

PROFESSIONAL TRAINING REPORT
at
Sathyabama Institute of Science and Technology
(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of

Bachelor of Engineering Degree in

Computer Science and Engineering

By

ALLEN V JOSE

(38110028)



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

MARCH 2022

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY



(Established under Section 3 of UGC Act, 1956)

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai -600 119

www.sathyabama.ac.in



SCHOOL OF COMPUTING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafide work of **ALLEN V JOSE (38110028)** who carried out the project entitled “**PREDICTION OF NETWORK ATTACKS USING SUPERVISED MACHINE LEARNING ALGORITHM**” under our supervision from NOVEMBER 2021 to MARCH 2022.

Internal Guide

Dr.MERCY PAUL, M.E.,Ph.D.

Head of the Department

Dr.S. VIGNESHWARI,ME.,Ph.D.

Submitted for Viva voce Examination held on

Internal Examiner

External Examiner

DECLARATION

I, **ALLEN V JOSE (38110028)** hereby declare that the Professional Training Report on **“PREDICTION OF NETWORK ATTACKS USING SUPERVISED MACHINE LEARNING ALGORITHM”** done by me under the guidance of **Dr.MERCY PAUL,M.E.,Ph.D.**, at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE:

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** and **Dr. S.Vigneshwari, M.E., Ph.D., Head of the Department, Department of Computer Science and Engineering** for providing me the 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.MERCY PAUL,M.E.,Ph.D.**, for her valuable guidance, suggestions and constant 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 project.

ABSTRACT

Intrusion Detection System (IDS) needs a data and for this it is important to keep the real working environment to find out all the possibilities of how an attack is about to happen and this seem to be expensive. A Software to detect network attacks in a computer network from unidentified users, including known personnel. The attack detector's learning task works up a predictive model which is a classifier in this case which differentiates the "bad" (i.e., intrusions or attacks) and "good" or "normal" connections. The primary aim is to use machine learning based techniques to provide packet connection transfer in a better way by predicting results with the at most accuracy. Comparing and discussing the outputs from the couple of machine learning algorithms used for the given dataset with evaluated classification report, find the confusion matrix and categorize the data from priority and the result which shows that the efficiency of the claimed machine learning algorithm method is to be compared with the best accuracy techniques such as Precision, Recall and F1 Score.

LIST OF FIGURES

Fig 1	Process of machine learning	13
Fig 2	Data Flow diagram	23
Fig 3	Data Frame	35
Fig 4	Percentage level of protocol type	36
Fig 5	Comparison of server and protocol type	37
Fig 6	Open anaconda navigator	40
Fig 7	Launch jupyter notebook	41
Fig 8	Open correspondent folder	41
Fig 9	Module 1 result	42
Fig 10	Module 2 result graph	42
Fig 11	Module 2 pie chart	43
Fig 12	Module 3 result code	43

LIST OF ABBREVIATIONS

Entity Relationship Diagram (ERD)

Intrusion Detection System (IDS)

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO.
	LIST OF FIGURES	ii
	LIST OF ABBREVIATIONS	iii
1	INTRODUCTION	1
	1.1 Aim	1
	1.2 Abstract	1
2	LITERATURE SURVEY	2
	2.1 General	2
	2.2 Review of literature survey	2
3	SYSTEM ANALYSIS	9
	3.1 Existing system	9
	3.2 Proposed system	9
	3.3 Requirement specification	11
	3.3.1 Project requirements	11
	3.3.2. Software and Hardware Requirements	12
	3.4 Technologies Used	12
	3.4.1 Introduction to machine learning	12
	3.4.2 Preparing the dataset	17
	3.4.3 Introduction to anaconda	17
	3.5 Outline of the project	22
	3.5.1 Overview of system	22
	3.5.2 Objective	26
	3.5.3 Project goals	27
	3.5.4 Problem description	27
	3.5.5 Scope	28

	3.6 Design engineering	28
	3.6.1 System architecture	29
	3.6.2 Data flow diagram	30
	3.6.3 Use case diagram	31
	3.6.4 Class diagram	32
	3.6.5 Entity relationship diagram	33
	3.7 Modules	34
	3.7.1 Modules 01	34
	3.7.2 Modules 02	35
	3.7.3 Modules 03	38
	3.8 Python packages used	39
4	RESULT AND DISCUSSION	40
	4.1 Software involvement steps	40
5	CONCLUSION	44
	5.1 Conclusion	44
	5.2 Future Work	47
	5.4 Appendices	45
	5.4.1 Sample code	48
	5.5 Publication	59
	SOURCE CODE	48
	REFERENCES	44

CHAPTER 1

1. INTRODUCTION

1.1 AIM:

The main aim of this project is to predict the network attacks in a chain of network using supervised machine learning language.

1.2 ABSTRACT:

To create data for the Intrusion Detection System (IDS), it is necessary to set the real working environment to explore all the possibilities of attacks, which is expensive. Software to detect network intrusions protects a computer network from unauthorized users, including perhaps insiders. The intrusion detector learning task is to build a predictive model (i.e. a classifier) capable of distinguishing between "bad" connections, called intrusions or attacks, and "good" normal connections. To prevent this problem in network sectors, have to predict whether the connection is attacked or not from KDDCup99 dataset using machine learning techniques. The aim is to investigate machine learning based techniques for better packet connection transfers forecasting by prediction results in best accuracy. To propose a machine learning-based method to accurately predict the overall attacks by prediction results in the form of best accuracy from comparing supervise classification machine learning algorithms. Additionally, to compare and discuss the performance of various machine learning algorithms from the given dataset with evaluation classification report, identify the confusion matrix and to categorizing data from priority and the result shows that the effectiveness of the proposed machine learning algorithm technique can be compared with best accuracy with precision, Recall and F1 Score.

CHAPTER 2

2. LITERATURE SURVEY

2.1 General

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 discusses 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.

2.2 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. 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.

Title : Apriori Viterbi Model for Prior Detection of Socio-Technical Attacks in a 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 starts 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 : 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

2. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

To focus on the conception of a monitoring system able to detect and classify jamming and protocol-based attacks and achieve this goal, we proposed to outsource the attack detection function from the network to protect and used an antenna to monitor the spectrum over the time. The Wi-Fi network and the attacks were carried out in an anechoic chamber to avoid disturbing other Wi-Fi communication networks in the vicinity. The spectra highlight that the frequencies of interest belong to the communication channel between 2.402 and 2.422 GHz. Focusing the analysis on this 20-MHz frequency band permits to construct a classification model to overcome the problems induced by the utilization of the adjacent channels that can be or not occupied by other Wi-Fi communications. On these frequencies, the proposed estimation model shows good results in the prediction of attacks. In addition, the correction using the K spectra nearest in time permits to correct most of the miss classification.

The development of connected devices and their daily use is presently at the origin of the omnipresence of Wi-Fi wireless networks. However, these Wi-Fi networks are often vulnerable, and can be used by malicious people to disturb services, intercept sensitive data, or to gain access to the system. In railways, trains are now equipped with wireless communication systems for

operational purposes or for passenger services. In both cases, defense strategies have to be developed to prevent the misuses of the networks.

The first objective of this study is to propose a monitoring solution, which is independent of the communication networks, to detect the occurrence of attacks. The second objective is to develop a method that is able to classify attacks of different types: the intentional electromagnetic interference, i.e., jamming attacks and the protocol-based attacks. To perform these analyses, we propose to monitor and to analyze electromagnetic (EM) signals received by a monitoring antenna and a receiver collecting the EM spectra. After that, we build a classification protocol following two steps: the first consists in the construction of a support vector machine (SVM) classification model using the collected spectra, and the second step uses this SVM model to predict the class of the attack (if any). A time-based correction of this prediction using the nearest neighbors is also included in this second step.

Drawbacks:

- It can't discuss to know how our model can evolve in the case where unknown attack occurs with all types of attacks by popular machine learning algorithms.
- Algorithm prediction results by best accuracy of classification algorithms with classification report of precision, recall and f1-score and additionally, to categorized other attacks of network connections.

3.2 PROPOSED SYSTEM

Exploratory Data Analysis:

This analysis is not meant to be providing a final conclusion on the reasons leading to network sector as it doesn't involve using any inferential statistics techniques/machine learning algorithms. Machine learning supervised classification algorithms will be used to give the network connection dataset and extract patterns, which would help in predicting the likely patient affected or not, thereby helping the attack of avoids for making better decisions in the future. Multiple datasets from different sources would be combined to form a generalized dataset, and

then different machine learning algorithms would be applied to extract patterns and to obtain results with maximum accuracy.

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.

Data collection:

The data set collected for predicting the network attacks 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 logistic Regression, are applied on the Training set and based on the test result accuracy, Test set prediction is done.

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.

The correlation among attributes can be identified using plot diagram in data visualization process. Data preprocessing is the most time-consuming phase of a data mining process. Data cleaning of connections, data removed several attributes that has no significance about the behavior of a packet transfers. Data integration, data reduction and data transformation are also to be applicable for network connections dataset. For easy analysis, the data is reduced to some minimum number of records. Initially the Attributes which are critical to make a loan

credibility prediction is identified with information gain as the attribute-evaluator and Ranker as the search-method.

3.3 REQUIREMENT SPECIFICATIONS

3.3.1. Project Requirements

General:

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

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.

