**Prediction of the network attacks by finding the best accuracy using supervised machine learning algorithm**

**Abstract:**

Generally, 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 DOS, R2L, U2R, Probe and 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.

**Keywords:** dataset, Machine learning-Classification method, python, Prediction of Accuracy result.

**INTRODUCTION**

**Domain overview**

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.

Analyses Predicts

Machine Learning

Result

Past Dataset

Trains

Fig: Process of Machine learning

[Supervised Machine Learning](https://www.geeksforgeeks.org/supervised-unsupervised-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 dumpdata, 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.

**Derived Features**

Stolfo et al. defined higher-level features that help in distinguishing normal connections from attacks. There are several categories of derived features. The same host‖ features examine only the connections in the past two seconds that have the same destination host as the current connection, and calculate statistics related to protocol behavior, service, etc. The similar ``same service'' features examine only the connections in the past two seconds that have the same service as the current connection. "Same host" and "same service" features are together called time-based traffic features of the connection records. 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 135 destination host, and features were constructed using a window of 100 connections to the same host instead of a time window. This yields a set of so-called host-based traffic features. Unlike most of the DOS and probing attacks, there appear to be no sequential patterns that are frequent in records of R2L and U2R attacks. This is because the DOS and probing attacks involve many connections to some host(s) in a very short period of time, but the R2L and U2R attacks are embedded in the data portions of packets, and normally involve only a single connection. Useful algorithms for mining the unstructured data portions of packets automatically are an open research question. Stolfo et al. used domain knowledge to add features that look for suspicious behavior in the data portions, such as the number of failed login attempts. These features are called ``content'' features.

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.

**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 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.

**LITERATURE SURVEY**

**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 survey’s to understand the past and present perspective of loan approval or not.

**Review of Literature Survey**

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

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

**Year :** 2008

The process of prediction analysis is a process of using some method or technology to explore or stimulate some unknown, undiscovered or complicated intermediate processes based on previous and present states and then speculated the results. In an early warning system, accurate prediction of DoS attacks is the prime aim in the network offence and defense task. Detection based on abnormity 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 kind of prediction may be drawn which will be helpful to take our decisions.

**Title** : New Attack Scenario Prediction Methodology

**Author:** Seraj Fayyad, Cristoph Meinel

**Year :** 2013

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

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

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

**Year :** 2012

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

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

**Author:** Wentao Zhao, Jianping Yin

**Year :** 2008

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

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

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

**Year :** 2019

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

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

**Author:** Wenying Xu , Guoqiang Hu

**Year :** 2019

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

**OUTLINE OF THE PROJECT**

**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 have 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 algorithms
* Improving results
* Predicting results

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

**Data collection**(Splitting Training &Test set)

**Building classification Model**

**Pre Processing**

**Prediction** (Network connection attacks or not)

Fig: data flow diagram for Machine learning model

#### Classification of Attacks:

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

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

Denial of Service:

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

* by targeting the implementation bugs
* by exploiting the misconfiguration of the systems

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

User to Root:

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

Remote to User:

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

Probing:

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

**Summary:**

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

Table: Attack Types Grouped to respective Class

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

Table: Description of Attacks

|  |  |
| --- | --- |
| **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 copy writed  software) onto FTP server |
| Imap | Remote buffer overflow using imap port leads to root shell |
| loadmodule | Non-stealthy loadmodule attack which resets IFS for a normal user and  creates a root shell |
| Perl | Perl attack which sets the user id to root in a perl script and creates a root shell |
| rootkit | Multi-day scenario where a user installs one or more components of a  rootkit |
| ipsweep | Surveillance sweep performing either a port sweep or ping on multiple  host addresses |
| nmap | Network mapping using the nmap tool. Mode of exploring network will  vary-options include SYN |
| satan | Network probing tool which looks for well-known weaknesses. Operates at three different levels. Level 0 is light |
| portsweep | Surveillance sweep through many ports to determine which services are  supported on a single host |
| dict | Guess passwords for a valid user using simple variants of the account  name over a telnet connection |
| eject | Buffer overflow using eject program on Solaris. Leads to a user->root  transition if successful |
| ffb | Buffer overflow using the ffbconfig UNIX system command leads to  root shell |
| format | Buffer overflow using the fdformat UNIX system command leads to  root shell |
| ftp-write | Remote FTP user creates .rhost file in world writable anonymous FTP  directory and obtains local login |
| guest | Try to guess password via telnet for guest account |
| syslog | Denial of service for the syslog service connects to port 514 with  unresolvable source ip |
| warez | User logs into anonymous FTP site and creates a hidden directory |

#### Objectives:

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

The repository is a learning exercise to:

* Apply the fundamental concepts of machine learning from an available dataset and Evaluate and interpret my results and justify my interpretation based on observed dataset.
* Create notebooks that serve as computational records and document my thought process and investigate the network connection whether attacked or not to analyses the data set.

## Evaluate and analyses statistical and visualized results, which find the standard patterns for all regiments.

**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 pairplot, 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

**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 are some predictive analytics solution 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 type of attacks to which their network is vulnerable to. But, 3 of them cause the maximum damage. In this challenge, you are given an anonymised sample dataset of server connections. You have to predict the type of attack(s) like Dos, R2L, U2R, Probe.

**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 identifying network connection is attacked or not.

**EXISTING SYSTEM**

To focuses 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 highlights 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.
* It can’t describe each categorized of DOS attacks like back, Neptune etc. based on the network connections.
* 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.

**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 Random Forest, logistic, Decision tree algorithms, K-Nearest Neighbor (KNN) and Support vector classifier (SVC) are applied on the Training set and based on the test result accuracy, Test set prediction is done.

**Preprocessing :**

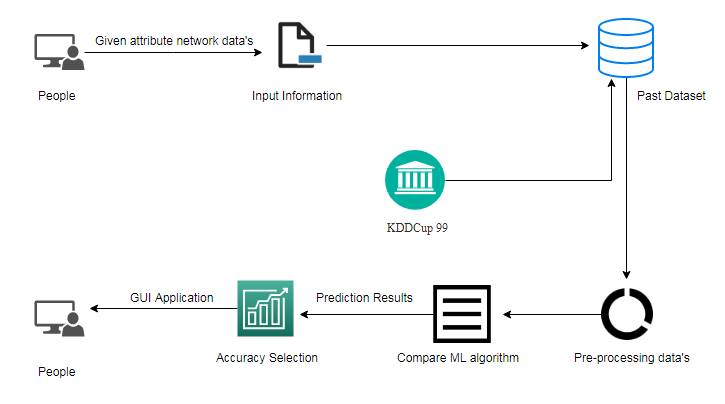
The data which was collected might contain missing values that may lead to inconsistency. To gain better results data need to be preprocessed so as to improve the efficiency of the algorithm. The outliers have to be removed and also variable conversion need to be done. The correlation among attributes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

**DESIGN ENGINEERING**

**General**

Design is meaningful engineering representation of something that is to be built. Software design is a process design is the perfect way to accurately translate requirements in to 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.

