

# **PREDICTION OF CYBER ATTACKS USING DATA SCIENCE TECHNIQUE**

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

by

**KORAPATI MAHESH (Reg. No. 38110288)**

**SILUVERU HAKHILESH RAM (Reg. No. 38110189)**



**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**

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119**

**MAY - 2022**



# **SATHYABAMA**

**INSTITUTE OF SCIENCE AND TECHNOLOGY  
(DEEMED TO BE UNIVERSITY)**

**Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE**

**[www.sathyabama.ac.in](http://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 **KORAPATI MAHESH (Reg. No: 38110288), SILUVERU HAKHILESH RAM (Reg. No: 38110189)** who carried out the project entitled "**PREDICTION OF CYBER ATTACKS USING DATA SCIENCE TECHNIQUE**" under my supervision from November 2021 to April 2022.

**Internal Guide  
Dr. G. Nagarajan M.E., Ph.D.,**

**External Guide (if Applicable)**

**Head of the Department  
Dr. L. LAKSHMANAN M.E., Ph.D.,**

---

**Submitted for Viva voce Examination held on \_\_\_\_\_**

**Internal Examiner**

**External Examiner**

## **DECLARATION**

I **KORAPATI MAHESH, SILUVERU HAKHILESH RAM** hereby declare that the Project Report entitled **PREDICTION OF CYBER ATTACKS USING DATA SCIENCE TECHNIQUE** done by me under the guidance of **Dr. G. Nagarajan M.E., Ph.D., (Internal)** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering.

**DATE:** 25-03-2022

K. Mahesh.

**PLACE:** CHENNAI

**SIGNATURE OF THE CANDIDATE**

## **ACKNOWLEDGEMENT**

**I am 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 Dr. L. Lakshmanan M.E., Ph.D., and Dr. S. Vigneshwari M.E., Ph.D. Head of the Department, Dept. 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. G. Nagarajan ME., Ph. D for his 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**

Cyber-attack, via cyberspace, targeting an enterprise's use of cyberspace for the purpose of disrupting, disabling, destroying, or maliciously controlling a computing environment/infrastructure; or destroying the integrity of the data or stealing controlled information. The state of the cyberspace portends uncertainty for the future Internet and its accelerated number of users. New paradigms add more concerns with big data collected through device sensors divulging large amounts of information, which can be used for targeted attacks. Though a plethora of extant approaches, models and algorithms have provided the basis for cyber-attack predictions, there is the need to consider new models and algorithms, which are based on data representations other than task-specific techniques. However, its non-linear information processing architecture can be adapted towards learning the different data representations of network traffic to classify type of network attack. In this paper, we model cyber-attack prediction as a classification problem, Networking sectors have to predict the type of Network attack from given dataset using machine learning techniques. The analysis of dataset by supervised machine learning technique (SMLT) to capture several information's like, variable identification, uni-variate analysis, bi-variate and multi-variate analysis, missing value treatments etc. A comparative study between machine learning algorithms had been carried out in order to determine which algorithm is the most accurate in predicting the type cyber-Attacks. We classify four types of attacks are DOS Attack, R2L Attack, U2R Attack, Probe attack. The results show that the effectiveness of the proposed machine learning algorithm technique can be compared with best accuracy with entropy calculation, precision, Recall, F1 Score, Sensitivity, Specificity and Entropy.

## TABLE OF CONTENTS

| CHAPTER NO | TITLE                                     | PAGE NO   |
|------------|-------------------------------------------|-----------|
|            | ABSTRACT                                  | i         |
|            | LIST OF FIGURES                           | vi        |
|            | LIST OF TABLES                            | viii      |
| <b>1</b>   | <b>INTRODUCTION</b>                       | <b>1</b>  |
|            | 1.1 Data Science                          | 1         |
|            | <i>1.1.1 Data Scientist</i>               | 1         |
|            | 1.2 Artificial Intelligence               | 2         |
|            | 1.3 Natural Language Processing           | 3         |
|            | 1.4 Machine Learning                      | 3         |
|            | 1.5 Objectives                            | 4         |
|            | 1.6 Project Goals                         | 5         |
| <b>2</b>   | <b>LITERATURE SURVEY</b>                  | <b>6</b>  |
|            | 2.1 Literature Survey                     | 6         |
| <b>3</b>   | <b>AIM AND SCOPE OF THE INVESTIGATION</b> | <b>14</b> |
|            | 3.1 Project Proposal                      | 14        |
|            | 3.2 Scope of the Project                  | 14        |
|            | 3.3 Existing System                       | 14        |
|            | 3.3.1 Disadvantages                       | 15        |
|            | 3.4 Preparing Dataset                     | 15        |
|            | 3.5 System Study                          | 15        |

|          |                                                               |           |
|----------|---------------------------------------------------------------|-----------|
| 3.5.1    | Classification of Attacks                                     | 15        |
| 3.5.2    | Attacks Summary                                               | 17        |
| 3.6      | Feasibility Study                                             | 21        |
| 3.6.1    | Data Wrangling                                                | 21        |
| 3.6.2    | Data Collection                                               | 21        |
| 3.6.3    | Preprocessing                                                 | 21        |
| 3.6.4    | Building the Classification model                             | 21        |
| 3.6.5    | Construction of a Predictive model                            | 22        |
| <b>4</b> | <b>EXPERIMENTAL OR MATERIALS AND METHODS, ALGORITHMS USED</b> | <b>23</b> |
| 4.1      | Project Requirements                                          | 23        |
| 4.1.1    | Functional Requirements                                       | 23        |
| 4.1.2    | Non-Functional Requirements                                   | 23        |
| 4.1.3    | Environment Requirements                                      | 24        |
| 4.2      | Software Description                                          | 24        |
| 4.2.1    | Anaconda Navigator                                            | 24        |
| 4.2.2    | Jupyter Notebook                                              | 26        |
| 4.2.3    | Python                                                        | 28        |
| 4.3      | Python Libraries Needed                                       | 30        |
| 4.3.1    | NUMPY Library                                                 | 30        |
| 4.3.2    | PANDAS Library                                                | 31        |
| 4.3.3    | MATPLOTLIB Library                                            | 31        |
| 4.3.4    | SCIKIT-LEARN Library                                          | 31        |
| 4.3.5    | TKINTER                                                       | 32        |
| 4.4      | System Architecture                                           | 32        |
| 4.5      | UML Diagrams                                                  | 33        |
| 4.5.1    | Class Diagram                                                 | 33        |

|          |                                                                             |           |
|----------|-----------------------------------------------------------------------------|-----------|
|          | 4.5.2 Use Case Diagram                                                      | 34        |
|          | 4.5.3 Workflow Diagram                                                      | 35        |
|          | 4.5.4 Activity Diagram                                                      | 36        |
|          | 4.5.5 Sequence Diagram                                                      | 37        |
|          | 4.5.6 Entity Relationship Diagram                                           | 38        |
|          | 4.6 Algorithms Used                                                         | 39        |
|          | 4.7 Comparing Algorithm with prediction in the form of best Accuracy result | 44        |
|          | 4.8 Module Description                                                      | 48        |
|          | 4.8.1 Data Validation process and Visualization                             | 48        |
|          | 4.8.2 Prediction of DOS Attacks                                             | 54        |
|          | 4.8.3 Prediction of R2L Attacks                                             | 55        |
|          | 4.8.4 Prediction of U2R Attacks                                             | 56        |
|          | 4.8.5 Prediction of Probe Attacks                                           | 57        |
|          | 4.8.6 Prediction of Overall Network Attacks                                 | 58        |
|          | 4.8.7 GUI based prediction results of Network Attacks                       | 59        |
| <b>5</b> | <b>RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS</b>                         | <b>61</b> |
|          | 5.1 Performance Analysis                                                    | 61        |
|          | 5.2 Discussion                                                              | 64        |
| <b>6</b> | <b>SUMMARY AND CONCLUSION</b>                                               | <b>65</b> |
|          | 6.1 Summary                                                                 | 65        |
|          | 6.2 Conclusion                                                              | 65        |
|          | 6.3 Future Work                                                             | 65        |
|          | <b>REFERENCES</b>                                                           | <b>66</b> |

|                 |     |
|-----------------|-----|
| <b>APPENDIX</b> | 67  |
| A. SOURCE CODE  | 67  |
| B. SCREENSHOTS  | 118 |

## LIST OF FIGURES

| FIGURE NO | FIGURE NAME                                  | PAGE NO |
|-----------|----------------------------------------------|---------|
| 1.1       | Process of Machine Learning                  | 4       |
| 3.1       | Process of data flow Diagram                 | 22      |
| 4.1       | Anaconda Navigator Interface                 | 26      |
| 4.2       | System Architecture                          | 33      |
| 4.3       | Class Diagram                                | 33      |
| 4.4       | Use Case Diagram                             | 34      |
| 4.5       | Workflow Diagram                             | 35      |
| 4.6       | Activity Diagram                             | 36      |
| 4.7       | Sequence Diagram                             | 37      |
| 4.8       | Entity Relationship Diagram                  | 38      |
| 4.9       | Logistic Regression                          | 40      |
| 4.10      | Decision Tree Classifier                     | 41      |
| 4.11      | Random Forest Classifier                     | 42      |
| 4.12      | Support Vector Classifier                    | 43      |
| 4.13      | Data Frame                                   | 49      |
| 4.14      | Percentage level of protocol type            | 50      |
| 4.15      | Comparison of service type and protocol type | 51      |
| 4.16      | Data Validation                              | 53      |
| 4.17      | Prediction of DOS Attacks                    | 54      |
| 4.18      | Prediction of R2L Attacks                    | 55      |
| 4.19      | Prediction of U2R Attacks                    | 56      |
| 4.20      | Prediction of Probe Attacks                  | 57      |
| 4.21      | GUI based prediction                         | 60      |
| 5.1       | Accuracy comparison of DOS Attack            | 61      |
| 5.2       | Accuracy comparison of R2L Attack            | 61      |
| 5.3       | Accuracy comparison of U2R Attack            | 62      |
| 5.4       | Accuracy comparison of Probe Attack          | 62      |

|     |                                       |     |
|-----|---------------------------------------|-----|
| 5.5 | Accuracy comparison of Overall Attack | 63  |
| 6.1 | GUI before input                      | 118 |
| 6.2 | GUI after giving input                | 118 |

## LIST OF TABLES

| TABLE NO | TABLE NAME                               | PAGE NO |
|----------|------------------------------------------|---------|
| 3.1      | Attack Types Grouped to respective class | 17      |
| 3.2      | Description of Attacks                   | 18      |

# CHAPTER-1

## INTRODUCTION

### DOMAIN OVERVIEW

#### 1.1 DATA SCIENCE

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data, and apply knowledge and actionable insights from data across a broad range of application domains.

The term "data science" has been traced back to 1974, when Peter Naur proposed it as an alternative name for computer science. In 1996, the International Federation of Classification Societies became the first conference to specifically feature data science as a topic. However, the definition was still in flux.

The term "data science" was first coined in 2008 by D.J. Patil, and Jeff Hammer Bacher, the pioneer leads of data and analytics efforts at LinkedIn and Facebook. In less than a decade, it has become one of the hottest and most trending professions in the market.

Data science is the field of study that combines domain expertise, programming skills, and knowledge of mathematics and statistics to extract meaningful insights from data.

Data science can be defined as a blend of mathematics, business acumen, tools, algorithms and machine learning techniques, all of which help us in finding out the hidden insights or patterns from raw data which can be of major use in the formation of big business decisions.

##### **1.1.1 Data Scientist:**

Data scientists examine which questions need answering and where to find the related data. They have business acumen and analytical skills as well as the ability to mine, clean, and present data.

Businesses use data scientists to source, manage, and analyze large amounts

of unstructured data.

### **Required Skills for a Data Scientist:**

- **Programming:** Python, SQL, Scala, Java, R, MATLAB.
- **Machine Learning:** Natural Language Processing, Classification, Clustering.
- **Data Visualization:** Tableau, SAS, D3.js, Python, Java, R libraries.
- **Big data platforms:** MongoDB, Oracle, Microsoft Azure, Cloudera.

## **1.2 ARTIFICIAL INTELLIGENCE**

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving.

Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to the natural intelligence displayed by humans or animals. Leading AI textbooks define the field as the study of "intelligent agents" any system that perceives its environment and takes actions that maximize its chance of achieving its goals. Some popular accounts use the term "artificial intelligence" to describe machines that mimic "cognitive" functions that humans associate with the human mind, such as "learning" and "problem solving", however this definition is rejected by major AI researchers.

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, speech recognition and machine vision.

AI programming focuses on three cognitive skills: learning, reasoning and self-correction.

**Learning processes.** This aspect of AI programming focuses on acquiring data and creating rules for how to turn the data into actionable information. The rules, which are called algorithms, provide computing devices with step-by-step

instructions for how to complete a specific task.

**Reasoning processes.** This aspect of AI programming focuses on choosing the right algorithm to reach a desired outcome.

**Self-correction processes.** This aspect of AI programming is designed to continually fine-tune algorithms and ensure they provide the most accurate results possible.

### **1.3 NATURAL LANGUAGE PROCESSING (NLP):**

Natural language processing (NLP) allows machines to read and understand human language. A sufficiently powerful natural language processing system would enable natural-language user interfaces and the acquisition of knowledge directly from human-written sources, such as newswire texts. Some straightforward applications of natural language processing include information retrieval, text mining, question answering and machine translation. Many current approaches use word co-occurrence frequencies to construct syntactic representations of text. "Keyword spotting" strategies for search are popular and scalable but dumb; a search query for "dog" might only match documents with the literal word "dog" and miss a document with the word "poodle". "Lexical affinity" strategies use the occurrence of words such as "accident" to assess the sentiment of a document. Modern statistical NLP approaches can combine all these strategies as well as others, and often achieve acceptable accuracy at the page or paragraph level. Beyond semantic NLP, the ultimate goal of "narrative" NLP is to embody a full understanding of commonsense reasoning. By 2019, transformer-based deep learning architectures could generate coherent text.

### **1.4 MACHINE LEARNING**

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of

Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms. It feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. Machine learning can be roughly separated in to three categories. There are supervised learning, unsupervised learning and reinforcement learning. Supervised learning program is both given the input data and the corresponding labeling to learn data has to be labeled by a human being beforehand. Unsupervised learning is no labels. It provided to the learning algorithm. This algorithm has to figure out the clustering of the input data. Finally, Reinforcement learning dynamically interacts with its environment and it receives positive or negative feedback to improve its performance.



***Fig 1.1: Process of Machine Learning***

## **1.5 OBJECTIVES:**

This analysis aims to observe which features are most helpful in predicting the network attacks of DOS, R2L, U2R, Probe and combination of attacks or not and to see the general trends that may help us in model selection and hyper parameter selection. To achieve used machine learning classification methods to fit a function that can predict the discrete class of new input.

The repository is a learning exercise to:

- Apply the fundamental concepts of machine learning from an available dataset and evaluate and interpret my results and justify my interpretation based on observed dataset.
- Create notebooks that serve as computational records and document my thought process and investigate the network connection whether attacked or not to analyses the data set.
- Evaluate and analyses statistical and visualized results, which find the standard patterns for all regiments.

## 1.6 PROJECT GOALS:

- Exploration data analysis of variable identification
  - Loading the given dataset
  - Import required libraries packages
  - Analyze the general properties
  - Find duplicate and missing values
  - Checking unique and count values
- Uni-variate data analysis
  - Rename, add data and drop the data
  - To specify data type
- Exploration data analysis of bi-variate and multi-variate
  - Plot diagram of pair plot, heatmap, bar chart and Histogram
- Method of Outlier detection with feature engineering
  - Pre-processing the given dataset
  - Splitting the test and training dataset
  - Comparing the Decision tree and Logistic regression model and random forest
- Comparing algorithm to predict the result
  - Based on the best accuracy

## **CHAPTER-2**

### **LITERATURE SURVEY**

#### **2.1 LITERATURE SURVEY**

A literature review is a body of text that aims to review the critical points of current knowledge on and/or methodological approaches to a particular topic. It is secondary sources and discuss published information in a particular subject area and sometimes information in a particular subject area within a certain time period. Its ultimate goal is to bring the reader up to date with current literature on a topic and forms the basis for another goal, such as future research that may be needed in the area and precedes a research proposal and may be just a simple summary of sources. Usually, it has an organizational pattern and combines both summary and synthesis.

A summary is a recap of important information about the source, but a synthesis is a re-organization, reshuffling of information. It might give a new interpretation of old material or combine new with old interpretations or it might trace the intellectual progression of the field, including major debates. Depending on the situation, the literature review may evaluate the sources and advise the reader on the most pertinent or relevant of them.

Loan default trends have been long studied from a socio-economic stand point. Most economics surveys believe in empirical modeling of these complex systems in order to be able to predict the loan default rate for a particular individual. The use of machine learning for such tasks is a trend which it is observing now. Some of the surveys to understand the past and present perspective of loan approval or not.

#### **Review of Literature Survey**

**Title** : A Prediction Model of DoS Attack's Distribution Discrete Probability

**Author** : Wentao Zhao, Jianping Yin and Jun Long

**Year** : 2008

The process of prediction analysis is a process of using some method or technology to explore or stimulate some unknown, undiscovered or complicated intermediate processes based on previous and present states and then speculated the results. In an early warning system, accurate prediction of DoS attacks is the prime aim in the network offence and defense task. Detection based on abnormality is effective to detect DoS attacks. A various studies focused on DoS attacks from different respects. However, these methods required a priori knowledge being a necessity and were difficult to discriminate between normal burst traffics and flux of DoS attacks. Moreover, they also required a large number of history records and cannot make the prediction for such attacks efficiently. Based on data from flux inspecting and intrusion detection, it proposed a prediction model of DOS attack's distribution discrete probability based on clustering method of genetic algorithm and Bayesian method and the clustering problem first, and then utilizes the genetic algorithm to implement the optimization of clustering methods. Based on the optimized clustering on the sample data, we get various categories of the relation between traffics and attack amounts, and then builds up several prediction sub-models about DoS attack. Furthermore, according to the Bayesian method and deduce discrete probability calculation about each sub-model and then get the distribution discrete probability prediction model for DoS attack. This paper begins with the relation exists between network traffic data and the amount of DoS attack, and then proposes a clustering method based on the genetic optimization algorithm to implement the classification of DoS attack data. This method first gets the proper partition of the relation between the network traffic and the amount of DoS attack based on the optimized clustering and builds the prediction sub-models of DoS attack.

**Title** : Apriori Viterbi Model for Prior Detection of Socio-Technical Attacks  
Social Network

**Author** : Preetish Ranjan, Abhishek Vaish

**Year** : 2014

Socio-technical attack is an organized approach which is defined by the interaction among people through maltreatment of technology with some of the malicious intent to attack the social structure based on trust and faith. Awful advertisement over internet and mobile phones may defame a person, organization, group and brand value in society which may be proved to be fatal. People are always very sensitive towards their religion therefore mass spread of manipulated information against their religious belief may create pandemonium in the society and can be one of the reasons for social riots, political misbalance etc. Cyber-attack on water, electricity, finance, healthcare, food and transportation system are may create chaos in society within few minutes and may prove even more destructive than that of a bomb as it does not attack physically but it attacks on the faith and trust which is the basic pillar of our social structure. Trust is a belief that the person who is being trusted will do what is being expected for and it starts from the family which grows to build a society. Trust for information may be established if it either comes from genuine source or information is validated by authentic body so that there is always a feeling of security and optimism. In the huge and complex social network formed using cyberspace or telecommunication technology, the identification or prediction of any kind of socio-technical attack is always difficult. This challenge creates an opportunity to explore different methodologies, concepts and algorithms used to identify these kinds of community on the basis of certain pattern, properties, structure and trend in their linkage. It tries to find the hidden information in huge social network by compressing it in small networks through apriori algorithm and then diagnosed using viterbi algorithm to predict the most probable pattern of conversation to be followed in the network and if this pattern matches with the existing pattern of criminals, terrorists and hijackers then it may be helpful to generate some kind of alert before crime.

Due to emergence of internet on mobile phone, the different social networks such as on social networking sites, blogs, opinion, ratings, review, serial bookmarking, social news, media sharing, Wikipedia led the people to disperse any kind of information very easily. Rigorous analysis of these patterns can reveal some very undisclosed and important information explicitly whether that person is conducting malignant or harmless communications with a particular user and may be a reason for any kind of

socio technical attacks. From the above simulation done on CDR, it may be concluded that if this kind of simulation applied on networks based on the internet and if we are in the position to get the data which could be transformed in transition and emission matrix then several kinds of prediction may be drawn which will be helpful to take our decisions.

**Title** : New Attack Scenario Prediction Methodology

**Author** : Seraj Fayyad, Cristoph Meinel

**Year** : 2013

Intrusion detection systems (IDS) are used to detect the occurrence of malicious activities against IT system. Through monitoring and analyzing of IT system activities the malicious activities will be detected. In ideal case IDS generate alert(s) for each detected malicious activity and store it in IDS database. Some of stored alerts in IDS database are related. Alerts relations are differentiated from duplication relation to same attack scenario relation. Duplication relation means that the two alerts generated as a result of same malicious activity. Where same attack scenario relation means that the two related alert are generated as a result of related malicious activities. Attack scenario or multi-step attack is a set of related malicious activities run by same attacker to reach specific goal. Normal relation between malicious activities belong to same attack scenario is causal relation. Causal relation means that current malicious activity output is pre-condition to run the next malicious activity. Possible multi-step attack against a network start with information gathering about network and the information gathering is done through network Reconnaissance and fingerprinting process. Through reconnaissance network configuration and running services are identified. Through fingerprint process Operating system type and version are identified. propose a real time prediction methodology for predicting most possible attack steps and attack scenarios. Proposed methodology benefits from attacks history against network and from attack graph source data. it comes without considerable computation overload such as checking of attack plans library. It provides parallel prediction for parallel attack scenarios. Possible third attack step is to identify attack plan based on the modeled

attack graph in the past step. The attack plan usually will include the exploiting of a sequence of founded vulnerabilities. Mostly this sequence is distributed over a set of network nodes. This sequence of nodes vulnerabilities is related through causal relation and connectivity. Lastly Attacker start orderly exploits the attack scenario sequences till reaching his/her goal. Attack plan consist of many correlated malicious activities end up with attacking goal.

**Title** : Cyber Attacks Prediction Model Based on Bayesian Network

**Author** : Jinyu W1, Lihua Yin and Yunchuan Guo

**Year** : 2012

The prediction results reflect the security situation of the target network in the future, and security administrators can take corresponding measures to enhance network security according to the results. To quantitatively predict the possible attack of the network in the future, attack probability plays a significant role. It can be used to indicate the possibility of invasion by intruders. As an important kind of network security quantitative evaluation measure, attack probability and its computing methods has been studied for a long time. Many models have been proposed for performing evaluation of network security. Graphical models such as attack graphs become the main-stream approach. Attack graphs which capture the relationships among vulnerabilities and exploits show us all the possible attack paths that an attacker can take to intrude all the targets in the network. The traffics to different hosts or servers may differ from each other. The hosts or servers with big traffic may be riskier since they are often important hosts or servers, and intruders may have more contacts and understanding with them. In our cyber-attack's prediction model, they used attack graph to capture the vulnerabilities in the network. In addition, we consider 3 environment factors that are the major impact factors of the cyber-attacks in the future. They are the value of assets in the network, the usage condition of the network and the attack history of the network. Cyber-attacks prediction is an important part of risk management. Existing cyber-attacks prediction methods did not fully consider the specific environment factors of the target network, which may make the results deviate from the

true situation. In this paper, we propose a cyber-attacks prediction model based on Bayesian network. We use attack graphs to represent all the vulnerabilities and possible attack paths. Then we capture the using environment factors using Bayesian network model. Cyber-attacks predictions are performed on the constructed Bayesian network.

**Title** : A Prediction Model of DoS Attack's Distribution Discrete Probability

**Author** : Wentao Zhao, Jianping Yin

**Year** : 2008

This paper begins with the relation exists between network traffic data and the amount of DoS attack, and then proposes a clustering method based on the genetic optimization algorithm to implement the classification of DoS attack data. This method first gets the proper partition of the relation between the network traffic and the amount of DoS attack based on the optimized clustering and builds the prediction sub-models of DoS attack. Meanwhile, with the Bayesian method, the calculation of the output probability corresponding to each sub-model is deduced and then the distribution of the amount of DoS attack in some range in future is obtained. This paper describes the clustering problem first, and then utilizes the genetic algorithm to implement the optimization of clustering methods. Based on the optimized clustering on the sample data, we get various categories of the relation between traffics and attack amounts, and then builds up several prediction sub-models about DoS attack. Furthermore, according to the Bayesian method, we deduce discrete probability calculation about each sub-model and then get the distribution discrete probability prediction model for DoS attack.

