self):
        header = list(string.ascii_lowercase[0:(self.gene_length+1)])
        kdd_train = pandas.read_csv(self.train_dataset, names=header)
        kdd_test = pandas.read_csv(self.test_dataset, names=header)
        selected_index= [header[x] for x, y in enumerate(self.chromosome)
if y==1]
        var_train, res_train = kdd_train[selected_index],
kdd_train[header[18]]
        var_test, res_test = kdd_test[selected_index], kdd_test[header[18]]
        self.fitness = self.__get_fitness(var_train, res_train, var_test,
res_test)*100

    def __get_fitness(self,var_train, res_train, var_test, res_test):
```

```
        return DecisionTree.get_fitness(var_train, res_train, var_test,
res_test)


import pyshark
import random
class Packet:
    packet_list = list()            #list is declare
    def initiating_packets(self):
        self.packet_list.clear()
        capture = pyshark.LiveCapture(interface="Wi-Fi")
        for packet in capture.sniff_continuously(packet_count=25):
            try:
                if "<UDP Layer>" in str(packet.layers) and "<IP Layer>" in
str(packet.layers):
                    self.packet_list.append(packet)
                elif "<TCP Layer>" in str(packet.layers) and "<IP Layer>"
in str(packet.layers):
                    self.packet_list.append(packet)
            except:
                print(f"No Attribute name 'ip' {packet.layers}")
    def udp_packet_attributes(self,packet):
        attr_list = list()
        a1 = packet.ip.ttl
        a2 = packet.ip.proto
        a3 = self._get_service(packet.udp.port, packet.udp.dstport)
        a4 = packet.ip.len
        a5 = random.randrange(0,1000)
        a6 = self.__get_land(packet,a2)
        a7 = 0        # urgent pointer not exist in udp layer
        a8, a10, a11 =
self._get_count_with_same_and_diff_service_rate(packet.udp.dstport, a3)
#23, 29, 30
        a9, a12 =
self._get_srv_count_and_srv_diff_host_rate(packet.ip.dst, a3) #24, 31
        a13, a15, a16 = self.__get_dst_host_count(packet.ip.dst, a3) #
32,34,35
        a14, a17, a18 = self._get_dst_host_srv_count(packet.udp.port,
packet.udp.dstport, packet.ip.dst) #33, 36, 37

attr_list.extend((a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15,a16,a1
7,a18))
        return self.get_all_float(attr_list)

    def tcp_packet_attributes(self,packet):
        attr_list = list()
        a1 = packet.ip.ttl #duration
        a2 = packet.ip.proto    #protocol
        a3 = self.__get_service(packet.tcp.port, packet.tcp.dstport) #
service
        a4 = packet.ip.len  #Src - byte
        a5 = random.randrange(0,1000) #dest_byte
        a6 = self.__get_land(packet,a2) #land
        a7 = packet.tcp.urgent_pointer #urgentpoint
        a8, a10, a11 =
self._get_count_with_same_and_diff_service_rate(packet.tcp.dstport, a3)
#23, 29, 30
        a9, a12 =
self._get_srv_count_and_srv_diff_host_rate(packet.ip.dst, a3) #24, 31
        a13, a15, a16 = self.__get_dst_host_count(packet.ip.dst, a3) #
32,34,35
```

```
        a14, a17, a18 = self._get_dst_host_srv_count(packet.tcp.port,
packet.tcp.dstport, packet.ip.dst) #33, 36, 37

attr_list.extend((a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15,a16,a1
7,a18))
        return self.get_all_float(attr_list)      # convert every attribute
to float data type

    def _get_service(self,src_port,dst_port):
        services = [80,443,53]
        if int(src_port) in services:
            return int(src_port)
        elif int(dst_port) in services:
            return int(dst_port)
        else:
            return 53

    def __get_land(self,packet, protocol):
        if int(protocol) == 6:
            if(packet.ip.dst == packet.ip.src and packet.tcp.port ==
packet.tcp.dstport):
                return 1
            else:
                return 0
        elif int(protocol) == 17:
            if(packet.ip.dst == packet.ip.src and packet.udp.port ==
packet.udp.dstport):
                return 1
            else:
                return 0


    def _get_count_with_same_and_diff_service_rate(self,dst_port,
service): #23, 29, 30
        count = 0
        packet_with_same_service = 0
        for p in self.packet_list:
                if "<UDP Layer>" in str(p.layers):
                    if (p.udp.dstport == dst_port):          #same
destination port
                        count+=1
                        if (self.__get_service(p.udp.port, p.udp.dstport)
== service): # same service
                            packet_with_same_service+=1
                elif "<TCP Layer>" in str(p.layers):
                    if (p.tcp.dstport == dst_port):
                        count+=1
                        if (self.__get_service(p.tcp.port, p.tcp.dstport)
== service):
                            packet_with_same_service+=1
        same_service_rate=0.0
        diff_service_rate = 1.0
        if not count==0:
            same_service_rate = ((packet_with_same_service*100)/count)/100
            diff_service_rate = diff_service_rate-same_service_rate
        return (count, same_service_rate, diff_service_rate)

    def __get_srv_count_and_srv_diff_host_rate(self,dst_ip, service): #24,
31
        diff_dst_ip = 0
        service_count = 0
```