**System Architecture**

****

**Work flow diagram**

Network connection data set

Data Processing and Cleaning

Testing Data set

Best Model by Accuracy

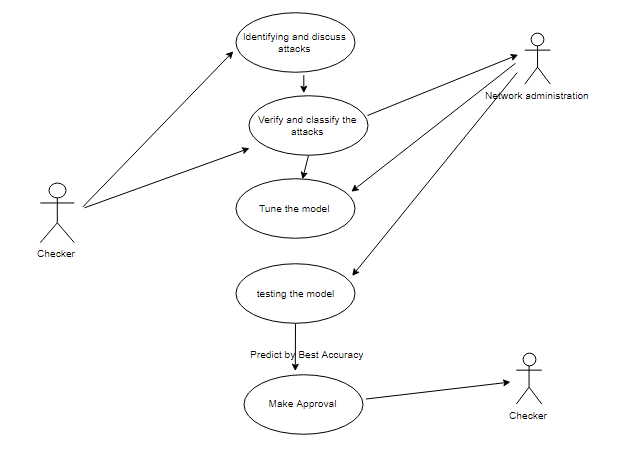
Finding the attack connections

Classification ML Algorithms

Training Data set

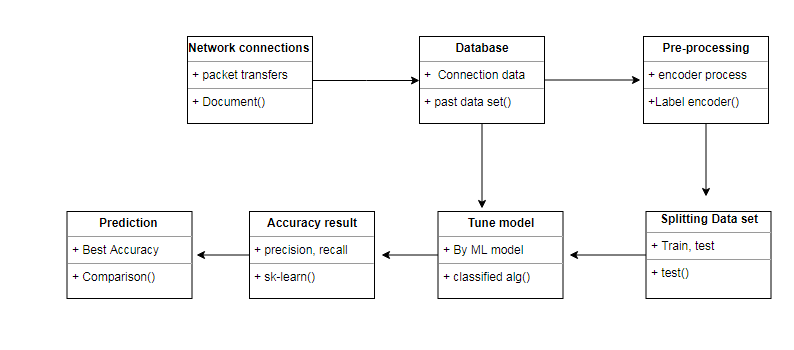
Fig: Workflow Diagram

**Use Case Diagram**



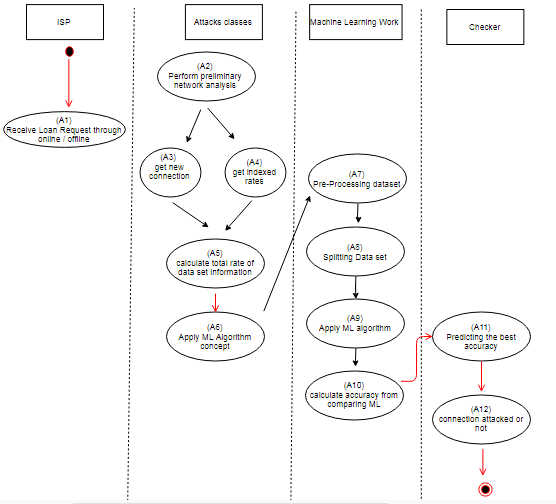
Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases. So, it can say that uses cases are nothing but the system functionalities written in an organized manner. Now the second things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

**Class Diagram**:



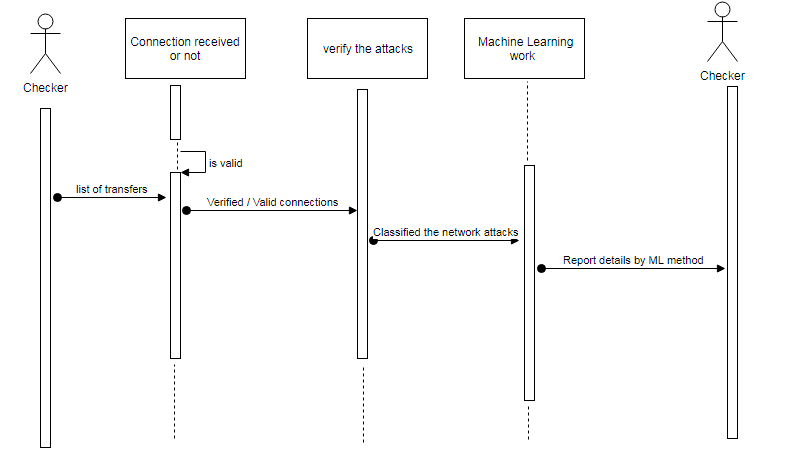
Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element 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.

**Activity Diagram**:



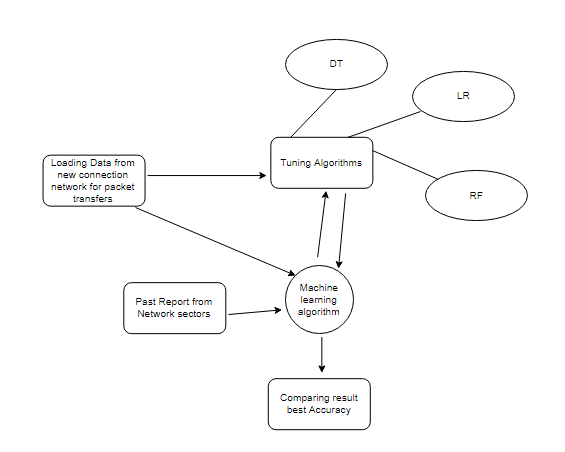
Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part.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

**Sequence Diagram**:



Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modeling, which focuses on identifying the behavior within your system. Other dynamic modeling techniques include [activity diagramming](http://agilemodeling.com/artifacts/activityDiagram.htm), [communication diagramming](http://agilemodeling.com/artifacts/communicationDiagram.htm), [timing diagramming](http://agilemodeling.com/artifacts/timingDiagram.htm), and [interaction overview diagramming](http://agilemodeling.com/artifacts/interactionOverviewDiagram.htm). Sequence diagrams, along with [class diagrams](http://agilemodeling.com/artifacts/classDiagram.htm) and [physical data models](http://agiledata.org/essays/dataModeling101.html) are in my opinion the most important design-level models for modern business application development.

**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](https://searchdatamanagement.techtarget.com/definition/data-modeling) technique that can help define business processes and be used as the foundation for a [relational database](https://searchdatamanagement.techtarget.com/definition/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 algorithms.
* 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.

**Project Requirements**

**General:**

Requirements are the basic constrains 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 define
2. Preparing data
3. Evaluating algorithms
4. Improving results
5. Prediction the result

**Environmental 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

**Software Description:**

Anaconda is a [free and open-source](https://en.wikipedia.org/wiki/Free_and_open-source" \o "Free and open-source) distribution of the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)" \o "Python (programming language)) and [R](https://en.wikipedia.org/wiki/R_(programming_language)" \o "R (programming language)) programming languages for [scientific computing](https://en.wikipedia.org/wiki/Scientific_computing" \o "Scientific computing) ([data science](https://en.wikipedia.org/wiki/Data_science" \o "Data science), [machine learning](https://en.wikipedia.org/wiki/Machine_learning" \o "Machine learning) applications, large-scale data processing, [predictive analytics](https://en.wikipedia.org/wiki/Predictive_analytics" \o "Predictive analytics), etc.), that aims to simplify [package management](https://en.wikipedia.org/wiki/Package_management" \o "Package management) and deployment. Package versions are managed by the [package management system](https://en.wikipedia.org/wiki/Package_manager" \o "Package manager) “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](https://en.wikipedia.org/wiki/Conda_(package_manager)" \o "Conda (package manager)) 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](https://en.wikipedia.org/wiki/Pip_(package_manager)" \o "Pip (package manager)) 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](https://en.wikipedia.org/wiki/Python_Package_Index" \o "Python Package Index) 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.