**Title** : Adversarial Examples: Attacks and Defenses for Deep Learning

**Author** : Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li

**Year** : 2019

It reviewed the recent findings of adversarial examples in DNNs. We investigated the existing methods for generating adversarial examples. A taxonomy of adversarial examples was proposed. We also explored the applications and

countermeasures for adversarial examples. This paper attempted to cover the state-of-the-art studies for adversarial examples in the DL domain. Compared with recent work on adversarial examples, we analyzed and discussed the current challenges and potential solutions in adversarial examples. However, deep neural networks (DNNs) have been recently found vulnerable to well-designed input samples called adversarial examples. Adversarial perturbations are imperceptible to human but can easily fool DNNs in the testing/deploying stage. The vulnerability to adversarial examples becomes one of the major risks for applying DNNs in safety-critical environments. Therefore, attacks and defenses on adversarial examples draw great attention. In this paper, we review recent findings on adversarial examples for DNNs, summarize the methods for generating adversarial examples, and propose taxonomy of these methods. Under the taxonomy, applications for adversarial examples are investigated. We further elaborate on countermeasures for adversarial examples. In addition, three major challenges in adversarial examples and the potential solutions are discussed.

**Title** : Distributed Secure Cooperative Control Under Denial-of-Service Attacks from Multiple Adversaries

**Author** : Wenying Xu, Guoqiang Hu

**Year** : 2019

This paper has investigated the distributed secure control of multiagent systems under DoS attacks. We focus on the investigation of a jointly adverse impact of distributed DoS attacks from multiple adversaries. In this scenario, two kinds of communication schemes, that is, sample-data and event-triggered communication schemes, have been discussed and, then, a fully distributed control protocol has been developed to guarantee satisfactory asymptotic consensus. Note that this protocol has strong robustness and high scalability. Its design does not involve any global information, and its efficiency has been proved. For the event-triggered case, two effective dynamical event conditions have been designed and implemented in a fully distributed way, and both of them have excluded Zeno behavior. Finally, a simulation example has been provided to verify the effectiveness of theoretical analysis. Our future

research topics focus on fully distributed event/self-triggered control for linear/nonlinear multiagent systems to gain a better understanding of fully distributed control.

## **CHAPTER-3**

### **AIM AND SCOPE OF THE PRESENT INVESTIGATION**

#### **3.1 PROJECT PROPOSAL**

The proposed model is to build a machine learning model for anomaly detection. Anomaly detection is an important technique for recognizing fraud activities, suspicious

activities, network intrusion, and other abnormal events that may have great significance but are difficult to detect. The machine learning model is built by applying proper data science techniques like variable identification that is the dependent and independent variables. Then the visualization of the data is done to insights of the data. The model is build based on the previous dataset where the algorithm learn data and get trained different algorithms are used for better comparisons. The performance metrics are calculated and compared.

### **3.2 SCOPE OF THE PROJECT**

The scope of this project is to investigate a dataset of network connection attacks for KDD records for medical sector using machine learning technique. To identifying network connection is attacked or not.

### **3.3 EXISTING SYSTEM**

They proposed first to create a contrastive self-supervised learning to the anomaly detection problem of attributed networks. CoLa, is mainly consists of three components: contrastive instance pair sampling, GNN-based contrastive learning model, and multi round sampling-based anomaly score computation. Their model captures the relationship between each node and its neighboring structure and uses an anomaly-related objective to train the contrastive learning model. We believe that the proposed framework opens a new opportunity to expand self-supervised learning and contrastive learning to increasingly graph anomaly detection applications. The multiround predicted scores by the contrastive learning model are further used to evaluate the abnormality of each node with statistical estimation. The training phase and the inference phase. In the training phase, the contrastive learning model is trained with sampled instance pairs in an unsupervised fashion. After that the anomaly score for each node is obtained in the inference phase.

#### **3.3.1 Disadvantages:**

1. The performance is not good and it get complicated for other networks.

2. The performance metrics like recall F1 score and comparison of machine learning algorithm is not done.

### **3.4 PREPARING DATASET:**

This Dataset contains 3000 records of features. It is classified into 4 classes.

- DOS Attack
- R2L Attack
- U2R Attack
- Probe Attack

### **3.5 SYSTEM STUDY**

#### ***3.5.1 Classification of Attacks:***

The data set in KDD Cup99 have normal and 22 attack type data with 41 features and all generated traffic patterns end with a label either as 'normal' or any type of 'attack' for upcoming analysis. There are varieties of attacks which are entering into the network over a period of time and the attacks are classified into the following four main classes.

- Denial of Service (DoS)
- User to Root (U2R)
- Remote to User (R2L)
- Probing

#### ***Denial of Service (DoS):***

Denial of Service is a class of attacks where an attacker makes some computing or memory resource too busy or too full to handle legitimate requests, denying legitimate users access to a machine. The different ways to launch a DoS attack are by abusing the computer's legitimate features,

- By targeting the implementation bugs
- By exploiting the misconfiguration of the systems

DoS attacks are classified based on the services that an attacker renders unavailable to legitimate users.

#### ***User to Root:***

In User to Root attack, an attacker starts with access to a normal user account on the system and gains root access. Regular programming mistakes and environment assumption give an attacker the opportunity to exploit the vulnerability of root access.

#### ***Remote to User:***

In Remote to User attack, an attacker sends packets to a machine over a network that exploits the machine's vulnerability to gain local access as a user illegally. There are different types of R2L attacks and the most common attack in this class is done by using social engineering.

#### ***Probing:***

Probing is a class of attacks where an attacker scans a network to gather information in order to find known vulnerabilities. An attacker with a map of machines and services that are available on a network can manipulate the information to look for exploits. There are different types of probes: some of them abuse the computer's legitimate features and some of them use social engineering techniques. This class of attacks is the most common because it requires very little technical expertise.

### ***3.5.2 Attacks Summary:***

This chapter outlines the structure of the dataset used in the proposed work. The various kinds of features such as discrete and continuous features are studied with a focus on their role in the attack. The attacks are classified with a brief introduction to each. The next chapter discusses the clustering and classification of the data with a direction to learning by machine.

| <b>Dos</b>    | <b>R2L</b>     | <b>U2R</b>      | <b>Probe</b> |
|---------------|----------------|-----------------|--------------|
| Back          | FTP Write      | Load module     | Ip sweep     |
| Neptune       | Multihop       | Rerl            | Nmap         |
| Land          | Phf            | Rootkit         | Satan        |
| Pod           | Spy            | Buffer overflow | Port sweep   |
| Smurf         | Warezcclient   | Ps              | Msscan       |
| Teardrop      | Warezmaster    | Sql attack      | saint        |
| Apache2       | Imap           | xterm           |              |
| Mail bomb     | Guess password |                 |              |
| Process table | http tunnel    |                 |              |
| UDP Storm     | named          |                 |              |
|               | send mail      |                 |              |
|               | snmpget attack |                 |              |
|               | snmp guess     |                 |              |
|               | worm           |                 |              |
|               | xlock          |                 |              |
|               | xsnoop         |                 |              |

**Table 3.1: Attack Types Grouped to respective Class**

| <b>Types of Attacks</b> | <b>Description</b>                                                                                           |
|-------------------------|--------------------------------------------------------------------------------------------------------------|
| back                    | Denial of service attack against apache web server where a client requests a URL containing many backslashes |
| neptune                 | Syn flood denial of service on one or more ports                                                             |
| land                    | Denial of service where a remote host is sent a UDP packet with the                                          |

|             |                                                                                                                                                                                                           |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             | same source and destination                                                                                                                                                                               |
|             |                                                                                                                                                                                                           |
| pod         | Denial of service ping of death                                                                                                                                                                           |
| smurf       | Denial of service icmp echo reply flood                                                                                                                                                                   |
| teardrop    | Denial of service where mis-fragmented UDP packets cause some systems to reboot                                                                                                                           |
| multihop    | Multi-day scenario in which a user first breaks into one machine                                                                                                                                          |
| phf         | Exploitable CGI script which allows a client to execute arbitrary commands on a machine with a mis-configured web server.                                                                                 |
| spy         | Multi-day scenario in which a user breaks into a machine with the purpose of finding important information where the user tries to avoid detection. Uses several different exploit methods to gain access |
| warezclient | Users downloading illegal software which was previously posted via anonymous FTP by the warezmaster                                                                                                       |
| warezmaster | Anonymous FTP upload of Warez (usually illegal copies of copy writed software) onto FTP server                                                                                                            |
| Imap        | Remote buffer overflow using imap port leads to root shell                                                                                                                                                |
| loadmodule  | Non-stealthy loadmodule attack which resets IFS for a normal user and creates a root shell                                                                                                                |

|           |                                                                                                                  |
|-----------|------------------------------------------------------------------------------------------------------------------|
| Perl      | Perl attack which sets the user id to root in a perl script and creates a root shell                             |
| rootkit   | Multi-day scenario where a user installs one or more components of a rootkit                                     |
| ipsweep   | Surveillance sweep performing either a port sweep or ping on multiple host addresses                             |
| nmap      | Network mapping using the nmap tool. Mode of exploring network will vary-options include SYN                     |
| satan     | Network probing tool which looks for well-known weaknesses. Operates at three different levels. Level 0 is light |
| portsweep | Surveillance sweep through many ports to determine which services are supported on a single host                 |
| dict      | Guess passwords for a valid user using simple variants of the account name over a telnet connection              |
| eject     | Buffer overflow using eject program on Solaris. Leads to a user->root transition if successful                   |
| ffb       | Buffer overflow using the ffbconfig UNIX system command leads to root shell                                      |

|           |                                                                                                       |
|-----------|-------------------------------------------------------------------------------------------------------|
| format    | Buffer overflow using the fdformat UNIX system command leads to root shell                            |
| ftp-write | Remote FTP user creates .rhost file in world writable anonymous FTP directory and obtains local login |
| guest     | Try to guess password via telnet for guest account                                                    |
| syslog    | Denial of service for the syslog service connects to port 514 with unresolvable source ip             |
| warez     | User logs into anonymous FTP site and creates a hidden directory                                      |

***Table 3.2: Description of Attacks***

## **3.6 FEASIBILITY STUDY**

### ***3.6.1 Data Wrangling:***

In this section of the report will load in the data, check for cleanliness, and then trim and clean given dataset for analysis. Make sure that the document steps carefully and justify for cleaning decisions.

### ***3.6.2 Data collection:***

The data set collected for predicting given data is split into Training set and Test set. Generally, 7:3 ratios are applied to split the Training set and Test set. The Data Model which was created using Random Forest, logistic, Decision tree algorithms and Support vector classifier (SVC) are applied on the Training set and based on the test result accuracy, Test set prediction is done.

### ***3.6.3 Preprocessing:***

The data which was collected might contain missing values that may lead to inconsistency. To gain better results data need to be preprocessed so as to improve the efficiency of the algorithm. The outliers have to be removed and also variable conversion need to be done.

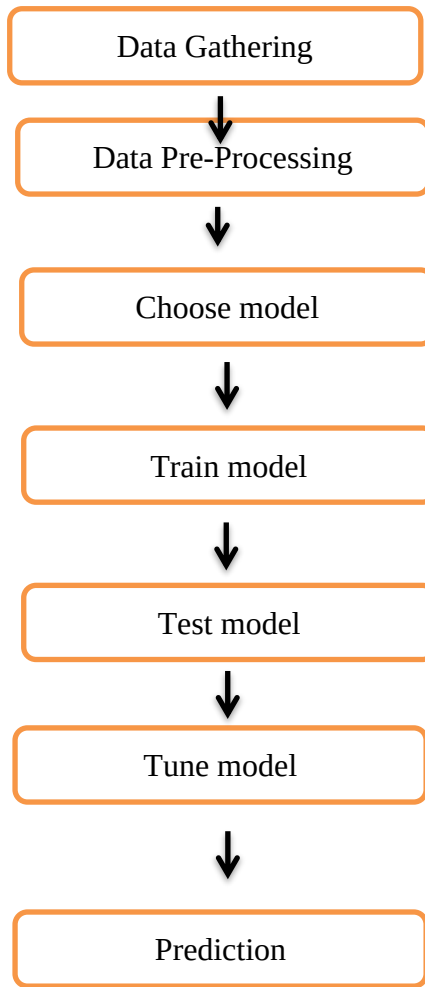
### ***3.6.4 Building the classification model:***

The prediction of Phishing Website, A Random Forest Algorithm prediction model is effective because of the following reasons: It provides better results in classification problem.

- It is strong in preprocessing outliers, irrelevant variables, and a mix of continuous, categorical and discrete variables.
- It produces out of bag estimate error which has proven to be unbiased in many tests and it is relatively easy to tune with.

### ***3.6.5 Construction of a Predictive Model:***

Machine learning needs data gathering have lot of past data. Data gathering have sufficient historical data and raw data. Before data pre-processing, raw data can't be used directly. It's used to preprocess then, what kind of algorithm with model. Training and testing this model working and predicting correctly with minimum errors. Tuned model involved by tuned time to time with improving the accuracy.



***Fig 3.1: Process of dataflow diagram***

## **CHAPTER-4**

### **EXPERIMENTAL OR MATERIALS AND METHODS; ALGORITHMS USED**

#### **4.1 PROJECT REQUIREMENTS**

Requirements are the basic constraints that are required to develop a system. Requirements are collected while designing the system. The following are the requirements that are to be discussed.

1. Functional requirements
2. Non-Functional requirements
3. Environment requirements
  - A. Hardware requirements
  - B. software requirements

#### **4.1.1 Functional Requirements:**

The software requirements specification is a technical specification of requirements for the software product. It is the first step in the requirements analysis process. It lists requirements of a particular software system. The following details to follow the special libraries like SK-learn, pandas, NumPy, matplotlib and seaborn.

#### **4.1.2 Non-Functional Requirements:**

Process of functional steps,

1. Problem defines
2. Preparing data
3. Evaluating algorithms
4. Improving results
5. Prediction the result

#### **4.1.3 Environment Requirements:**

1. Software Requirements:

|                  |                                  |
|------------------|----------------------------------|
| Operating System | : Windows                        |
| Tool             | : Anaconda with Jupyter Notebook |

2. Hardware requirements:

|           |                  |
|-----------|------------------|
| Processor | : Pentium IV/III |
| Hard disk | : minimum 80 GB  |
| RAM       | : minimum 2 GB   |

## **4.2 SOFTWARE DESCRIPTION:**

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system “Conda”. The Anaconda distribution is used by over 12 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS. So, Anaconda distribution comes with more than 1,400 packages as well as the Conda package and virtual environment manager called Anaconda Navigator and it eliminates the need to learn to install each library independently.

### **4.2.1 Anaconda Navigator:**

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository. Anaconda. Now, if you are primarily doing data science work, Anaconda is also a great option. Anaconda is created by Continuum Analytics, and it is a Python distribution that comes preinstalled with lots of useful python libraries for data science.

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.

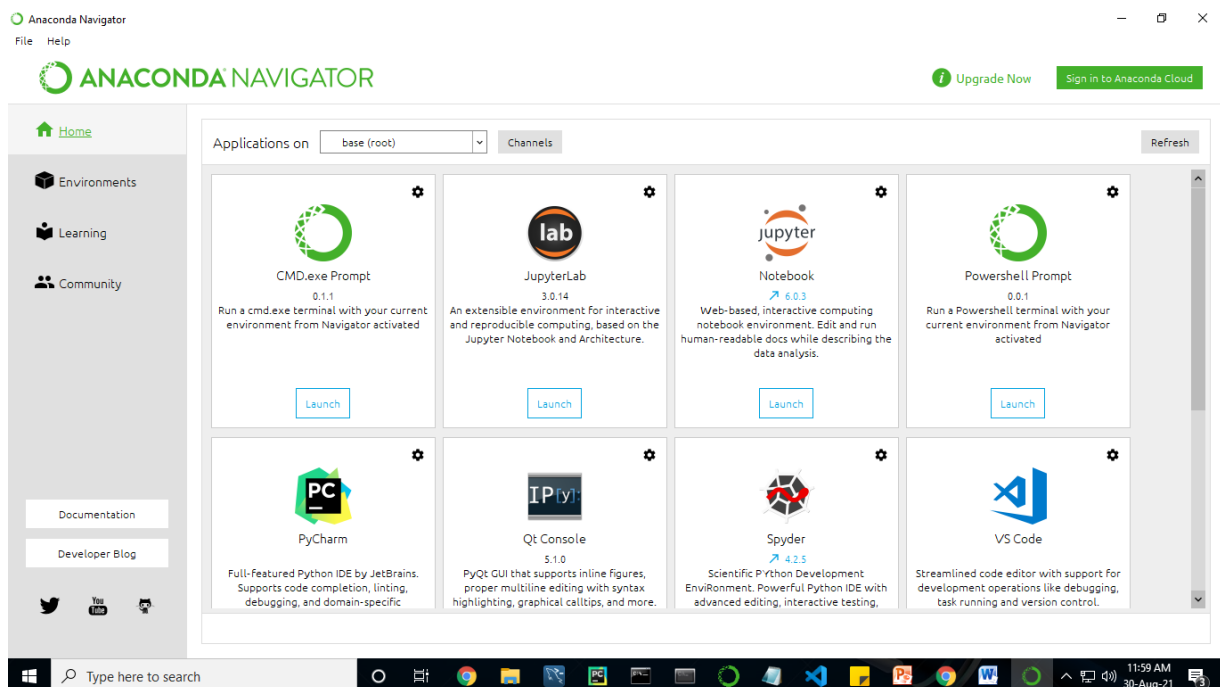
The command-line program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find

the packages you want, install them in an environment, run the packages, and update them – all inside Navigator.

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- Spyder
- PyCharm
- VSCode
- Glueviz
- Orange 3 App
- RStudio
- Anaconda Prompt (Windows only)
- Anaconda PowerShell (Windows only)



**Fig 4.1: Anaconda Navigator Interface**

### **4.2.2 Jupyter Notebook:**

This website acts as “meta” documentation for the Jupyter ecosystem. It has a collection of resources to navigate the tools and communities in this ecosystem, and to help you get started. Project Jupyter is a project and community whose goal is to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages". It was spun off from IPython in 2014 by Fernando Perez.

Notebook documents are documents produced by the Jupyter Notebook App, which contain both computer code (e.g., python) and rich text elements (paragraph, equations, figures, links, etc....). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc.) as well as executable documents which can be run to perform data analysis.

**Installation:** The easiest way to install the Jupyter Notebook App is installing a scientific python distribution which also includes scientific python packages. The most common distribution is called Anaconda

### **Running the Jupyter Notebook:**

**Launching Jupyter Notebook App:** The Jupyter Notebook App can be launched by clicking on the Jupyter Notebook icon installed by Anaconda in the start menu (Windows) or by typing in a terminal (cmd on Windows): “jupyter notebook”

This will launch a new browser window (or a new tab) showing the Notebook Dashboard, a sort of control panel that allows (among other things) to select which notebook to open.

When started, the Jupyter Notebook App can access only files within its start-up folder (including any sub-folder). No configuration is necessary if you place your notebooks in your home folder or subfolders. Otherwise, you need to choose a Jupyter Notebook App start-up folder which will contain all the notebooks.

Save notebooks: Modifications to the notebooks are automatically saved every few minutes. To avoid modifying the original notebook, make a copy of the notebook document (menu file -> make a copy...) and save the modifications on the copy.

**Executing a notebook:** Download the notebook you want to execute and put it in your notebook folder (or a sub-folder of it).

- ❖ Launch the jupyter notebook app
- ❖ In the Notebook Dashboard navigate to find the notebook: clicking on its name will open it in a new browser tab.
- ❖ Click on the menu Help -> User Interface Tour for an overview of the Jupyter Notebook App user interface.
- ❖ You can run the notebook document step-by-step (one cell a time) by pressing shift + enter.
- ❖ You can run the whole notebook in a single step by clicking on the menu Cell -> Run All.

To restart the kernel (i.e. the computational engine), click on the menu Kernel -> Restart. This can be useful to start over a computation from scratch (e.g. variables are deleted, open files are closed, etc...).

**Purpose:** To support interactive data science and scientific computing across all programming languages.

**File Extension:** An IPYNB file is a notebook document created by Jupyter Notebook, an interactive computational environment that helps scientists manipulate and analyze data using Python.

### **4.2.3 PYTHON:**

#### **Introduction:**

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and

functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible. Python 2 was discontinued with version 2.7.18 in 2020. Python consistently ranks as one of the most popular programming languages

### ***History:***

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to ABC programming language, which was inspired by SETL, capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's Benevolent Dictator for Life, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. In January 2019, active Python core developers elected a 5-member "Steering Council" to lead the project. As of 2021, the current members of this council are Barry Warsaw, Brett Cannon, Carol Willing, Thomas Wouters, and Pablo Galindo Salgado.

Python 2.0 was released on 16 October 2000, with many major new features, including a cycle-detecting garbage collector and support for Unicode.

Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely backward-compatible. Many of its major features were backported to Python 2.6.x and 2.7.x version series. Releases of Python 3 include the 2 to 3 utilities, which automates (at least partially) the translation of Python 2 code to Python 3.

Python 2.7's end-of-life date was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to

Python 3. No more security patches or other improvements will be released for it. With Python 2's end-of-life, only Python 3.6.x and later are supported.

Python 3.9.2 and 3.8.8 were expedited as all versions of Python (including 2.7) had security issues, leading to possible remote code execution and web cache poisoning.

### ***Design Philosophy & Feature:***

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by meta-programming and meta-objects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has filter, map and reduce functions; list comprehensions, dictionaries, sets, and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

The language's core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as:

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

Rather than having all of its functionality built into its core, Python was designed to be highly extensible (with modules). This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily

extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

### 4.3 PYTHON LIBRARIES NEEDED

There are many libraries in python. In those we only use few main libraries needed.

#### 4.3.1 *NUMPY LIBRARY:*

NumPy is an open-source numerical Python library. NumPy contains a multi-dimensional array and matrix data structures. It can be utilized to perform a number of mathematical operations on arrays such as trigonometric, statistical, and algebraic routines like mean, mode, standard deviation etc.,

Installation- (<https://numpy.org/install/>)

```
pip install NUMPY
```

Here we mainly use array, to find mean and standard deviation.

#### 4.3.2 *PANDAS LIBRARY:*

Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the DataFrame. DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables. There are several ways to create a DataFrame.

Installation- ([https://pandas.pydata.org/getting\\_started.html](https://pandas.pydata.org/getting_started.html))

```
pip install PANDAS
```

Here we use pandas for reading the csv files, for grouping the data, for cleaning the data using some operations.

#### 4.3.3 *MATPLOTLIB LIBRARY:*

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible. Use interactive figures that can zoom, pan, update, visualize etc.,

Installation- (<https://matplotlib.org/users/installing.html>)

**pip install Matplotlib**

Here we use pyplot mainly for plotting graphs.

matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

#### **4.3.4 SCIKIT-LEARN LIBRARY:**

Scikit-learn is a free machine learning library for the Python. It features various algorithms like support vector machine, random forests, regression and k-neighbors, and it also supports Python numerical and scientific libraries like NumPy and SciPy.

Installation- (<https://scikit-learn.org/stable/install.html>)

**Pip install Scikit-Learn**

Here use scikit-learns regression methods for prediction purpose.

#### **4.3.5 TKINTER:**

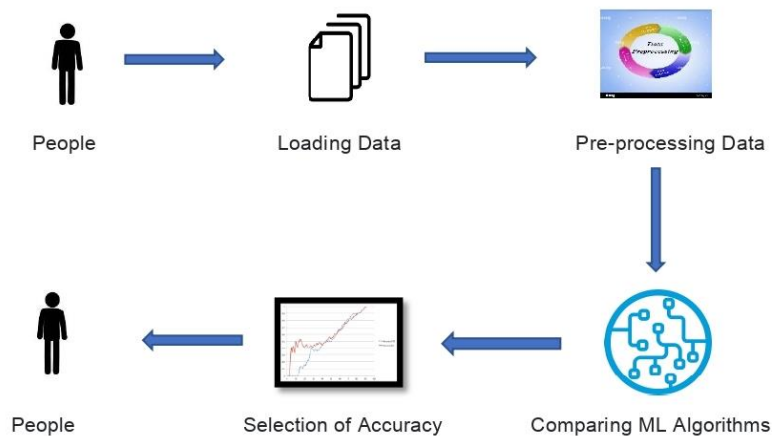
- Standard python interface to the GUI toolkit.
- Accessible to everybody and reusable in various contexts.

### **4.4 SYSTEM ARCHITECTURE**

A modular design reduces complexity, facilitates change (a critical aspect of software maintainability), and results in easier implementation by encouraging parallel development of different part of system. Software with effective modularity is easier to develop because function may be compartmentalized and interfaces are simplified. Software architecture embodies modularity that is software is divided into separately

named and addressable components called modules that are integrated to satisfy problem requirements.