```
        for p in self.packet_list:
                if "<UDP Layer>" in str(p.layers):
                        if (self.__get_service(p.udp.port, p.udp.dstport)
== service): # same service
                            service_count+=1
                            if not (p.ip.dst == dst_ip):            #
different destination ip if udp
                                diff_dst_ip+=1
                elif "<TCP Layer>" in str(p.layers):
                    if (self.__get_service(p.tcp.port, p.tcp.dstport) ==
service):
                            service_count+=1
                            if not (p.ip.dst == dst_ip):             # #
different destination ip if tcp
                                diff_dst_ip+=1
        srv_diff_host_rate = 0.0
        if not(service_count == 0):
            srv_diff_host_rate = ((diff_dst_ip*100)/service_count)/100
        return (service_count, srv_diff_host_rate)

    def __get_dst_host_count(self,dst_ip, service): #32, 34, 35
        same_dst_ip = 0
        same_service=0
        for p in self.packet_list:
            if(p.ip.dst == dst_ip): # same destination ip
                same_dst_ip+=1
                if "<UDP Layer>" in str(p.layers):
                    if (self._get_service(p.udp.port, p.udp.dstport) ==
service): # same service if udp
                            same_service+=1
                elif "<TCP Layer>" in str(p.layers):
                    if (self._get_service(p.tcp.port, p.tcp.dstport) ==
service): # same service if tcp
                            same_service+=1
        dst_host_same_srv_rate = 0.0
        dst_host_diff_srv_rate = 1.0
        if not same_dst_ip==0:
            dst_host_same_srv_rate = ((same_service*100)/same_dst_ip)/100
            dst_host_diff_srv_rate = dst_host_diff_srv_rate-
dst_host_same_srv_rate
        return (same_dst_ip, dst_host_same_srv_rate,
dst_host_diff_srv_rate)

    def __get_dst_host_srv_count(self,src_port, dst_port, dst_ip): #33, 36,
37
        dst_host_srv_count = 0
        same_src_port = 0
        diff_dst_ip = 0
        for p in self.packet_list:
            if "<UDP Layer>" in str(p.layers):
                if (p.udp.dstport == dst_port):       # same destination
port
                    dst_host_srv_count+=1
                    if (p.udp.port == src_port):     # same src port
                        same_src_port+=1
                    if not (p.ip.dst == dst_ip):          # different
destination Ip
                        diff_dst_ip+=1

            elif "<TCP Layer>" in str(p.layers):
                if (p.tcp.dstport == dst_port):      # same destination port
```

```
                    dst_host_srv_count+=1
                    if (p.tcp.port == src_port):      # same src port
                        same_src_port+=1
                    if not (p.ip.dst == dst_ip):          #different
destination ip
                        diff_dst_ip+=1
        dst_host_same_src_port_rate = 0.0
        dst_host_srv_diff_host_rate = 0.0
        if not dst_host_srv_count==0:
            dst_host_same_src_port_rate =
((same_src_port*100)/dst_host_srv_count)/100
            dst_host_srv_diff_host_rate =
((diff_dst_ip*100)/dst_host_srv_count)/100
        return (dst_host_srv_count, dst_host_same_src_port_rate,
dst_host_srv_diff_host_rate)
```

```
    def get_all_float(self,l):

        all_float = list()
        for x in l:
            all_float.append(round(float(x),1))
        return all_float
```

```
# Change testing panel to avoid segmentation fault
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtGui import QIcon, QPixmap
from PyQt5.QtWidgets import
qApp,QFileDialog,QMessageBox,QMainWindow,QDialog,QDialogButtonBox,QVBoxLayo
ut, QHeaderView, QMessageBox
import os
import time
import pyshark
import matplotlib.pyplot as plt
import threading
import packet as pack
import GAAlgorithm
import Preprocess as data
import classifier


class Ui_MainWindow(object):
    def __init__(self):
        self.tree_classifier = classifier.DecisionTree()
        self.packet = pack.Packet()
        self.trained = False
        self.stop = False
        self.threadActive = False
        self.pause = False
    def plot_graph(self):
        x = ['Normal','DoS','Prob']
        normal,dos,prob = self.tree_classifier.get_class_count()
        y = [normal,dos,prob]
        plt.bar(x,y,width=0.3,label="BARCHART")
        plt.xlabel('Classes')
        plt.ylabel('Count')
        plt.title('Graph Plotting')
        plt.legend()
        plt.show()

    def train_model(self):
```

```
        try:
            train_dataset, train_dataset_type =
QFileDialog.getOpenFileName(MainWindow, "Select Training Dataset","","All
Files (*);;CSV Files (*.csv)")
            if train_dataset:
                os.chdir(os.path.dirname(train_dataset))
                test_dataset, test_dataset_type =
QFileDialog.getOpenFileName(MainWindow, "Select Testing Dataset","","All
Files (*);;CSV Files (*.csv)")
            if train_dataset and test_dataset:
                generation  = 0
                train_dataset = data.Dataset.refine_dataset(train_dataset,
"Train Preprocess.txt")

                test_dataset = data.Dataset.refine_dataset(test_dataset,
"Test Preprocess.txt")
                #Start Genetic Algorithm
                ga =
GAAlgorithm.GAAlgorithm(train_dataset,test_dataset,population_size=5,mutati
on_rate=65)
                ga.initialization() # if error occur due to invalid dataset
population needs to be clear to avoid append of new population
                ga.calculate_fitness()
                while(ga.population.max_fitness<93 and generation<1):
                    print(f"Generation = {generation}")
                    generation+=1
                    parents = ga.selection()
                    ga.cross_over(parents)
                    ga.mutation()
                    ga.calculate_fitness()
                max_fitest = ga.population.max_fittest
                max_fitness = round(ga.population.max_fitness,1)

self.tree_classifier.train_classifier(train_dataset,max_fitest)
                self.trained = True
                ga.clear_population()
                self.progressBar.setProperty("value", 100)
                self.showdialog('Model train',f'Model trained
successfully',1)

        except:
            try:
                ga.clear_population()
            except:
                print("Err 00")
            finally:
                self.showdialog('Model train','Model trained
unsuccessfully',2)


    def static_testing(self):
        if self.isModelTrained():
            if (self.threadActive):
                self.showdialog('Warning','Please stop currently
testing',3)
            else:
                test_dataset, train_dataset_type =
QFileDialog.getOpenFileName(MainWindow, "Select Testing Dataset","","All
Files (*);;CSV Files (*.csv)")
                if test_dataset:
                    try:
```