Anaconda Navigator:

Anaconda Navigator is a desktop [graphical user interface (GUI)](https://en.wikipedia.org/wiki/Graphical_user_interface" \o "Graphical user interface) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using [command-line commands](https://en.wikipedia.org/wiki/Command-line_interface" \o "Command-line interface). 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](https://en.wikipedia.org/wiki/Windows" \o "Windows), [macOS](https://en.wikipedia.org/wiki/MacOS" \o "MacOS) and [Linux](https://en.wikipedia.org/wiki/Linux" \o "Linux).

The following applications are available by default in Navigator:

* [JupyterLab](https://en.wikipedia.org/wiki/Project_Jupyter" \l "Jupyter_Lab" \o "Project Jupyter)
* [Jupyter Notebook](https://en.wikipedia.org/wiki/Project_Jupyter" \l "Jupyter_Notebook" \o "Project Jupyter)
* [QtConsole](https://qtconsole.readthedocs.io/en/latest/)
* [Spyder](https://en.wikipedia.org/wiki/Spyder_(software)" \o "Spyder (software))
* [Glueviz](http://glueviz.org/)
* [Orange](https://en.wikipedia.org/wiki/Orange_(software)" \o "Orange (software))
* [Rstudio](https://en.wikipedia.org/wiki/Rstudio" \o "Rstudio)
* [Visual Studio Code](https://en.wikipedia.org/wiki/Visual_Studio_Code" \o "Visual Studio Code)

### Conda:

Conda is an [open source](https://en.wikipedia.org/wiki/Open-source_software" \o "Open-source software), [cross-platform](https://en.wikipedia.org/wiki/Cross-platform" \o "Cross-platform), language-agnostic [package manager](https://en.wikipedia.org/wiki/Package_manager" \o "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](https://en.wikipedia.org/wiki/R_(programming_language)" \o "R (programming language))), including multi-languages. The Conda package and environment manager is included in all versions of Anaconda, Miniconda, and Anaconda Repository.

### 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.

## [Notebook document](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html" \l "id5):

Notebook documents (or “notebooks”, all lower case) are documents produced by the [Jupyter Notebook App](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html" \l "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.

## [Jupyter Notebook App](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html" \l "id6):

The Jupyter Notebook App is a server-client application that allows editing and running [notebook documents](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html" \l "notebook-document) 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](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html" \l "dashboard)), a “control panel” showing local files and allowing to open notebook documents or shutting down their [kernels](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html" \l "kernel).

## [Kernel](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html" \l "id7):

A notebook kernel is a “computational engine” that executes the code contained in a [Notebook document](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html" \l "notebook-document). The ipython kernel, referenced in this guide, executes python code. Kernels for many other languages exist ([official kernels](http://jupyter.readthedocs.org/en/latest/" \l "kernels)). When you open a [Notebook document](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html" \l "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.

## [Notebook Dashboard](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html" \l "id8):

The Notebook Dashboard is the component which is shown first when you launch [Jupyter Notebook App](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html" \l "notebook-app). The Notebook Dashboard is mainly used to open [notebook documents](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html" \l "notebook-document), and to manage the running [kernels](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html" \l "kernel) (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.
* Present 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 some algorithms.
6. Making some predictions.

**Modules:**

* Data validation process by each attack (Module-01)
* Performance measurements of DoS attacks (Module-02)
* Performance measurements of R2L attacks (Module-03)
* Performance measurements of U2R attacks (Module-04)
* Performance measurements of Probe attacks (Module-05)
* Performance measurements of overall network attacks (Module-06)
* GUI based prediction results of Network attacks (Module-07)

**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)



Fig. 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.

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.

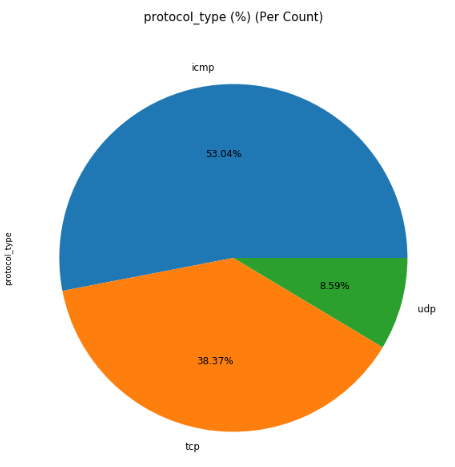


Fig. Percentage level of protocol type

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.

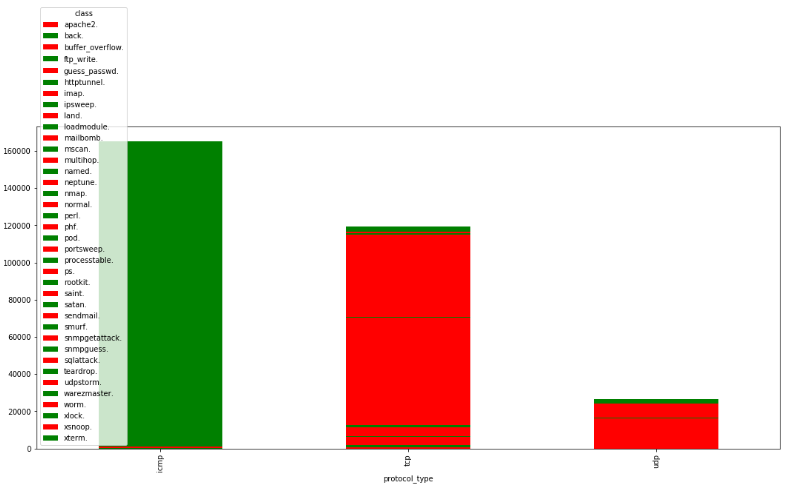


Fig. Comparison of service type and protocol type

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

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

It couldn’t fit the model on the training data and can’t say that the model will work accurately for the real data. For this, we must assure that our model got the correct patterns from the data, and it is not getting up too much noise. Cross-validation is a technique in which we train our model using the subset of the data-set and then evaluate using the complementary subset of the data-set.

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

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

Advantages of train/test split:

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

Advantages of cross-validation:

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

Data Pre-processing:

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

**Module-02:**

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

A distributed denial-of-service (DDoS) is a large-scale DoS attack where the perpetrator uses more than one unique [IP address](https://en.wikipedia.org/wiki/IP_address" \o "IP address), often thousands of them.[[10]](https://en.wikipedia.org/wiki/Denial-of-service_attack" \l "cite_note-10) A distributed denial of service attack typically involves more than around 3–5 nodes on different networks; fewer nodes may qualify as a DoS attack but is not a DDoS attack.[[11]](https://en.wikipedia.org/wiki/Denial-of-service_attack" \l "cite_note-Infosec7Layer-11)[[12]](https://en.wikipedia.org/wiki/Denial-of-service_attack" \l "cite_note-12) Since the incoming traffic flooding the victim originates from different sources, it may be impossible to stop the attack simply by using [ingress filtering](https://en.wikipedia.org/wiki/Ingress_filtering" \o "Ingress filtering). It also makes it difficult to distinguish legitimate user traffic from attack traffic when spread across multiple points of origin. As an alternative or augmentation of a DDoS, attacks may involve forging of IP sender addresses ([IP address spoofing](https://en.wikipedia.org/wiki/IP_address_spoofing" \o "IP address spoofing)) further complicating identifying and defeating the attack. An application layer DDoS attack (sometimes referred to as layer 7 DDoS attack) is a form of DDoS attack where attackers target [application-layer](https://en.wikipedia.org/wiki/Application_layer" \o "Application layer) processes. The attack over-exercises specific functions or features of a website with the intention to disable those functions or features. This application-layer attack is different from an entire network attack, and is often used against financial institutions to distract IT and security personnel from security breaches.