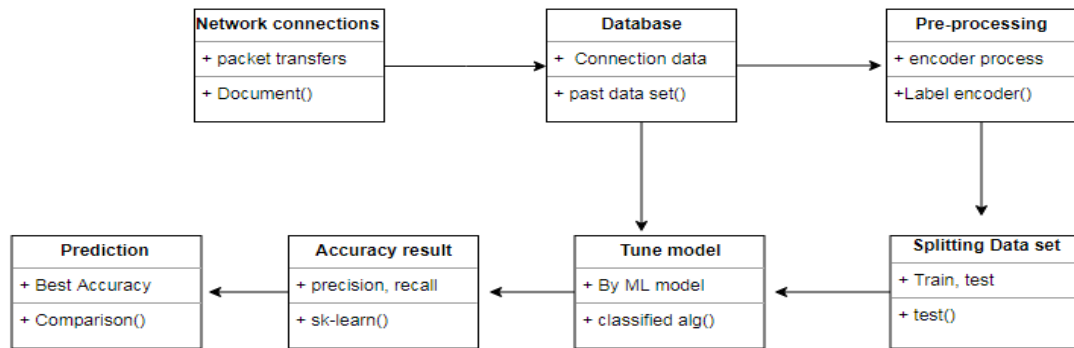
Modularity is the single attribute of software that allows a program to be intellectually manageable. The five important criteria that enable us to evaluate a design method with respect to its ability to define an effective modular design are: Modular decomposability, Modular Comps ability, Modular Understand ability, Modular continuity, Modular Protection.



**Fig 4.2: System Architecture**

## **4.5 UML DIAGRAMS**

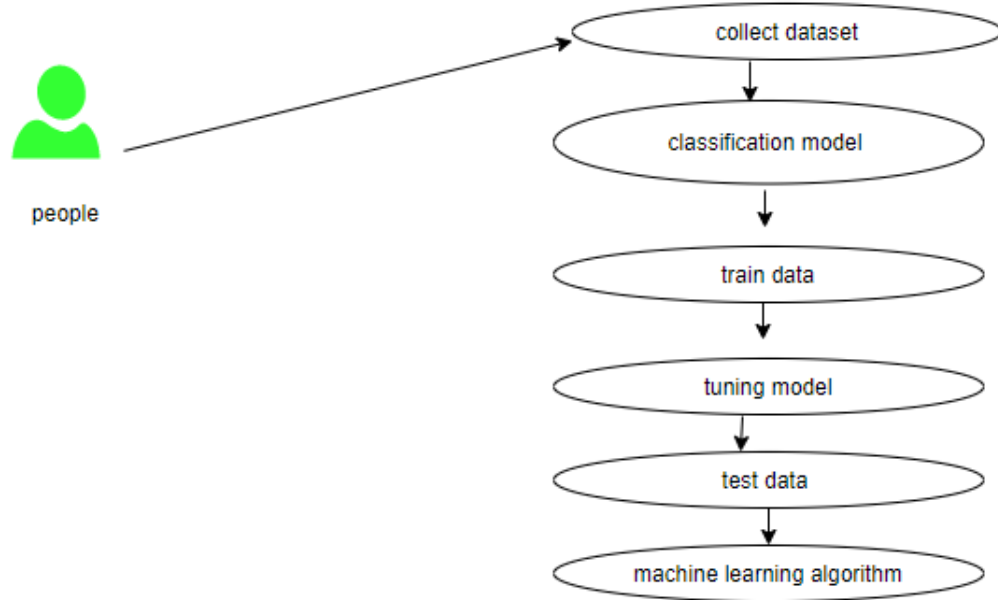
### **4.5.1 Class Diagram**



**Fig 4.3: Class Diagram**

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So, a collection of class diagrams represents the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance Responsibility (attributes and methods) of each class should be clearly identified for each class minimum number of properties should be specified and because, unnecessary properties will make the diagram complicated. Use notes whenever required to describe some aspect of the diagram and at the end of the drawing it should be understandable to the developer/coder. Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

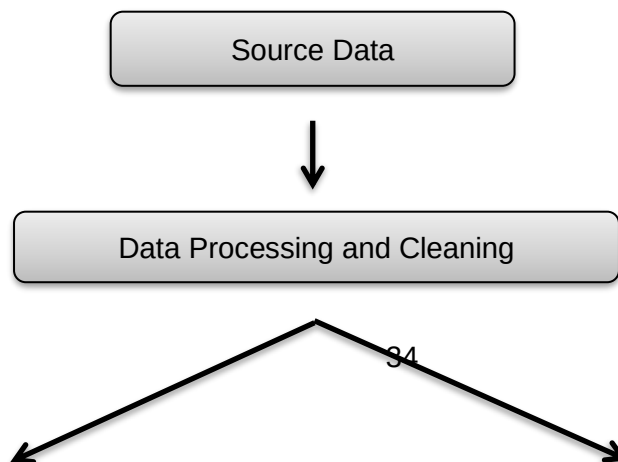
#### **4.5.2 Use Case Diagram:**

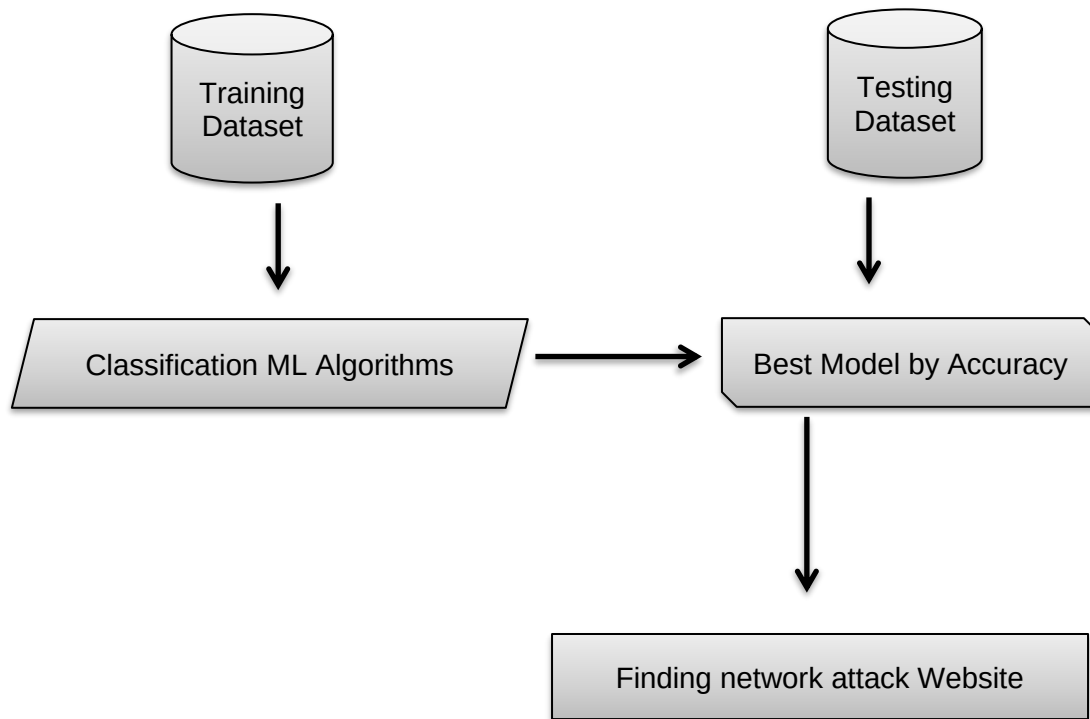


**Fig 4.4: Use Case Diagram**

Use case diagrams are considered for high level requirement analysis of a system. So, when the requirements of a system are analyzed, the functionalities are captured in use cases. So, it can say that uses cases are nothing but the system functionalities written in an organized manner.

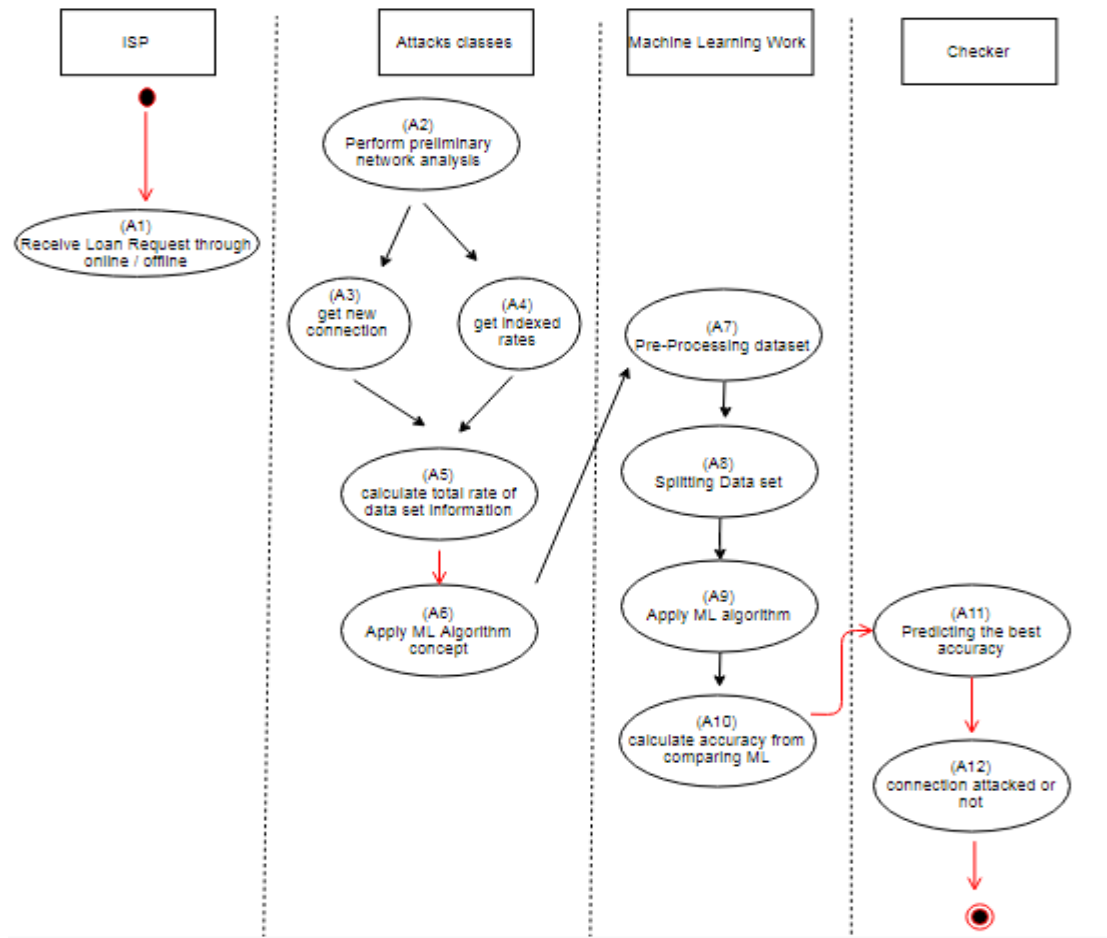
#### **4.5.3 Workflow Diagram:**





***Fig 4.5: Workflow Diagram***

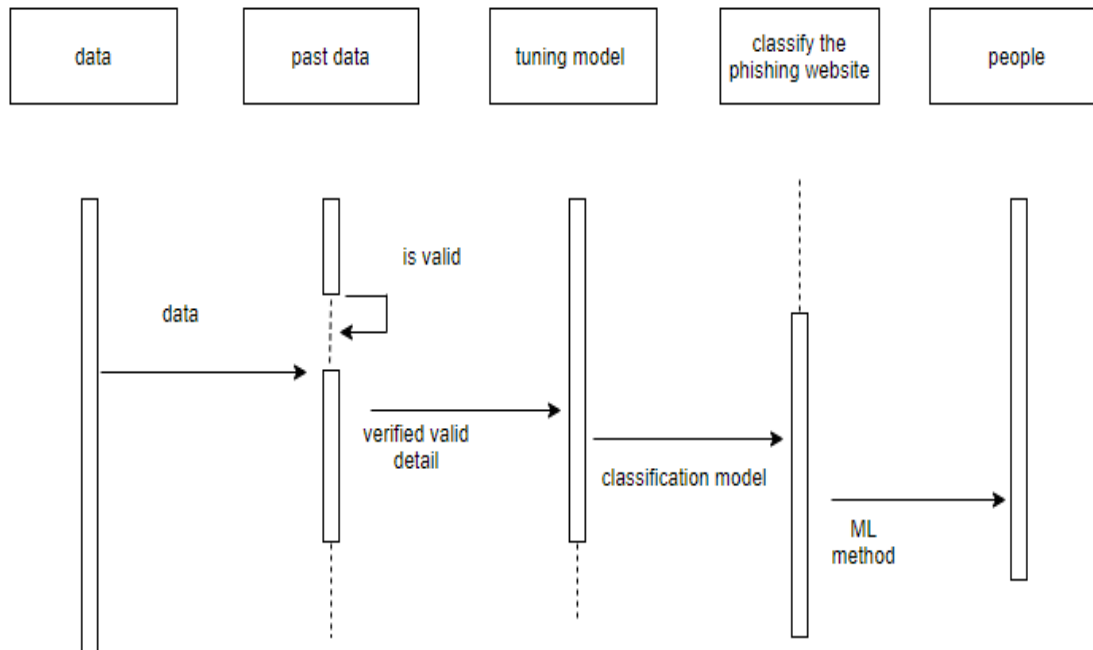
#### ***4.5.4 Activity Diagram:***



**Fig 4.6: Activity Diagram**

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is some time considered as the flow chart. Although the diagrams look like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.

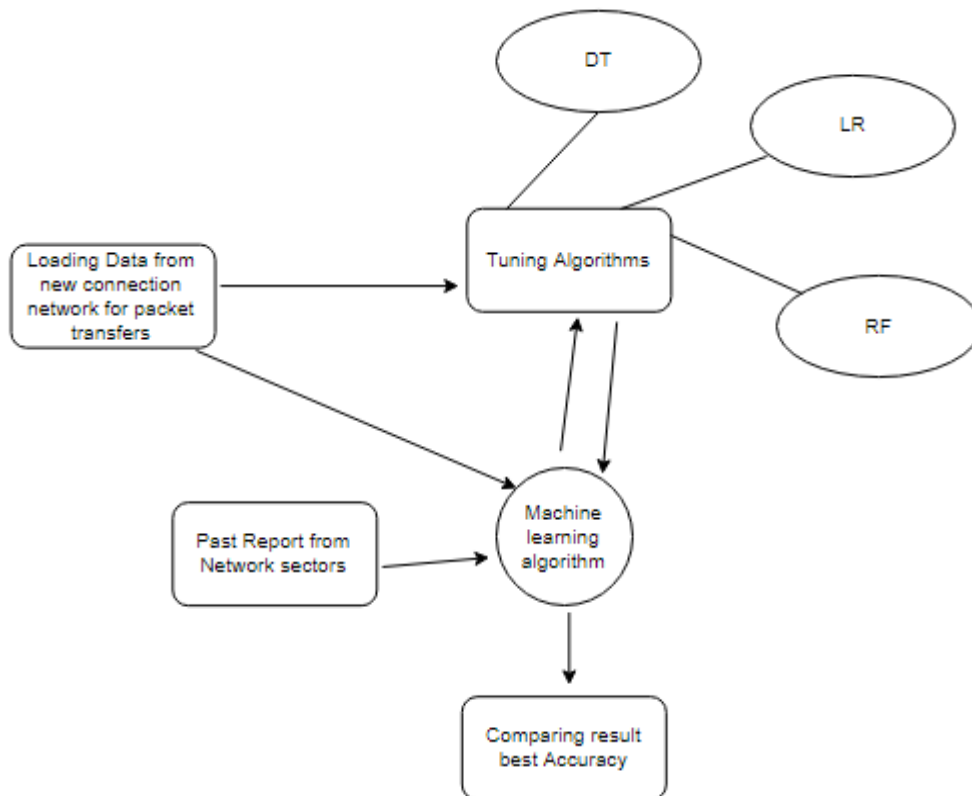
#### 4.5.5 Sequence Diagram:



**Fig 4.7: Sequence Diagram**

Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modeling, which focuses on identifying the behavior within your system. Other dynamic modeling techniques include activity diagramming, communication diagramming, timing diagramming, and interaction overview diagramming. Sequence diagrams, along with class diagrams and physical data models are in my opinion the most important design-level models for modern business application development.

#### 4.5.6 Entity Relationship Diagram (ERD):



**Fig 4.8: Entity Relationship Diagram (ERD)**

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An ERD is a data modeling technique that can help define business processes and be used as the foundation for a relational database. Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational

database is rolled out, an ERD can still serve as a referral point, should any debugging or business process re-engineering be needed later.

## **4.6 ALGORITHMS USED**

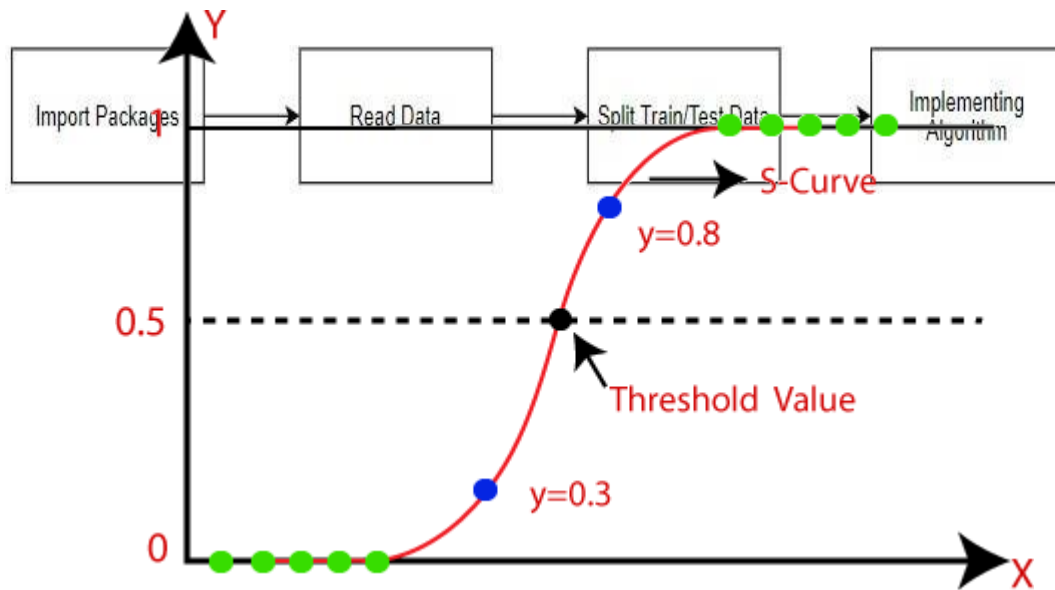
### ***Logistic Regression:***

It is a statistical method for analyzing a data set in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). The goal of logistic regression is to find the best fitting model to describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables. Logistic regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.).

In other words, the logistic regression model predicts  $P(Y=1)$  as a function of  $X$ . Logistic regression Assumptions:

- Binary logistic regression requires the dependent variable to be binary.
- For a binary regression, the factor level 1 of the dependent variable should represent the desired outcome.
- Only the meaningful variables should be included.
- The independent variables should be independent of each other. That is, the model should have little.
- The independent variables are linearly related to the log odds.

- Logistic regression requires quite large sample sizes.



**Fig 4.9: Logistic Regression**

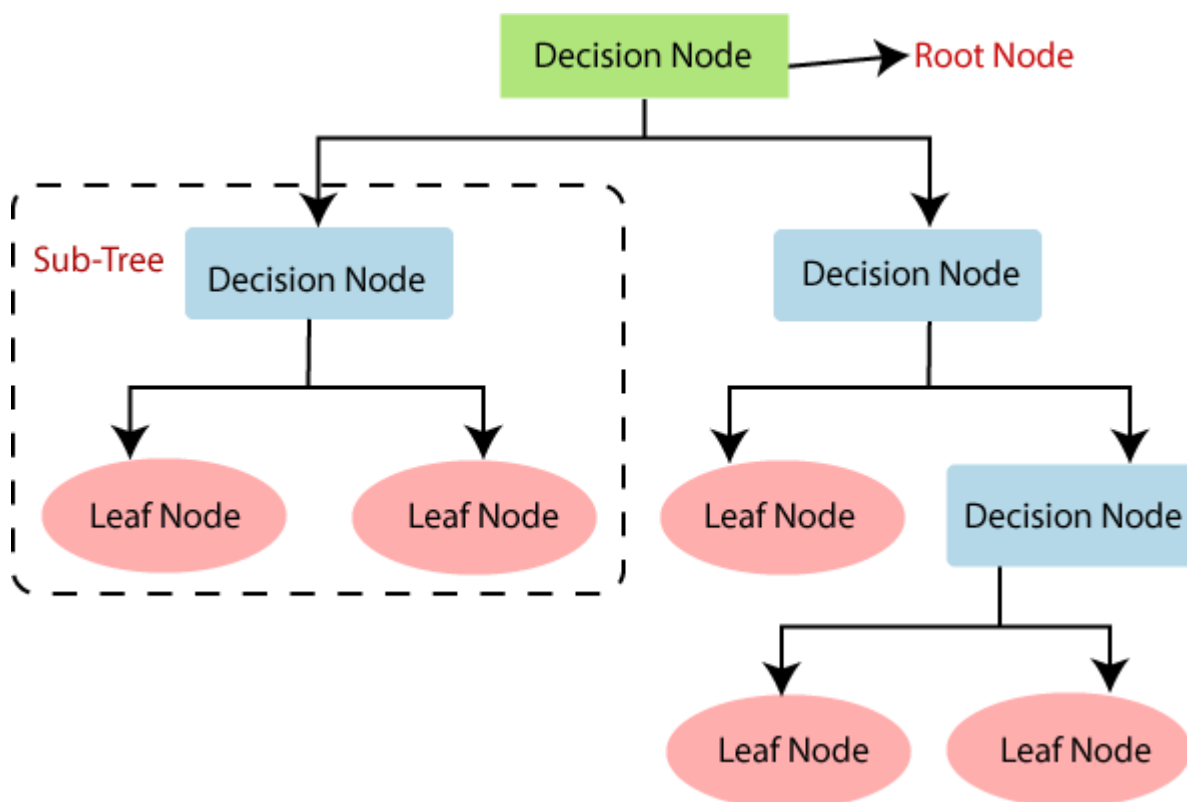
### **Decision Tree Classifier:**

It is one of the most powerful and popular algorithms. Decision-tree algorithm falls under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables. Assumptions of Decision tree:

- At the beginning, we consider the whole training set as the root.
- Attributes are assumed to be categorical for information gain, attributes are assumed to be continuous.
- On the basis of attribute values records are distributed recursively.
- We use statistical methods for ordering attributes as root or internal node.

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a data set into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. A decision node has two or more

branches and a leaf node 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. Decision tree builds classification or regression models in the form of a tree structure. It utilizes an if-then rule set which is mutually exclusive and exhaustive for classification. The rules are learned sequentially using the training data one at a time. Each time a rule is learned, the tuples covered by the rules are removed. This process is continued on the training set until meeting a termination condition. It is constructed in a top-down recursive divide-and-conquer manner. All the attributes should be categorical. Otherwise, they should be discretized in advance. Attributes in the top of the tree have more impact towards in the classification and they are identified using the information gain concept. A decision tree can be easily over-fitted generating too many branches and may reflect anomalies due to noise or outliers.



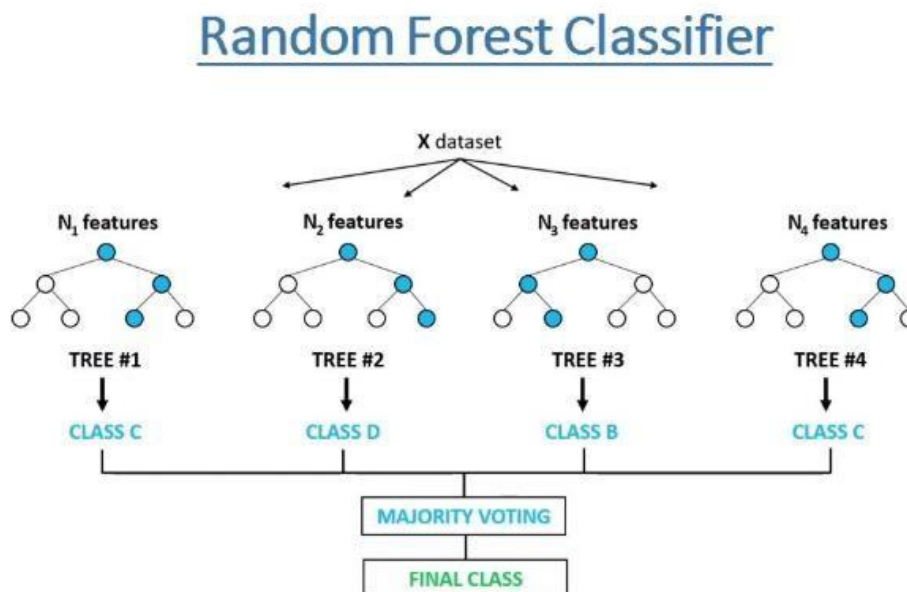
**Fig 4.10: Decision Tree Classifier**

**Random Forest Classifier:**

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of over fitting to their training set. Random forest is a type of supervised machine learning algorithm based on ensemble learning. Ensemble learning is a type of learning where you join different types of algorithms or same algorithm multiple times to form a more powerful prediction model. The random forest algorithm combines multiple algorithms of the same type i.e., multiple decision trees, resulting in a forest of trees, hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks.