Non-Functional Requirements:

Process of functional steps,

1. Problem defines

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

3.3.2 SOFTWARE AND HARWARE 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

3.4 TECHNOLOGIES USED:

3.4.1 Introduction to 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.

Data scientists use many different kinds of machine learning algorithms to discover patterns in python that lead to actionable insights. At a high level, these different algorithms can be classified into two groups based on the way they “learn” about data to make predictions: supervised and unsupervised learning. Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function from input variables(X) to discrete output variables(y). In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc.



Fig 1: Process of Machine learning

Supervised Machine Learning is the majority of practical machine learning uses supervised learning. Supervised learning is where have input variables (X) and an output variable (y) and use an algorithm to learn the mapping function from the input to the output is $y = f(X)$. The goal is to approximate the mapping function so well that when you have new input data (X) that you can predict the output variables (y) for that data. Techniques of Supervised Machine Learning algorithms include logistic regression, multi-class classification, Decision Trees and support vector machines etc. Supervised learning requires that the data used to train the algorithm is already labeled with correct answers. Supervised learning problems can be further grouped into Classification problems. This problem has as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for categorical for classification. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. A classification problem is when the output variable is a category, such as “red” or “blue”.

As networked systems become more and more pervasive and businesses continue to move more and more of their sensitive data online, the number and sophistication of cyber-attacks and network security breaches has risen dramatically. As FBI Director James Comey stated earlier year, “There are two kinds of big companies in the United States. There are those who’ve been hacked... and those who don’t yet know they’ve been hacked.”

In order to secure their infrastructure and protect sensitive assets, organizations are increasingly relying on network intrusion detection systems (NIDS) to automatically monitor their network traffic and report suspicious or anomalous behavior. Historically, most NIDS operate in one of two styles: misuse detection and anomaly detection. Misuse detection searches for precise signatures of known malicious behavior, while anomaly detection tries to build a model for what constitutes “normal” network traffic patterns and then flag deviations from those patterns. For all the same reasons that signature-based antivirus software is becoming obsolete (the ease of spoofing signatures and the increasing diversity and sophistication of new attacks), misuse-detection is struggling to remain relevant in today’s threat landscape. Anomaly-based

intrusion detection offers the enticing prospect of being able to detect novel attacks even before they've been studied and characterized by security analysts, as well as being able to detect variations on existing attack methods. In our project we focus on classifying anomalies using both supervised and unsupervised learning techniques.

In order to create data for the IDS, it is necessary to set the real working environment to explore all the possibilities of attacks, which is expensive. Data analysis phase systematically identifies the patterns in the gathered information, and narrates them to the defined issue. It is a process of examining, transforming and modeling of data and deciding how to organize, classify, interrelate, compare and display it. Data quality focuses the correctness and reliability of information gathered and utilized in an evaluation. Data quantity deals with the quantity of information gathered for the evaluation.

This task requires various ground truth databases in its region and the experimentation would be completed effectively if the quality and features of data for the specific region are good. Image processing, web site analysis, medical applications, remote sensing, etc. and have standard and authorized ground truth databases for analysis. Likewise, most of the computer network intrusion detection systems use the KDD Cup99 for the classification analysis of network traffic and it explains the formation of KDD Cup99 dataset and its features. The KDDCup99 data set stems from data gathered at MIT Lincoln Laboratory under sponsorship of the Defense Advanced Research Projects Agency (DARPA) to evaluate Intrusion Detection Systems (IDSs) in 1998 and 1999. These two data sets are referred to as DARPA98 and DARPA99, which consist of raw TCP dump data from a simulated medium sized US air force base. The KDDCup99 data set was provided by Stolfo and Lee for the Knowledge Discovery and Data Mining Tools competition (and associated conference) in 1999. This is a transformed version of the DARPA TCP dump data, consisting of a set of features considered suitable for classification with machine learning algorithms. The data set consists of 41 features, some of which are intrinsic to the network connections, whilst other are created using domain knowledge.

Features are grouped into four categories:

Basic Features: These features are directly obtained from packet headers.

Content Features: Domain knowledge is applied to assess data portion of the TCP packets. Features like number of failed login attempts are content features.

Time-based Traffic Features: These features are designed to capture properties that mature over a 2 second temporal window. One example of such a feature would be the number of connections to the same host over the 2 second interval.

Host-based Traffic Features: Some probing attacks scan the hosts (or ports) using a much larger time interval than two seconds, for example once per minute. Therefore, connection records were also sorted by destination host, and features were constructed using a window of 100 connections to the same host instead of a time window.

3.4.2 Preparing the Dataset:

The dataset is now supplied to machine learning model on the basis of this data set the model is trained. In the first step of accumulating information, data from previously Attacks affected patients' datasets from online sources are gathered together. These datasets are merged to form a common dataset, on which analysis will be done. The Association for Computing Machinery (ACM) has a special interest group on Knowledge Discovery and Data mining (KDD) 38 (<http://www.sigkdd.org/kddcup>) which is the most popular professional organization of data miners. The KDD organized the annual Data Mining and Knowledge Discovery competition called KDD Cup in different areas. The data (referring to a group of packets over a time duration of 2 seconds, also named as packet data in upcoming discussion) set in KDD Cup99 have 41 features. Among the 41 features, 1-9 are used to represent the basic features of a packet, 10-22 employ the content features, 23-31 are used for traffic features with two seconds of time window and 32-41 for host-based features.

They are basically grouped into three categories: basic features of individual connection, content features within a connection, and traffic features which are computed using a two seconds time window of 2 seconds' window. 41 features collected from the packet data observed over the time window of 2 seconds' window.

Some of the terminologies associated with the data set are:

- The term 'same host' refers to the connections in the past two seconds that have the same destination host as the current connection, and is attached to the features like protocol activities, service etc.
- The term 'same service' refers to the connections in the past two seconds that have the same service as the current connection.
- The features based on 'same host' and 'same service' are collectively called as time-based traffic features of the connection records

3.4.3 Introduction to Anaconda:

Anaconda 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. The open-source packages can be individually installed from the Anaconda repository with the Conda install command or using the pip install command that is installed with Anaconda. Pip packages provide many of the features of Conda packages and in most cases they can work together. Custom packages can be made using the Conda build command, and can be shared with others by uploading them to Anaconda Cloud, PyPI or other repositories.

The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, you can create new environments that include any version of Python packaged with Conda.

a) Anaconda Navigator:

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage Conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux.

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- QtConsole
- Spyder
- Glueviz
- Orange
- Rstudio
- Visual Studio Code

b) Conda:

Conda is an open source, cross-platform, language-agnostic package manager and environment management system that installs, runs and updates packages and their dependencies.

It was created for Python programs, but it can package and distribute software for any language (e.g., R), including multi-languages. The Conda package and environment manager is included in all versions of Anaconda, Miniconda, and Anaconda Repository.

c) The Jupyter Notebook:

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

d) Notebook document:

Notebook documents (or “notebooks”, all lower case) 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.

e) Jupyter Notebook App:

The Jupyter Notebook App is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet. In addition to displaying/editing/running notebook documents, the Jupyter Notebook App has a “Dashboard” (Notebook Dashboard), a “control panel” showing local files and allowing to open notebook documents or shutting down their kernels.

f) Kernel:

A notebook kernel is a “computational engine” that executes the code contained in a Notebook document. The ipython kernel referenced in this guide, executes python code. Kernels for many other languages exist (official kernels). When you open a Notebook document, the associated kernel is automatically launched. When the notebook is executed (either cell-by-cell or with menu Cell -> Run All), the kernel performs the computation and produces the results. Depending on the type of computations, the kernel may consume significant CPU and RAM. Note that the RAM is not released until the kernel is shut-down.

g) Notebook Dashboard:

The Notebook Dashboard is the component which is shown first when you launch Jupyter Notebook App. The Notebook Dashboard is mainly used to open notebook documents, and to manage the running kernels (Visualize and shutdown). The Notebook Dashboard has other features similar to a file manager, namely navigating folders and renaming/deleting files.

Working Process:

- Download and install anaconda and get the most useful package for machine learning in Python.
- Load a dataset and understand its structure using statistical summaries and data visualization.
- Machine learning models, pick the best and build confidence that the accuracy is reliable.

Python is a popular and powerful interpreted language. Unlike R, Python is a complete language and platform that you can use for both research and development and developing production systems. There are also a lot of modules and libraries to choose from, providing multiple ways to do each task. It can feel overwhelming.

The best way to get started using Python for machine learning is to complete a project.

- It will force you to install and start the Python interpreter (at the very least).
- It will give you a bird's eye view of how to step through a small project.
- It will give you confidence, maybe to go on to your own small projects.

When you are applying machine learning to your own datasets, you are working on a project. A machine learning project may not be linear, but it has a number of well-known steps:

- Define Problem.
- Prepare Data.
- Evaluate Algorithms.
- Improve Results.

The best way to really come to terms with a new platform or tool is to work through a machine learning project end-to-end and cover the key steps. Namely, from loading data, summarizing data, evaluating algorithms and making some predictions.