**Module-03:**

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

**Module-04:**

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

**Module-05:**

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

**Module-06:**

Accuracy calculation:

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

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

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

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

Comparing Algorithm with prediction in the form of best accuracy result:

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

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

In the example below 5 different algorithms are compared:

* Logistic Regression
* Random Forest
* K-Nearest Neighbors
* Decision tree
* Support Vector Machines
* Naive Bayes

Training the Dataset:

* The first line imports iris data set which is already predefined in sklearn module and raw data set is basically a table which contains information about various varieties.
* For example, to import any algorithm and train\_test\_split class from sklearn and numpy module for use in this program.
* To encapsulate load\_data() method in data\_dataset variable. Further divide the dataset into training data and test data using train\_test\_split method. The X prefix in variable denotes the feature values and y prefix denotes target values.
* This method divides dataset into training and test data randomly in ratio of 67:33 / 70:30. Then we encapsulate any algorithm.
* In the next line, we fit our training data into this algorithm so that computer can get trained using this data. Now the training part is complete.

Testing the Dataset:

* Now, the dimensions of new features in a numpy array called ‘n’ and it want to predict the species of this features and to do using the predict method which takes this array as input and spits out predicted target value as output.
* So, the predicted target value comes out to be 0. Finally to find the test score which is the ratio of no. of predictions found correct and total predictions made and finding accuracy score method which basically compares the actual values of the test set with the predicted values.

Sensitivity:

Sensitivity is a measure of the proportion of actual positive cases that got predicted as positive (or true positive). Sensitivity is also termed as Recall. This implies that there will be another proportion of actual positive cases, which would get predicted incorrectly as negative (and, thus, could also be termed as the false negative). This can also be represented in the form of a false negative rate. The sum of sensitivity and false negative rate would be 1. Let's try and understand this with the model used for predicting whether a person is suffering from the disease. Sensitivity is a measure of the proportion of people suffering from the disease who got predicted correctly as the ones suffering from the disease. In other words, the person who is unhealthy actually got predicted as unhealthy.

Mathematically, sensitivity can be calculated as the following:

Sensitivity = (True Positive) / (True Positive + False Negative)

The following is the details in relation to True Positive and False Negative used in the above equation.

* True Positive = Persons predicted as suffering from the disease (or unhealthy) are actually suffering from the disease (unhealthy); In other words, the true positive represents the number of persons who are unhealthy and are predicted as unhealthy.
* False Negative = Persons who are actually suffering from the disease (or unhealthy) are actually predicted to be not suffering from the disease (healthy). In other words, the false negative represents the number of persons who are unhealthy and got predicted as healthy. Ideally, we would seek the model to have low false negatives as it might prove to be life-threatening or business threatening.

The higher value of sensitivity would mean higher value of true positive and lower value of false negative. The lower value of sensitivity would mean lower value of true positive and higher value of false negative. For healthcare and financial domain, models with high sensitivity will be desired.

Specificity:

Specificity is defined as the proportion of actual negatives, which got predicted as the negative (or true negative). This implies that there will be another proportion of actual negative, which got predicted as positive and could be termed as false positives. This proportion could also be called a false positive rate. The sum of specificity and false positive rate would always be 1. Let's try and understand this with the model used for predicting whether a person is suffering from the disease. Specificity is a measure of the proportion of people not suffering from the disease who got predicted correctly as the ones who are not suffering from the disease. In other words, the person who is healthy actually got predicted as healthy is specificity.

Mathematically, specificity can be calculated as the following:

Specificity = (True Negative) / (True Negative + False Positive)

The following is the details in relation to True Negative and False Positive used in the above equation.

* True Negative = Persons predicted as not suffering from the disease (or healthy) are actually found to be not suffering from the disease (healthy); In other words, the true negative represents the number of persons who are healthy and are predicted as healthy.
* False Positive = Persons predicted as suffering from the disease (or unhealthy) are actually found to be not suffering from the disease (healthy). In other words, the false positive represents the number of persons who are healthy and got predicted as unhealthy.

The higher value of specificity would mean higher value of true negative and lower false positive rate. The lower value of specificity would mean lower value of true negative and higher value of false positive.

Prediction result by accuracy:

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

True Positive Rate(TPR) = TP / (TP + FN)

False Positive rate(FPR) = FP / (FP + TN)

Accuracy: The Proportion of the total number of predictions that is correct otherwise overall how often the model predicts correctly defaulters and non-defaulters.

Accuracy calculation:

Accuracy = (TP + TN) / (TP + TN + FP + FN)

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

Precision: The proportion of positive predictions that are actually correct. (When the model predicts default: how often is correct?)

Precision = TP / (TP + FP)

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

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

Recall = TP / (TP + FN)

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

F1 Scoreis the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it’s better to look at both Precision and Recall.

General Formula: F- Measure = 2TP / (2TP + FP + FN)

F1-Score Formula: F1 Score = 2\*(Recall \* Precision) / (Recall + Precision)

**Module-07:**

Tkinter is a python library for developing GUI (Graphical User Interfaces). We use the tkinter library for creating an application of UI (User Interface), to create windows and all other graphical user interface and Tkinter will come with Python as a standard package, it can be used for security purpose of each users or accountants.

Algorithm Explanation:

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. In Supervised Learning, algorithms learn from labeled data. After understanding the data, the algorithm determines which label should be given to new data based on pattern and associating the patterns to the unlabeled new data.

**[Logistic Regression](https://en.wikipedia.org/wiki/Logistic_regression" \t "_blank):**

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.

[Decision Tree](https://www.geeksforgeeks.org/decision-tree/):

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

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

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a data set into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. A decision node has two or more branches and a leaf node represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data. Decision tree builds classification or regression models in the form of a tree structure. It utilizes an if-then rule set which is mutually exclusive and exhaustive for classification. The rules are learned sequentially using the training data one at a time. Each time a rule is learned, the tuples covered by the rules are removed.

This process is continued on the training set until meeting a termination condition. It is constructed in a top-down recursive divide-and-conquer manner. All the attributes should be categorical. Otherwise, they should be discretized in advance. Attributes in the top of the tree have more impact towards in the classification and they are identified using the information gain concept. A decision tree can be easily over-fitted generating too many branches and may reflect anomalies due to noise or outliers.

K-Nearest Neighbor (KNN):

K-Nearest Neighbor is a supervised machine learning algorithm which stores all instances correspond to training data points in n-dimensional space. When an unknown discrete data is received, it analyzes the closest k number of instances saved (nearest neighbors) and returns the most common class as the prediction and for real-valued data it returns the mean of k nearest neighbors. In the distance-weighted nearest neighbor algorithm, it weights the contribution of each of the k neighbors according to their distance using the following query giving greater weight to the closest neighbors.