The following are the basic steps involved in performing the random forest algorithm:

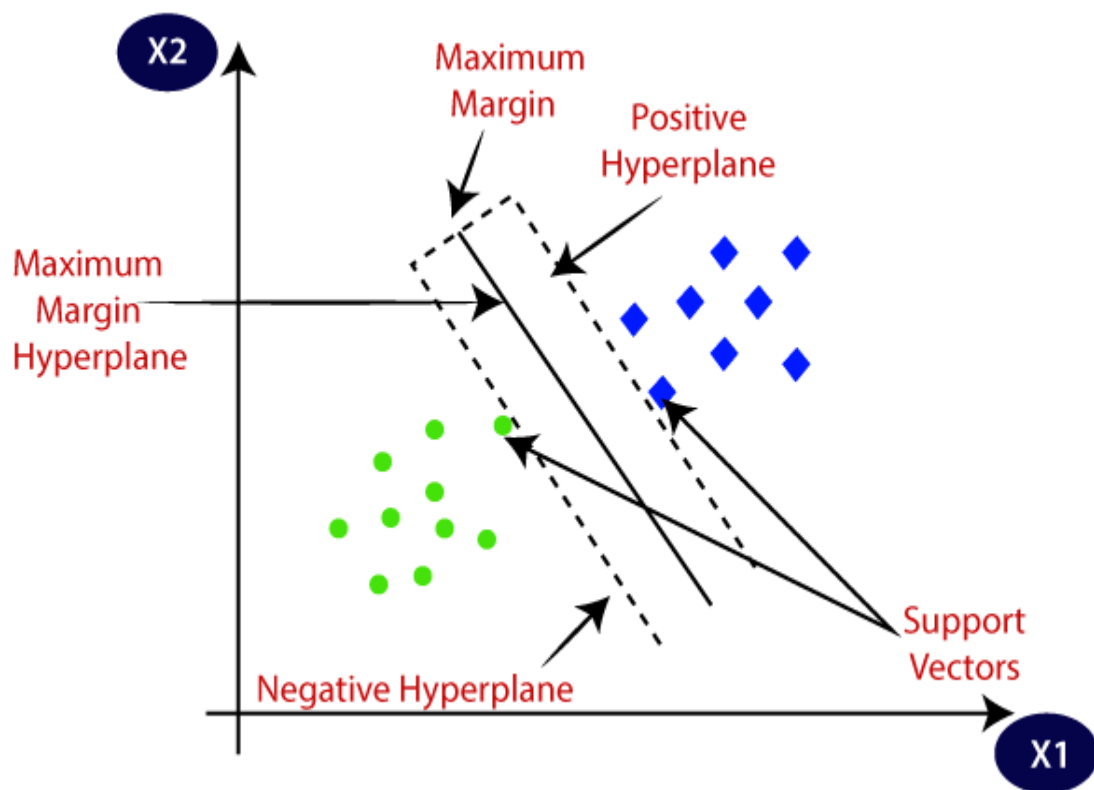
- Pick N random records from the dataset.
- Build a decision tree based on these N records.
- Choose the number of trees you want in your algorithm and repeat steps 1 and 2.



**Fig 4.11: Random Forest Classifier**

### **Support Vector Machine:**

Support Vector Classifier or SVC is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. Primarily, it is used for Classification problems in Machine Learning. The goal of the SVC algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes. So that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVC chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.



**Fig 4.12: Support Vector Classifier**

### ***Graphical Neural Network (GNN):***

A graph neural network (GNN) is a class of neural network for processing data best represented by graph data structures. They were popularized by their use in supervised learning on properties of various molecules.

- Since their inception, variants of the message passing neural network (MPNN) framework have been proposed. These models optimize GNNs for use on larger graphs and apply them to domains such as social networks, citation networks, and online communities.
- GNNs are a weak form of the Weisfeiler–Lehman graph isomorphism test, so any GNN model is at least as powerful as this test. Researcher are attempting to unite GNNs with other "geometric deep learning models to better understand how and why these models work.
- In the case of the absence of a known graph structure for example a k-nearest neighbor graph can be heuristically induced.

## **4.7 COMPARING ALGORITHM WITH PREDICTION IN THE FORM OF BEST ACCURACY RESULT**

It is important to compare the performance of multiple different machine learning algorithms consistently and it will discover to create a test harness to compare multiple different machine learning algorithms in Python with scikit-learn. It can use this test harness as a template on your own machine learning problems and add more and different algorithms to compare. Each model will have different performance characteristics. Using resampling methods like cross validation, you can get an estimate for how accurate each model may be on unseen data. It needs to be able to use these estimates to choose one or two best models from the suite of models that you have created. When have a new dataset, it is a good idea to visualize the data using different techniques in order to look at the data from different perspectives. The same idea applies to model selection. You should use a number of different ways of looking at the estimated accuracy of your machine learning algorithms in order to

choose the one or two to finalize. A way to do this is

to use different visualization methods to show the average accuracy, variance and other properties of the distribution of model accuracies.

In the next section you will discover exactly how you can do that in Python with scikit-learn. The key to a fair comparison of machine learning algorithms is ensuring that each algorithm is evaluated in the same way on the same data and it can achieve this by forcing each algorithm to be evaluated on a consistent test harness.

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. To achieving better results from the applied model in Machine Learning method of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format, for example, Random Forest algorithm does not support null values. Therefore, to execute random forest algorithm null values have to be managed from the original raw data set. And another aspect is that data set should be formatted in such a way that more than one Machine Learning and Deep Learning algorithms are executed in given dataset.

In the example below these 5 different algorithms are compared:

- Logistic Regression
- Random Forest
- Decision Tree Classifier
- Support Vector Classifier
- Graphical Neural Network

The K-fold cross validation procedure is used to evaluate each algorithm, importantly configured with the same random seed to ensure that the same splits to the training data are performed and that each algorithm is evaluated in precisely

the same way. Before that comparing algorithm, building a Machine Learning Model using install Scikit-Learn libraries. In this library package have to done preprocessing, linear model with logistic regression method, cross validating by KFold method, ensemble with random forest method and tree with decision tree classifier. Additionally, splitting the train set and test set. To predicting the result by comparing accuracy.

**False Positives (FP):** A person who will pay predicted as defaulter. When actual class is no and predicted class is yes. E.g., if actual class says this passenger did not survive but predicted class tells you that this passenger will survive.

**False Negatives (FN):** A person who default predicted as payer. When actual class is yes but predicted class in no. E.g., if actual class value indicates that this passenger survived and predicted class tells you that passenger will die.

**True Positives (TP):** A person who will not pay predicted as defaulter. These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes. E.g., if actual class value indicates that this passenger survived and predicted class tells you the same thing.

**True Negatives (TN):** A person who default predicted as payer. These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no. E.g., if actual class says this passenger did not survive and predicted class tells you the same thing.

#### **Prediction result by accuracy:**

Logistic regression algorithm also uses a linear equation with independent predictors to predict a value. The predicted value can be anywhere between negative infinity to positive infinity. It needs the output of the algorithm to be classified variable data. Higher accuracy predicting result is logistic regression model by comparing the best accuracy.

**True Positive Rate (TPR)** =  $TP / (TP + FN)$  **False Positive Rate (FPR)** =  $FP / (FP + TN)$

**Accuracy:** The Proportion of the total number of predictions that is correct otherwise

overall how often the model predicts correctly defaulters and non- defaulters.

***Accuracy calculation:***

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN})$$

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same.

***Precision:*** The proportion of positive predictions that are actually correct.  $\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. The question that this metric answer is of all passengers that labeled as survived, how many actually survived? High precision relates to the low false positive rate. We have got 0.788 precision which is pretty good.

***Recall:*** The proportion of positive observed values correctly predicted. (The proportion of actual defaulters that the model will correctly predict)

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

Recall (Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

***F1 Score*** is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as

easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.

***General Formula:***

F- Measure =  $2TP / (2TP + FP + FN)$

***F1-Score Formula:***

F1 Score =  $2 * (\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$

## **4.8 MODULES DESCRIPTION**

1. Data validation process and Visualization
2. Prediction of DoS attacks
3. Prediction of R2L attacks
4. Prediction of U2R attacks
5. Prediction of Probe attacks
6. Prediction of overall network attacks
7. GUI based prediction results of Network attacks

### ***4.8.1 Data validation process and Visualization (Module-01):***

Validation techniques in machine learning are used to get the error rate of the Machine Learning (ML) model, which can be considered as close to the true error rate of the dataset. If the data volume is large enough to be representative of the population, you may not need the validation techniques. However, in real-world scenarios, to work with samples of data that may not be a true representative of the population of given dataset. To finding the missing value, duplicate value and description of data type whether it is float variable or integer. The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. It as machine learning engineers uses this data to fine-tune the model hyper parameters. Data collection, data analysis, and the process of addressing data content, quality, and structure can add up to a time-consuming to-do list. During the process of data identification, it helps to understand your data and its properties; this knowledge will help you choose which algorithm to use to build your

model. For example, time series data can be analyzed by regression algorithms; classification algorithms can be used to analyze discrete data. (For example, to show the data type format of given dataset)

|   | duration | protocol_type | service  | flag | src_bytes | dst_bytes | land | Wrong_fragment | Urgent | hot | ... | dst_host_srv_count | dst_host_same_srv_rate |
|---|----------|---------------|----------|------|-----------|-----------|------|----------------|--------|-----|-----|--------------------|------------------------|
| 0 | 0        | udp           | private  | SF   | 105       | 146       | 0    | 0              | 0      | 0   | ... | 254                | 1.00                   |
| 1 | 0        | udp           | private  | SF   | 105       | 146       | 0    | 0              | 0      | 0   | ... | 254                | 1.00                   |
| 2 | 0        | udp           | private  | SF   | 105       | 146       | 0    | 0              | 0      | 0   | ... | 254                | 1.00                   |
| 3 | 0        | udp           | private  | SF   | 105       | 146       | 0    | 0              | 0      | 0   | ... | 254                | 1.00                   |
| 4 | 0        | udp           | private  | SF   | 105       | 146       | 0    | 0              | 0      | 0   | ... | 254                | 1.00                   |
| 5 | 0        | udp           | private  | SF   | 105       | 146       | 0    | 0              | 0      | 0   | ... | 255                | 1.00                   |
| 6 | 0        | udp           | domain_u | SF   | 29        | 0         | 0    | 0              | 0      | 0   | ... | 3                  | 0.30                   |
| 7 | 0        | udp           | private  | SF   | 105       | 146       | 0    | 0              | 0      | 0   | ... | 253                | 0.99                   |
| 8 | 0        | udp           | private  | SF   | 105       | 146       | 0    | 0              | 0      | 0   | ... | 254                | 1.00                   |
| 9 | 0        | tcp           | http     | SF   | 223       | 185       | 0    | 0              | 0      | 0   | ... | 255                | 1.00                   |

**Fig 4.13: Data Frame**

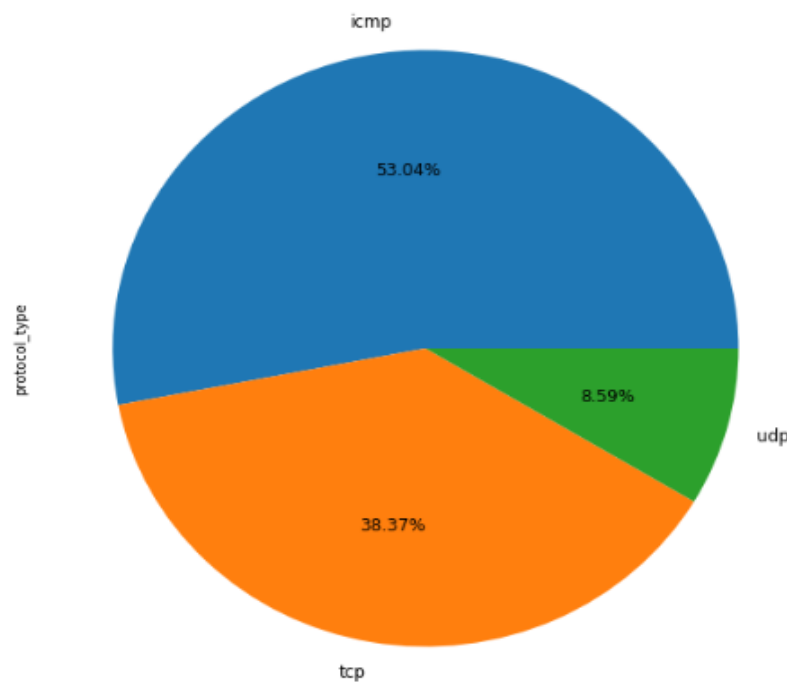
#### **Data Validation/ Cleaning/Preparing Process:**

Importing the library packages with loading given dataset. To analyzing the variable identification by data shape, data type and evaluating the missing values, duplicate values. A validation dataset is a sample of data held back from training your model that is used to give an estimate of model skill while tuning model's and procedures that you can use to make the best use of validation and test datasets when evaluating your models. Data cleaning / preparing by rename the given dataset and drop the column etc. to analyze the uni-variate, bi-variate and multi-variate process. The steps and techniques for data cleaning will vary from dataset to dataset. The primary goal of data cleaning is to detect and remove errors and anomalies to increase the value of data in analytics and decision making.

#### **Exploration data analysis of visualization:**

Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. Data visualization provides an important suite of tools for gaining a qualitative understanding. This can be helpful when exploring and getting to know a dataset and can help with

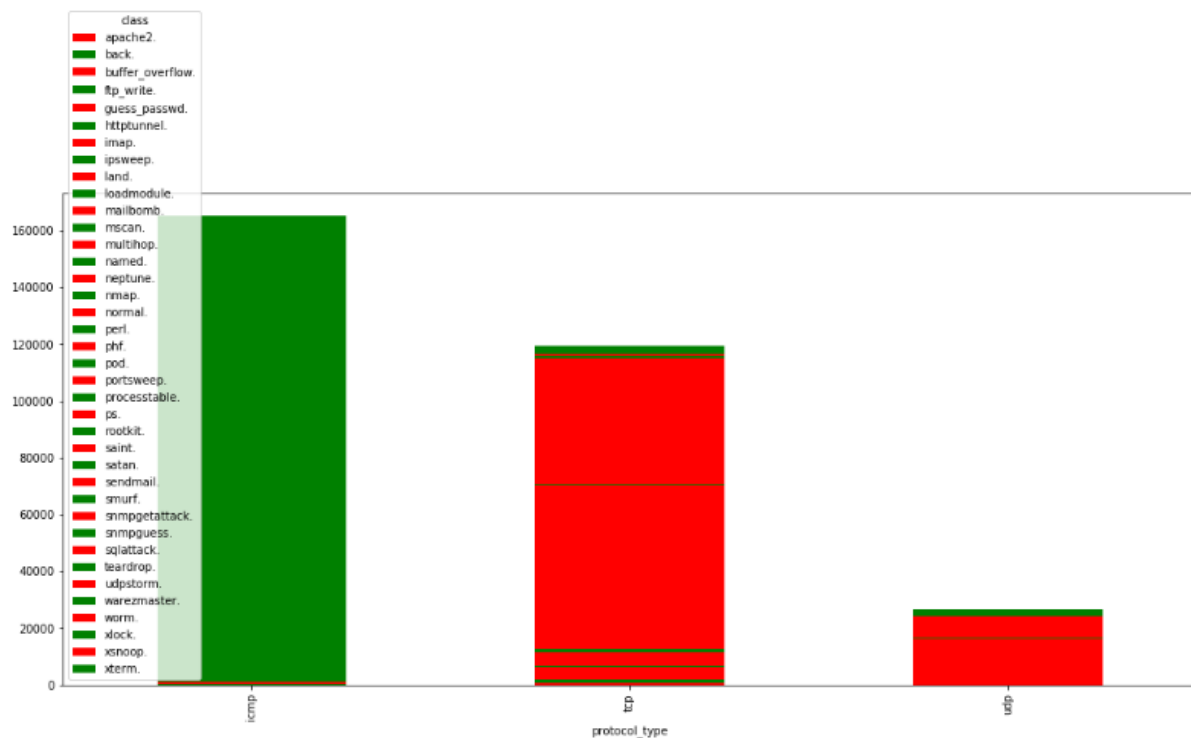
identifying patterns, corrupt data, outliers, and much more. With a little domain knowledge, data visualizations can be used to express and demonstrate key relationships in plots and charts that are more visceral and stakeholders than measures of association or significance. Data visualization and exploratory data analysis are whole fields themselves and it will recommend a deeper dive into some the books mentioned at the end.



***Fig 4.14: Percentage level of protocol type***

Sometimes data does not make sense until it can look at in a visual form, such as with charts and plots. Being able to quickly visualize of data samples and others is an important skill both in applied statistics and in applied machine learning. It will discover the many types of plots that you will need to know when visualizing data in Python and how to use them to better understand your own data.

- How to chart time series data with line plots and categorical quantities with bar charts.
- How to summarize data distributions with histograms and box plots.
- How to summarize the relationship between variables with scatter plots.



**Fig 4.15: Comparison of service type and protocol type**

Many machine learning algorithms are sensitive to the range and distribution of attribute values in the input data. Outliers in input data can skew and mislead the training process of machine learning algorithms resulting in longer training times, less accurate models and ultimately poorer results.

Even before predictive models are prepared on training data, outliers can result in misleading representations and in turn misleading interpretations of collected data. Outliers can skew the summary distribution of attribute values in descriptive statistics like mean and standard deviation and in plots such as histograms and scatterplots, compressing the body of the data. Finally, outliers can represent examples of data instances that are relevant to the problem such as anomalies in the case of fraud detection and computer security.

It couldn't fit the model on the training data and can't say that the model will work accurately for the real data. For this, we must assure that our model got the correct

patterns from the data, and it is not getting up too much noise. Cross-validation is a technique in which we train our model using the subset of the data-set and then evaluate using the complementary subset of the data-set.

The three steps involved in cross-validation are as follows:

1. Reserve some portion of sample data-set.
2. Using the rest data-set train the model.
3. Test the model using the reserve portion of the data-set.

***Advantages of train/test split:***

1. This runs K times faster than Leave One Out cross-validation because K-fold cross-validation repeats the train/test split K-times.
2. Simpler to examine the detailed results of the testing process.

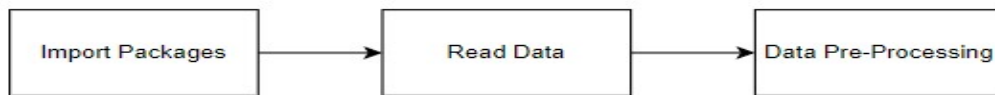
***Advantages of cross-validation:***

1. More accurate estimate of out-of-sample accuracy.
2. More “efficient” use of data as every observation is used for both training and testing.

***Data Pre-processing:***

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. To achieving better results from the applied model in Machine Learning method of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format; for example, Random Forest algorithm does not support null values. Therefore, to execute random forest algorithm null values have to be managed from the original raw data set. And another aspect is that data set should be formatted in such a way that more than one Machine Learning and Deep Learning algorithms are executed in given dataset.

MODULE DIAGRAM

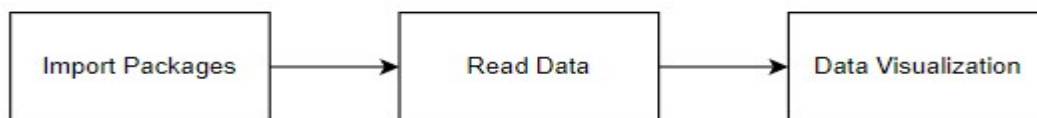


## GIVEN INPUT EXPECTED OUTPUT

Input : Data

Output : Removing noisy data

## MODULE DIAGRAM



## GIVEN INPUT EXPECTED OUTPUT

input : Data

output : Visualized data

**Module 1: Data validation and pre-processing by each attack**

```

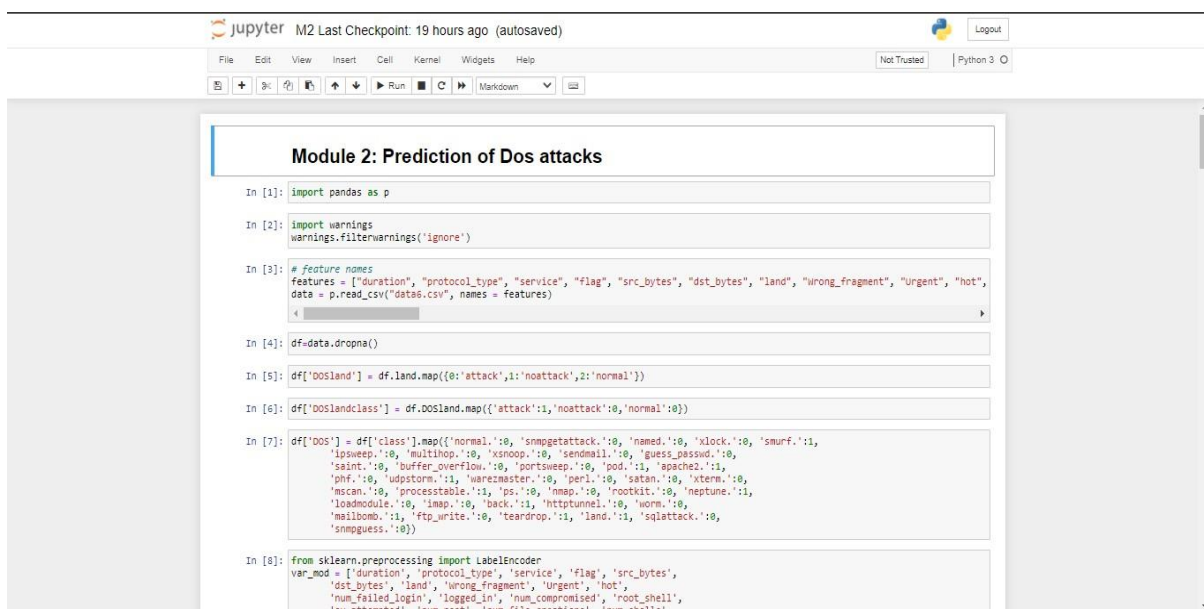
In [1]: #import library packages
import pandas as p
import numpy as n

In [2]: #feature names
features = ["duration", "protocol_type", "service", "flag", "src_bytes", "dst_bytes", "land", "wrong_fragment", "urgent", "hot",
<
Out[4]:
duration  protocol_type  service  flag  src_bytes  dst_bytes  land  Wrong_fragment  Urgent  hot  ...  dst_host_srv_count  dst_host_same_srv_rate  dst_host_d
srv_r
0      0      icmp      eor_j  SF      1032      0      0      0      0      0  ...      255      1.0
1      0      icmp      eor_j  SF      1032      0      0      0      0      0  ...      255      1.0
2      0      icmp      eor_j  SF      1032      0      0      0      0      0  ...      255      1.0
3      0      icmp      eor_j  SF      1032      0      0      0      0      0  ...      255      1.0
4      0      icmp      eor_j  SF      1032      0      0      0      0      0  ...      255      1.0
5      0      icmp      eor_j  SF      1032      0      0      0      0      0  ...      255      1.0
6      0      icmp      eor_j  SF      1032      0      0      0      0      0  ...      255      1.0
7      0      icmp      eor_j  SF      1032      0      0      0      0      0  ...      255      1.0
8      0      icmp      eor_j  SF      1032      0      0      0      0      0  ...      255      1.0
9      0      icmp      eor_j  SF      1032      0      0      0      0      0  ...      255      1.0
  