```
                        test_dataset =
data.Dataset.refine_dataset(test_dataset, "Test Dataset.txt")
                        t1 =
threading.Thread(target=self.static_testing_thread, name = 'Static
testing', args=(test_dataset,))
                        t1.start()
                        self.threadActive = True
                    except:
                        self.showdialog('Error','Invalid Dataset',2)
        else:
            self.showdialog('Warning','Model not trained',3)

    def static_testing_thread(self,dataset):
        row = 0
        self.reset_all_content()
        with open(dataset,"r") as file:
            for line in file.readlines():
                try:
                    line = line.split(',')
                    result, result_type =
self.tree_classifier.test_dataset(line)
                    self.insert_data(line,result,result_type,row)

                    row+=1
                    if self.pause:
                        while(self.pause):
                            pass
                    if self.isStop():
                        self.stop=False
                        break
                    time.sleep(0.05)
                except:
                    print("Err")
        self.threadActive = False


    def realtime_testing(self):
        if self.isModelTrained():
            if (self.threadActive):
                self.showdialog('Warning','Please stop currently
testing',3)
            else:
                t2 = threading.Thread(target=self.realtime_testing_thread,
name = 'Realtime testing')
                t2.start()
                self.threadActive = True
        else:
            self.showdialog('Warning','Model not trained',3)
    def realtime_testing_thread(self):
        self.reset_all_content()
        self.packet.initiating_packets()
        t1 = time.time()
        attr_list  = list()
        capture = pyshark.LiveCapture(interface='Wi-Fi')
        row = 0
        try:
            for p in capture.sniff_continuously():
                try:
                    if "<UDP Layer>" in str(p.layers) and "<IP Layer>" in
str(p.layers):
                        attr_list = self.packet.udp_packet_attributes(p)
```

```
                        result, result_type =
self.tree_classifier.test_dataset(attr_list)
                        self.insert_data(attr_list,result,result_type,row)
                        print(attr_list)
                        row+=1
                    elif "<TCP Layer>" in str(p.layers) and "<IP Layer>" in
str(p.layers):
                        attr_list = self.packet.tcp_packet_attributes(p)
                        result, result_type =
self.tree_classifier.test_dataset(attr_list)
                        self.insert_data(attr_list,result,result_type,row)
                        print(attr_list)
                        row+=1
                    if (time.time()-t1) > 5 and not self.isStop:  #
5Seconds
                        print("Updateing List")
                        self.packet.initiating_packets()
                        t1 = time.time()
                    if self.pause:
                            while(self.pause):
                                pass
                    if self.isStop():
                        self.stop=False
                        break
                except :
                    print("Err")
        except :
                print("Error in looooooop")

    def pause_resume(self):
        if self.pause:
            self.pause = False
            self.btn_start.setText("Pause")
        else:
            self.pause = True
            self.btn_start.setText("Resume")


    def save_log_file(self):
        log = self.tree_classifier.get_log()
        url = QFileDialog.getSaveFileName(None, 'Save Log', 'untitled',
"Text file (*.txt);;All Files (*)")
        if url[0]:
            try:
                name = url[1]
                url = url[0]
                with open(url, 'w') as file:
                    file.write(log)
                self.showdialog('Saved',f'File saved as {url}',1)
            except:
                self.showdialog('Error','File not saved',2)

    def stop_capturing_testing(self):
        if self.pause:
            self.pause = False
            self.btn_start.setText('Pause')
        if not self.stop:
            self.stop = True
        if self.threadActive:
            self.threadActive = False
    def reset_all_content(self):
```

```
        if self.pause:
            self.pause = False
            self.btn_start.setText('Pause')
        self.stop=False
        self.tree_classifier.reset_class_count()
        self.panel_capturing.clearContents()
        self.panel_capturing.setRowCount(0)
        self.panel_result.clearContents()
        self.panel_result.setRowCount(0)
        self.panel_testing.clear()


    def insert_data(self,line,result,result_type,row):
        self.panel_capturing.insertRow(row)
        for column, item  in enumerate(line[0:4:1]):
self.panel_capturing.setItem(row,column,QtWidgets.QTableWidgetItem(str(item
)))
            self.panel_capturing.scrollToBottom()
        self.panel_testing.clear()
        self.panel_testing.addItem(str(line[0:4:1]))
        if not result==0:
            result_row = self.panel_result.rowCount()
            self.panel_result.insertRow(result_row)
            x = [row+1, line[1], line[2], result_type]
            for column, item  in enumerate(x):

self.panel_result.setItem(result_row,column,QtWidgets.QTableWidgetItem(str(
item)))
                self.panel_result.scrollToBottom()

    def clickexit(self):
        buttonReply = QMessageBox.question(MainWindow, 'Exit', "Are ou sure
to exit?", QMessageBox.Yes | QMessageBox.No, QMessageBox.No)
        if buttonReply == QMessageBox.Yes:
            if self.threadActive:
                self.pause = False
                self.stop = True
            qApp.quit()
        else:
            print('No clicked.')



    def isStop(self):
        return self.stop
    def showdialog(self,title,text, icon_type):
        msg = QMessageBox()
        if icon_type==1:
            msg.setIcon(QMessageBox.Information)
        elif icon_type==2:
            msg.setIcon(QMessageBox.Critical)
        elif icon_type==3:
            msg.setIcon(QMessageBox.Warning)
        msg.setText(text)
        msg.setWindowTitle(title)
        msg.setStandardButtons(QMessageBox.Ok)
        msg.buttonClicked.connect(self.msgbtn)
        retval = msg.exec_()
```