Usually KNN is robust to noisy data since it is averaging the k-nearest neighbors. The k-nearest-neighbors algorithm is a classification algorithm, and it is supervised: it takes a bunch of labeled points and uses them to learn how to label other points. To label a new point, it looks at the labeled points closest to that new point (those are its nearest neighbors), and has those neighbors vote, so whichever label the most of the neighbors have is the label for the new point (the “k” is the number of neighbors it checks). Makes predictions about the validation set using the entire training set. KNN makes a prediction about a new instance by searching through the entire set to find the k “closest” instances. “Closeness” is determined using a proximity measurement (Euclidean) across all features.

**Random Forest:**

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees’ habit of over fitting to their training set. Random forest is a type of supervised machine learning algorithm based on [ensemble learning](https://en.wikipedia.org/wiki/Ensemble_learning" \t "_blank). Ensemble learning is a type of learning where you join different types of algorithms or same algorithm multiple times to form a more powerful prediction model. The [random forest](https://en.wikipedia.org/wiki/Random_forest" \t "_blank) algorithm combines multiple algorithm of the same type i.e. multiple decision trees, resulting in a forest of trees, hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks.

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

* Pick N random records from the dataset.
* Build a decision tree based on these N records.
* Choose the number of trees you want in your algorithm and repeat steps 1 and 2.
* In case of a regression problem, for a new record, each tree in the forest predicts a value for Y (output). The final value can be calculated by taking the average of all the values predicted by all the trees in forest. Or, in case of a classification problem, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote.

Naive Bayes algorithm:

The Naive Bayes algorithm is an intuitive method that uses the probabilities of each attribute belonging to each class to make a prediction. It is the supervised learning approach you would come up with if you wanted to model a predictive modeling problem probabilistically. Naive bayes simplifies the calculation of probabilities by assuming that the probability of each attribute belonging to a given class value is independent of all other attributes. This is a strong assumption but results in a fast and effective method. The probability of a class value given a value of an attribute is called the conditional probability. By multiplying the conditional probabilities together for each attribute for a given class value, we have a probability of a data instance belonging to that class. To make a prediction we can calculate probabilities of the instance belonging to each class and select the class value with the highest probability.

Naive Bayes is a statistical classification technique based on Bayes Theorem. It is one of the simplest supervised learning algorithms. Naive Bayes classifier is the fast, accurate and reliable algorithm. Naive Bayes classifiers have high accuracy and speed on large datasets. Naive Bayes classifier assumes that the effect of a particular feature in a class is independent of other features. For example, a loan applicant is desirable or not depending on his/her income, previous loan and transaction history, age, and location. Even if these features are interdependent, these features are still considered independently. This assumption simplifies computation, and that's why it is considered as naive. This assumption is called class conditional independence.

**Support Vector Machines:**

A classifier that categorizes the data set by setting an optimal hyper plane between data. I chose this classifier as it is incredibly versatile in the number of different kernel functions that can be applied and this model can yield a high predictability rate. Support Vector Machines are perhaps one of the most popular and talked about machine learning algorithms. They were extremely popular around the time they were developed in the 1990s and continue to be the go-to method for a high-performing algorithm with little tuning.

* How to disentangle the many names used to refer to support vector machines.
* The representation used by SVM when the model is actually stored on disk.
* How a learned SVM model representation can be used to make predictions for new data.
* How to learn an SVM model from training data.
* How to best prepare your data for the SVM algorithm.
* Where you might look to get more information on SVM.

**Used Python Packages:**

**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.

**tkinter:**

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

**SAMPLE CODE**

#import library packages

import pandas as p

import numpy as n

# feature names

features = ["duration", "protocol\_type", "service", "flag", "src\_bytes", "dst\_bytes", "land", "Wrong\_fragment", "Urgent", "hot", "num\_failed\_login", "logged\_in", "num\_compromised", "root\_shell", "su\_attempted", "num\_root", "num\_file\_creations", "num\_shells", "num\_access\_files", "num\_outbound\_cmds", "is\_host\_login", "is\_guest\_login", "count", "srv\_count", "serror\_rate", "srv\_serror\_rate", "rerror\_rate", "srv\_rerror\_rate", "same\_srv\_rate", "diff\_srv\_rate", "srv\_diff\_host\_rate", "dst\_host\_count", "dst\_host\_srv\_count", "dst\_host\_same\_srv\_rate", "dst\_host\_diff\_ srv\_rate", "dst\_host\_same\_src\_port\_rate", "dst\_host\_srv\_diff\_host \_rate", "dst\_host\_serror\_rate", "dst\_host\_srv\_serror\_rate", "dst\_host\_rerror\_rate", "dst\_host\_srv\_rerror\_rate","class"]

data = p.read\_csv("data.csv", names = features)

data.head(10)

#shape

data.shape

df=data.dropna()

#show columns

df.columns

#To describe the dataframe

df.describe()

#Checking datatype and information about dataset

df.info()

#Checking for duplicate data

df.duplicated()

#find sum of duplicate data

sum(df.duplicated())

#Checking sum of missing values

df.isnull().sum()

d = p.crosstab(df['protocol\_type'], df['class'])

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

import matplotlib.pyplot as plt

pr = df["protocol\_type"]

fl = df["flag"]

plt.plot(fl, pr, color='g')

plt.xlabel('Flag Types')

plt.ylabel('Protocol Types')

plt.title('Flag Details by protocol type')

plt.show()

df["class"].unique()

import numpy as n

def PropByVar(df, variable):

dataframe\_pie = df[variable].value\_counts()

ax = dataframe\_pie.plot.pie(figsize=(10,10), autopct='%1.2f%%', fontsize = 12);

ax.set\_title(variable + ' (%) (Per Count)\n', fontsize = 15);

return n.round(dataframe\_pie/df.shape[0]\*100,2)

PropByVar(df, 'protocol\_type')

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

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

df['DOS'] = df['class'].map({'normal.':0, 'snmpgetattack.':0, 'named.':0, 'xlock.':0, 'smurf.':1,

'ipsweep.':0, 'multihop.':0, 'xsnoop.':0, 'sendmail.':0, 'guess\_passwd.':0,

'saint.':0, 'buffer\_overflow.':0, 'portsweep.':0, 'pod.':1, 'apache2.':1,

'phf.':0, 'udpstorm.':1, 'warezmaster.':0, 'perl.':0, 'satan.':0, 'xterm.':0,

'mscan.':0, 'processtable.':1, 'ps.':0, 'nmap.':0, 'rootkit.':0, 'neptune.':1,

'loadmodule.':0, 'imap.':0, 'back.':1, 'httptunnel.':0, 'worm.':0,

'mailbomb.':1, 'ftp\_write.':0, 'teardrop.':1, 'land.':1, 'sqlattack.':0,

'snmpguess.':0})

df['R2L'] = df['class'].map({'normal.':0, 'snmpgetattack.':1, 'named.':1, 'xlock.':1, 'smurf.':0,