```

**Fig 4.16: Data Validation**

## 4.8.2 Prediction of DOS attacks (Module-02):

In computing, a denial-of-service attack (DoS attack) is a cyber-attack in which the perpetrator seeks to make a machine or network resource unavailable to its intended users by temporarily or indefinitely disrupting services of a host connected to the Internet. Denial of service is typically accomplished by flooding the targeted machine or resource with superfluous requests in an attempt to overload systems and prevent some or all legitimate requests from being fulfilled. In a distributed denial-of-service attack (DDoS attack), the incoming traffic flooding the victim originates from many different sources. This effectively makes it impossible to stop the attack simply by blocking a single source. A DoS or DDoS attack is analogous to a group of people crowding the entry door of a shop, making it hard for legitimate customers to enter, disrupting trade.



```
jupyter M2 Last Checkpoint: 19 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Not Trusted Python 3
In [1]: import pandas as p
In [2]: import warnings
        warnings.filterwarnings('ignore')
In [3]: # feature names
        features = ["duration", "protocol_type", "service", "flag", "src_bytes", "dst_bytes", "land", "wrong_fragment", "urgent", "hot",
        data = p.read_csv("data6.csv", names = features)
In [4]: df=data.dropna()
In [5]: df['DOSland'] = df.land.map({0:'attack',1:'noattack',2:'normal'})
In [6]: df['DOSlandclass'] = df.DOSland.map({'attack':1,'noattack':0,'normal':0})
In [7]: df['DOS'] = df['class'].map({'normal':0, 'snmpgetattack':0, 'named':0, 'xlock':0, 'smurf':1,
        'ipsweep':0, 'multihop':0, 'xsnmp':0, 'sendmail':0, 'guess_passwd':0,
        'saint':0, 'buffer_overflow':0, 'portsweep':0, 'pod':1, 'apache2':1,
        'shf':0, 'udpstorm':1, 'warezmaster':0, 'perl':0, 'satan':0, 'xterm':0,
        'mscan':0, 'processtable':1, 'ps':0, 'nmap':0, 'rootkit':0, 'neptune':1,
        'loadmodule':0, 'lmap':0, 'back':1, 'httptunnel':0, 'worm':0,
        'mailbomb':1, 'ftp_write':0, 'teardrop':1, 'land':1, 'sqlattack':0,
        'smmpguess':0})
In [8]: from sklearn.preprocessing import LabelEncoder
        var_mod = ['duration', 'protocol_type', 'service', 'flag', 'src_bytes',
        'dst_bytes', 'land', 'wrong_fragment', 'urgent', 'hot',
        'num_failed_login', 'logged_in', 'num_compromised', 'root_shell',
        'su_attempted', 'num root', 'num file creations', 'num shells',
```

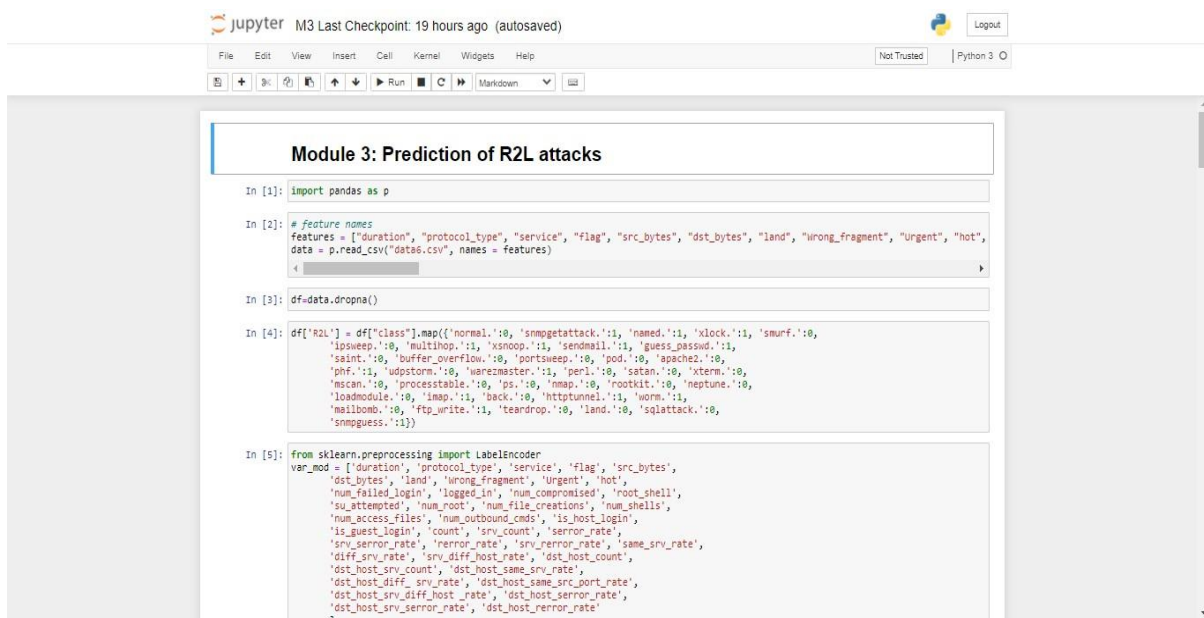
**Fig 4.17: Prediction of DOS Attacks**

### MODULE DIAGRAM



### 4.8.3 Prediction of R2L attacks (Module-03):

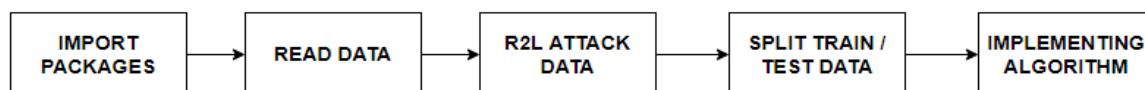
Now-a-days, it is very important to maintain a high-level security to ensure safe and trusted communication of information between various organizations. But secured data communication over internet and any other network is always under threat of intrusions and misuses. To control these threats, recognition of attacks is critical matter. Probing, Denial of Service (DoS), Remote To User (R2L) attacks is some of the attacks which affect large number of computers in the world daily. Detection of these attacks and prevention of computers from it is a major research topic for researchers throughout the world.



```
jupyter M3 Last Checkpoint: 19 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3
Module 3: Prediction of R2L attacks
In [1]: import pandas as p
In [2]: # feature names
features = ["duration", "protocol_type", "service", "flag", "src_bytes", "dst_bytes", "land", "wrong_fragment", "urgent", "hot",
data = p.read_csv("data6.csv", names = features)
In [3]: df=data.dropna()
In [4]: df['R2L'] = df['class'].map({'normal':0, 'smmpgetattack':1, 'named':1, 'xlock':1, 'smurf':0,
'ipsweep':0, 'multihop':1, 'xsnoot':1, 'sendmail':1, 'guess_passwd':1,
'saInt':0, 'buffer_overflow':0, 'portsweep':0, 'pod':0, 'apache2':0,
'phf':1, 'udpostorm':0, 'warezmaster':1, 'perl':0, 'satan':0, 'xterm':0,
'mscan':0, 'processtable':0, 'ps':0, 'nmap':0, 'rootkit':0, 'neptune':0,
'loadmodule':0, 'imap':1, 'back':0, 'httptunnel':1, 'worm':1,
'mailbomb':0, 'ftp_write':1, 'teardrop':0, 'land':0, 'sqlattack':0,
'smmpguess':1})
In [5]: from sklearn.preprocessing import LabelEncoder
var_mod = ['duration', 'protocol_type', 'service', 'flag', 'src_bytes',
'dst_bytes', 'land', 'wrong_fragment', 'urgent', 'hot',
'num_failed_login', 'logged_in', 'num_compromised', 'root_shell',
'su_attempted', 'num_root', 'num_file_creations', 'num_shells',
'num_access_files', 'num_outbound_cmds', 'is_host_login',
'is_guest_login', 'count', 'srv_count', 'serror_rate',
'srv_serror_rate', 'serror_rate', 'srv_serror_rate', 'same_srv_rate',
'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_count',
'dst_host_srv_count', 'dst_host_same_srv_rate',
'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate',
'dst_host_srv_diff_host_rate', 'dst_host_serror_rate',
'dst_host_srv_serror_rate', 'dst_host_rerror_rate']
```

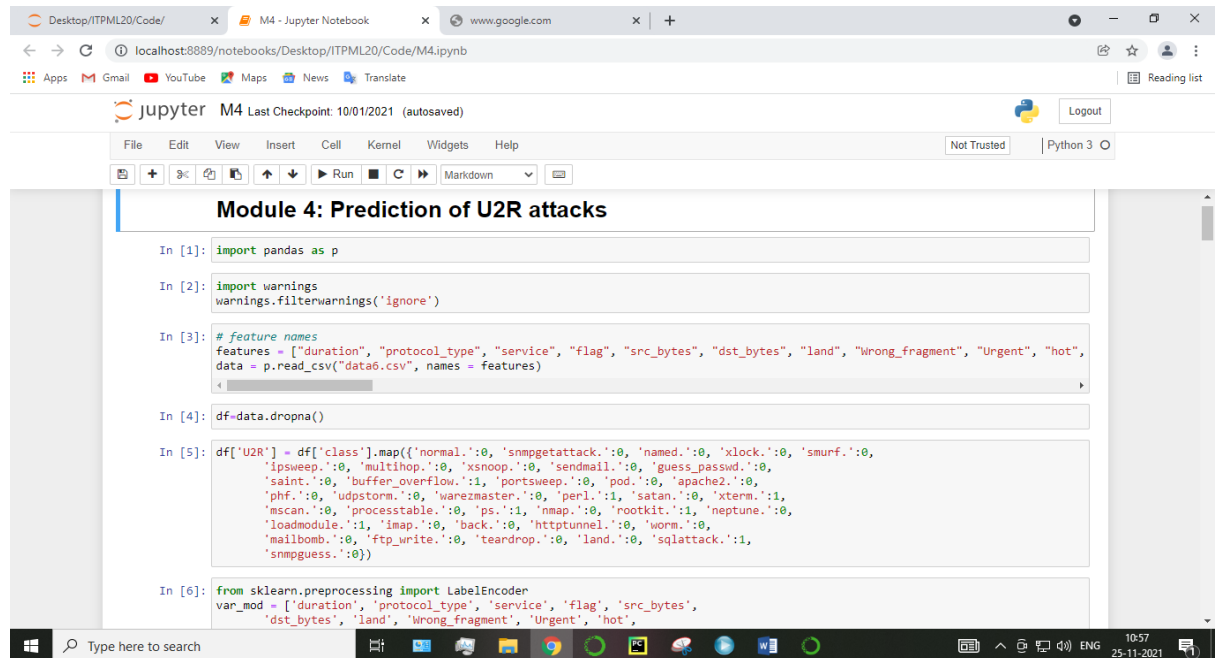
Fig 4.18: Prediction of R2L Attacks

#### MODULE DIAGRAM



#### 4.8.4 Prediction of U2R attacks (Module-04):

Remote to local attack (r2l) has been widely known to be launched by an attacker to gain unauthorized access to a victim machine in the entire network. Similarly, user to root attack (u2r) is usually launched for illegally obtaining the root's privileges when legally accessing a local machine. Buffer overflow is the most common of U2R attacks. This class begins by gaining access to a normal user while sniffing around for passwords to gain access as a root user to a computer resource. Detection of these attacks and prevention of computers from it is a major research topic for researchers throughout the world.



```
In [1]: import pandas as p

In [2]: import warnings
warnings.filterwarnings('ignore')

In [3]: # feature names
features = ["duration", "protocol_type", "service", "flag", "src_bytes", "dst_bytes", "land", "wrong_fragment", "urgent", "hot",
data = p.read_csv("data6.csv", names = features)

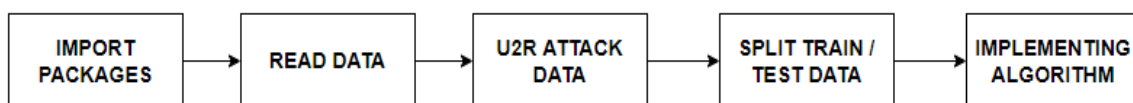
In [4]: df=data.dropna()

In [5]: df['U2R'] = df['class'].map({'normal':0, 'snmpgetattack':0, 'named':0, 'xlock':0, 'smurf':0,
'ipsweep':0, 'multihop':0, 'xsnoop':0, 'sendmail':0, 'guess_passwd':0,
'saint':0, 'buffer_overflow':1, 'portsweep':0, 'pod':0, 'apache2':0,
'phf':0, 'udpstorm':0, 'warezmaster':0, 'perl':1, 'satan':0, 'xterm':1,
'mscan':0, 'processtable':0, 'ps':1, 'nmap':0, 'rootkit':1, 'neptune':0,
'loadmodule':1, 'imap':0, 'back':0, 'httptunnel':0, 'worm':0,
'mailbomb':0, 'ftp_write':0, 'teardrop':0, 'land':0, 'sqlattack':1,
'snmpguess':0})

In [6]: from sklearn.preprocessing import LabelEncoder
var_mod = ['duration', 'protocol_type', 'service', 'flag', 'src_bytes',
'dst_bytes', 'land', 'wrong_fragment', 'urgent', 'hot',
```

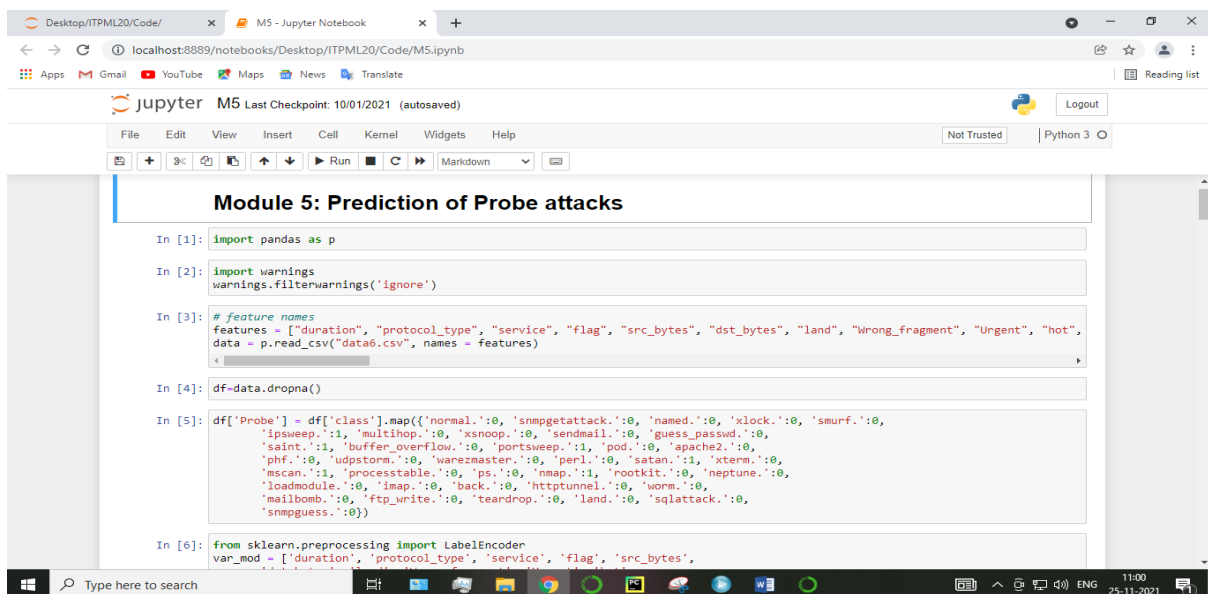
**Fig 4.19: Prediction of U2R Attacks**

#### MODULE DIAGRAM



#### 4.8.5 Prediction of Probe attacks (Module-05):

Probing attacks are an invasive method for bypassing security measures by observing the physical silicon implementation of a chip. As an invasive attack, one directly accesses the internal wires and connections of a targeted device and extracts sensitive information. A probe is an attack which is deliberately crafted so that its target detects and reports it with a recognizable “fingerprint” in the report. The attacker then uses the collaborative infrastructure to learn the detector's location and defensive capabilities from this report. This is an attack where the attacker attempts to gather information about the target machine or the network, to map out the network. Information about target may reveal useful information such as open ports, its IP address, hostname, and operating system. Network Probe is the ultimate network monitor and protocol analyzer to monitor network traffic in real-time, and will help you find the sources of any network slow-downs in a matter of seconds.



```
Module 5: Prediction of Probe attacks

In [1]: import pandas as p

In [2]: import warnings
warnings.filterwarnings('ignore')

In [3]: # feature names
features = ["duration", "protocol_type", "service", "flag", "src_bytes", "dst_bytes", "land", "wrong_fragment", "urgent", "hot",
data = p.read_csv("data6.csv", names = features)

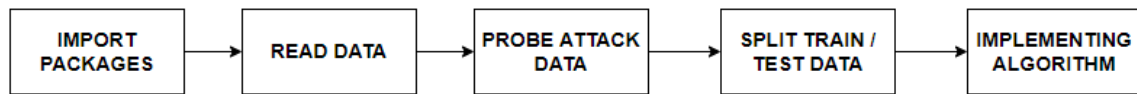
In [4]: df=data.dropna()

In [5]: df['Probe'] = df['class'].map({'normal':0, 'snmpgetattack':0, 'named':0, 'xlock':0, 'smurf':0,
'ipsweep':1, 'multihop':0, 'xsnoot':0, 'sendmail':0, 'guess_passwd':0,
'saint':1, 'buffer_overflow':0, 'portsweep':1, 'pod':0, 'apache2':0,
'phf':0, 'udpsonm':0, 'warezmaster':0, 'perl':0, 'satan':1, 'xterm':0,
'mscan':1, 'processtable':0, 'ps':0, 'nmap':1, 'rootkit':0, 'neptune':0,
'loadmodule':0, 'imap':0, 'back':0, 'httptunnel':0, 'worm':0,
'mailbomb':0, 'ftp_write':0, 'teardrop':0, 'land':0, 'sqlattack':0,
'snmpguess':0})

In [6]: from sklearn.preprocessing import LabelEncoder
var_mod = ['duration', 'protocol_type', 'service', 'flag', 'src_bytes',
```

**Fig 4.20: Prediction of Probe Attacks**

#### MODULE DIAGRAM



#### **4.8.6 Prediction of overall network attacks (Module-06):**

Increasingly, attacks are executed in multiple steps, making them harder to detect. Such complex attacks require that defenders recognize the separate stages of an attack, possibly carried out over a longer period, as belonging to the same attack. Complex attacks can be divided into exploration and exploitation phases. Exploration involves identifying vulnerabilities and scanning and testing a system. It is how an attacker gathers information about the system. Exploitation involves gaining and maintaining access. At this stage, the attacker applies the know-how gathered during the exploration stage. An example of a complex attack that combines exploration and exploitation is a sequence of a phishing attack, followed by an exfiltration attack. First, attackers will attempt to collect information on the organization they intend to attack, e.g., names of key employees. Then, they will craft a targeted phishing attack. The phishing attack allows the attackers to gain access to the user's system and install malware.

The purpose of the malware could be to extract files from the user's machine or to use the user's machine as an attack vector to attack other machines in the organization's network. A phishing attack is usually carried out by sending an email purporting to come from a trusted source and tricking its receiver to click on a URL that results in installing malware on the user's system. This malware then creates a backdoor into the user's system for staging a more complex attack. Phishing attacks can be recognized both by the types of keywords used in the email (as with a spam email), as well as by the characteristics of URLs included in the message. Features that have been used successfully to detect phishing attacks include URLs that include IP addresses, the age of a linked-to domain, and a mismatch between anchor and text of a link.

## MODULE DIAGRAM



### ***4.8.7 GUI based prediction results of Network attacks (Module-07):***

GUI means Graphical User Interface. It is the common user Interface that includes Graphical representation like buttons and icons, and communication can be performed by interacting with these icons rather than the usual text-based or command-based communication. A common example of a GUI is Microsoft operating systems.

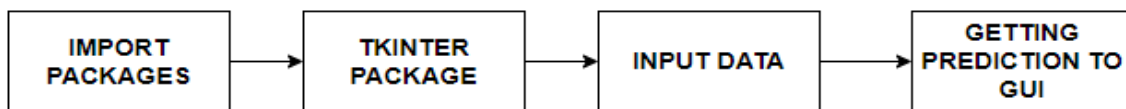
The graphical user interface (GUI) is a form of user interface that allows users to interact with electronic devices through graphical icons and audio indicator such as primary notation, instead of text-based user interfaces, typed command labels or text navigation. GUIs were introduced in reaction to the perceived steep learning curve of command-line interfaces (CLIs) which require commands to be typed on a computer keyboard.

The actions in a GUI are usually performed through direct manipulation of the graphical elements. Beyond computers, GUIs are used in many handheld mobile devices such as MP3 players, portable media players, gaming devices, smartphones and smaller household, office and industrial controls. The term GUI tends not to be applied to other lower-display resolution types of interfaces, such as video games (where head-up display (HUD) is preferred), or not including flat screens, like volumetric displays because the term is restricted to the scope of two-dimensional display screens able to describe generic information, in the tradition of the computer science research at the Xerox Palo Alto Research Center.

Graphical user interface (GUI) wrappers find a way around the command-line interface versions (CLI) of (typically) Linux and Unix-like software applications and their text-based user interfaces or typed command labels. While command-line or text-based applications allow users to run a program non-interactively, GUI wrappers atop them avoid the steep learning curve of the command-line, which requires commands to be

typed on the keyboard. By starting a GUI wrapper, users can intuitively interact with, start, stop, and change its working parameters, through graphical icons and visual indicators of a desktop environment, for example. Applications may also provide both interfaces, and when they do the GUI is usually a WIMP wrapper around the command-line version. This is especially common with applications designed for Unix-like operating systems. The latter used to be implemented first because it allowed the developers to focus exclusively on their product's functionality without bothering about interface details such as designing icons and placing buttons. Designing programs this way also allows users to run the program in a shell script.

## MODULE DIAGRAM



**Module - 07: GUI**

```

In [1]: #import library packages
import pandas as p
import matplotlib.pyplot as plt
import seaborn as s
import numpy as n

In [2]: import warnings
warnings.filterwarnings('ignore')

In [3]: # feature names
features = ["duration", "protocol_type", "service", "flag", "src_bytes", "dst_bytes", "land", "Wrong_fragment", "Urgent", "hot",
df = p.read_csv("demo3.csv", names = features)

In [4]: df.head()

Out[4]:
  duration  protocol_type  service  flag  src_bytes  dst_bytes  land  Wrong_fragment  Urgent  hot  ...  dst_host_srv_count  dst_host_same_srv_rate  dst_host_d
0         0             tcp    ecr_I  SF      1032         0     0             0         0     0  ...             255                 1.0
1         0             tcp    ecr_I  SF      1032         0     0             0         0     0  ...             255                 1.0

```

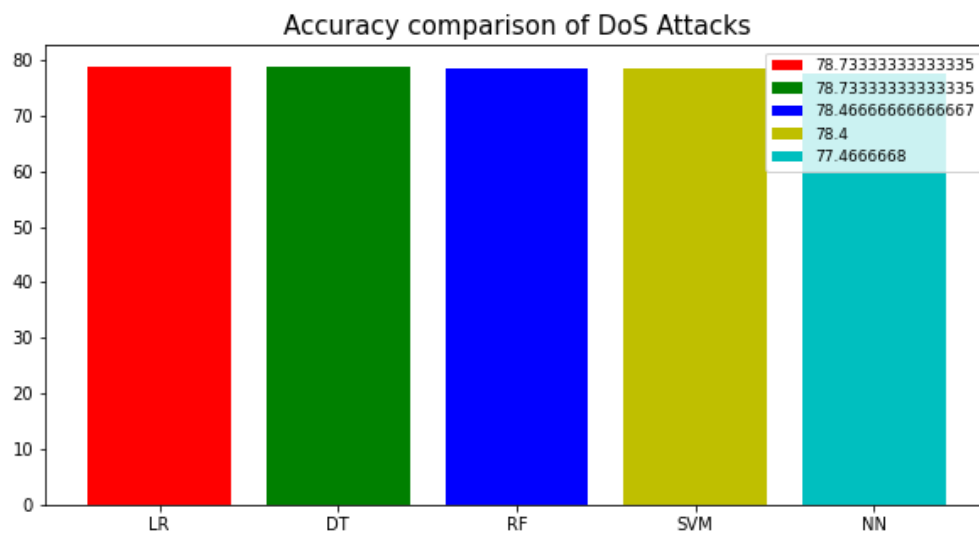
**Fig 4.21: GUI based prediction**

## CHAPTER-5

### RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS

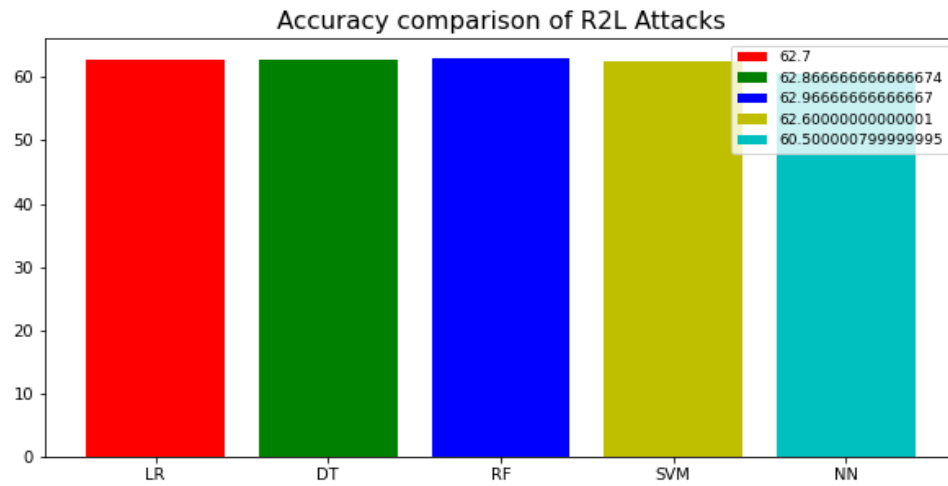
#### 5.1 PERFORMANCE ANALYSIS

*Performance Comparison of DOS Attack:*



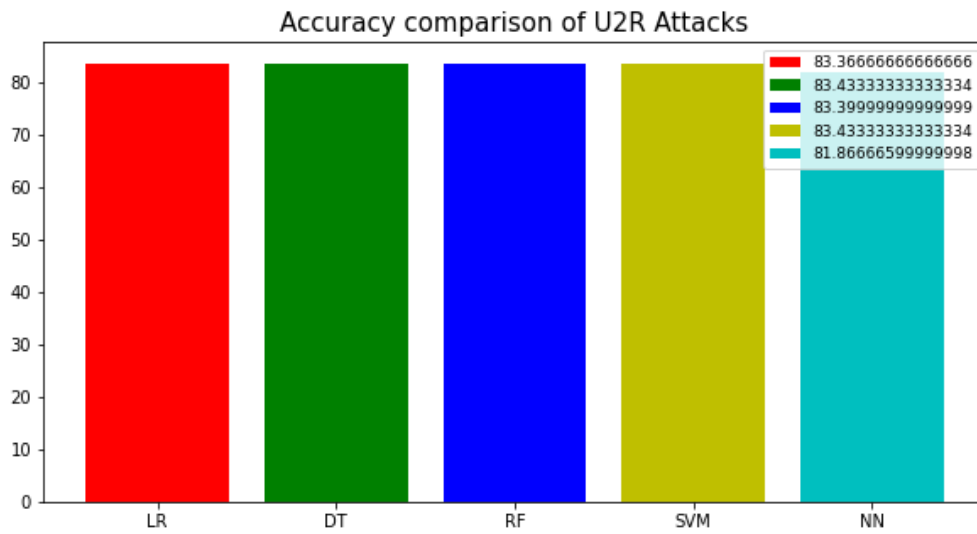
**Fig 5.1: Accuracy comparison of DOS Attack**