Here is an overview of what we are going to cover:

1. Installing the Python anaconda platform.
2. Loading the dataset.
3. Summarizing the dataset.
4. Visualizing the dataset.
5. Evaluating algorithm

3.5 OUTLINE OF THE PROJECT

3.5.1 Overview of the system:

Due to the large volumes of data as well as the complex and dynamic properties of intrusion behaviors, data-mining based Intrusion Detection techniques have been applied to network-based traffic data. With recent advances in computer technology large amounts of data could be collected and stored. Machine Learning techniques can help the integration of computer-based systems in the network environment providing opportunities to facilitate and enhance the work of network security experts. It ultimately improves the efficiency and quality of data and information. Network Intrusion Detection aims at distinguishing the behavior of the network. This paper presents the implementation of four supervised learning algorithms, C4.5 Decision tree Classifier (J48), Instance Based Learning (IBK), Naive Bayes (NB) and Multilayer Perceptron (MLP) in WEKA environment, in an Offline environment. The classification models were trained using the data collected from Knowledge Discovery Databases (KDD) for Intrusion Detection. The trained models were then used for predicting the risk of the attacks in a web server environment or by any network administrator or any Security Experts. The Prediction Accuracy of the Classifiers was evaluated using K-fold Cross Validation and the results have been compared to obtain the accuracy.

It has to find Accuracy of the training dataset, Accuracy of the testing dataset, Specification, False Positive rate, precision and recall by comparing algorithm using python code. The following Involvement steps are,

- Define a problem
- Preparing data
- Evaluating algorithm

The steps involved in Building the data model is depicted below.

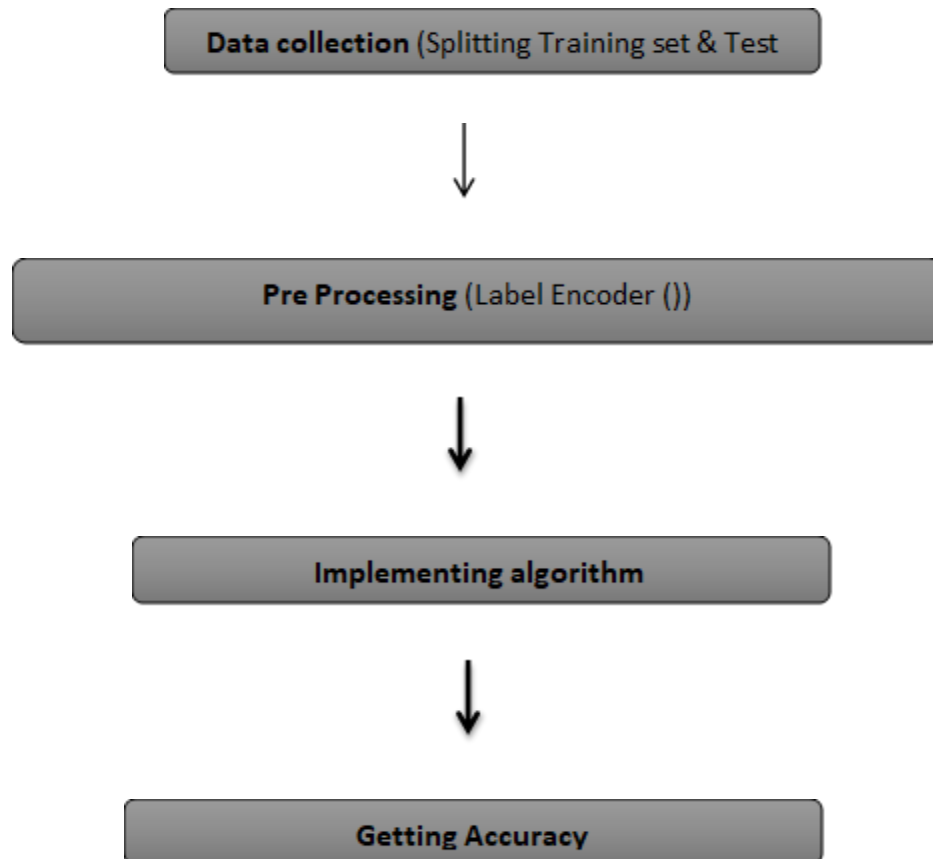


Fig 2: data flow diagram for Machine learning model

Types of Attacks	Description
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 copywritten 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 sweeps 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

3.5.2 Objectives:

This analysis aims to observe which features are most helpful in predicting the network 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.

3.5.3 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

3.5.4 Problem Description:

Lately, an internet network company in Japan has been facing huge losses due to malicious server attacks. They've encountered breach in data security, reduced data transfer speed and intermittent breakdowns in user-user & user-network connections. When asked, a company official said, "there's a significant dip in the number of active users on our network".

The company is looking for some predictive analytics solutions to help them understand, detect and counter the attacks and make their network connection secure. Think of a connection as a sequence of TCP packets starting and ending at some well-defined times, between which data flows to and from a source IP address to a target IP address under some well-defined protocol. In total, there are 3 major types of attacks to which their network is vulnerable to. But, 3 of them cause the maximum damage. In this challenge, you are given an anonymized sample dataset of server connections. You have to predict the type of attack.

3.5.5 Scope:

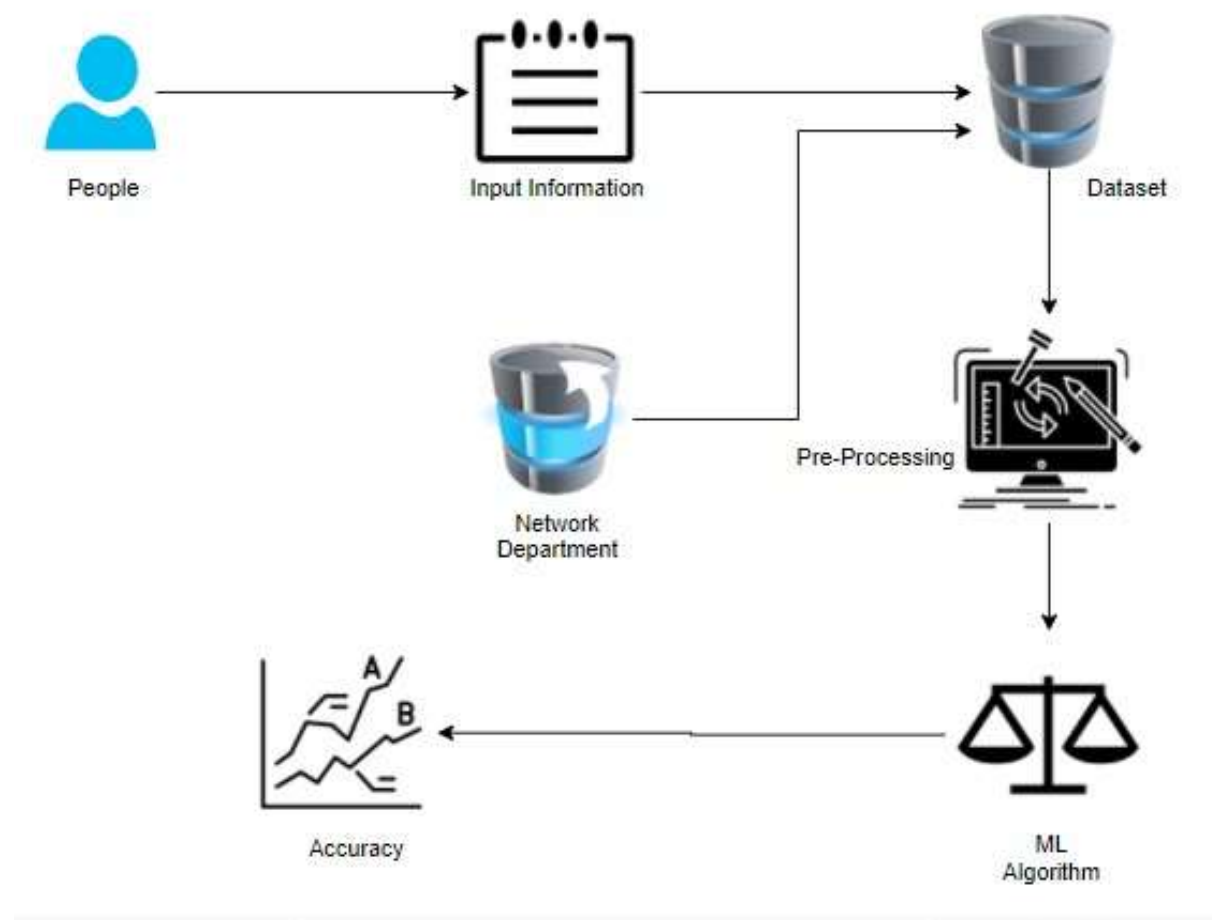
The scope of this project is to investigate a dataset of network connection attacks for KDD records for medical sector using machine learning technique. To identify if a network connection is attacked or not.