'ipsweep.':0, 'multihop.':1, 'xsnoop.':1, 'sendmail.':1, 'guess\_passwd.':1,

'saint.':0, 'buffer\_overflow.':0, 'portsweep.':0, 'pod.':0, 'apache2.':0,

'phf.':1, 'udpstorm.':0, 'warezmaster.':1, 'perl.':0, 'satan.':0, 'xterm.':0,

'mscan.':0, 'processtable.':0, 'ps.':0, 'nmap.':0, 'rootkit.':0, 'neptune.':0,

'loadmodule.':0, 'imap.':1, 'back.':0, 'httptunnel.':1, 'worm.':1,

'mailbomb.':0, 'ftp\_write.':1, 'teardrop.':0, 'land.':0, 'sqlattack.':0,

'snmpguess.':1})

df['U2R'] = df['class'].map({'normal.':0, 'snmpgetattack.':0, 'named.':0, 'xlock.':0, 'smurf.':0,

'ipsweep.':0, 'multihop.':0, 'xsnoop.':0, 'sendmail.':0, 'guess\_passwd.':0,

'saint.':0, 'buffer\_overflow.':1, 'portsweep.':0, 'pod.':0, 'apache2.':0,

'phf.':0, 'udpstorm.':0, 'warezmaster.':0, 'perl.':1, 'satan.':0, 'xterm.':1,

'mscan.':0, 'processtable.':0, 'ps.':1, 'nmap.':0, 'rootkit.':1, 'neptune.':0,

'loadmodule.':1, 'imap.':0, 'back.':0, 'httptunnel.':0, 'worm.':0,

'mailbomb.':0, 'ftp\_write.':0, 'teardrop.':0, 'land.':0, 'sqlattack.':1,

'snmpguess.':0})

df['Probe'] = df['class'].map({'normal.':0, 'snmpgetattack.':0, 'named.':0, 'xlock.':0, 'smurf.':0,

'ipsweep.':1, 'multihop.':0, 'xsnoop.':0, 'sendmail.':0, 'guess\_passwd.':0,

'saint.':1, 'buffer\_overflow.':0, 'portsweep.':1, 'pod.':0, 'apache2.':0,

'phf.':0, 'udpstorm.':0, 'warezmaster.':0, 'perl.':0, 'satan.':1, 'xterm.':0,

'mscan.':1, 'processtable.':0, 'ps.':0, 'nmap.':1, 'rootkit.':0, 'neptune.':0,

'loadmodule.':0, 'imap.':0, 'back.':0, 'httptunnel.':0, 'worm.':0,

'mailbomb.':0, 'ftp\_write.':0, 'teardrop.':0, 'land.':0, 'sqlattack.':0,

'snmpguess.':0})

df['attack'] = df['class'].map({'normal.':0, 'snmpgetattack.':1, 'named.':1, 'xlock.':1, 'smurf.':1,

'ipsweep.':1, 'multihop.':1, 'xsnoop.':1, 'sendmail.':1, 'guess\_passwd.':1,

'saint.':1, 'buffer\_overflow.':1, 'portsweep.':1, 'pod.':1, 'apache2.':1,

'phf.':1, 'udpstorm.':1, 'warezmaster.':1, 'perl.':1, 'satan.':1, 'xterm.':1,

'mscan.':1, 'processtable.':1, 'ps.':1, 'nmap.':1, 'rootkit.':1, 'neptune.':1,

'loadmodule.':1, 'imap.':1, 'back.':1, 'httptunnel.':1, 'worm.':1,

'mailbomb.':1, 'ftp\_write.':1, 'teardrop.':1, 'land.':1, 'sqlattack.':1,

'snmpguess.':1})

from sklearn.preprocessing import LabelEncoder

var\_mod = ['duration', 'protocol\_type', 'service', 'flag', 'src\_bytes',

'dst\_bytes', 'land', 'Wrong\_fragment', 'Urgent', 'hot',

'num\_failed\_login', 'logged\_in', 'num\_compromised', 'root\_shell',

'su\_attempted', 'num\_root', 'num\_file\_creations', 'num\_shells',

'num\_access\_files', 'num\_outbound\_cmds', 'is\_host\_login',

'is\_guest\_login', 'count', 'srv\_count', 'serror\_rate',

'srv\_serror\_rate', 'rerror\_rate', 'srv\_rerror\_rate', 'same\_srv\_rate',

'diff\_srv\_rate', 'srv\_diff\_host\_rate', 'dst\_host\_count',

'dst\_host\_srv\_count', 'dst\_host\_same\_srv\_rate',

'dst\_host\_diff\_ srv\_rate', 'dst\_host\_same\_src\_port\_rate',

'dst\_host\_srv\_diff\_host \_rate', 'dst\_host\_serror\_rate',

'dst\_host\_srv\_serror\_rate', 'dst\_host\_rerror\_rate',

'dst\_host\_srv\_rerror\_rate', ]

le = LabelEncoder()

for i in var\_mod:

df[i] = le.fit\_transform(df[i]).astype(str)

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

from sklearn.metrics import confusion\_matrix, classification\_report, matthews\_corrcoef, cohen\_kappa\_score, accuracy\_score, average\_precision\_score, roc\_auc\_score

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

#Response variable

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

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=1, stratify=y)

# Prevent view warnings

X\_train.is\_copy = False

X\_test.is\_copy = False

from sklearn.linear\_model import LogisticRegression

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))

x = (accuracy\_score(y\_test,predictR)\*100)

print('Accuracy result of Logistic Regression is:', x)

print("")

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("")

from sklearn.tree import DecisionTreeClassifier

dtree = DecisionTreeClassifier()

dtree.fit(X\_train, y\_train)

predictDT = dtree.predict(X\_test)

print("")

print('Classification report of Decision Tree Classifier Results:')

print("")

print(classification\_report(y\_test,predictDT))

x = (accuracy\_score(y\_test,predictDT)\*100)

print('Accuracy result of Decision Tree Classifier is', x)

print("")

cm2=confusion\_matrix(y\_test,predictDT)

print('Confusion Matrix result of Decision Tree Classifier is:\n', confusion\_matrix(y\_test,predictDT))

print("")

sensitivity1 = cm2[0,0]/(cm2[0,0]+cm2[0,1])

print('Sensitivity : ', sensitivity1 )

print("")

specificity1 = cm2[1,1]/(cm2[1,0]+cm2[1,1])

print('Specificity : ', specificity1)

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier()

rfc.fit(X\_train,y\_train)

predictR = rfc.predict(X\_test)

print("")

print('Classification report of Random Forest Results:')

print("")

print(classification\_report(y\_test,predictR))

x = (accuracy\_score(y\_test,predictR)\*100)

print('Accuracy result of Random Forest is:', x)

print("")

cm1=confusion\_matrix(y\_test,predictR)

print('Confusion Matrix result of Random Forest is:\n',cm1)

print("")

sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])

print('Sensitivity : ', sensitivity1 )

print("")

specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])

print('Specificity : ', specificity1)

print("")

from sklearn.svm import SVC

sv = SVC()

sv.fit(X\_train, y\_train)

predictSVC = sv.predict(X\_test)

print("")

print('Classification report of Support Vector Classifier Results:')

print("")

print(classification\_report(y\_test,predictSVC))

x = (accuracy\_score(y\_test,predictSVC)\*100)

print('Accuracy result of Support Vector Classifier is', x)

print("")

cm4=confusion\_matrix(y\_test,predictSVC)

print('Confusion Matrix result of Support Vector Classifier is:\n', confusion\_matrix(y\_test,predictSVC))

print("")

sensitivity1 = cm4[0,0]/(cm4[0,0]+cm4[0,1])