*Performance Comparison of R2L Attack:*



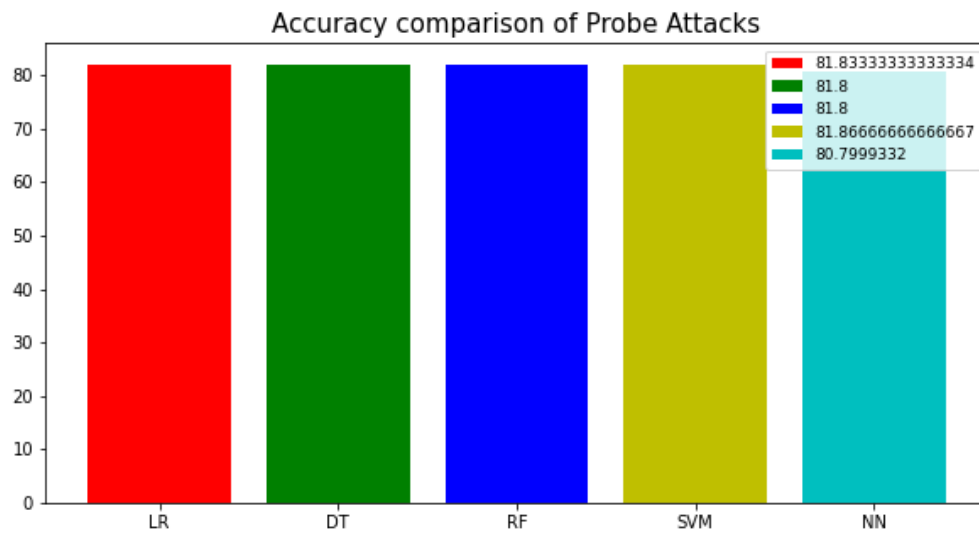
**Fig 5.2: Accuracy Comparison of R2L Attacks**

***Performance Comparison of U2R Attack:***



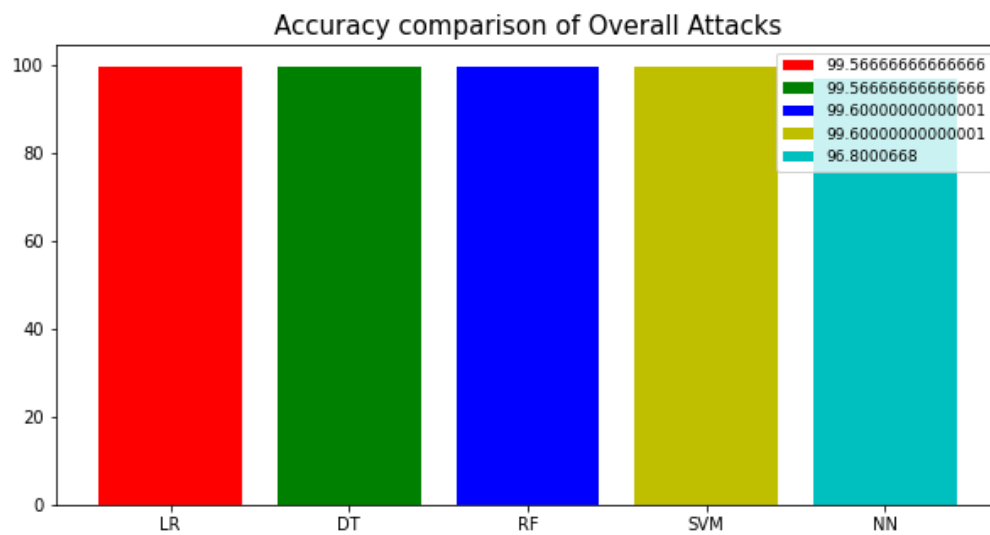
**Fig 5.3: Accuracy Comparison of U2R Attacks**

***Performance Comparison of Probe Attack:***



**Fig 5.4: Accuracy Comparison of Probe Attacks**

***Performance Comparison of Overall Attacks:***



***Fig 5.5: Accuracy Comparison of Overall Attacks***

## **Deployment of GUI:**

We are making a GUI in python using Tkinter which is used to predict the attack type and applied some Machine Learning and Deep Learning models to it. The tkinter package (“Tk interface”) is the standard Python interface to the Tcl/Tk GUI toolkit. Both Tk and tkinter are available on most Unix platforms, including macOS, as well as on Windows systems. The GUI have had some performance issues when packing many widgets onto the screen, for example packing a grid of buttons takes a few seconds. It seems to be the process of drawing the widgets onto the screen which takes the time. We have tried using both the grid and place geometry managers. We would welcome any other suggestions for any way in which this could be significantly sped up.

## **5.2 DISCUSSION**

While discussions provide avenues for exploration and discovery, leading a discussion can be anxiety-producing: discussions are, by their nature, unpredictable, and require us as instructors to surrender a certain degree of control over the flow of information. Fortunately, careful planning can help us ensure that discussions are lively without being chaotic and exploratory without losing focus. When planning a discussion, it is helpful to consider not only cognitive, but also social/emotional, and physical factors that can either foster or inhibit the productive exchange of ideas.

## **CHAPTER-6**

### **SUMMARY AND CONCLUSION**

#### **6.1 SUMMARY**

The scope of this project is to investigate a dataset of network connection attacks for KDD records for medical sector using machine learning technique. To identifying network connection is attacked or not.

#### **6.2 CONCLUSION**

The analytical process started from data cleaning and processing, missing value, exploratory analysis and finally model building and evaluation. The best accuracy on public test set is higher accuracy score will be find out by comparing each algorithm with type of all network attacks for future prediction results by finding best connections. This brings some of the following insights about diagnose the network

attack of each new connection. To presented a prediction model with the aid of artificial intelligence to improve over human accuracy and provide with the scope of early detection. It can be inferred from this model that, area analysis and use of machine learning technique is useful in developing prediction models that can helps to network sectors reduce the long process of diagnosis and eradicate any human error.

### 6.3 FUTURE WORK

- Network sector want to automate the detecting the attacks of packet transfers from eligibility process (real time) based on the connection detail.
- To automate this process by show the prediction result in web application or desktop application.
- To optimize the work to implement in Artificial Intelligence environment.

### REFERENCES

- [1] A. Kurakin, I. Goodfellow, and S. Bengio. (2016). "Adversarial examples in the physical world." [Online]. Available: <https://arxiv.org/abs/1607.02533>.
- [2] G. Zhang, C. Yan, X. Ji, T. Zhang, T. Zhang, and W. Xu. (2017). "Dolphin Attack: Inaudible voice commands." [Online]. Available: <https://arxiv.org/abs/1708.09537>.
- [3] P. He, W. Huang, T. He, Q. Zhu, Y. Qiao, and X. Li, "Single shot text detector with regional attention," in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 3066–3074.
- [4] C. Yan, H. Xie, J. Chen, Z. Zha, X. Hao, and Y. Zhang, "A fast uyghur text detector for complex background images," IEEE Trans. Multimedia, vol. 20, no. 12, pp. 3389–3398, Dec. 2018.
- [5] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in Proc. IEEE Int. Conf. Comput. Vis., 2015, pp. 1026–1034.
- [6] D. Storcheus, A. Rostamizadeh, and S. Kumar, "A survey of modern questions and challenges in feature extraction," in Proc. NIPS Workshop, Dec. 2015, pp. 1–18.

- [7] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in Proc. IEEE Symp. Secur. Privacy (SP), May 2017, pp. 3–18.
- [8] Y. Bengio and Y. LeCun, "Scaling learning algorithms towards AI," Large-Scale Kernel Mach., vol. 34, no. 5, pp. 1–41, 2007.
- [9] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman. (2017). "PixelDefend: Leveraging generative models to understand and defend against adversarial examples." [Online]. Available: <https://arxiv.org/abs/1710.10766>
- [10] A. Kurakin et al. (2018). "Adversarial attacks and defences competition." [Online]. Available: <https://arxiv.org/abs/1804.00097>
- [11] A. Fawzi, O. Fawzi, and P. Frossard, "Fundamental limits on adversarial robustness," in Proc. ICML Workshop Deep Learn., 2015, pp. 1–7.

## APPENDIX

### A. SOURCE CODE

#### MODULE – 1:

```
#import library packages
import pandas as p
import numpy as n
# feature names
features = ["duration", "protocol_type", "service", "flag", "src_bytes", "dst_bytes",
"land", "Wrong_fragment", "Urgent", "hot", "num_failed_login", "logged_in",
"num_compromised", "root_shell", "su_attempted", "num_root",
"num_file_creations", "num_shells", "num_access_files", "num_outbound_cmds",
"is_host_login", "is_guest_login", "count", "srv_count", "serror_rate",
"srv_serror_rate", "error_rate", "srv_error_rate", "same_srv_rate", "diff_srv_rate",
"srv_diff_host_rate", "dst_host_count", "dst_host_srv_count",
"dst_host_same_srv_rate", "dst_host_diff_srv_rate",
"dst_host_same_src_port_rate", "dst_host_srv_diff_host_rate",
```

```

"dst_host_serror_rate",    "dst_host_srv_serror_rate",    "dst_host_rerror_rate",
"dst_host_srv_rerror_rate","class"]
data = p.read_csv("data6.csv", names = features)
Before drop the given dataset
data.head(10)
After drop the given dataset
df=data.dropna()
df.head(10)
#show columns
df.columns
Index(['duration', 'protocol_type', 'service', 'flag', 'src_bytes',
      'dst_bytes', 'land', 'Wrong_fragment', 'Urgent', 'hot',
      'num_failed_login', 'logged_in', 'num_compromised', 'root_shell',
      'su_attempted', 'num_root', 'num_file_creations', 'num_shells',
      'num_access_files', 'num_outbound_cmds', 'is_host_login',
      'is_guest_login', 'count', 'srv_count', 'serror_rate',
      'srv_serror_rate', 'rerror_rate', 'srv_rerror_rate', 'same_srv_rate',
      'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_count',
      'dst_host_srv_count', 'dst_host_same_srv_rate',
      'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate',
      'dst_host_srv_diff_host_rate', 'dst_host_serror_rate',
      'dst_host_srv_serror_rate', 'dst_host_rerror_rate',
      'dst_host_srv_rerror_rate', 'class'],
      dtype='object')
#To describe the dataframe
df.describe()
#Checking datatype and information about dataset
df.info()
Checking duplicate values of dataframe
#Checking for duplicate data
df.duplicated()

```

```

#find sum of duplicate data
sum(df.duplicated())

#Checking sum of missing values
df.isnull().sum()

d = p.crosstab(df['protocol_type'], df['class'])
d.plot(kind='bar', stacked=True, color=['red','green'], grid=False, figsize=(18,8))

import matplotlib.pyplot as plt
pr = df["protocol_type"]
fl = df["flag"]
plt.plot(fl, pr, color='g')
plt.xlabel('Flag Types')
plt.ylabel('Protocol Types')
plt.title('Flag Details by protocol type')
plt.show()

df["class"].unique()
df['land'].value_counts()
df['service'].value_counts()
df['protocol_type'].value_counts()

import numpy as n
def PropByVar(df, variable):
    dataframe_pie = df[variable].value_counts()
    ax = dataframe_pie.plot.pie(figsize=(10,10), autopct='%1.2f%%', fontsize = 12);
    ax.set_title(variable + ' (%) (Per Count)\n', fontsize = 15);
    return n.round(dataframe_pie/df.shape[0]*100,2)

PropByVar(df, 'protocol_type')
df['DOSland'] = df.land.map({0:'attack',1:'noattack',2:'normal'})
df['DOSlandclass'] = df.DOSland.map({'attack':1,'noattack':0,'normal':0})
df['DOSlandclass'].value_counts()
df['DOS'] = df['class'].map({'normal.':0, 'snmpgetattack.':0, 'named.':0, 'xlock.':0,
'smurf.':1,

```

```

'ipsweep.':0, 'multihop.':0, 'xsnoop.':0, 'sendmail.':0, 'guess_passwd.':0,
'saint.':0, 'buffer_overflow.':0, 'portsweep.':0, 'pod.':1, 'apache2.':1,
'phf.':0, 'udpstorm.':1, 'warezmaster.':0, 'perl.':0, 'satan.':0, 'xterm.':0,
'mscan.':0, 'processtable.':1, 'ps.':0, 'nmap.':0, 'rootkit.':0, 'neptune.':1,
'loadmodule.':0, 'imap.':0, 'back.':1, 'httptunnel.':0, 'worm.':0,
'mailbomb.':1, 'ftp_write.':0, 'teardrop.':1, 'land.':1, 'sqlattack.':0,
'snmpguess.':0})
df.head()
df['R2L'] = df['class'].map({'normal.':0, 'snmpgetattack.':1, 'named.':1, 'xlock.':1,
'smurf.':0,
'ipsweep.':0, 'multihop.':1, 'xsnoop.':1, 'sendmail.':1, 'guess_passwd.':1,
'saint.':0, 'buffer_overflow.':0, 'portsweep.':0, 'pod.':0, 'apache2.':0,
'phf.':1, 'udpstorm.':0, 'warezmaster.':1, 'perl.':0, 'satan.':0, 'xterm.':0,
'mscan.':0, 'processtable.':0, 'ps.':0, 'nmap.':0, 'rootkit.':0, 'neptune.':0,
'loadmodule.':0, 'imap.':1, 'back.':0, 'httptunnel.':1, 'worm.':1,
'mailbomb.':0, 'ftp_write.':1, 'teardrop.':0, 'land.':0, 'sqlattack.':0,
'snmpguess.':1})
df['U2R'] = df['class'].map({'normal.':0, 'snmpgetattack.':0, 'named.':0, 'xlock.':0,
'smurf.':0,
'ipsweep.':0, 'multihop.':0, 'xsnoop.':0, 'sendmail.':0, 'guess_passwd.':0,
'saint.':0, 'buffer_overflow.':1, 'portsweep.':0, 'pod.':0, 'apache2.':0,
'phf.':0, 'udpstorm.':0, 'warezmaster.':0, 'perl.':1, 'satan.':0, 'xterm.':1,
'mscan.':0, 'processtable.':0, 'ps.':1, 'nmap.':0, 'rootkit.':1, 'neptune.':0,
'loadmodule.':1, 'imap.':0, 'back.':0, 'httptunnel.':0, 'worm.':0,
'mailbomb.':0, 'ftp_write.':0, 'teardrop.':0, 'land.':0, 'sqlattack.':1,
'snmpguess.':0})
df['Probe'] = df['class'].map({'normal.':0, 'snmpgetattack.':0, 'named.':0, 'xlock.':0,
'smurf.':0,
'ipsweep.':1, 'multihop.':0, 'xsnoop.':0, 'sendmail.':0, 'guess_passwd.':0,
'saint.':1, 'buffer_overflow.':0, 'portsweep.':1, 'pod.':0, 'apache2.':0,
'phf.':0, 'udpstorm.':0, 'warezmaster.':0, 'perl.':0, 'satan.':1, 'xterm.':0,

```

```

'mscan.':1, 'processtable.':0, 'ps.':0, 'nmap.':1, 'rootkit.':0, 'neptune.':0,
'loadmodule.':0, 'imap.':0, 'back.':0, 'httptunnel.':0, 'worm.':0,
'mailbomb.':0, 'ftp_write.':0, 'teardrop.':0, 'land.':0, 'sqlattack.':0,
'snmpguess.':0})
df['attack'] = df['class'].map({'normal.':0, 'snmpgetattack.':1, 'named.':1, 'xlock.':1,
'smurf.':1,
'ipsweep.':1, 'multihop.':1, 'xsnoop.':1, 'sendmail.':1, 'guess_passwd.':1,
'saint.':1, 'buffer_overflow.':1, 'portsweep.':1, 'pod.':1, 'apache2.':1,
'phf.':1, 'udpstorm.':1, 'warezmaster.':1, 'perl.':1, 'satan.':1, 'xterm.':1,
'mscan.':1, 'processtable.':1, 'ps.':1, 'nmap.':1, 'rootkit.':1, 'neptune.':1,
'loadmodule.':1, 'imap.':1, 'back.':1, 'httptunnel.':1, 'worm.':1,
'mailbomb.':1, 'ftp_write.':1, 'teardrop.':1, 'land.':1, 'sqlattack.':1,
'snmpguess.':1})
df.head()
df.corr()
Before Pre-Processing:
df.head()
After Pre-Processing:
df.columns

```

```

from sklearn.preprocessing import LabelEncoder
var_mod = ['duration', 'protocol_type', 'service', 'flag', 'src_bytes',
'dst_bytes', 'land', 'Wrong_fragment', 'Urgent', 'hot',
'num_failed_login', 'logged_in', 'num_compromised', 'root_shell',
'su_attempted', 'num_root', 'num_file_creations', 'num_shells',
'num_access_files', 'num_outbound_cmds', 'is_host_login',
'is_guest_login', 'count', 'srv_count', 'serror_rate',
'srv_serror_rate', 'error_rate', 'srv_error_rate', 'same_srv_rate',
'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_count',
'dst_host_srv_count', 'dst_host_same_srv_rate',
'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate',

```

```

'dst_host_srv_diff_host_rate', 'dst_host_serror_rate',
'dst_host_srv_serror_rate', 'dst_host_rerror_rate',
'dst_host_srv_rerror_rate', ]
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i]).astype(str)
df.head()

```

## MODULE-2:

```

import pandas as p
import warnings
warnings.filterwarnings('ignore')
# feature names
features = ["duration", "protocol_type", "service", "flag", "src_bytes", "dst_bytes",
"land", "Wrong_fragment", "Urgent", "hot", "num_failed_login", "logged_in",
"num_compromised", "root_shell", "su_attempted", "num_root",
"num_file_creations", "num_shells", "num_access_files", "num_outbound_cmds",
"is_host_login", "is_guest_login", "count", "srv_count", "serror_rate",
"srv_serror_rate", "rerror_rate", "srv_rerror_rate", "same_srv_rate", "diff_srv_rate",
"srv_diff_host_rate", "dst_host_count", "dst_host_srv_count",
"dst_host_same_srv_rate", "dst_host_diff_srv_rate",
"dst_host_same_src_port_rate", "dst_host_srv_diff_host_rate",
"dst_host_serror_rate", "dst_host_srv_serror_rate", "dst_host_rerror_rate",
"dst_host_srv_rerror_rate", "class"]
data = p.read_csv("data6.csv", names = features)
df=data.dropna()
df['DOSland'] = df.land.map({0:'attack',1:'noattack',2:'normal'})
df['DOSlandclass'] = df.DOSland.map({'attack':1,'noattack':0,'normal':0})
df['DOS'] = df['class'].map({'normal':0, 'snmpgetattack':0, 'named':0, 'xlock':0,
'smurf':1,
'ipsweep':0, 'multihop':0, 'xsnoop':0, 'sendmail':0, 'guess_passwd':0,

```

```
'saint.':0, 'buffer_overflow.':0, 'portsweep.':0, 'pod.':1, 'apache2.':1,
'phf.':0, 'udpstorm.':1, 'warezmaster.':0, 'perl.':0, 'satan.':0, 'xterm.':0,
'mscan.':0, 'processtable.':1, 'ps.':0, 'nmap.':0, 'rootkit.':0, 'neptune.':1,
'loadmodule.':0, 'imap.':0, 'back.':1, 'httptunnel.':0, 'worm.':0,
'mailbomb.':1, 'ftp_write.':0, 'teardrop.':1, 'land.':1, 'sqlattack.':0,
'snmpguess.':0})
```

```
from sklearn.preprocessing import LabelEncoder
var_mod = ['duration', 'protocol_type', 'service', 'flag', 'src_bytes',
'dst_bytes', 'land', 'Wrong_fragment', 'Urgent', 'hot',
'num_failed_login', 'logged_in', 'num_compromised', 'root_shell',
'su_attempted', 'num_root', 'num_file_creations', 'num_shells',
'num_access_files', 'num_outbound_cmds', 'is_host_login',
'is_guest_login', 'count', 'srv_count', 'serror_rate',
'srv_serror_rate', 'error_rate', 'srv_error_rate', 'same_srv_rate',
'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_count',
'dst_host_srv_count', 'dst_host_same_srv_rate',
'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate',
'dst_host_srv_diff_host_rate', 'dst_host_serror_rate',
'dst_host_srv_serror_rate', 'dst_host_error_rate'
]
```

```
le = LabelEncoder()
```

```
for i in var_mod:
```

```
    df[i] = le.fit_transform(df[i]).astype(int)
```

```
del df['DOSland']
```

```
del df['dst_host_srv_error_rate']
```

```
del df['DOSlandclass']
```

```
del df['class']
```

#According to the cross-validated MCC scores, the random forest is the best-performing model, so now let's evaluate its performance on the test set.

```

from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score
X = df.drop(labels='DOS', axis=1)
#Response variable
y = df.loc[:, 'DOS']
#We'll use a test size of 30%. We also stratify the split on the response variable,
which is very important to do because there are so few fraudulent transactions.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=1, stratify=y)
#According to the cross-validated MCC scores, the random forest is the best-
performing model, so now let's evaluate its performance on the test set.