3.6 DESIGN ENGINEERING

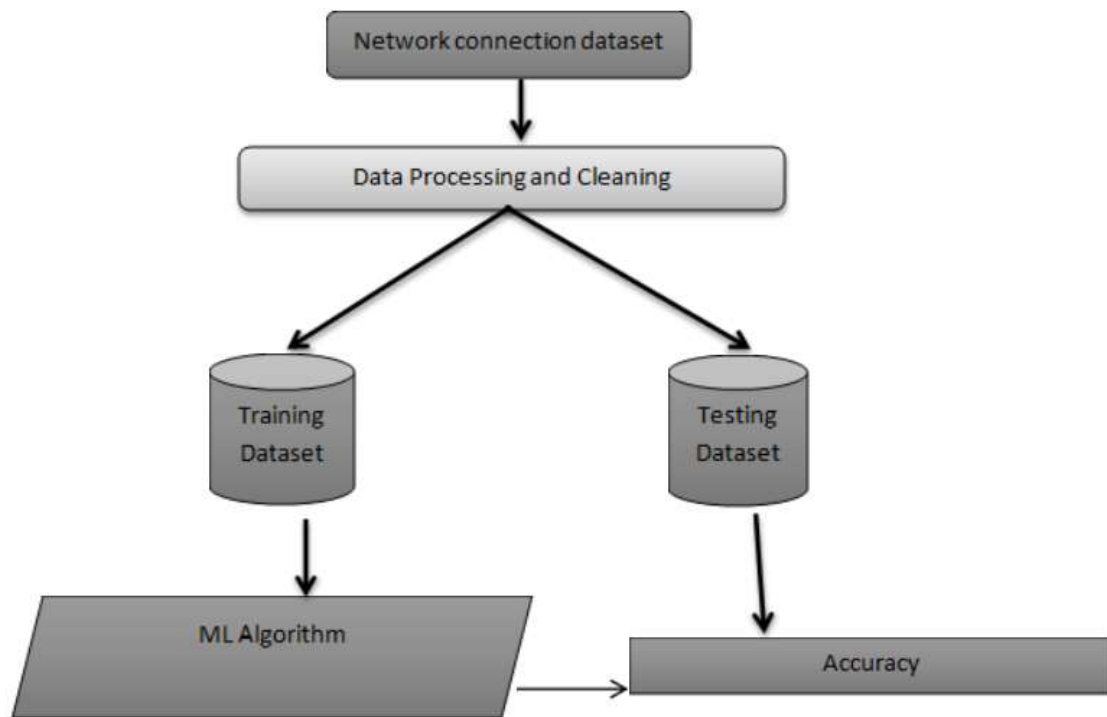
a) General

Design is a meaningful engineering representation of something that is to be built. Software design is a process design is the perfect way to accurately translate requirements into a finished software product. Design creates a representation or model, provides detail about software data structure, architecture, interfaces and components that are necessary to implement a system.

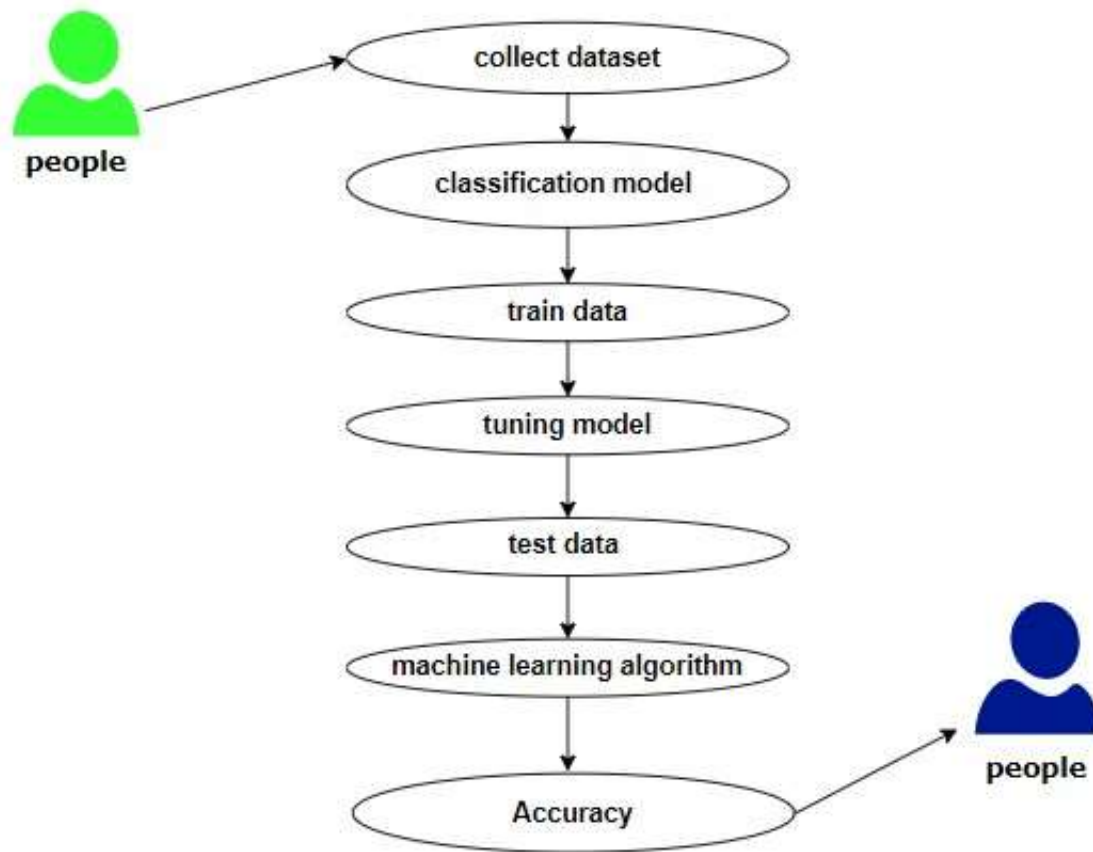
3.6.1 System Architecture



3.6.2 Data flow diagram:

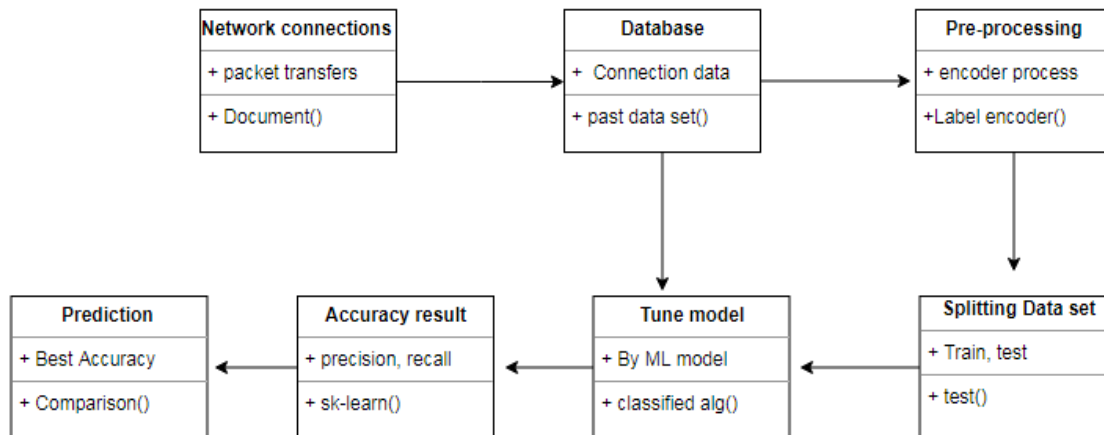


3.6.3 Use Case Diagram:



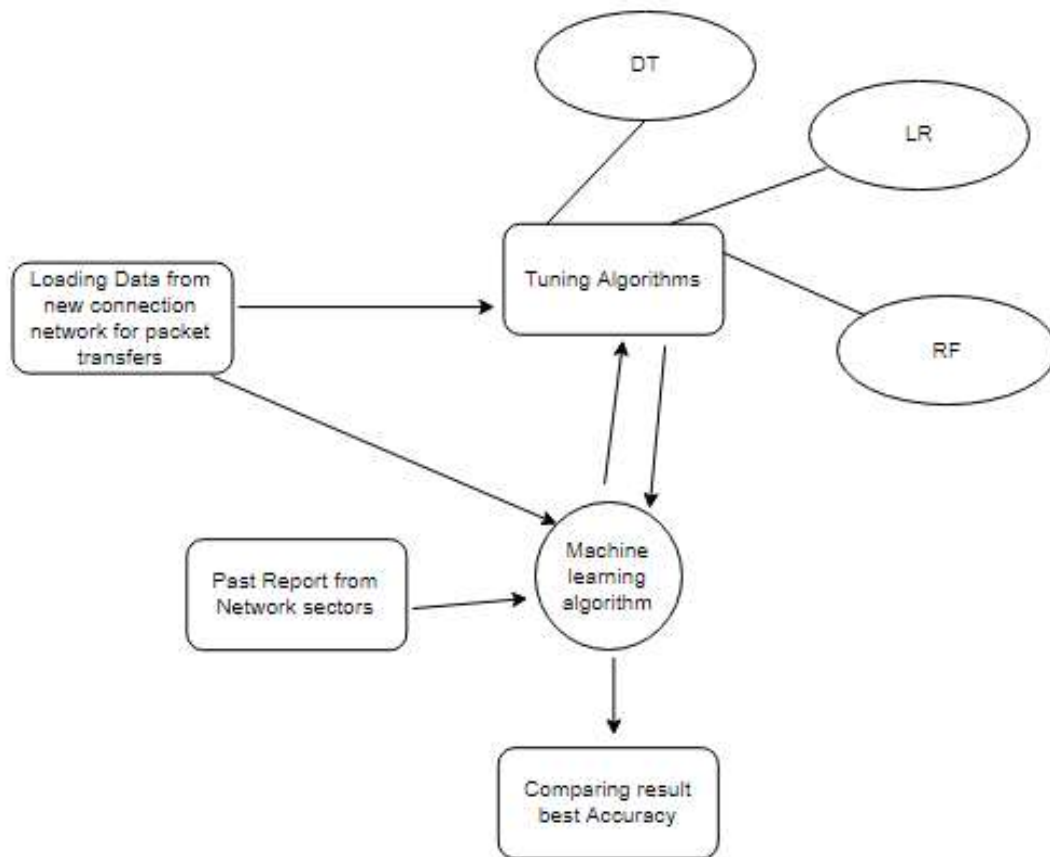
Use case diagrams are considered for high level requirement analysis of a system. So, when the requirements of a system are analyzed the functionalities are captured in use cases. So, it can say that uses cases are nothing but the system functionalities written in an organized manner. Now the second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

3.6.4 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.