```
    def msgbtn(self):
        self.progressBar.setProperty("value", 0)
    def isModelTrained(self):
        return self.trained
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        path = os.path.dirname(os.path.abspath(__file__))

MainWindow.setWindowIcon(QtGui.QIcon(os.path.join(path,'icon.png')))
        MainWindow.resize(908, 844)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(MainWindow.sizePolicy().hasHeightForWidth())
        MainWindow.setSizePolicy(sizePolicy)
        MainWindow.setIconSize(QtCore.QSize(30, 30))
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.gridLayout = QtWidgets.QGridLayout(self.centralwidget)
        self.gridLayout.setObjectName("gridLayout")
        spacerItem = QtWidgets.QSpacerItem(10, 10,
QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)
        self.gridLayout.addItem(spacerItem, 1, 0, 1, 1)
        spacerItem1 = QtWidgets.QSpacerItem(20, 20,
QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Maximum)
        self.gridLayout.addItem(spacerItem1, 4, 1, 1, 1)
        spacerItem2 = QtWidgets.QSpacerItem(20, 10,
QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Fixed)
        self.gridLayout.addItem(spacerItem2, 6, 1, 1, 1)
        self.horizontalLayout_2 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_2.setObjectName("horizontalLayout_2")
        spacerItem3 = QtWidgets.QSpacerItem(15, 10,
QtWidgets.QSizePolicy.Ignored, QtWidgets.QSizePolicy.Minimum)
        self.horizontalLayout_2.addItem(spacerItem3)
        self.btn_start = QtWidgets.QPushButton(self.centralwidget)

        self.btn_start.setObjectName("btn_start")
        self.btn_start.setText('Pause')
        self.btn_start.clicked.connect(self.pause_resume)
        self.horizontalLayout_2.addWidget(self.btn_start)

        # ######################################################
        self.btn_pause = QtWidgets.QPushButton(self.centralwidget)
        self.btn_pause.setText("Stop Capturing/Testing")

        self.btn_pause.setObjectName("btn_pause")
        self.btn_pause.clicked.connect(self.stop_capturing_testing)
        self.horizontalLayout_2.addWidget(self.btn_pause)
        self.gridLayout.addLayout(self.horizontalLayout_2, 8, 1, 1, 1)
        self.horizontalLayout = QtWidgets.QHBoxLayout()
        self.horizontalLayout.setObjectName("horizontalLayout")
        # ######################################################
        self.btn_modeltrain = QtWidgets.QPushButton(self.centralwidget)
        self.btn_modeltrain.setText("Train Model")

        self.btn_modeltrain.setObjectName("btn_modeltrain")
        self.btn_modeltrain.clicked.connect(self.train_model)
        self.horizontalLayout.addWidget(self.btn_modeltrain)
        # ######################################################
```

```
        self.btn_statictesting = QtWidgets.QPushButton(self.centralwidget)
        self.btn_statictesting.setText("Static Testing")

        self.btn_statictesting.setObjectName("btn_statictesting")
        self.btn_statictesting.clicked.connect(self.static_testing)
        self.horizontalLayout.addWidget(self.btn_statictesting)
        # ########################################################
        self.btn_realtimetesting =
QtWidgets.QPushButton(self.centralwidget)
        self.btn_realtimetesting.setText("")


        self.btn_realtimetesting.setObjectName("")
        self.btn_realtimetesting.clicked.connect(self.realtime_testing)
        self.horizontalLayout.addWidget(self.btn_realtimetesting)

        # ########################################################
        self.btn_savelog = QtWidgets.QPushButton(self.centralwidget)
        self.btn_savelog.setText("Save Log")
        icon5 = QtGui.QIcon()

        self.btn_savelog.setObjectName("btn_savelog")
        self.btn_savelog.clicked.connect(self.save_log_file)
        self.horizontalLayout.addWidget(self.btn_savelog)

        # ########################################################
        self.btn_graph = QtWidgets.QPushButton(self.centralwidget)
        self.btn_graph.setText("Plot Graph")

        self.btn_graph.setObjectName("btn_graph")
        self.btn_graph.clicked.connect(self.plot_graph)
        self.horizontalLayout.addWidget(self.btn_graph)

        # ########################################################
        self.btn_exit = QtWidgets.QPushButton(self.centralwidget)
        self.btn_exit.setText("Exit")


        self.btn_exit.setObjectName("btn_exit")
        self.btn_exit.clicked.connect(self.clickexit)
        self.horizontalLayout.addWidget(self.btn_exit)
        # ########################################################
        self.gridLayout.addLayout(self.horizontalLayout, 3, 1, 1, 2)
        spacerItem4 = QtWidgets.QSpacerItem(20, 10,
QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Fixed)
        self.gridLayout.addItem(spacerItem4, 8, 1, 1, 1)
        spacerItem5 = QtWidgets.QSpacerItem(20, 10,
QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Fixed)
        self.gridLayout.addItem(spacerItem5, 0, 1, 1, 1)
        self.panel_capturing = QtWidgets.QTableWidget(self.centralwidget)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(10)
        sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.panel_capturing.sizePolicy().hasHeightFor
Width())
        self.panel_capturing.setSizePolicy(sizePolicy)
        self.panel_capturing.setRowCount(0)
        self.panel_capturing.setColumnCount(4)
        self.panel_capturing.setObjectName("panel_capturing")
```