```

```

from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score

```

### **Logistic Regression:**

```

from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
logR= LogisticRegression()
logR.fit(X_train,y_train)
predictR = logR.predict(X_test)
print("")
print('Classification report of Logistic Regression Results:')
print("")
print(classification_report(y_test,predictR))

cm1=confusion_matrix(y_test,predictR)

```

```

print('Confusion Matrix result of Logistic Regression is:\n',cm1)
print("")
sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
print("")

accuracy = cross_val_score(logR, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Logistic Regression is:",accuracy.mean() * 100)
lr=accuracy.mean() * 100

```

### **Decision Tree:**

```

from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import cross_val_score

dt= DecisionTreeClassifier()
dt.fit(X_train,y_train)
predictR = dt.predict(X_test)
print("")
print('Classification report of Decision Tree Results:')
print("")
print(classification_report(y_test,predictR))

cm1=confusion_matrix(y_test,predictR)

```

```

print('Confusion Matrix result of Decision Tree is:\n',cm1)
print("")
sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
print("")

accuracy = cross_val_score(dt, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Decision Tree is:",accuracy.mean() * 100)
dt=accuracy.mean() * 100

```

### **Random Forest:**

```

from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(X_train,y_train)
predictR = rfc.predict(X_test)
print("")
print('Classification report of Random Forest Results:')
print("")
print(classification_report(y_test,predictR))
print("")
cm1=confusion_matrix(y_test,predictR)
print('Confusion Matrix result of Random Forest is:\n',cm1)
print("")
sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])

```

```

print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
print("")

accuracy = cross_val_score(rfc, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Random Forest is:",accuracy.mean() * 100)
rf=accuracy.mean() * 100

```

### **Support Vector Classifier:**

```

from sklearn.svm import SVC
sv = SVC()
sv.fit(X_train, y_train)
predictSVC = sv.predict(X_test)
print("")
print('Classification report of Support Vector Classifier Results:')
print("")
print(classification_report(y_test,predictSVC))

print("")
cm4=confusion_matrix(y_test,predictSVC)
print('Confusion Matrix result of Support Vector Classifier is:\n',
confusion_matrix(y_test,predictSVC))
print("")
sensitivity1 = cm4[0,0]/(cm4[0,0]+cm4[0,1])
print('Sensitivity : ', sensitivity1 )

```

```

print("")
specificity1 = cm4[1,1]/(cm4[1,0]+cm4[1,1])
print('Specificity : ', specificity1)

accuracy = cross_val_score(sv, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Support Vector Classifier is:",accuracy.mean() * 100)
sv=accuracy.mean() * 100
def graph():
    import matplotlib.pyplot as plt
    data=[lr,dt,rf,sv]
    alg="LR","DT","RF","SVM"
    plt.figure(figsize=(10,5))
    b=plt.bar(alg,data,color=("r","g","b","y"))
    plt.title("Accuracy comparison of DoS Attacks",fontsize=15)
    plt.legend(b,data,fontsize=9)
    plt.savefig('DOS.png')
graph()

import tkinter
from matplotlib.backends.backend_tkagg import (FigureCanvasTkAgg,
NavigationToolbar2Tk)
from matplotlib.backend_bases import key_press_handler
from matplotlib.figure import Figure
import numpy as np
root = tkinter.Tk()
root.wm_title("Accuracy plot for DoS Attacks")
fig = Figure(figsize=(10,10),dpi=1)

```

```

canvas = FigureCanvasTkAgg(fig, master=root)
canvas.draw()
canvas.get_tk_widget().pack(side=tkinter.TOP, fill=tkinter.BOTH, expand=1)
icon=tkinter.PhotoImage(file='DOS.png')
label=tkinter.Label(root,image=icon)
label.pack()
root.mainloop()

```

### **MODULE-3:**

```

import pandas as p
# feature names
features = ["duration", "protocol_type", "service", "flag", "src_bytes", "dst_bytes",
"land", "Wrong_fragment", "Urgent", "hot", "num_failed_login", "logged_in",
"num_compromised", "root_shell", "su_attempted", "num_root",
"num_file_creations", "num_shells", "num_access_files", "num_outbound_cmds",
"is_host_login", "is_guest_login", "count", "srv_count", "serror_rate",
"srv_serror_rate", "error_rate", "srv_error_rate", "same_srv_rate", "diff_srv_rate",
"srv_diff_host_rate", "dst_host_count", "dst_host_srv_count",
"dst_host_same_srv_rate", "dst_host_diff_srv_rate",
"dst_host_same_src_port_rate", "dst_host_srv_diff_host_rate",
"dst_host_serror_rate", "dst_host_srv_serror_rate", "dst_host_error_rate",
"dst_host_srv_error_rate","class"]
data = p.read_csv("data6.csv", names = features)
df=data.dropna()

```

```
df['R2L'] = df["class"].map({'normal.':0, 'snmpgetattack.':1, 'named.':1, 'xlock.':1,
'smurf.':0,
    'ipsweep.':0, 'multihop.':1, 'xsnoop.':1, 'sendmail.':1, 'guess_passwd.':1,
    'saint.':0, 'buffer_overflow.':0, 'portsweep.':0, 'pod.':0, 'apache2.':0,
    'phf.':1, 'udpstorm.':0, 'warezmaster.':1, 'perl.':0, 'satan.':0, 'xterm.':0,
    'mscan.':0, 'processtable.':0, 'ps.':0, 'nmap.':0, 'rootkit.':0, 'neptune.':0,
    'loadmodule.':0, 'imap.':1, 'back.':0, 'httptunnel.':1, 'worm.':1,
    'mailbomb.':0, 'ftp_write.':1, 'teardrop.':0, 'land.':0, 'sqlattack.':0,
    'snmpguess.':1})
```

```
from sklearn.preprocessing import LabelEncoder
var_mod = ['duration', 'protocol_type', 'service', 'flag', 'src_bytes',
    'dst_bytes', 'land', 'Wrong_fragment', 'Urgent', 'hot',
    'num_failed_login', 'logged_in', 'num_compromised', 'root_shell',
    'su_attempted', 'num_root', 'num_file_creations', 'num_shells',
    'num_access_files', 'num_outbound_cmds', 'is_host_login',
    'is_guest_login', 'count', 'srv_count', 'serror_rate',
    'srv_serror_rate', 'error_rate', 'srv_error_rate', 'same_srv_rate',
    'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_count',
    'dst_host_srv_count', 'dst_host_same_srv_rate',
    'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate',
    'dst_host_srv_diff_host_rate', 'dst_host_serror_rate',
    'dst_host_srv_serror_rate', 'dst_host_error_rate'
]
```

```
le = LabelEncoder()
```

```
for i in var_mod:
```

```
    df[i] = le.fit_transform(df[i]).astype(int)
```

```
del df['dst_host_srv_error_rate']
```

```
del df["class"]
```

#According to the cross-validated MCC scores, the random forest is the best-performing model, so now let's evaluate its performance on the test set.

```

from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score
X = df.drop(labels='R2L', axis=1)
#Response variable
y = df.loc[:, 'R2L']
#We'll use a test size of 30%. We also stratify the split on the response variable,
which is very important to do because there are so few fraudulent transactions.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=1, stratify=y)
import warnings
warnings.filterwarnings('ignore')
#According to the cross-validated MCC scores, the random forest is the best-
performing model, so now let's evaluate its performance on the test set.
from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score

```

### **Logistic Regression:**

```

from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score

logR= LogisticRegression()
logR.fit(X_train,y_train)
predictR = logR.predict(X_test)
print("")
print('Classification report of Logistic Regression Results:')
print("")
print(classification_report(y_test,predictR))

```

```

cm1=confusion_matrix(y_test,predictR)
print('Confusion Matrix result of Logistic Regression is:\n',cm1)
print("")
sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
print("")

accuracy = cross_val_score(logR, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Logistic Regression is:",accuracy.mean() * 100)
lr=accuracy.mean() * 100

```

### **Decision Tree:**

```

from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import cross_val_score

dt= DecisionTreeClassifier()
dt.fit(X_train,y_train)
predictR = dt.predict(X_test)
print("")
print('Classification report of Decision Tree Results:')
print("")
print(classification_report(y_test,predictR))

```

```

cm1=confusion_matrix(y_test,predictR)
print('Confusion Matrix result of Decision Tree is:\n',cm1)
print("")
sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
print("")

accuracy = cross_val_score(dt, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Decision Tree is:",accuracy.mean() * 100)
dt=accuracy.mean() * 100

```

### **Random Forest:**

```

from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(X_train,y_train)
predictR = rfc.predict(X_test)
print("")
print('Classification report of Random Forest Results:')
print("")
print(classification_report(y_test,predictR))
print("")
cm1=confusion_matrix(y_test,predictR)
print('Confusion Matrix result of Random Forest is:\n',cm1)

```

```

print("")
sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
print("")

accuracy = cross_val_score(rfc, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Random Forest is:",accuracy.mean() * 100)
rf=accuracy.mean() * 100

```

### **Support Vector Classifier:**

```

from sklearn.svm import SVC
sv = SVC()
sv.fit(X_train, y_train)
predictSVC = sv.predict(X_test)
print("")
print('Classification report of Support Vector Classifier Results:')
print("")
print(classification_report(y_test,predictSVC))

print("")
cm4=confusion_matrix(y_test,predictSVC)
print('Confusion Matrix result of Support Vector Classifier is:\n',
confusion_matrix(y_test,predictSVC))
print("")

```

```

sensitivity1 = cm4[0,0]/(cm4[0,0]+cm4[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm4[1,1]/(cm4[1,0]+cm4[1,1])
print('Specificity : ', specificity1)

accuracy = cross_val_score(sv, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Support Vector Classifier is:",accuracy.mean() * 100)
sv=accuracy.mean() * 100
def graph():
    import matplotlib.pyplot as plt
    data=[lr,dt,rf,sv]
    alg="LR","DT","RF","SVM"
    plt.figure(figsize=(10,5))
    b=plt.bar(alg,data,color=("r","g","b","y"))
    plt.title("Accuracy comparison of R2L Attacks",fontsize=15)
    plt.legend(b,data,fontsize=9)
    plt.savefig('R2L.png')
graph()

import tkinter
from matplotlib.backends.backend_tkagg import (FigureCanvasTkAgg,
NavigationToolbar2Tk)
from matplotlib.backend_bases import key_press_handler
from matplotlib.figure import Figure
import numpy as np
root = tkinter.Tk()

```

```

root.wm_title("Accuracy plot for R2L Attacks")
fig = Figure(figsize=(10,10),dpi=1)
canvas = FigureCanvasTkAgg(fig, master=root)
canvas.draw()
canvas.get_tk_widget().pack(side=tkinter.TOP, fill=tkinter.BOTH, expand=1)
icon=tkinter.PhotoImage(file='R2L.png')
label=tkinter.Label(root,image=icon)
label.pack()
root.mainloop()

```

#### **MODULE-4:**

```

import pandas as p
import warnings
warnings.filterwarnings('ignore')
# feature names
features = ["duration", "protocol_type", "service", "flag", "src_bytes", "dst_bytes",
"land", "Wrong_fragment", "Urgent", "hot", "num_failed_login", "logged_in",
"num_compromised", "root_shell", "su_attempted", "num_root",
"num_file_creations", "num_shells", "num_access_files", "num_outbound_cmds",
"is_host_login", "is_guest_login", "count", "srv_count", "error_rate",
"srv_error_rate", "error_rate", "srv_error_rate", "same_srv_rate", "diff_srv_rate",
"srv_diff_host_rate", "dst_host_count", "dst_host_srv_count",
"dst_host_same_srv_rate", "dst_host_diff_srv_rate",
"dst_host_same_src_port_rate", "dst_host_srv_diff_host_rate",
"dst_host_error_rate", "dst_host_srv_error_rate", "dst_host_error_rate",
"dst_host_srv_error_rate","class"]

```

```

data = p.read_csv("data6.csv", names = features)
df=data.dropna()
df['U2R'] = df['class'].map({'normal.':0, 'snmpgetattack.':0, 'named.':0, 'xlock.':0,
'smurf.':0,
    'ipsweep.':0, 'multihop.':0, 'xsnoop.':0, 'sendmail.':0, 'guess_passwd.':0,
    'saint.':0, 'buffer_overflow.':1, 'portsweep.':0, 'pod.':0, 'apache2.':0,
    'phf.':0, 'udpstorm.':0, 'warezmaster.':0, 'perl.':1, 'satan.':0, 'xterm.':1,
    'mscan.':0, 'processtable.':0, 'ps.':1, 'nmap.':0, 'rootkit.':1, 'neptune.':0,
    'loadmodule.':1, 'imap.':0, 'back.':0, 'httptunnel.':0, 'worm.':0,
    'mailbomb.':0, 'ftp_write.':0, 'teardrop.':0, 'land.':0, 'sqlattack.':1,
    'snmpguess.':0})

```

```

from sklearn.preprocessing import LabelEncoder
var_mod = ['duration', 'protocol_type', 'service', 'flag', 'src_bytes',
    'dst_bytes', 'land', 'Wrong_fragment', 'Urgent', 'hot',
    'num_failed_login', 'logged_in', 'num_compromised', 'root_shell',
    'su_attempted', 'num_root', 'num_file_creations', 'num_shells',
    'num_access_files', 'num_outbound_cmds', 'is_host_login',
    'is_guest_login', 'count', 'srv_count', 'serror_rate',
    'srv_serror_rate', 'error_rate', 'srv_error_rate', 'same_srv_rate',
    'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_count',
    'dst_host_srv_count', 'dst_host_same_srv_rate',
    'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate',
    'dst_host_srv_diff_host_rate', 'dst_host_serror_rate',
    'dst_host_srv_serror_rate', 'dst_host_rerror_rate'
]
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i]).astype(str)
del df['dst_host_srv_rerror_rate']
del df["class"]

```

#According to the cross-validated MCC scores, the random forest is the best-performing model, so now let's evaluate its performance on the test set.

```
from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score
```

```
X = df.drop(labels='U2R', axis=1)
```

```
#Response variable
```

```
y = df.loc[:, 'U2R']
```

#We'll use a test size of 30%. We also stratify the split on the response variable, which is very important to do because there are so few fraudulent transactions.

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=1, stratify=y)
```

#According to the cross-validated MCC scores, the random forest is the best-performing model, so now let's evaluate its performance on the test set.

```
from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score
```

### **Logistic Regression:**

```
from sklearn.metrics import accuracy_score, confusion_matrix
```

```
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.model_selection import cross_val_score
```

```
logR= LogisticRegression()
```

```
logR.fit(X_train,y_train)
```

```
predictR = logR.predict(X_test)
```

```
print("")
```

```

print('Classification report of Logistic Regression Results:')
print("")
print(classification_report(y_test,predictR))

cm1=confusion_matrix(y_test,predictR)
print('Confusion Matrix result of Logistic Regression is:\n',cm1)
print("")
sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
print("")

accuracy = cross_val_score(logR, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Logistic Regression is:",accuracy.mean() * 100)
lr=accuracy.mean() * 100

```

### **Decision Tree:**

```

from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import cross_val_score

dt= DecisionTreeClassifier()
dt.fit(X_train,y_train)
predictR = dt.predict(X_test)
print("")

```

```

print('Classification report of Decision Tree Results:')
print("")
print(classification_report(y_test,predictR))

cm1=confusion_matrix(y_test,predictR)
print('Confusion Matrix result of Decision Tree is:\n',cm1)
print("")
sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
print("")

accuracy = cross_val_score(dt, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Decision Tree is:",accuracy.mean() * 100)
dt=accuracy.mean() * 100

```

### **Random Forest:**

```

from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(X_train,y_train)
predictR = rfc.predict(X_test)
print("")
print('Classification report of Random Forest Results:')
print("")
print(classification_report(y_test,predictR))

```

```

print("")
cm1=confusion_matrix(y_test,predictR)
print('Confusion Matrix result of Random Forest is:\n',cm1)
print("")
sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
print("")

accuracy = cross_val_score(rfc, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Random Forest is:",accuracy.mean() * 100)
rf=accuracy.mean() * 100

```

### **Support Vector Classifier:**

```

from sklearn.svm import SVC
sv = SVC()
sv.fit(X_train, y_train)
predictSVC = sv.predict(X_test)
print("")
print('Classification report of Support Vector Classifier Results:')
print("")
print(classification_report(y_test,predictSVC))

print("")
cm4=confusion_matrix(y_test,predictSVC)

```

```

print('Confusion Matrix result of Support Vector Classifier is:\n',
confusion_matrix(y_test,predictSVC))
print("")
sensitivity1 = cm4[0,0]/(cm4[0,0]+cm4[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm4[1,1]/(cm4[1,0]+cm4[1,1])
print('Specificity : ', specificity1)

accuracy = cross_val_score(sv, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Support Vector Classifier is:",accuracy.mean() * 100)
sv=accuracy.mean() * 100
def graph():

    import matplotlib.pyplot as plt
    data=[lr,dt,rf,sv]
    alg="LR","DT","RF","SVM"
    plt.figure(figsize=(10,5))
    b=plt.bar(alg,data,color=("r","g","b","y"))
    plt.title("Accuracy comparison of U2R Attacks",fontsize=15)
    plt.legend(b,data,fontsize=9)
    plt.savefig('U2R.png')
graph()

import tkinter
from matplotlib.backends.backend_tkagg import (FigureCanvasTkAgg,
NavigationToolbar2Tk)

```

```

from matplotlib.backend_bases import key_press_handler
from matplotlib.figure import Figure
import numpy as np
root = tkinter.Tk()
root.wm_title("Accuracy plot for U2R Attacks")
fig = Figure(figsize=(10,10),dpi=1)
canvas = FigureCanvasTkAgg(fig, master=root)
canvas.draw()
canvas.get_tk_widget().pack(side=tkinter.TOP, fill=tkinter.BOTH, expand=1)
icon=tkinter.PhotoImage(file='U2R.png')
label=tkinter.Label(root,image=icon)
label.pack()
root.mainloop()

```

## MODULE-5:

```

import pandas as p
import warnings
warnings.filterwarnings('ignore')
# feature names
features = ["duration", "protocol_type", "service", "flag", "src_bytes", "dst_bytes",
"land", "Wrong_fragment", "Urgent", "hot", "num_failed_login", "logged_in",
"num_compromised", "root_shell", "su_attempted", "num_root",
"num_file_creations", "num_shells", "num_access_files", "num_outbound_cmds",
"is_host_login", "is_guest_login", "count", "srv_count", "error_rate",
"srv_error_rate", "error_rate", "srv_error_rate", "same_srv_rate", "diff_srv_rate",
"srv_diff_host_rate", "dst_host_count", "dst_host_srv_count",
"dst_host_same_srv_rate", "dst_host_diff_srv_rate",
"dst_host_same_src_port_rate", "dst_host_srv_diff_host_rate",
"dst_host_error_rate", "dst_host_srv_error_rate", "dst_host_error_rate",
"dst_host_srv_error_rate","class"]

```

```

data = p.read_csv("data6.csv", names = features)
df=data.dropna()
df['Probe'] = df['class'].map({'normal.':0, 'snmpgetattack.':0, 'named.':0, 'xlock.':0,
'smurf.':0,
    'ipsweep.':1, 'multihop.':0, 'xsnoop.':0, 'sendmail.':0, 'guess_passwd.':0,
    'saint.':1, 'buffer_overflow.':0, 'portsweep.':1, 'pod.':0, 'apache2.':0,
    'phf.':0, 'udpstorm.':0, 'warezmaster.':0, 'perl.':0, 'satan.':1, 'xterm.':0,
    'mscan.':1, 'processtable.':0, 'ps.':0, 'nmap.':1, 'rootkit.':0, 'neptune.':0,
    'loadmodule.':0, 'imap.':0, 'back.':0, 'httptunnel.':0, 'worm.':0,
    'mailbomb.':0, 'ftp_write.':0, 'teardrop.':0, 'land.':0, 'sqlattack.':0,
    'snmpguess.':0})

```

```

from sklearn.preprocessing import LabelEncoder
var_mod = ['duration', 'protocol_type', 'service', 'flag', 'src_bytes',
    'dst_bytes', 'land', 'Wrong_fragment', 'Urgent', 'hot',
    'num_failed_login', 'logged_in', 'num_compromised', 'root_shell',
    'su_attempted', 'num_root', 'num_file_creations', 'num_shells',
    'num_access_files', 'num_outbound_cmds', 'is_host_login',
    'is_guest_login', 'count', 'srv_count', 'serror_rate',
    'srv_serror_rate', 'error_rate', 'srv_error_rate', 'same_srv_rate',
    'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_count',
    'dst_host_srv_count', 'dst_host_same_srv_rate',
    'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate',
    'dst_host_srv_diff_host_rate', 'dst_host_serror_rate',
    'dst_host_srv_serror_rate', 'dst_host_rerror_rate'
]
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i]).astype(int)
del df['dst_host_srv_rerror_rate']
del df["class"]

```

#According to the cross-validated MCC scores, the random forest is the best-performing model, so now let's evaluate its performance on the test set.

```
from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score
```

```
X = df.drop(labels='Probe', axis=1)
```

```
#Response variable
```

```
y = df.loc[:, 'Probe']
```

#We'll use a test size of 30%. We also stratify the split on the response variable, which is very important to do because there are so few fraudulent transactions.

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=1, stratify=y)
```

#According to the cross-validated MCC scores, the random forest is the best-performing model, so now let's evaluate its performance on the test set.

```
from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score
```

### **Logistic Regression:**

```
from sklearn.metrics import accuracy_score, confusion_matrix
```

```
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.model_selection import cross_val_score
```

```
logR= LogisticRegression()
```

```
logR.fit(X_train,y_train)
```

```
predictR = logR.predict(X_test)
```

```

print("")
print('Classification report of Logistic Regression Results:')
print("")
print(classification_report(y_test,predictR))

cm1=confusion_matrix(y_test,predictR)
print('Confusion Matrix result of Logistic Regression is:\n',cm1)
print("")
sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
print("")

accuracy = cross_val_score(logR, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Logistic Regression is:",accuracy.mean() * 100)
lr=accuracy.mean() * 100

```

### **Decision Tree:**

```

from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import cross_val_score

dt= DecisionTreeClassifier()
dt.fit(X_train,y_train)
predictR = dt.predict(X_test)

```

```

print("")
print('Classification report of Decision Tree Results:')
print("")
print(classification_report(y_test,predictR))

cm1=confusion_matrix(y_test,predictR)
print('Confusion Matrix result of Decision Tree is:\n',cm1)
print("")
sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
print("")

accuracy = cross_val_score(dt, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Decision Tree is:",accuracy.mean() * 100)
dt=accuracy.mean() * 100

```

### **Random Forest:**

```

from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(X_train,y_train)
predictR = rfc.predict(X_test)
print("")
print('Classification report of Random Forest Results:')
print("")

```

```

print(classification_report(y_test,predictR))
print("")
cm1=confusion_matrix(y_test,predictR)
print('Confusion Matrix result of Random Forest is:\n',cm1)
print("")
sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
print("")

accuracy = cross_val_score(rfc, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Random Forest is:",accuracy.mean() * 100)
rf=accuracy.mean() * 100

```

### **Support Vector Classifier:**

```

from sklearn.svm import SVC
sv = SVC()
sv.fit(X_train, y_train)
predictSVC = sv.predict(X_test)
print("")
print('Classification report of Support Vector Classifier Results:')
print("")
print(classification_report(y_test,predictSVC))

print("")

```

```

cm4=confusion_matrix(y_test,predictSVC)
print('Confusion Matrix result of Support Vector Classifier is:\n',
confusion_matrix(y_test,predictSVC))
print("")
sensitivity1 = cm4[0,0]/(cm4[0,0]+cm4[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm4[1,1]/(cm4[1,0]+cm4[1,1])
print('Specificity : ', specificity1)


accuracy = cross_val_score(sv, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Support Vector Classifier is:",accuracy.mean() * 100)
sv=accuracy.mean() * 100
def graph():
    import matplotlib.pyplot as plt
    data=[lr,dt,rf,sv]
    alg="LR","DT","RF","SVM"
    plt.figure(figsize=(10,5))
    b=plt.bar(alg,data,color=("r","g","b","y"))
    plt.title("Accuracy comparison of Probe Attacks",fontsize=15)
    plt.legend(b,data,fontsize=9)
    plt.savefig('Probe.png')
graph()


import tkinter
from matplotlib.backends.backend_tkagg import (FigureCanvasTkAgg,
NavigationToolbar2Tk)

```

```

from matplotlib.backend_bases import key_press_handler
from matplotlib.figure import Figure
import numpy as np
root = tkinter.Tk()
root.wm_title("Accuracy plot for Probe Attacks")
fig = Figure(figsize=(10,10),dpi=1)
canvas = FigureCanvasTkAgg(fig, master=root)
canvas.draw()
canvas.get_tk_widget().pack(side=tkinter.TOP, fill=tkinter.BOTH, expand=1)
icon=tkinter.PhotoImage(file='Probe.png')
label=tkinter.Label(root,image=icon)
label.pack()
root.mainloop()

```

#### **MODULE-6:**

```

import pandas as p
import warnings
warnings.filterwarnings('ignore')
# feature names
features = ["duration", "protocol_type", "service", "flag", "src_bytes", "dst_bytes",
"land", "Wrong_fragment", "Urgent", "hot", "num_failed_login", "logged_in",
"num_compromised", "root_shell", "su_attempted", "num_root",
"num_file_creations", "num_shells", "num_access_files", "num_outbound_cmds",
"is_host_login", "is_guest_login", "count", "srv_count", "error_rate",
"srv_error_rate", "error_rate", "srv_error_rate", "same_srv_rate", "diff_srv_rate",
"srv_diff_host_rate", "dst_host_count", "dst_host_srv_count",
"dst_host_same_srv_rate", "dst_host_diff_srv_rate",
"dst_host_same_src_port_rate", "dst_host_srv_diff_host_rate",
"dst_host_error_rate", "dst_host_srv_error_rate", "dst_host_error_rate",
"dst_host_srv_error_rate","class"]
data = p.read_csv("data6.csv", names = features)

```

```

df=data.dropna()

df['attack'] = df['class'].map({'normal.':0, 'snmpgetattack.':1, 'named.':1, 'xlock.':1,
'smurf.':1,
    'ipsweep.':1, 'multihop.':1, 'xsnoop.':1, 'sendmail.':1, 'guess_passwd.':1,
    'saint.':1, 'buffer_overflow.':1, 'portsweep.':1, 'pod.':1, 'apache2.':1,
    'phf.':1, 'udpstorm.':1, 'warezmaster.':1, 'perl.':1, 'satan.':1, 'xterm.':1,
    'mscan.':1, 'processtable.':1, 'ps.':1, 'nmap.':1, 'rootkit.':1, 'neptune.':1,
    'loadmodule.':1, 'imap.':1, 'back.':1, 'httptunnel.':1, 'worm.':1,
    'mailbomb.':1, 'ftp_write.':1, 'teardrop.':1, 'land.':1, 'sqlattack.':1,
    'snmpguess.':1})

from sklearn.preprocessing import LabelEncoder
var_mod = ['duration', 'protocol_type', 'service', 'flag', 'src_bytes',
    'dst_bytes', 'land', 'Wrong_fragment', 'Urgent', 'hot',
    'num_failed_login', 'logged_in', 'num_compromised', 'root_shell',
    'su_attempted', 'num_root', 'num_file_creations', 'num_shells',
    'num_access_files', 'num_outbound_cmds', 'is_host_login',
    'is_guest_login', 'count', 'srv_count', 'serror_rate',
    'srv_serror_rate', 'rerror_rate', 'srv_rerror_rate', 'same_srv_rate',
    'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_count',
    'dst_host_srv_count', 'dst_host_same_srv_rate',
    'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate',
    'dst_host_srv_diff_host_rate', 'dst_host_serror_rate',
    'dst_host_srv_serror_rate', 'dst_host_rerror_rate'
    ]
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i]).astype(int)
del df['dst_host_srv_rerror_rate']
del df["class"]

#According to the cross-validated MCC scores, the random forest is the best-

```

performing model, so now let's evaluate its performance on the test set.

```
from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score
```

```
X = df.drop(labels='attack', axis=1)
```

```
#Response variable
```

```
y = df.loc[:, 'attack']
```

#We'll use a test size of 30%. We also stratify the split on the response variable, which is very important to do because there are so few fraudulent transactions.