3.6.5 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.

Advantages:

- It improves accuracy score by comparing popular machine learning algorithm.
- These reports are to the investigation of applicability of machine learning techniques for diabetes stages forecasting in operational conditions by attribute prediction.
- Finally, it highlights some observations on future research issues, challenges, and needs.

3.7 Modules:

- Data Pre-Processing.
- Data visualization.
- Implementing Algorithm in the dataset.

3.7.1 Module-01:**Variable Identification Process / data validation process:**

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 3 . Given 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.

3.7.2 Module-02:

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.

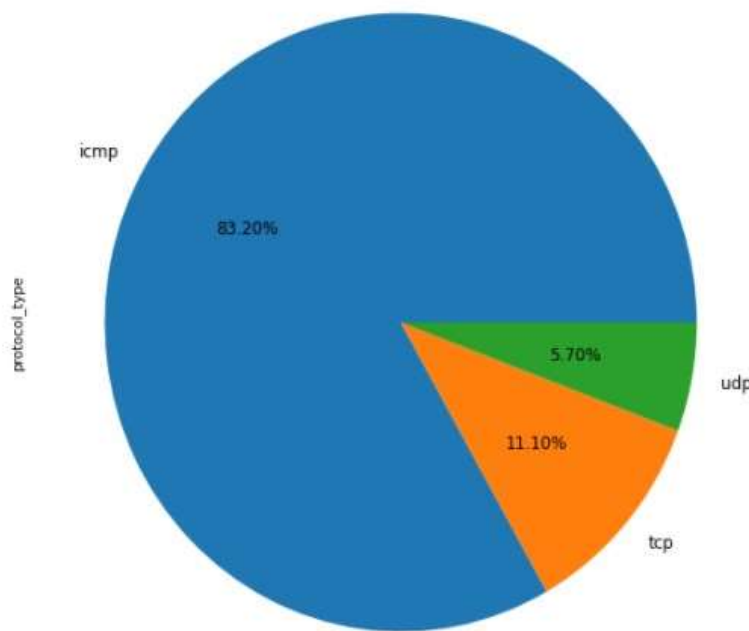


Fig 4. Percentage level of protocol type

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.

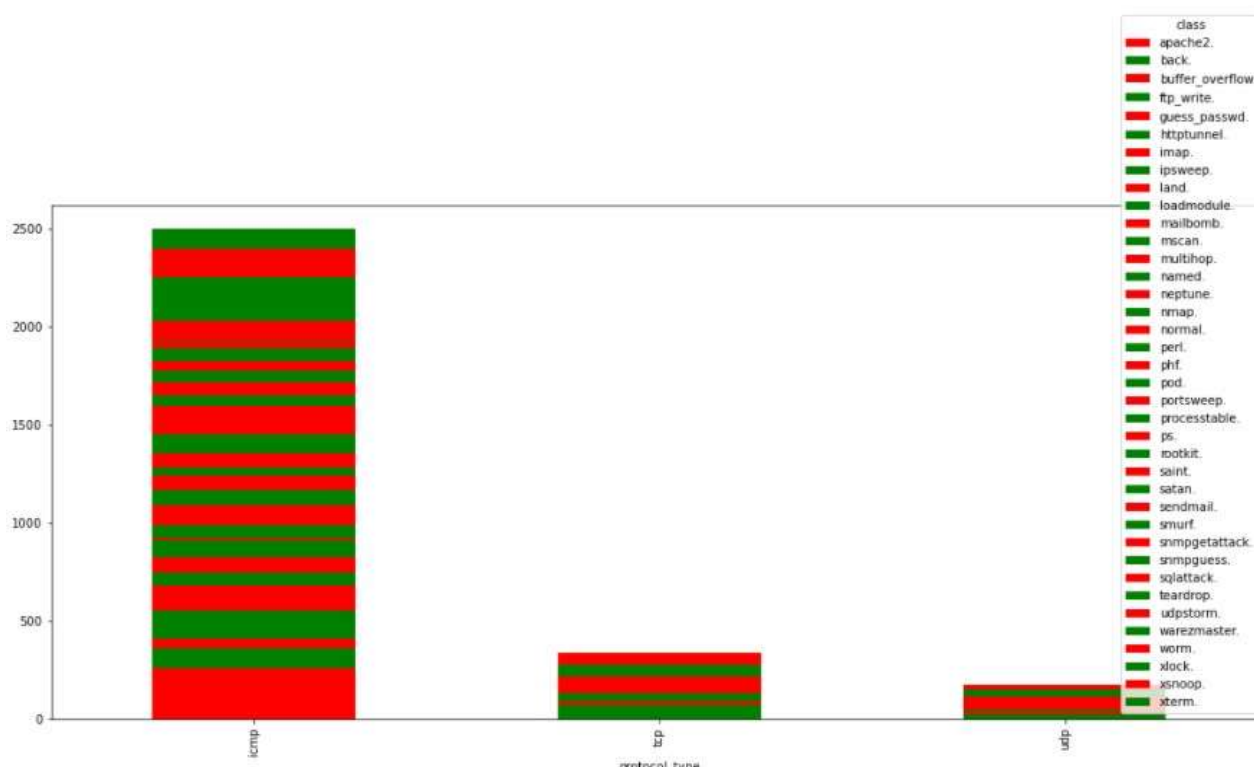


Fig 5. Comparison of service type and protocol type

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.

3.7.3 Module-03:

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.

3.8 Python Packages used:

sklearn:

- In python, sklearn is a machine learning package which include a lot of ML algorithms.
- Here, we are using some of its modules like `train_test_split`, `DecisionTreeClassifier` or `Logistic Regression` and `accuracy_score`.

NumPy:

- It is a numeric python module which provides fast maths functions for calculations.
- It is used to read data in numpy arrays and for manipulation purpose.

Pandas:

- Used to read and write different files.
- Data manipulation can be done easily with data frames.

Matplotlib:

- Data visualization is a useful way to help with identify the patterns from given dataset.
- Data manipulation can be done easily with data frames.