```
        item = QtWidgets.QTableWidgetItem()
        self.panel_capturing.setHorizontalHeaderItem(0, item)
        item = QtWidgets.QTableWidgetItem()
        self.panel_capturing.setHorizontalHeaderItem(1, item)
        item = QtWidgets.QTableWidgetItem()
        self.panel_capturing.setHorizontalHeaderItem(2, item)
        item = QtWidgets.QTableWidgetItem()
        self.panel_capturing.setHorizontalHeaderItem(3, item)
        self.gridLayout.addWidget(self.panel_capturing, 4, 1, 4, 1)
        self.label = QtWidgets.QLabel(self.centralwidget)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.label.sizePolicy().hasHeightForWidth())
        self.label.setSizePolicy(sizePolicy)
        self.label.setLayoutDirection(QtCore.Qt.LeftToRight)
        self.label.setAutoFillBackground(False)
        self.label.setText("")
        path = os.path.dirname(os.path.abspath(_file_))
        path = path + r'\icons'
        self.label.setPixmap(QtGui.QPixmap(os.path.join(path,'logo.jpg')))
        self.label.setScaledContents(True)
        self.label.setAlignment(QtCore.Qt.AlignCenter)
        self.label.setObjectName("label")
        self.gridLayout.addWidget(self.label, 1, 1, 1, 1)
        spacerItem6 = QtWidgets.QSpacerItem(10, 20,
QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Fixed)
        self.gridLayout.addItem(spacerItem6, 2, 1, 1, 1)
        self.panel_testing = QtWidgets.QListWidget(self.centralwidget)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.panel_testing.sizePolicy().hasHeightForWi
dth())
        self.panel_testing.setSizePolicy(sizePolicy)

self.panel_testing.setVerticalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOn)

self.panel_testing.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarAsNeeded
)
        self.panel_testing.setObjectName("panel_testing")
        self.gridLayout.addWidget(self.panel_testing, 9, 1, 1, 1)
        self.progressBar = QtWidgets.QProgressBar(self.centralwidget)
        self.progressBar.setProperty("value", 0)
        self.progressBar.setObjectName("progressBar")
        self.gridLayout.addWidget(self.progressBar, 10, 1, 1, 2)
        # ------------------------------------------------------------------------------------
#

        self.panel_result = QtWidgets.QTableWidget(self.centralwidget)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(10)
        sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.panel_result.sizePolicy().hasHeightForWid
th())
```

```
        self.panel_result.setSizePolicy(sizePolicy)
        self.panel_result.setRowCount(0)
        self.panel_result.setColumnCount(4)
        self.panel_result.setObjectName("panel_result")
        item = QtWidgets.QTableWidgetItem()
        self.panel_result.setHorizontalHeaderItem(0, item)
        item = QtWidgets.QTableWidgetItem()
        self.panel_result.setHorizontalHeaderItem(1, item)
        item = QtWidgets.QTableWidgetItem()
        self.panel_result.setHorizontalHeaderItem(2, item)
        item = QtWidgets.QTableWidgetItem()
        self.panel_result.setHorizontalHeaderItem(3, item)
        self.gridLayout.addWidget(self.panel_result, 4,2,6,1)
        # ------------------------------------------------------------------------------------
#
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 908, 26))
        self.menubar.setObjectName("menubar")
        self.menuFile = QtWidgets.QMenu(self.menubar)
        self.menuFile.setObjectName("menuFile")
        self.menuAbout = QtWidgets.QMenu(self.menubar)
        self.menuAbout.setObjectName("menuAbout")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)
        self.actionNew = QtWidgets.QAction(MainWindow)
        self.actionNew.setObjectName("actionNew")
        self.actionOpen = QtWidgets.QAction(MainWindow)
        self.actionOpen.setObjectName("actionOpen")
        self.actionExit = QtWidgets.QAction(MainWindow)
        self.actionExit.setObjectName("actionExit")
        self.actionHelp = QtWidgets.QAction(MainWindow)
        self.actionHelp.setObjectName("actionHelp")
        self.menuFile.addAction(self.actionNew)
        self.menuFile.addAction(self.actionOpen)
        self.menuFile.addSeparator()
        self.menuFile.addAction(self.actionExit)
        self.actionExit.triggered.connect(qApp.quit)
        self.menuAbout.addAction(self.actionHelp)
        self.menubar.addAction(self.menuFile.menuAction())
        self.menubar.addAction(self.menuAbout.menuAction())

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Intrusion
Detection"))
        self.btn_start.setStatusTip(_translate("MainWindow",
"Pause/Resume"))
        self.btn_pause.setStatusTip(_translate("MainWindow", "Stop"))
        self.btn_modeltrain.setStatusTip(_translate("MainWindow", "Train
Model"))
        self.btn_statictesting.setToolTip(_translate("MainWindow", "Stactic
Testing"))
        self.btn_statictesting.setStatusTip(_translate("MainWindow",
"Static Testing"))
```

```
        self.btn_savelog.setToolTip(_translate("MainWindow", "Real Time
Capturing"))
        self.btn_savelog.setStatusTip(_translate("MainWindow", "Real Time
Capturing"))
        self.btn_graph.setStatusTip(_translate("MainWindow", "Graph"))
        self.btn_exit.setStatusTip(_translate("MainWindow", "Exit"))
        item = self.panel_capturing.horizontalHeaderItem(0)
        item.setText(_translate("MainWindow", "Duration"))
        item = self.panel_capturing.horizontalHeaderItem(1)
        item.setText(_translate("MainWindow", "Protocol"))
        item = self.panel_capturing.horizontalHeaderItem(2)
        item.setText(_translate("MainWindow", "Service"))
        item = self.panel_capturing.horizontalHeaderItem(3)
        item.setText(_translate("MainWindow", "Src_Bytes"))
        # ------------------------------------------------------------------ #
        item = self.panel_result.horizontalHeaderItem(0)
        item.setText(_translate("MainWindow", "Packet #"))
        item = self.panel_result.horizontalHeaderItem(1)
        item.setText(_translate("MainWindow", "Protocol"))
        item = self.panel_result.horizontalHeaderItem(2)
        item.setText(_translate("MainWindow", "Service"))
        item = self.panel_result.horizontalHeaderItem(3)
        item.setText(_translate("MainWindow", "Class"))
        # ------------------------------------------------------------------ #
        self.menuFile.setTitle(_translate("MainWindow", "File"))
        self.menuAbout.setTitle(_translate("MainWindow", "About"))
        self.actionNew.setText(_translate("MainWindow", "New"))
        self.actionOpen.setText(_translate("MainWindow", "Open"))
        self.actionExit.setText(_translate("MainWindow", "Exit"))
        self.actionHelp.setText(_translate("MainWindow", "Help"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