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=1, stratify=y)
```

#According to the cross-validated MCC scores, the random forest is the best-performing model, so now let's evaluate its performance on the test set.

```
from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score
```

### **Logistic Regression:**

```
from sklearn.metrics import accuracy_score, confusion_matrix
```

```
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.model_selection import cross_val_score
```

```
logR= LogisticRegression()
```

```
logR.fit(X_train,y_train)
```

```
predictR = logR.predict(X_test)
```

```
print("")
```

```
print('Classification report of Logistic Regression Results:')
```

```

print("")
print(classification_report(y_test,predictR))

cm1=confusion_matrix(y_test,predictR)
print('Confusion Matrix result of Logistic Regression is:\n',cm1)
print("")
sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
print("")

accuracy = cross_val_score(logR, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Logistic Regression is:",accuracy.mean() * 100)
lr=accuracy.mean() * 100

```

### **Decision Tree Classifier:**

```

from sklearn.tree import DecisionTreeClassifier

dt= DecisionTreeClassifier()
dt.fit(X_train,y_train)
predictR = dt.predict(X_test)
print("")
print('Classification report of Decision Tree Results:')
print("")
print(classification_report(y_test,predictR))

```

```

cm1=confusion_matrix(y_test,predictR)
print('Confusion Matrix result of Decision Tree is:\n',cm1)
print("")
sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
print("")

accuracy = cross_val_score(dt, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Decision Tree is:",accuracy.mean() * 100)
dt=accuracy.mean() * 100

```

### **Random Forest:**

```

from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(X_train,y_train)
predictR = rfc.predict(X_test)
print("")
print('Classification report of Random Forest Results:')
print("")
print(classification_report(y_test,predictR))
print("")
cm1=confusion_matrix(y_test,predictR)
print('Confusion Matrix result of Random Forest is:\n',cm1)

```

```

print("")
sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
print("")

accuracy = cross_val_score(rfc, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Random Forest is:",accuracy.mean() * 100)
rf=accuracy.mean() * 100

```

### **Support Vector Classifier:**

```

from sklearn.svm import SVC
sv = SVC()
sv.fit(X_train, y_train)
predictSVC = sv.predict(X_test)
print("")
print('Classification report of Support Vector Classifier Results:')
print("")
print(classification_report(y_test,predictSVC))

print("")
cm4=confusion_matrix(y_test,predictSVC)
print('Confusion Matrix result of Support Vector Classifier is:\n',
confusion_matrix(y_test,predictSVC))
print("")

```

```

sensitivity1 = cm4[0,0]/(cm4[0,0]+cm4[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm4[1,1]/(cm4[1,0]+cm4[1,1])
print('Specificity : ', specificity1)

accuracy = cross_val_score(sv, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Support Vector Classifier is:",accuracy.mean() * 100)
sv=accuracy.mean() * 100
def graph():
    import matplotlib.pyplot as plt
    data=[lr,dt,rf,sv]
    alg="LR","DT","RF","SVM"
    plt.figure(figsize=(10,5))
    b=plt.bar(alg,data,color=("r","g","b","y"))
    plt.title("Accuracy comparison of Overall Attacks",fontsize=15)
    plt.legend(b,data,fontsize=9)
    plt.savefig('overallattack.png')
graph()

import tkinter
from matplotlib.backends.backend_tkagg import (FigureCanvasTkAgg,
NavigationToolbar2Tk)
from matplotlib.backend_bases import key_press_handler
from matplotlib.figure import Figure
import numpy as np
root = tkinter.Tk()

```

```

root.wm_title("Accuracy plot for Overall Attacks")
fig = Figure(figsize=(10,10),dpi=1)
canvas = FigureCanvasTkAgg(fig, master=root)
canvas.draw()
canvas.get_tk_widget().pack(side=tkinter.TOP, fill=tkinter.BOTH, expand=1)
icon=tkinter.PhotoImage(file='overallattack.png')
label=tkinter.Label(root,image=icon)
label.pack()
root.mainloop()

```

## MODULE-7:

```

#import library packages
import pandas as p
import matplotlib.pyplot as plt
import seaborn as s
import numpy as n
import warnings
warnings.filterwarnings('ignore')
# feature names
features = ["duration", "protocol_type", "service", "flag", "src_bytes", "dst_bytes",
"land", "Wrong_fragment", "Urgent", "hot", "num_failed_login", "logged_in",
"num_compromised", "root_shell", "su_attempted", "num_root",
"num_file_creations", "num_shells", "num_access_files", "num_outbound_cmds",
"is_host_login", "is_guest_login", "count", "srv_count", "serror_rate",
"srv_serror_rate", "error_rate", "srv_error_rate", "same_srv_rate", "diff_srv_rate",
"srv_diff_host_rate", "dst_host_count", "dst_host_srv_count",

```

```

"dst_host_same_srv_rate",          "dst_host_diff_          srv_rate",
"dst_host_same_src_port_rate",    "dst_host_srv_diff_host          _rate",
"dst_host_serror_rate",    "dst_host_srv_serror_rate",    "dst_host_rerror_rate",
"dst_host_srv_rerror_rate","class"]
df = p.read_csv("demo3.csv", names = features)
df.head()
df['class'].unique()
df.head()

```

```

from tkinter import *
del df["duration"]
del df["land"]
del df["Urgent"]
del df["hot"]
del df["num_failed_login"]
del df["logged_in"]
del df["num_compromised"]
del df["root_shell"]
del df["is_host_login"]
del df["is_guest_login"]

del df['num_root']
del df['num_file_creations']
del df['num_shells']
del df['num_outbound_cmds']
del df['count']
del df['srv_count']
del df['srv_serror_rate']
del df['srv_rerror_rate']
del df['same_srv_rate']
del df['diff_srv_rate']

```

```

del df['srv_diff_host_rate']
del df['dst_host_count']
del df['dst_host_srv_count']
del df['dst_host_same_srv_rate']
del df['dst_host_diff_srv_rate']
del df['dst_host_same_src_port_rate']
del df['dst_host_srv_diff_host_rate']
del df['dst_host_serror_rate']
del df['dst_host_srv_serror_rate']
del df['dst_host_rerror_rate']
del df['dst_host_srv_rerror_rate']

del df['su_attempted']
del df['num_access_files']
df.columns
df['protocol_type'].unique()
df['UDP'] = df.protocol_type.map({'udp':1, 'tcp':0, 'icmp':0})
df['TCP'] = df.protocol_type.map({'udp':0, 'tcp':1, 'icmp':0})
df['ICMP'] = df.protocol_type.map({'udp':0, 'tcp':0, 'icmp':1})
del df['protocol_type']
df['service'].unique()
df['private'] = df.service.map({'ecr_i':0, 'http':0, 'private':1})

df['http'] = df.service.map({'ecr_i':0, 'http':1, 'private':0})
df['ecr_i'] = df.service.map({'ecr_i':1, 'http':0, 'private':0})
df['http'].unique()
del df['service']
df['flag'].unique()
df['SF'] = df.flag.map({'SF':1, 'S0':0, 'REJ':0, 'S1':0})
df['S1'] = df.flag.map({'SF':0, 'S0':0, 'REJ':0, 'S1':1})
df['REJ'] = df.flag.map({'SF':0, 'S0':0, 'REJ':1, 'S1':0})

```

```
df['S0'] = df.flag.map({'SF':0, 'S0':0, 'REJ':0, 'S1':1})
df['S0'].unique()
```

```
del df['flag']
df.columns
df['src_bytes'].unique()
```

```
df['SRC_BY_BL_50'] = df.src_bytes.map({1032:0, 283:0, 252:0, 0:1, 105:0,
303:0, 42:1, 45:1, 213:0, 285:0, 5050:0,
212:0, 184:0, 289:0, 291:0, 246:0, 175:0, 241:0, 293:0, 245:0, 249:0,
225:0,
305:0, 3894:0, 320:0, 162:0, 206:0, 353:0, 1:1})
```

```
df['SRC_BY_AB_50'] = df.src_bytes.map({1032:0, 283:0, 252:0, 0:0, 105:1,
303:0, 42:0, 45:0, 213:0, 285:0, 5050:0,
212:1, 184:1, 289:0, 291:0, 246:1, 175:1, 241:1, 293:0, 245:1, 249:1,
225:1,
305:0, 3894:0, 320:0, 162:1, 206:1, 353:0, 1:0})
```

```
df['SRC_BY_AB_250'] = df.src_bytes.map({1032:0, 283:1, 252:1, 0:0, 105:1,
303:0, 42:0, 45:0, 213:0, 285:1, 5050:0,
212:0, 184:0, 289:1, 291:1, 246:0, 175:1, 241:0, 293:1, 245:0, 249:0,
225:0,
305:1, 3894:0, 320:1, 162:1, 206:0, 353:1, 1:0})
```

```
df['SRC_BY_AB_450'] = df.src_bytes.map({1032:0, 283:1, 252:1, 0:0, 105:0,
303:0, 42:0, 45:0, 213:1, 285:1, 5050:0,
212:1, 184:0, 289:1, 291:1, 246:1, 175:0, 241:1, 293:1, 245:1, 249:1,
225:1,
305:0, 3894:0, 320:0, 162:0, 206:1, 353:0, 1:0})
```

```
df['SRC_BY_AB_650'] = df.src_bytes.map({1032:0, 283:0, 252:0, 0:0, 105:0,
```

```

303:0, 42:0, 45:0, 213:0, 285:0, 5050:0,
    212:0, 184:0, 289:0, 291:0, 246:0, 175:0, 241:0, 293:0, 245:0, 249:0,
225:0,
    305:0, 3894:0, 320:0, 162:0, 206:0, 353:0, 1:0})

```

```

df['SRC_BY_AB_850'] = df.src_bytes.map({1032:0, 283:0, 252:0, 0:0, 105:0,
303:0, 42:0, 45:0, 213:0, 285:0, 5050:0,
    212:0, 184:0, 289:0, 291:0, 246:0, 175:0, 241:0, 293:0, 245:0, 249:0,
225:0,
    305:0, 3894:0, 320:0, 162:0, 206:0, 353:0, 1:0})

```

```

df['SRC_BY_AB_1000'] = df.src_bytes.map({1032:1, 283:0, 252:0, 0:0, 105:0,
303:0, 42:0, 45:0, 213:0, 285:0, 5050:1,
    212:0, 184:0, 289:0, 291:0, 246:0, 175:0, 241:0, 293:0, 245:0, 249:0,
225:0,
    305:0, 3894:1, 320:0, 162:0, 206:0, 353:0, 1:0})

```

```
del df['src_bytes']
```

```
df.columns
```

```
df['dst_bytes'].unique()
```

```

df['DST_BY_BL_50'] = df.dst_bytes.map({ 0:1, 903:0, 1422:0, 146:0, 1292:0,
42:1, 115:0, 4996:0, 145:0,
    5200:0, 329:0, 341:0, 128:0, 721:0, 331:0, 753:0, 38352:0, 722:0,
    1965:0, 634:0, 628:0, 188:0, 47582:0, 489:0, 105:0, 486:0, 2940:0,
    209:0, 1401:0, 292:0, 1085:0, 1:1})

```

```

df['DST_BY_AB_50'] = df.dst_bytes.map({ 0:0, 903:0, 1422:0, 146:1, 1292:0,
42:0, 115:1, 4996:0, 145:1,
    5200:0, 329:0, 341:0, 128:1, 721:0, 331:0, 753:0, 38352:0, 722:0,
    1965:0, 634:0, 628:0, 188:1, 47582:0, 489:0, 105:1, 486:0, 2940:0,

```

```

209:1, 1401:0, 292:1, 1085:0, 1:0})

df['DST_BY_AB_250'] = df.dst_bytes.map({ 0:0, 903:0, 1422:0, 146:0, 1292:0,
42:0, 115:0, 4996:0, 145:0,
5200:0, 329:1, 341:1, 128:0, 721:0, 331:1, 753:0, 38352:0, 722:0,
1965:0, 634:0, 628:0, 188:0, 47582:0, 489:0, 105:0, 486:0, 2940:0,
209:1, 1401:0, 292:1, 1085:0, 1:0})

df['DST_BY_AB_450'] = df.dst_bytes.map({ 0:0, 903:0, 1422:0, 146:0, 1292:0,
42:0, 115:0, 4996:0, 145:0,
5200:0, 329:0, 341:0, 128:0, 721:0, 331:0, 753:0, 38352:0, 722:0,
1965:0, 634:1, 628:1, 188:0, 47582:0, 489:1, 105:0, 486:1, 2940:0,
209:0, 1401:0, 292:0, 1085:0, 1:0})

df['DST_BY_AB_650'] = df.dst_bytes.map({ 0:0, 903:0, 1422:0, 146:0, 1292:0,
42:0, 115:0, 4996:0, 145:0,
5200:0, 329:0, 341:0, 128:0, 721:1, 331:0, 753:1, 38352:0, 722:1,
1965:0, 634:0, 628:0, 188:0, 47582:0, 489:0, 105:0, 486:0, 2940:0,
209:0, 1401:0, 292:0, 1085:0, 1:0})

df['DST_BY_AB_850'] = df.dst_bytes.map({ 0:0, 903:1, 1422:0, 146:0, 1292:0,
42:0, 115:0, 4996:0, 145:0,
5200:0, 329:0, 341:0, 128:0, 721:0, 331:0, 753:0, 38352:0, 722:0,
1965:0, 634:0, 628:0, 188:0, 47582:0, 489:0, 105:0, 486:0, 2940:0,
209:0, 1401:0, 292:0, 1085:0, 1:0})

df['DST_BY_AB_1000'] = df.dst_bytes.map({ 0:0, 903:0, 1422:1, 146:0, 1292:1,
42:0, 115:0, 4996:1, 145:0,
5200:1, 329:0, 341:0, 128:0, 721:0, 331:0, 753:0, 38352:1, 722:0,
1965:1, 634:0, 628:0, 188:0, 47582:1, 489:0, 105:0, 486:0, 2940:1,
209:0, 1401:0, 292:0, 1085:1, 1:0})

```

```

df['DST_BY_AB_1000'].unique()
del df['dst_bytes']
del df['Wrong_fragment']
del df['serror_rate']
del df['rerror_rate']
df.head()
df.columns

```

```

l1=['SRC_BY_BL_50', 'SRC_BY_AB_50', 'SRC_BY_AB_250','SRC_BY_AB_450',
'SRC_BY_AB_650', 'SRC_BY_AB_850', 'SRC_BY_AB_1000']

```

```

l2=['DST_BY_BL_50',          'DST_BY_AB_50',          'DST_BY_AB_250',
'DST_BY_AB_450','DST_BY_AB_650', 'DST_BY_AB_850', 'DST_BY_AB_1000']

```

```

l3=['UDP', 'TCP', 'ICMP']

```

```

l4=['SF', 'S1', 'REJ', 'S0']

```

```

l5=['private','http', 'ecr_i']

```

```

l6=['UDP', 'TCP', 'ICMP', 'private', 'http', 'ecr_i', 'SF', 'S1','REJ', 'S0',
'SRC_BY_BL_50', 'SRC_BY_AB_50', 'SRC_BY_AB_250',
'SRC_BY_AB_450', 'SRC_BY_AB_650', 'SRC_BY_AB_850', 'SRC_BY_AB_1000',
'DST_BY_BL_50',          'DST_BY_AB_50',          'DST_BY_AB_250',
'DST_BY_AB_450','DST_BY_AB_650', 'DST_BY_AB_850', 'DST_BY_AB_1000']
df['class'].unique()

```

```

decision = ['smurf', 'perl', 'xlock', 'xsnoop', 'xterm', 'satan', 'neptune','nmap', 'back',
'apache2', 'multihop', 'worm',
'buffer overflow','sql attack', 'saint', 'Nmap', 'ipsweep']

```

```

l7=[]

```

```

for x in range(0,len(l6)):

```

```

    l7.append(0)

```

```

df['class'].unique()

```

```

df.replace({'class':{'smurf':0, 'perl':1, 'xlock':2, 'xsnoop':3, 'xterm':4, 'satan':5,

```

```
'neptune':6,
    'nmap':7, 'back':8, 'apache2':9, 'multihop':10, 'worm':11, 'buffer overflow':12,
    'sql attack':13, 'saint':14, 'Nmap':15, 'ipsweep':16}},inplace=True)
```

```
import numpy as np
```

```
Xd= df[l6]
```

```
yd = df[["class"]]
```

```
np.ravel(yd)
```

```
import numpy as np
```

```
X_testd= df[l6]
```

```
y_testd = df[["class"]]
```

```
np.ravel(y_testd)
```

```
from sklearn.svm import SVC
```

```
from sklearn.model_selection import cross_val_score
```

```
from sklearn.metrics import accuracy_score
```

```
def over():
```

```
    clf = SVC()
```

```
    gnb=clf.fit(Xd,np.ravel(yd))
```

```
    # calculating accuracy-----
```

```
    from sklearn.metrics import accuracy_score
```

```
    y_predd=gnb.predict(X_testd)
```

```
    print(accuracy_score(y_testd, y_predd))
```

```
    print(accuracy_score(y_testd, y_predd,normalize=False))
```

```
    # -----
```

```
terms = [src.get(),dst.get(),prt.get(),fl.get(),ser.get()]
```

```
for k in range(0,len(l6)):
```

```
    for z in terms:
```

```
        if(z==l6[k]):
```

```
            l7[k]=1
```

```
inputtest = [l7]
```

```
predict = gnb.predict(inputtest)
```

```
predicted=predict[0]
```

```
h='no'
```

```
for a in range(0,len(decision)):
```

```
    if(predicted == a):
```

```
        h='yes'
```

```
        break
```

```
if (h=='yes'):
```

```
    t1.delete("1.0", END)
```

```
    t1.insert(END, decision[a])
```

```
else:
```

```
    t1.delete("1.0", END)
```

```
    t1.insert(END, "Not Found")
```

```
root1 = Tk()
```

```
root1.title("Prediction of Network Attacks")
```

```
#root1.configure(background='black')
```

```
root = Canvas(root1,width=1620,height=1800)
```

```
root.pack()
```

```

photo = PhotoImage(file ='im3.png')
root.create_image(0,0,image=photo,anchor=NW)

src = StringVar()
src.set(None)

dst = StringVar()
dst.set(None)

prt = StringVar()
prt.set(None)

fl = StringVar()
fl.set(None)

ser = StringVar()
ser.set(None)
# Heading
w2 = Label(root, justify=LEFT, text="Network attack prediction ", fg="red",
bg="white")
w2.config(font=("Elephant", 20))
w2.grid(row=1, column=0, columnspan=2, padx=100)
w2 = Label(root, justify=LEFT, text="DoS, R2L, U2R and Probe Types ", fg="blue")
w2.config(font=("Aharoni", 15))
w2.grid(row=2, column=0, columnspan=2, padx=100)
# labels
srcLb = Label(root, text="Source File Size(in BY):")
srcLb.grid(row=6, column=0, pady=15, sticky=W)

dstLb = Label(root, text="Destination File Size(in BY):")
dstLb.grid(row=7, column=0, pady=15, sticky=W)

```

```
prtLb = Label(root, text="Protocol Type:")
prtLb.grid(row=8, column=0, pady=15, sticky=W)
```

```
flLb = Label(root, text="Flag Type:")
flLb.grid(row=9, column=0, pady=15, sticky=W)
```

```
serLb = Label(root, text="Select services:")
serLb.grid(row=10, column=0, pady=15, sticky=W)
```

```
lrdLb = Label(root, text="Attack_Type", fg="white", bg="red")
lrdLb.grid(row=13, column=0, pady=10, sticky=W)
```

```
# entries
```

```
OPTIONSsrc = sorted(l1)
```

```
OPTIONSdst = sorted(l2)
```

```
OPTIONSprt = sorted(l3)
```

```
OPTIONSfl = sorted(l4)
```

```
OPTIONSser = sorted(l5)
```

```
srcEn = OptionMenu(root, src,*OPTIONSsrc)
```

```
srcEn.grid(row=6, column=1)
```

```
dstEn = OptionMenu(root, dst,*OPTIONSdst)
```

```
dstEn.grid(row=7, column=1)
```

```
prtEn = OptionMenu(root, prt,*OPTIONSprt)
```

```
prtEn.grid(row=8, column=1)
```

```
flEn = OptionMenu(root, fl,*OPTIONSfl)
```

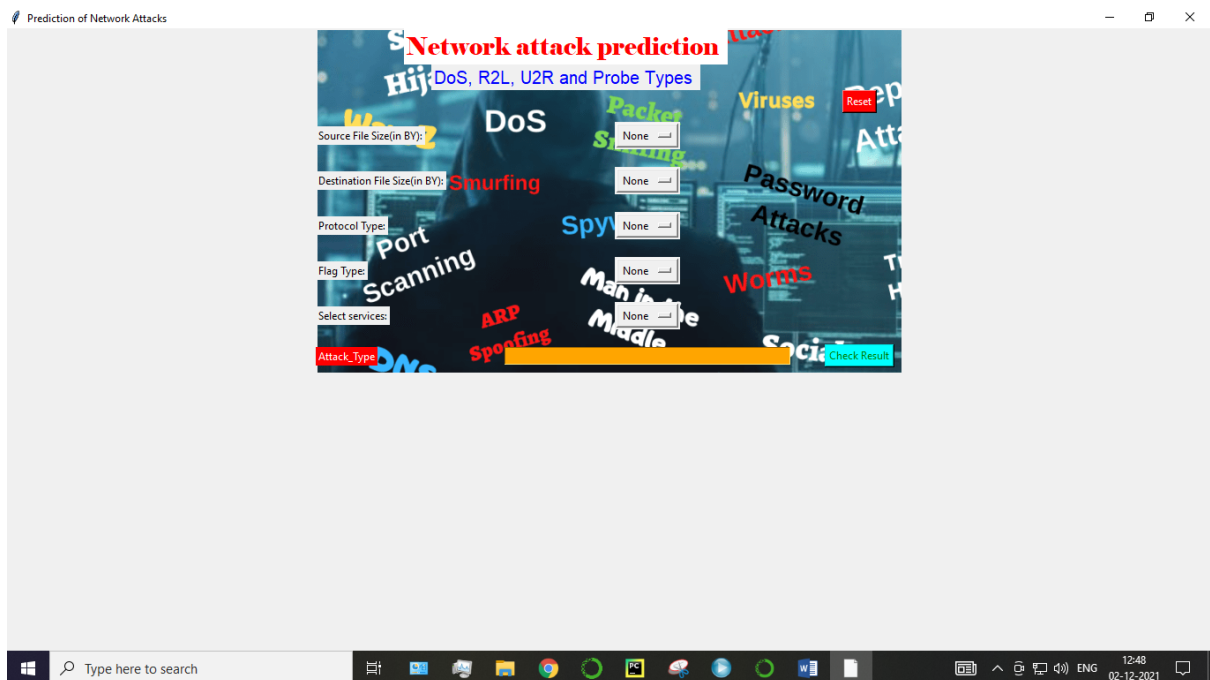
```
flEn.grid(row=9, column=1)
```

```

serEn = OptionMenu(root, ser,*OPTIONSser)
serEn.grid(row=10, column=1)
def clear_display_result():
    t1.delete('1.0',END)
lrd = Button(root, text="Check Result", command=over,bg="cyan",fg="green")
lrd.grid(row=13, column=3,padx=10)
b = Button(root, text="Reset", command=clear_display_result,bg="red",fg="white")
b.grid(row=5, column=3,padx=10)
t1 = Text(root, height=1, width=40,bg="orange",fg="black")
t1.grid(row=13, column=1 , padx=10)
root1.mainloop()

```

## B. SCREENSHOTS



**Fig 6.1: GUI before input**