print('Sensitivity : ', sensitivity1 )

print("")

specificity1 = cm4[1,1]/(cm4[1,0]+cm4[1,1])

print('Specificity : ', specificity1)

from sklearn.naive\_bayes import GaussianNB

gnb = GaussianNB()

gnb.fit(X\_train,y\_train)

predictR = gnb.predict(X\_test)

print("")

print('Classification report of Naive Bayes Results:')

print("")

print(classification\_report(y\_test,predictR))

x = (accuracy\_score(y\_test,predictR)\*100)

print('Accuracy result of Naive Bayes is:', x)

print("")

cm1=confusion\_matrix(y\_test,predictR)

print('Confusion Matrix result of Naive Bayes 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("")

from sklearn.neighbors import KNeighborsClassifier

knnc = KNeighborsClassifier()

knnc.fit(X\_train,y\_train)

predictR = knnc.predict(X\_test)

print("")

print('Classification report of K-Nearest Neighbor Results:')

print("")

print(classification\_report(y\_test,predictR))

x = (accuracy\_score(y\_test,predictR)\*100)

print('Accuracy result of K-Nearest Neighbor is:', x)

print("")

cm2=confusion\_matrix(y\_test,predictR)

print('Confusion Matrix result of K-Nearest Neighbor is:\n',cm2)

print("")

sensitivity1 = cm2[0,0]/(cm2[0,0]+cm2[0,1])

print('Sensitivity : ', sensitivity1 )

print("")

specificity1 = cm2[1,1]/(cm2[1,0]+cm2[1,1])

print('Specificity : ', specificity1)

print("")

l1=['SRC\_BY\_BL\_50', 'SRC\_BY\_AB\_100', 'SRC\_BY\_AB\_200', 'SRC\_BY\_AB\_3000',

'SRC\_BY\_AB\_700', 'SRC\_BY\_AB\_30000', 'SRC\_BY\_AB\_8000', 'SRC\_BY\_AB\_500',

'SRC\_BY\_AB\_300', 'SRC\_BY\_AB\_1000', 'SRC\_BY\_AB\_600', 'SRC\_BY\_AB\_900',

'SRC\_BY\_AB\_800', 'SRC\_BY\_AB\_10000', 'SRC\_BY\_AB\_2000',

'SRC\_BY\_AB\_200000', 'SRC\_BY\_AB\_400', 'SRC\_BY\_AB\_50000',

'SRC\_BY\_AB\_5000', 'SRC\_BY\_AB\_4000', 'SRC\_BY\_AB\_30000000',

'SRC\_BY\_AB\_7000', 'SRC\_BY\_AB\_1000000', 'SRC\_BY\_AB\_6000',

'SRC\_BY\_AB\_60000', 'SRC\_BY\_AB\_100000', 'SRC\_BY\_AB\_500000',

'SRC\_BY\_AB\_40000', 'SRC\_BY\_AB\_9000', 'SRC\_BY\_AB\_20000',

'SRC\_BY\_AB\_80000', 'SRC\_BY\_AB\_90000', 'SRC\_BY\_AB\_70000']

l2=['DST\_BY\_AB\_100', 'DST\_BY\_BL\_50', 'DST\_BY\_AB\_200', 'DST\_BY\_AB\_300',

'DST\_BY\_AB\_10000', 'DST\_BY\_AB\_3000', 'DST\_BY\_AB\_700', 'DST\_BY\_AB\_9000',

'DST\_BY\_AB\_8000', 'DST\_BY\_AB\_600', 'DST\_BY\_AB\_2000', 'DST\_BY\_AB\_7000',

'DST\_BY\_AB\_1000', 'DST\_BY\_AB\_4000', 'DST\_BY\_AB\_5000', 'DST\_BY\_AB\_500',

'DST\_BY\_AB\_400', 'DST\_BY\_AB\_900', 'DST\_BY\_AB\_6000', 'DST\_BY\_AB\_300000',

'DST\_BY\_AB\_20000', 'DST\_BY\_AB\_800', 'DST\_BY\_AB\_30000',

'DST\_BY\_AB\_60000', 'DST\_BY\_AB\_200000', 'DST\_BY\_AB\_40000',

'DST\_BY\_AB\_50000']

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

l4=['SF', 'RSTR', 'S1', 'REJ', 'S3', 'RSTO', 'S0', 'S2']

l5=['private', 'domain\_u', 'http', 'smtp', 'ftp\_data', 'ftp','eco\_i', 'other', 'auth', 'ecr\_i', 'IRC', 'X11', 'finger', 'time',

'domain', 'telnet', 'pop\_3', 'ldap', 'login', 'name', 'ntp\_u',

'http\_443', 'sunrpc', 'printer', 'systat', 'tim\_i','netstat', 'remote\_job',

'link', 'urp\_i', 'sql\_net', 'bgp', 'pop\_2', 'tftp\_u', 'uucp',

'imap4', 'pm\_dump', 'nnsp', 'courier', 'daytime', 'iso\_tsap',

'echo', 'discard', 'ssh', 'whois', 'mtp', 'gopher', 'rje', 'ctf',

'supdup', 'hostnames', 'csnet\_ns', 'uucp\_path', 'nntp',

'netbios\_ns', 'netbios\_dgm', 'netbios\_ssn', 'vmnet', 'Z39\_50',

'exec', 'shell', 'efs', 'klogin', 'kshell', 'icmp']

l6=['SRC\_BY\_BL\_50', 'SRC\_BY\_AB\_100', 'SRC\_BY\_AB\_200', 'SRC\_BY\_AB\_3000',

'SRC\_BY\_AB\_700', 'SRC\_BY\_AB\_30000', 'SRC\_BY\_AB\_8000', 'SRC\_BY\_AB\_500',

'SRC\_BY\_AB\_300', 'SRC\_BY\_AB\_1000', 'SRC\_BY\_AB\_600', 'SRC\_BY\_AB\_900',

'SRC\_BY\_AB\_800', 'SRC\_BY\_AB\_10000', 'SRC\_BY\_AB\_2000',

'SRC\_BY\_AB\_200000', 'SRC\_BY\_AB\_400', 'SRC\_BY\_AB\_50000',

'SRC\_BY\_AB\_5000', 'SRC\_BY\_AB\_4000', 'SRC\_BY\_AB\_30000000',

'SRC\_BY\_AB\_7000', 'SRC\_BY\_AB\_1000000', 'SRC\_BY\_AB\_6000',

'SRC\_BY\_AB\_60000', 'SRC\_BY\_AB\_100000', 'SRC\_BY\_AB\_500000',

'SRC\_BY\_AB\_40000', 'SRC\_BY\_AB\_9000', 'SRC\_BY\_AB\_20000',

'SRC\_BY\_AB\_80000', 'SRC\_BY\_AB\_90000', 'SRC\_BY\_AB\_70000','DST\_BY\_AB\_100', 'DST\_BY\_BL\_50', 'DST\_BY\_AB\_200', 'DST\_BY\_AB\_300',