RESEARCH PAPER

DETECTION OF ATTACKS USING RFA

ABSTRACT:

The entry level setup IDS is right now exceptionally intriguing as the main part of framework security. The IDS takes traffic data from line or framework then uses it to build more secure network. Attacks are really challenging and dangerous to eliminate street exercises. The entrance framework (IDS), which is a key component of framework security, is currently quite intriguing. The IDS collects traffic data from the network or framework and then uses it to improve security. Assaults are often extremely difficult and time-consuming to separate from street exercises. The examiner should look over all of the vast and varied material in order to screen the organisation association. The frequency of traffic is thus anticipated to be determined by an organisation search strategy. Another method for finding IDS identifiers was developed in this study by focusing on information mining techniques from a computation machine. Sorting the decision tree and the computation is the method used to establish the principles. A wide range of known (irresistible) assaults can be distinguished by evaluating the normal interruption pace of the framework for checking the means of misconception. In the case of something surprising occurs, the framework initially learns the ordinary profile and afterward records every one of the components of the framework that don't match the setup profile. The main benefit of discovery is its inability to discern between fresh or unexpected assaults at a high rate of occurrence.

The upside of having the option to identify uncommon things is the capacity to recognize new (or startling) assaults that convey many advantages. Procedures dependent on innovation pipelines utilized in different ventures. We give general data to the investigation of traffic data and for the location of street mishaps utilizing the significant distance-course of-the-street .The proposed technique utilizes tests dependent on removing traffic statistics from internet media (such as Facebook and Twitter): This movement collects phrases related to any type of traffic workout, like traffic pauses or street terminations. The quantity of starting handling strategies is presently executed. breathing, signal presentation, POS signal, partition, and so forth to change the data acquired in the inherent structure. T information is then consequently shown as "traffic" or "traffic" utilizing the latent Dirichlet allocation (LDA) calculation. Vehicle enrollment data is isolated into three kinds; great, terrible and impartial. The response to this classification is the expression enraptured (positive, negative, or unbiased) as for street sentences, contingent upon whether or not it is traffic. To manage bi-directional LSTM organisations, each sentence is now converted to a single hot code using the bag-of-words (Bow) technique (Bi- LSTM). In the wake of preparing, a multi- stage muscle network utilizes soft max to arrange sentences as indicated by area, vehicle experience, and sort of polarization. The suggested approach compares the high-level preparing approaches with the preparation of various machines in terms of precision, F scores, and numerous standards. Building a Machine Learning-Based Network Intrusion Detection System for Software Defined Networks.

LITERATURE REVIEW

Designing a Network Intrusion Detection System Based on Machine Learning for Software Defined Networks

Software-defined Networking (SDNs) have as of late been created as a feasible and promising answer for the eventual fate of the Internet. Networks are made due. incorporated; and observed and adjusted utilizing SDN. These advantages, then again, bring us ecological dangers, for example, network crashes, framework incapacities, internet banking misrepresentation, and robbery. Families, businesses, and the economy may all suffer as a result of these problems. Truth, superior execution, and the genuine framework are fundamental to accomplishing this objective. The extension of wise Al calculations into the network intrusion detection system (NIDS) through a software-defined network (SDN) has been extremely invigorating over the previous decade. The accessibility of data, the distinction in information investigation. and the many advances in Al calculations us with making a superior, more dependable, and solid framework for distinguishing the various sorts of organization assaults. The review was essential for the NIDS SDN survey.

A Deep Learning Approach for Network Intrusion Detection System

Network Intrusion Detection Systems (NIDSs) are a significant device for network framework overseers

to decide network security. NIDS screens and examines approaching and active calls from family network gadgets and cautions assuming that entrance is identified. As far as access control, NIDS is separated into two classifications: I) NIDS (SNIDS) based mark (abuse), and ii) NIDS (ADNIDS) based secrecy location. SNIDS and Drinking put assault marks first in NIDS. The helpful plan is made of against slip vehicle to permit admittance to the organization. Interestingly. ADNIDS permits network traffic to stream in when it. is going to split away from typical traffic. Significant in characterizing SNIDS. notable, notable assault, non-salvage assault. Nonetheless, its unmistakable makes it extremely challenging to distinguish obscure or new assaults on the grounds that the marks of pre-introduced assaults on the IDS are decreased. However, ADNIDS is titical to be familiar with obscure and new assaults. In spite of the fact that ADNIDS estimates its adequacy well, its capacity to identify new assaults has prompted its far and wide acknowledgment. There are two issues that function admirably in the advancement of NIDS: gentle and direct assaults. Above all else, the strategy for choosing the right traffic information from the informational index line is hard to distinguish peculiarities. Because of steady vacillations and changes, the capacities chose at a similar assault level may not be reasonable for other assault classes. Second, there is an absence of a bunch of traffic information from the genuine line of NIDS improvement. It requires a ton of work to separate a bunch of genuine or ongoing recorded information from the crude line of the gathered way.