CHAPTER 4

4. RESULT AND DISCUSSION

4.1. Software involvement steps:

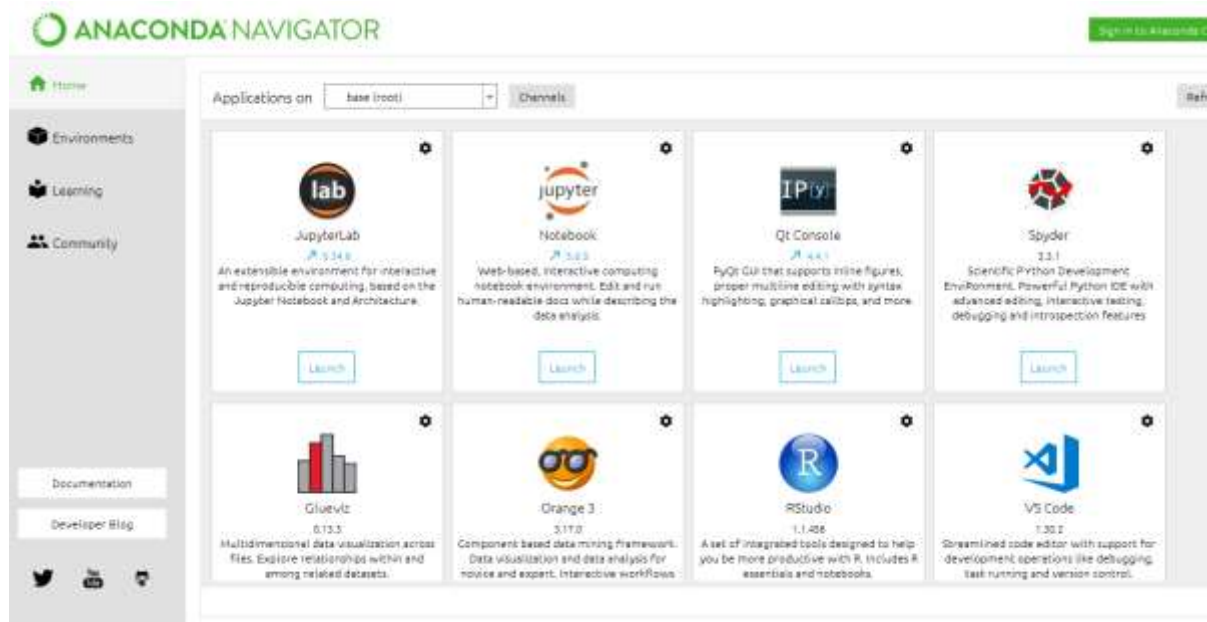


Fig 6: Open the anaconda navigator

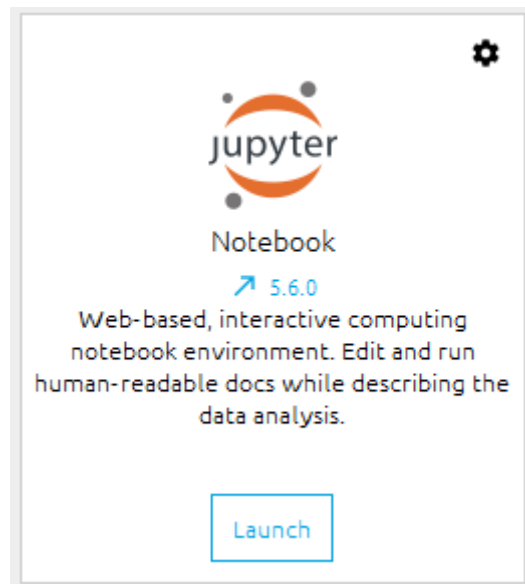


Fig 7: Launch the jupyter notebook platform



Fig 8: Open the correspondent result folder

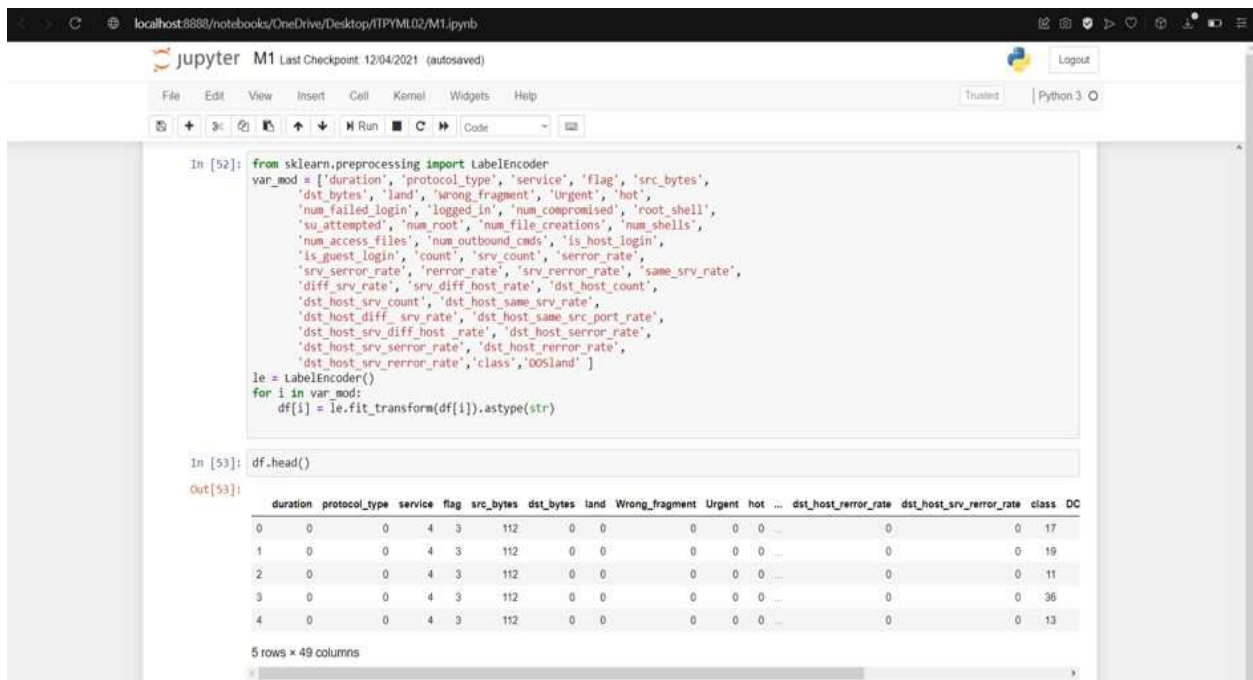


Fig 9: Module 1 resulting code

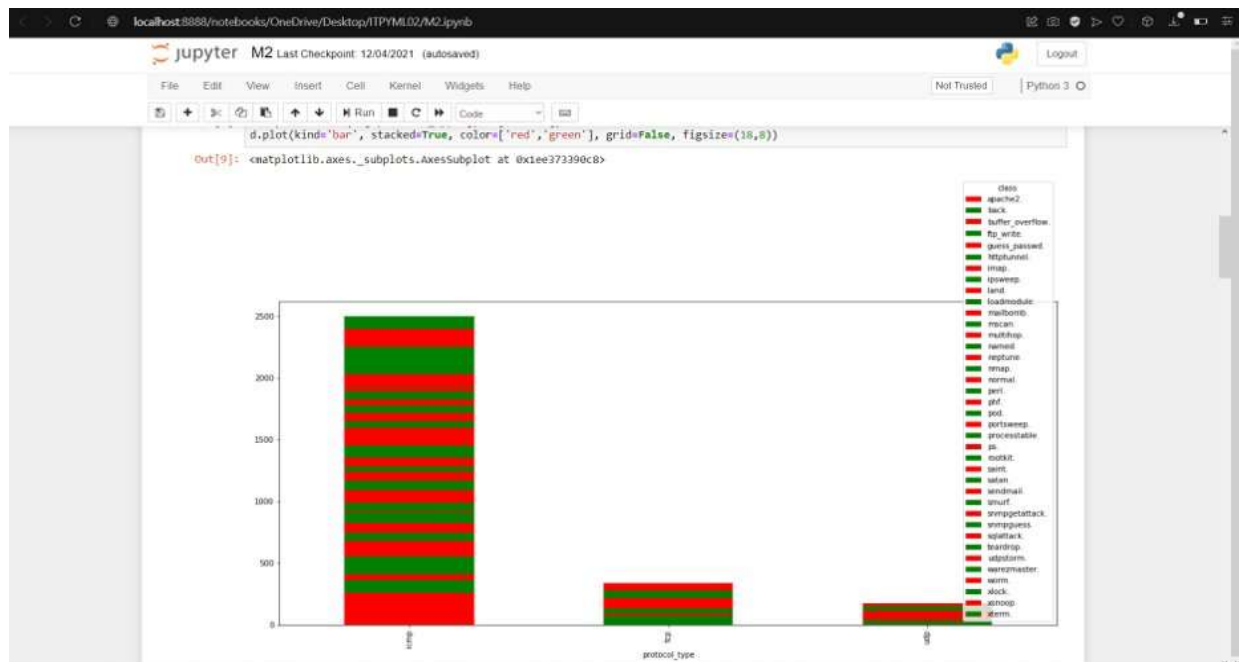


Fig 10: Module 2 graph result

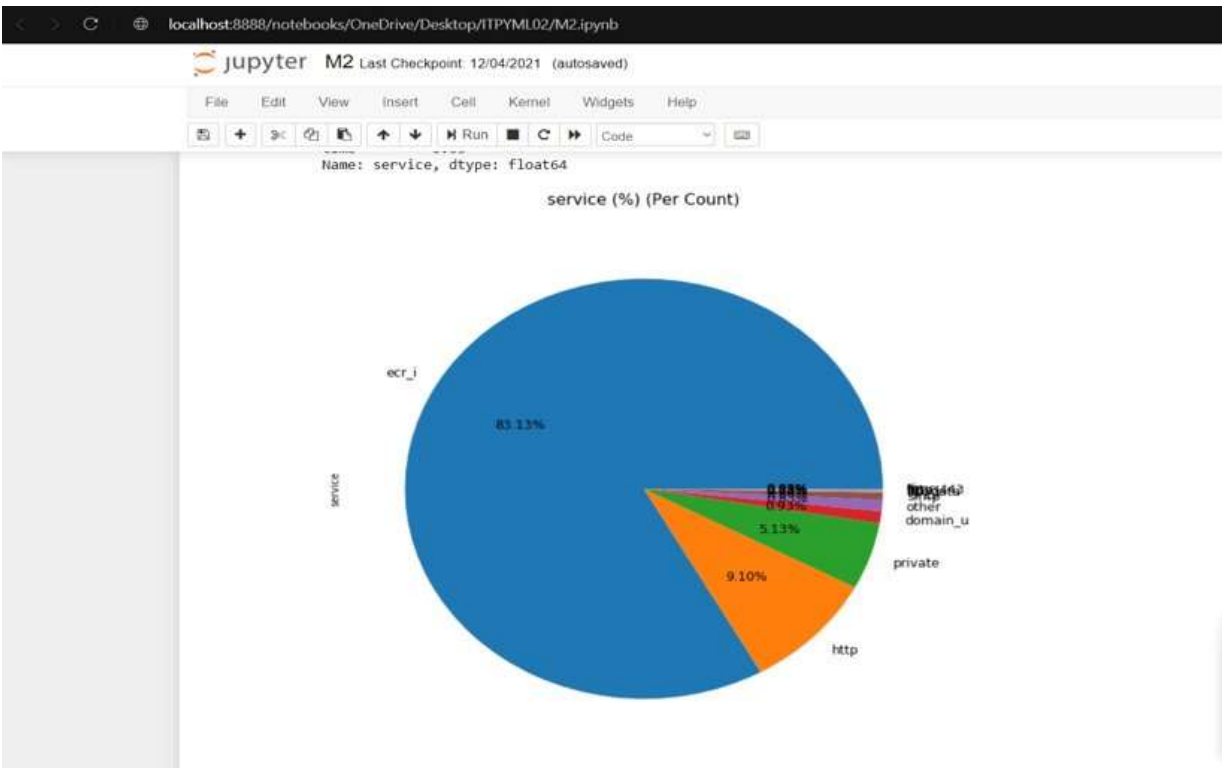


Fig 11: Module 2 pie chart result

```
print("")
accuracy = cross_val_score(logit, 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

Classification report of Logistic Regression Results:
      precision    recall  f1-score   support

     0       0.00      0.00      0.00         4
     1       1.00      1.00      1.00      896

 accuracy: 1.00      900
macro avg: 0.50      0.50      0.50      900
weighted avg: 0.99      1.00      0.99      900

Confusion Matrix result of Logistic Regression is:
[[ 0  4]
 [ 0 896]]

Sensitivity : 0.0
Specificity : 1.0

Cross validation test results of accuracy:
[0.996 0.996 0.994]

Accuracy result of Logistic Regression is: 99.53333333333333
```

Fig 12: Module 3 resulting code

CHAPTER 5

5.1 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 is will be found out by algorithm with 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.