Intrusion Preventing System using Intrusion Detection System Decision Tree Data Mining

With worldwide availability, network security has become more associated with innovative work. As the quantity of assaults builds, the firewall has turned into a significant security strategy issue overall. Firewalls can be permitted or denied over the organization, however since firewalls can't be recognized or assaulted, signing in and applying to a firewall is a method for controlling how you forestall it. Access location Firewall innovation is viewed as an extra answer for identify interruptions in an organization without a firewall. Firewalls and IDS address the old as far as data innovation security. A firewall is great for ensuring frameworks and networks and lessens the danger of organization assaults. IDS can identify endurance or assault. Capacity to interface IDS and firewalls called IPS. That is the fair thing to do, and it should end there. There are at least one distinct standard for every retailer. Each organization parcel that arrives at the firewall should be tried by characterized rules until an appropriate rule is found. Under current law, bundles will be permitted or restricted from arriving at the line. Every law determines a particular kind of vehicle. The points of interest of how the pipeline will be sold should be visible from the lines of vehicles from people's perspective. This review plans to try not to attempt to sign in to look for Internet-based substance, like IDS. and afterward implementing firewall rules like impeding Need to find out about our information mining machine security strategy The technique used to make the standard is to rank the ID3 calculation by tree endorsement. It's a decent and great practice to implement firewalls.

A Deep Learning Approach to Network Intrusion Detection

The Network Access System (NIDS) assumes a significant part in ensuring PC organizations. Be that as it may, there are worries about the accessibility and maintainability of current innovation to meet present day network necessities. Specifically, these worries are connected with the increment in individuals' level of correspondence and the lessening in their level of information. This paper presents new top to bottom examination techniques to comprehend and resolve these issues. We plainly characterize non-standard encoder (NDAE) prerequisites for the investigation of uncontrolled items. Furthermore, we suggest a top to bottom investigation of the classes made utilizing the NDAE. Our proposals were carried out in GPU-TensorFlow and assessed utilizing the KDD Cup '99 scale and the NSL-KDD informational index.

EXISTING SYSTEM:

- Today, pipelines have turned into a significant piece of public foundation and the computation of public or private mists.
- Techniques Traditional organization network has turned into a test.
- These troubles have forestalled the foundation of new and forward- thinking administrations in a similar organization, making it hard to associate organizations, business associations, and the Internet overall.

Problem Statement:

- Attacks are genuinely difficult, common, and tiresome to separate street workouts.

- Utilizes To assess the danger of pipelines, analysts must consider vast and extensive data.
- Technique The method utilised to identify the pipelines is anticipated to determine how the traffic moves forward.

Proposed System:

- Hereditary Algorithms are one of the most generally utilized techniques for AI as far as availability.
- Cold The choice sheet looks at the test to one of the qualities of a specific case, while the leaf shows the possibility of whether the result is in the ordinary or typical period of the assault (potentially a potential assault).
- Strategy A better approach to observe IDS tokens utilizing an authentication tree. A strategy for Al has been given. The technique utilized in law making is to sort the choice tree and calculation.

Advantages:

- Attack location should be possible physically or consequently.
- IDS should have the option to adapt to the hours of development and exposure.
- It is vital to utilize a choice tree. The importance of comprehending programmed attacks and how to respond is growing.

HARDWARE REQUIREMENTS:

The absolute most broadly utilized calculations.

- K-Neighbour
- Blameless Bays
- Choice tree/Natural woodland
- Support for vector machines

System: Pentium i3 Processor.

HDD: 500 GB.

Screen: 15" LED

Devices: Keyboard, Mouse

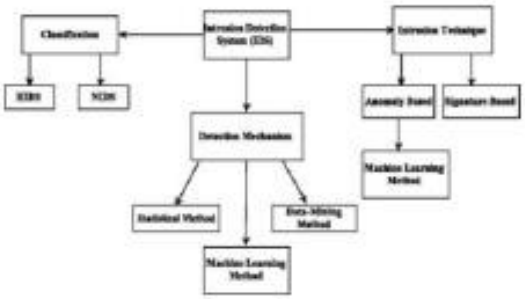
Random Access Memory: 2 GB

SOFTWARE REQUIREMENTS:

Software: Windows 10.

Language: Python.

BLOCK DIAGRAM :



FLOW DIAGRAM :

- Intercession

Decision tree

Introduction:

Up until this point, we have figured out how to go this way and that, and it has been hard to

comprehend. Presently how about we start with "Tree Decision", I guarantee you it very well may be a straightforward calculation in Machine Learning. There aren't so numerous here. It is one of the most broadly utilized and common sense strategies for Al since it is not difficult to utilize and clarify.

What is a Decision Tree?

It is an instrument with applications running in better places. The testament tree can be utilized in similar class as obsolete issues. The actual name recommends that it utilizes plans, for example, trees to show prescience from the request in which things are isolated. It begins at the root and finishes with the choice to get away. Before we study the choice tree, how about we investigate a few words.

Root Nodes The top of this hub is toward the start of the choice tree, and the public starts to isolate it as indicated by different elements.

Decision Nodes - The gatherings we see subsequent to isolating the root are called Resolutions.

Leaf Nodes an indivisible head called a leaf or leaf

Sub-tree- 33% of the sub-tree plan, a large portion of the exactness of the sub- tree.

Pruning: In order to stop overworking yourself, all that is left to do is remove the head.

MODULES:

- Dataset collection
- Data Cleaning
- Feature Extraction
- Model training
- Testing model
- Performance Evaluation
- Prediction

Dataset collection:

Informational index assortment:

Information assortment can assist you with tracking down ways of following previous occasions utilizing information examination to record them. This permits you to foresee the way and make prescient models utilizing Al devices to anticipate future changes. Since the prescient model is just pretty much as great as the data acquired, the most effective way to gather information is to further develop execution. The data ought to be faultless (garbage. open air trash) and should include information on the task you are undertaking. For instance, a non-performing advance might not profit from the money received, but might profit over time from gas prices. We compile information from the Kaggle data set in this module. These figures contain data on yearly contrasts.