5.2 FUTURE WORK:

- Network sector want to automate the detecting the attacks of packet transfers from eligibility process (real time) based on the connection detail.
- Comparing the data with some more algorithms.
- 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.

5.3 REFERENCES:

- [1] Swetha Arra, K. Rekha Devi “Evaluation, prediction and implementation patterns of network traffic malware using machine learning” 2021
- [2] Preetish Ranjan, Abhishek Vaish “Apriori Viterbi Model for Prior Detection of Socio-Technical Attacks in a Social Network” 2014
- [3] Seraj Fayyad, Cristoph Meinel “New Attack Scenario Prediction Methodology” 2013

[4] Jinyu W1, Lihua Yin and Yunchuan Guo “Cyber Attacks Prediction Model Based on Bayesian Network” 2012

[5] Xiaoyong Yuan , Pan He, Qile Zhu, and Xiaolin Li “Adversarial Examples: Attacks and Defenses for Deep Learning” 2019

[6] Wenying Xu , Guoqiang Hu “Distributed Secure Cooperative Control Under Denial-of-Service Attacks From Multiple Adversaries” 2019

[7] Wentao Zhao, Jianping Yin and Jun Long “A Prediction Model of DoS Attack’s Distribution Discrete Probability” 2008

[8] Timothy Chadza, Konstantinos G. Kriyakopoulos, Sangarapillai Lambotharan “Analysis of hidden Markov model learning algorithms for the detection and prediction of multi-stage network attacks” 2020

[9] Neamat Al Sarah, Fahmida Yasmin Rifat, Hasnu S. Narman “An Efficient Android Malware Prediction Using Ensemble machine learning algorithms” 2021

[10] Onkar Thorat, Nirali Parekh, Ramachandra Mangrulkar “TaxoDaCML: Taxonomy based Divide and Conquer using machine learning approach for DDoS attack classification” 2021

[11] Raj Kumar Batchu, Hari Seetha “A generalized machine learning model for DDoS attacks detection using hybrid feature selection and hyperparameter tuning” 2021

5.4 Appendices:

5.4.1 SAMPLE CODE:

A) MODULE-1

```
#import library packages
```

```
import pandas as p
import numpy as n
```

In []:

```
# 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"]
```

In []:

```
data = p.read_csv("data6.csv", names = features)
```

Before drop the given dataset

In []:

```
data.head(10)
```

After drop the given dataset

In []:

```
df=data.dropna()
```

In []:

```
df.head(10)
```

In []:

```
#show columns
```

```
df.columns
```

```
In [ ]:
```

```
#To describe the data frame
```

```
df.describe()
```

```
In [ ]:
```

```
#Checking datatype and information about dataset
```

```
df.info()
```

```
Checking duplicate values of data frame
```

```
In [ ]:
```

```
#Checking for duplicate data
```

```
df.duplicated()
```

```
In [ ]:
```

```
#find sum of duplicate data
```

```
sum(df.duplicated())
```

```
In [ ]:
```

```
#Checking sum of missing values
```

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

```
In [ ]:
```

```
df["class"].unique()
```

```
In [ ]:
```

```
df['land'].value_counts()
```

```
In [ ]:
```

```
df['service'].value_counts()
```

```
In [ ]:
```

```
df['protocol_type'].value_counts()
```

In []:

```
df['DOSland'] = df.land.map({0:'attack',1:'noattack',2:'normal'})
```

In []:

```
df['DOSlandclass'] = df.DOSland.map({'attack':1,'noattack':0,'normal':0})
```

In []:

```
df['DOSlandclass'].value_counts()
```

In []:

```
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})
```

In []:

```
df.head()
```

In []:

```
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})
```

In []:

```
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 []:

```
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})
```

In []:

```
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})
```

In []:

```
df.head()
```

In []:

```
df.corr()
```

Before Pre-Processing:

In []:

```
df.head()
```

After Pre-Processing:

In []:

```
df.columns
```

In []:

```
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', 'error_rate',
           'srv_error_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_error_rate',
           'dst_host_srv_error_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)
```

In []:

```
df.head()
```

In []:

B) MODULE-2

```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

In []:

```
# 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"]
```

In []:

```
data = pd.read_csv("data6.csv", names = features)
```

In []:

```
data.head(10)
```

In []:

```
df=data.dropna()
```

In []:

```
#Checking for duplicate data
```

```
df.duplicated().sum()
```

In []:

```
#Checking sum of missing values
```

```
a = df.isnull().sum()
```

In []:

```
d = pd.crosstab(df['protocol_type'], df['class'])
```

```
d.plot(kind='bar', stacked=True, color=['red','green'], grid=False, figsize=(18,8))
```

In []:

```
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()
```

In []:

```
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')
```

In []:

```
# Heatmap plot diagram
```

```
fig, ax = plt.subplots(figsize=(14,10))
```

```
sns.heatmap(data.corr(), ax=ax, annot=True)
```

In []:

```
#Histogram Plot of Applicant income
```

```
df["class"].hist(figsize=(10,8), color="red")
```

```
plt.title("class")
```

```
plt.xlabel("Output")
```

```
plt.show()
```

In []:

```
df
```

In []:

```
#Histogram Plot of Applicant income
```

```
df["dst_host_srv_count"].hist(figsize=(10,8), color="red")
```

```
plt.title("dst_host_srv_count")
```

```
plt.show()
```

C) MODULE-3

```
import pandas as p
```

In []:

```
import warnings
```

```
warnings.filterwarnings('ignore')
```

In []:

```
# 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_rerror_rate", "dst_host_srv_rerror_rate", "class"]  
data = p.read_csv("data6.csv", names = features)
```

In []:

```
df=data.dropna()
```

In []:

```
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})
```

In []:

```
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', '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'
          ]

```

```
le = LabelEncoder()
```

```
for i in var_mod:
```

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

In []:

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

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

In []:

#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

```

In []:

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

```
#Response variable
```

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

In []:

#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)
```

In []:

#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 :

In []:

```
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
```

In []:

```
def graph():
    import matplotlib.pyplot as plt
    data=[lr]
    alg="LR"
    plt.figure(figsize=(10,5))
    b=plt.bar(alg,data,color=("r"))
    plt.title("Accuracy of Overall Attacks",fontsize=15)
    plt.legend(b,data,fontsize=9)
    plt.savefig('overallattack.png')
```

In []:

```
graph()
```

In []:

```
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()
```

In []:

5.5 Publication:

PREDICTION OF NETWORK ATTACKS USING SUPERVISED MACHINE LEARNING ALGORITHM

ABSTRACT:

Intrusion Detection System (IDS) needs a data and for this it is important to keep the real working environment to find out all the possibilities of how an attack is about to happen and this seem to be expensive. A Software to detect network attacks in a computer network from unidentified users, including known personnel. The attack detector's learning task works up a predictive model which is a classifier in this case which differentiates the "bad" (i.e., intrusions or attacks) and "good" or "normal" connections. The primary aim is to use machine learning based techniques to provide packet connection transfer in a better way by predicting results with the at most accuracy. Comparing and discussing the outputs from the couple of machine learning algorithms used for the given dataset with evaluated classification report, find the confusion matrix and categorize the data from priority and the result which shows that the efficiency of the claimed machine learning algorithm method is to be compared with the best accuracy techniques such as Precision, Recall and F1 Score.

1. INTRODUCTION:

Domain overview:

Machine learning is used here on past data to predict the future. It acts like an artificial intelligence which gives computers the talent to learn without being programmed accordingly. Machine learning targets the growth of Computer Programs that can adapt when it is exposed to new data and a simple machine learning algorithm using python will be in effect. Training data and prediction of attacks here involves the use of some specialized algorithms.

The algorithm learns from the training data to give accurate predictions on a new test data. Machine learning is set into three categories. There are supervised, unsupervised and reinforcement learning. Supervised learning program is delivered both to the input data and to the learning data which is separated from the dataset by the user prior to the attack prediction. Machine learning algorithms used:

1. Logistic regression
2. Multi-Class classification
3. Decision Trees



Supervised Machine Learning is the most practical machine learning out there which is suitable to use in most conditions. This type of machine learning is where input variable (X) and output variable (Y) and the algorithm works on the mapping function from the input to the output which generates to $Y = f(X)$. The goal is to map accurately so that when a new input data (X) is introduced it predicts the output (Y) for that respective data.

The goal of this problems is to construct a succinct model which predicts the value of the dependent attribute from a set of variables. The fact that the dependent attribute is numerical for categorical for classification is the main difference between the two tasks. A classification model attempts to draw Conclusions from observed values are drawn as attempts from a classification model. The number of inputs does not matter as the output for that input is easily predictable.

2. DATASET USED:

Most computers use KDD Cup99 as the dataset for network intrusions as the classification analysis of network traffic and it explains the formation of this dataset and its features. The same dataset is used here. The data is gathered from MIT Lincoln Laboratory which worked on the evaluation of Intrusion Detection System under the sponsorship of Defence Advanced Research Projects Agency. This data set consists of 41 features, some of which are created using domain knowledge, whilst other are intrinsic to the network connections.

3. DATASET PREPARATION:

The model is trained with the dataset which is supplied to machine learning model. The first step of collecting information, data is gathered from online sources i.e., previously Attacks infected patients. Analysis is done on a common dataset which is formed by combining these datasets. There are 41 features in this KDD Cup99 dataset.

The features from 1-9 are the basic features of a packet, 10-22 consist of content features, 23-31 consist of traffic features which has a two second time window and 32-41 entirely consist of host-based features.

The terms associated with this data set are:

1. The connections in the past two seconds is referred to as “same host” and they have the same connection as the destination host.
2. The connections in the past two seconds is referred to as “same service” and the current connections have the same service.
3. “Same host” and “same service” features are together called as time-based traffic features of the connection records.

4. LITERATURE SURVEY:

[1] Wi-Fi users are very much exposed and vulnerable to malware. Data can capture from the users who connect to public wireless hotspots. Machine learning algorithms need work and to discover accurately the existence of malware in the wireless environment. The key focus is to predict random network traffic so attacks can be predicted with better level of accuracy.

[2] The prediction of socio-technical attack is considered to be difficult always. It explores different algorithms to find out the kinds of community based on specific pattern, structure, trend and concepts which explore different opportunities. Hidden information is found within enormous social network by compiling it into small networks through Apriori algorithm and the end result using Viterbi algorithm which uses pattern of conversation.

[3] The occurrence of harmful activities against IT systems is detected by Intrusion detection systems (IDS). Malicious activities in IT systems are detected through constant monitoring and analysing the systems. An alert is generated for every harmful activity detected by the IDS and it is stored in the database.

[4] The future of the target network's security situation reflects the prediction result and according to the results necessary measures need to be taken by the security administrators. Attack probability plays a vital role to predict the possibility of an attack quantitatively. Invasion by intruders is detected by the indication of a possibility.

[5] By applying for DNNs the only main risk is the vulnerability to adversarial examples in safety important environments. Recently revealed on adversarial examples for DNNs puts in a nutshell the methods to generate adversarial examples as attacks and defences find out all the possibilities.

[6] Through DoS attacks secure control of multilevel agent systems are distributed. The investigation is focused on the aftermath of distributed DoS attacks from many adversaries. Event- triggered and sample – data communication schemes are discussed and later on a proper working control protocol is developed to provide an assured asymptotic consensus.

[7] The relationship between the amount of DoS attacks and network traffic data is shown, and a clustering method is proposed regarding the genetic optimization algorithm which is used to classify DoS attack data. The prediction of sub – models of DoS attack and based on the optimized clustering the number of DoS attacks is found when the proper relation between the latter and former is found.

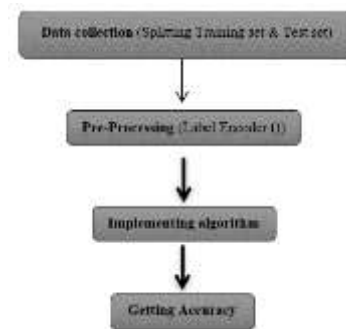
[8] An alert sequence is initiated to evaluate the performance of the algorithms both in a hybrid and individual approach for detecting, predicting the next state (NS) and also to predict the next observation (NO). To generate this alert sequence an intrusion detection system is utilised called Snot signature.

5. SYSTEM OVERVIEW:

Using python code to compare algorithm the recall, precision, Accuracy of the testing dataset, False Positive rate and Specification are found. The steps involved in Building the data model is depicted below:

The Involvement steps are;

1. Define a problem
2. Preparing data
3. Evaluating algorithm

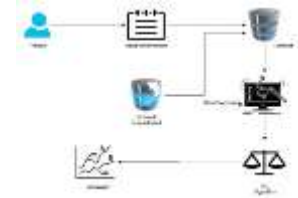


In a network-based environment the integration of computer-based system by enhancing the work of network security experts and providing opportunities to facilitate is done by Machine Learning techniques

6. ARCHITECTURE DIAGRAM:

Patterns are extracted and the network connection dataset is given by using Supervised classification algorithms. This predicts if the network is affected or not and later it trains the algorithm to predict accurately so the network can avoid attacks in the future. A common dataset is formed from multiple datasets which is used by various algorithms to provide results with a high – tier accuracy rate.

Some missing values might be present in the dataset which may lead to inconsistency; hence it is pre-processed which will provide with a more efficient data which returns the algorithm to be efficient. The time consuming of data mining phase is data pre-processing. Unsignificant data which has no part in the behaviour of a packet transfer is removed during the data cleaning phase.



7. RESULTS AND DISCUSSION:

Module – 1:

Data validation Process:

Under this process, data is identified, this helps to identify the data and its properties. The knowledge adopted from this will help in choosing which algorithm should be used to build a model. The data frame is given below.

duration	protocol_type	service	flag	src_bytes	dst_bytes	land	Wrong_fragment	Urgent	hot	...	dst_host_src_count	dst_host_name_src_rate
0	0	udp	private	SP	100	140	0	0	0	0	204	1.00
1	0	udp	private	SP	100	140	0	0	0	0	204	1.00
2	0	udp	private	SP	100	140	0	0	0	0	204	1.00
3	0	udp	private	SP	100	140	0	0	0	0	204	1.00
4	0	udp	private	SP	100	140	0	0	0	0	204	1.00
5	0	udp	private	SP	100	140	0	0	0	0	204	1.00
6	0	udp	private	SP	100	140	0	0	0	0	204	1.00
7	0	udp	private	SP	100	140	0	0	0	0	204	1.00
8	0	udp	private	SP	100	140	0	0	0	0	204	1.00
9	0	tcp	help	SP	223	100	0	0	0	0	204	1.00

Library packages are imported by loading the given dataset. Data type, data shape and the evaluation of missing values analyses the identification of variables. This valid dataset acts as a sample to train the model which estimates the skill of the model.

Visualization of Data:

Figure 1 consists of two panels. The left panel is a pie chart showing the distribution of the 100 most abundant proteins in the proteome. The chart is divided into three segments: a large blue segment labeled 'emp' representing 92.20%, an orange segment labeled 'gcp' representing 11.10%, and a green segment labeled 'udp' representing 6.70%. The right panel is a stacked bar chart showing the distribution of the 100 most abundant proteins across three categories: 1, 2, and 3. The y-axis represents the count of proteins, ranging from 0 to 3000. The x-axis shows the categories 1, 2, and 3. The legend on the right lists 100 proteins, including emp, udp, gcp, and many others. The bars are stacked with red and green segments, indicating the distribution of proteins across the three categories.

Module – 3:

Logistic Regression:

The machine learning algorithm used here is Logistic regression which predicts the probability of a categorical dependent variable. Binary characters used here as dependent variable which contains data that is coded as 1 (yes) or 0 (no). The statistical method to analyze data used here has one or more independent variables which determines an output.

```
Classification report of Logistic Regression Results:
              precision    recall  f1-score   support

     0       0.00        0.00        0.00         4
     1       1.00        1.00        1.00      896

 accuracy          0.50
 macro avg         0.50        0.50        0.50      900
weighted avg         0.99        1.00        0.99      900

Confusion Matrix result of Logistic Regression is:
[[ 0  4]
 [ 0 896]]

Sensitivity : 0.0
Specificity : 1.0

Cross validation test results of accuracy:
[0.99666667 0.99666667 0.99666667 0.995      0.99333333]

Accuracy result of Logistic Regression is: 99.56666666666666
```

CONCLUSION:

The process starts from data cleaning and pre-processing, which eliminates the missing value, model building, evaluation and exploratory analysis. The algorithm predicts the attacks with a high accuracy rate by finding the best connections. A prediction model with the help of AI increases the scope of early detection of attacks. It is understandable from the model that the use of machine learning technique and area analysis helps network sections decreases the diagnosis time and eliminates any human error.

REFERENCES:

- [1] Swetha Arra, K. Rekha Devi “Evaluation, prediction and implementation patterns of network traffic malware using machine learning” 2021
- [2] Preetish Ranjan, Abhishek Vaish “Apriori Viterbi Model for Prior Detection of Socio-Technical Attacks in a Social Network” 2014
- [3] Seraj Fayyad, Cristoph Meinel “New Attack Scenario Prediction Methodology” 2013

- [4] Jinyu W1, Lihua Yin and Yunchuan Guo “Cyber Attacks Prediction Model Based on Bayesian Network” 2012
- [5] Xiaoyong Yuan , Pan He, Qile Zhu, and Xiaolin Li “Adversarial Examples: Attacks and Defenses for Deep Learning” 2019
- [6] Wenying Xu , Guoqiang Hu “Distributed Secure Cooperative Control Under Denial-of-Service Attacks From Multiple Adversaries” 2019
- [7] Wentao Zhao, Jianping Yin and Jun Long “A Prediction Model of DoS Attack’s Distribution Discrete Probability” 2008
- [8] Timothy Chadza, Konstantinos G. Kriyakopoulos, Sangarapillai Lambotharan “Analysis of hidden Markov model learning algorithms for the detection and prediction of multi-stage network attacks” 2020
- [9] Neamat Al Sarah, Fahmida Yasmin Rifat, Hasnu S. Narman “An Efficient Android Malware Prediction Using Ensemble machine learning algorithms” 2021
- [10] Onkar Thorat, Nirali Parekh, Ramachandra Mangrulkar “TaxoDaCML: Taxonomy based Divide and Conquer using machine learning approach for DDoS attack classification” 2021
- [11] Raj Kumar Batchu, Hari Seetha “A generalized machine learning model for DDoS attacks detection using hybrid feature selection and hyperparameter tuning” 2021