Data cleaning:

Data cleanliness is a significant piece of all Al exercises. The data cleanliness of this module is expected for the arrangement of information for the annihilation and transformation of wrong, inadequate, deluding or misdirecting data. You can utilize it to look for data. Discover what cleaning you can do.

Feature Extraction:

This is done to lessen the quantity of capacities in the informational index, which will accelerate preparing and increment proficiency.

Mining begins at the front line of estimated, usable data (ascribes) targeted toward securing, altering, and following in AI, image acknowledgment, and picture handling. following. and normalizing data, and now and again prompting more prominent clearness. Take out the properties related with aspect decrease

On the off chance that the calculation's feedback is excessively enormous, it won't be handled, and assuming it is suspected to be excessively huge (like estimating one foot and meter, or rehashing the picture displayed in pixels), it tends to be switched. properties (likewise called vector properties).

Characterize the initial segment, called highlight choice. The chose things ought to contain data about the data got so they can fill the ideal role utilizing this portrayal rather than complete data.

Model training:

An illustration of this preparation is the informational collection used to prepare the ML calculation. It comprises of significant info definitions that influence information inspecting and yield.

The preparation model is utilized to utilize the information through the result and result calculations. The This connection's results will be used to change the layout. This strategy for assault is designated "matching model". Information preparing definition or informational collection approval is significant for demonstrating. Plan

language preparing is a method for giving data about the ML calculation and assist with deciding and become familiar with the best significance of every one of its highlights. There are many kinds of AI, the majority of which are controlled an uncontrolled.

Testing model:

In this module, we test an AI machine planned utilizing research information Quality protection is needed to make the product framework work appropriately. All chances settled upon? Does the program fill in true to form? All program testing standards should be remembered for the specialized detail.

What's more, programming testing can uncover every one of the defects and shortcomings that have happened during improvement. Once the application is delivered, you don't need your clients to come to your home together. Various kinds of tests just take into account recognition of blunders during activity.

Performance Evaluation:

In this module, e audit the presentation of an AI framework utilizing execution assessment measures, for example, Fl scores, exactness, mistake.

At the point when the model performs inadequately, we change the AI to further develop execution.

Execution examination is characterized as a norm and productive method for estimating representative execution dependent on worker obligations. It is utilized to gauge the worth of representatives by expanding their business pay contrasted with industry and all out venture (ROI).

All associations that have taken in the specialty of "mutual benefit" depend on the presentation of their workers dependent on an exhibition exahsination framework to continually survey and assess the presentation of its representatives. In a perfect world, workers are evaluated yearly upon the arrival of the occasion, in view of advancement or compensation increment. Execution examination plays an immediate part to play in giving input to workers to all the more likely comprehend their principles.

Prediction:

Consistency refers to the results after making a computation based on the historical context of the set and executing it out when you anticipate the probability of a particular outcome, for example, deciding whether the client will remain for 30 days.

The worth-based calculation can be changed for each new thing composed, permitting the author to decide the worth that is destined to be. "Speculation" can be misdirecting. Now and then, this implies foreseeing the future, like utilizing a machine to decide the following game-plan.

In different cases, "prescience" is connected, for instance, in the event that the item has as of now been created. For this situation, the move has as of now been made, however it will assist you with giving input on whether it is satisfactory and to make a proper move. In this module, we utilize an organized, AI technique to decide whether the patient will respond to a portion of the inquiries.

CONCLUSION:

Detours depict personal conduct standards that happen during street mishaps and typical exercises. The tree manating method is the most ideal to the working of the IDS access street and is executed in the hereditary calculation of avoidance. Then again, this innovation functions admirably and maintains a strategic distance from over-the-top guidelines, like firewalls.

REFERENCES:

[1] R. M. A. Ujjan, Z. Pervez, K. Dahal, A. K. Bashir, R. Mumtaz, and J. González, "Towards sFlow and adaptive polling for deep learning based DDoS detection in SDN," Future Generation Computer Systems, vol. 111, pp. 763-779, 2020, doi: 10.1016/j.future.2019.10.015.

[2]'Software Defined No working Definition."

https://www.opennetworking.org/sin- definition/ (accessed March, 2, 2020).

[3] S. Garg, K. Kaur, N. Kumar, and J. J. P. C. Rodrigues, "Hybrid Deep-Learning- Based Anomaly Detection Scheme for Suspicious Flow Detection in SDN: A Social Multimedia Perspective," IEEE Transactions on Multimedia, vol. 21, no. 3, PP. 566-578, 2019, doi: 10.1109/tmm 2019.2893549.

[4] M. Nobakht, V. Sivaraman, and R. Boreli, "A Host-Based Intrusion Detection and Mitigation Framework for Smart Home IoT Using OpenFlow, presented at the 2016 11th International Conference on Availability, Reliability and Security (ARES). 2016.

[5] M. S. Elsayed, N. Le-Khac, S. Dev, and A. D. Jurcut, "Machine Learning Techniques for Detecting Attacks in SDN," in 2019 IEEE 7th International Conference on Computer Science and Network (ICCSNT), 19-20 Oct. 2019 2019, pp. 277-281, 10.1109/ICCSNT47585.2019.8962519.

DETECTION OF ATTACKS USING RFA RESEARCH PAPER.docx

Scanned on: 11:27 January 28, 2023 UTC

	Word count
Identical	91
Minor Changes	33
Paraphrased	205
Omitted	464