'DST\_BY\_AB\_10000', 'DST\_BY\_AB\_3000', 'DST\_BY\_AB\_700', 'DST\_BY\_AB\_9000',

'DST\_BY\_AB\_8000', 'DST\_BY\_AB\_600', 'DST\_BY\_AB\_2000', 'DST\_BY\_AB\_7000',

'DST\_BY\_AB\_1000', 'DST\_BY\_AB\_4000', 'DST\_BY\_AB\_5000', 'DST\_BY\_AB\_500',

'DST\_BY\_AB\_400', 'DST\_BY\_AB\_900', 'DST\_BY\_AB\_6000', 'DST\_BY\_AB\_300000',

'DST\_BY\_AB\_20000', 'DST\_BY\_AB\_800', 'DST\_BY\_AB\_30000',

'DST\_BY\_AB\_60000', 'DST\_BY\_AB\_200000', 'DST\_BY\_AB\_40000',

'DST\_BY\_AB\_50000','UDP', 'TCP','ICMP','SF', 'RSTR', 'S1', 'REJ', 'S3', 'RSTO', 'S0', 'S2','private', 'domain\_u', 'http', 'smtp', 'ftp\_data', 'ftp','eco\_i', 'other', 'auth', 'ecr\_i', 'IRC', 'X11', 'finger', 'time',

'domain', 'telnet', 'pop\_3', 'ldap', 'login', 'name', 'ntp\_u',

'http\_443', 'sunrpc', 'printer', 'systat', 'tim\_i' ,'netstat', 'remote\_job',

'link', 'urp\_i', 'sql\_net', 'bgp', 'pop\_2', 'tftp\_u', 'uucp',

'imap4', 'pm\_dump', 'nnsp', 'courier', 'daytime', 'iso\_tsap',

'echo', 'discard', 'ssh', 'whois', 'mtp', 'gopher', 'rje', 'ctf',

'supdup', 'hostnames', 'csnet\_ns', 'uucp\_path', 'nntp',

'netbios\_ns', 'netbios\_dgm', 'netbios\_ssn', 'vmnet', 'Z39\_50',

'exec', 'shell', 'efs', 'klogin', 'kshell', 'icmp']

decision = ['No\_Attack\_Occurs', 'SNMPget\_Attack', 'Named\_Attack',

'Xlock\_Attack', 'Smurf\_Attack', 'IPsweep\_Attack',

'Multihop\_Attack', 'Sendmail\_Attack', 'Saint\_Attack',

'Portsweep\_Attack', 'Pod\_Attack', 'Apache2\_Attack', 'PHF\_Attack',

'UDPstorm\_Attack', 'Warezmaster\_Attack', 'Satan\_Attack',

'Mscan\_Attack', 'Processtable\_Attack', 'Rootkit\_Attack',

'Neptune\_Attack', 'PS\_Attack', 'HTTPtunnel\_Attack',

'Guess\_passwd\_Attack', 'FTP\_Write\_Attack', 'Land\_Attack',

'SNMPguess\_Attack']

l7=[]

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

l7.append(0)

df.replace({'Attack\_Type':{'No\_Attack\_Occurs':0, 'SNMPget\_Attack':1, 'Named\_Attack':2,

'Xlock\_Attack':3, 'Smurf\_Attack':4, 'IPsweep\_Attack':5,

'Multihop\_Attack':6, 'Sendmail\_Attack':7, 'Saint\_Attack':8,

'Portsweep\_Attack':9, 'Pod\_Attack':10, 'Apache2\_Attack':11, 'PHF\_Attack':12,

'UDPstorm\_Attack':13, 'Warezmaster\_Attack':14, 'Satan\_Attack':15,

'Mscan\_Attack':16, 'Processtable\_Attack':17, 'Rootkit\_Attack':18,

'Neptune\_Attack':19, 'PS\_Attack':20, 'HTTPtunnel\_Attack':21,

'Guess\_passwd\_Attack':22, 'FTP\_Write\_Attack':23, 'Land\_Attack':24,

'SNMPguess\_Attack':25}},inplace=True)

import numpy as np

Xd= df[l6]

yd = df[["Attack\_Type"]]

np.ravel(yd)

import numpy as np

X\_testd= df[l6]

y\_testd = df[["Attack\_Type"]]

np.ravel(y\_testd)

def RandomForestClassifier():

from sklearn.ensemble import RandomForestClassifier

gnb = RandomForestClassifier()

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

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

from sklearn.metrics import accuracy\_score

y\_predd=gnb.predict(X\_testd)

print(accuracy\_score(y\_testd, y\_predd))

print(accuracy\_score(y\_testd, y\_predd,normalize=False))

# -----------------------------------------------------

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

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

for z in terms:

if(z==l6[k]):

l7[k]=1

inputtest = [l7]

predict = gnb.predict(inputtest)

predicted=predict[0]

h='no'

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

if(predicted == a):

h='yes'

break

if (h=='yes'):

t1.delete("1.0", END)

t1.insert(END, decision[a])

else:

t1.delete("1.0", END)

t1.insert(END, "Not Found")

root1 = Tk()

root1.title("Prediction of Network Attacks")

#root1.configure(background='black')

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

root.pack()

photo = PhotoImage(file ='im3.png')

root.create\_image(0,0,image=photo,anchor=NW)

src = StringVar()

src.set(None)

dst = StringVar()

dst.set(None)

prt = StringVar()

prt.set(None)

fl = StringVar()

fl.set(None)

ser = StringVar()

ser.set(None)

# Heading

w2 = Label(root, justify=LEFT, text="Network attack prediction using Machine Learning", fg="red", bg="white")

w2.config(font=("Elephant", 20))

w2.grid(row=1, column=0, columnspan=2, padx=100)

w2 = Label(root, justify=LEFT, text="DoS, R2L, U2R and Probe Types ", fg="blue")

w2.config(font=("Aharoni", 15))

w2.grid(row=2, column=0, columnspan=2, padx=100)

# labels

srcLb = Label(root, text="Source File Size(in BY):")

srcLb.grid(row=6, column=0, pady=15, sticky=W)

dstLb = Label(root, text="Destination File Size(in BY):")

dstLb.grid(row=7, column=0, pady=15, sticky=W)

prtLb = Label(root, text="Protocol Type:")

prtLb.grid(row=8, column=0, pady=15, sticky=W)

flLb = Label(root, text="Flag Type:")

flLb.grid(row=9, column=0, pady=15, sticky=W)

serLb = Label(root, text="Select services:")

serLb.grid(row=10, column=0, pady=15, sticky=W)

lrdLb = Label(root, text="Attack\_Type", fg="white", bg="red")

lrdLb.grid(row=13, column=0, pady=10, sticky=W)

# entries

OPTIONSsrc = sorted(l1)

OPTIONSdst = sorted(l2)

OPTIONSprt = sorted(l3)

OPTIONSfl = sorted(l4)

OPTIONSser = sorted(l5)

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

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

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

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

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

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

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

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

serEn = OptionMenu(root, ser,\*OPTIONSser)

serEn.grid(row=10, column=1)

def clear\_display\_result():

t1.delete('1.0',END)

lrd = Button(root, text="Check Result", command=RandomForestClassifier,bg="cyan",fg="green")

lrd.grid(row=13, column=3,padx=10)

button3=Button(root,text="Reset", command=clear\_display\_result,width=12,bg="red",fg="white")

button3.grid(row=6,column=3,padx=10,pady=10)

t1 = Text(root, height=1, width=40,bg="orange",fg="black")

t1.grid(row=13, column=1 , padx=10)

root1.mainloop()

**RESULT AND DISCUSSION**

**Software involvement steps:**

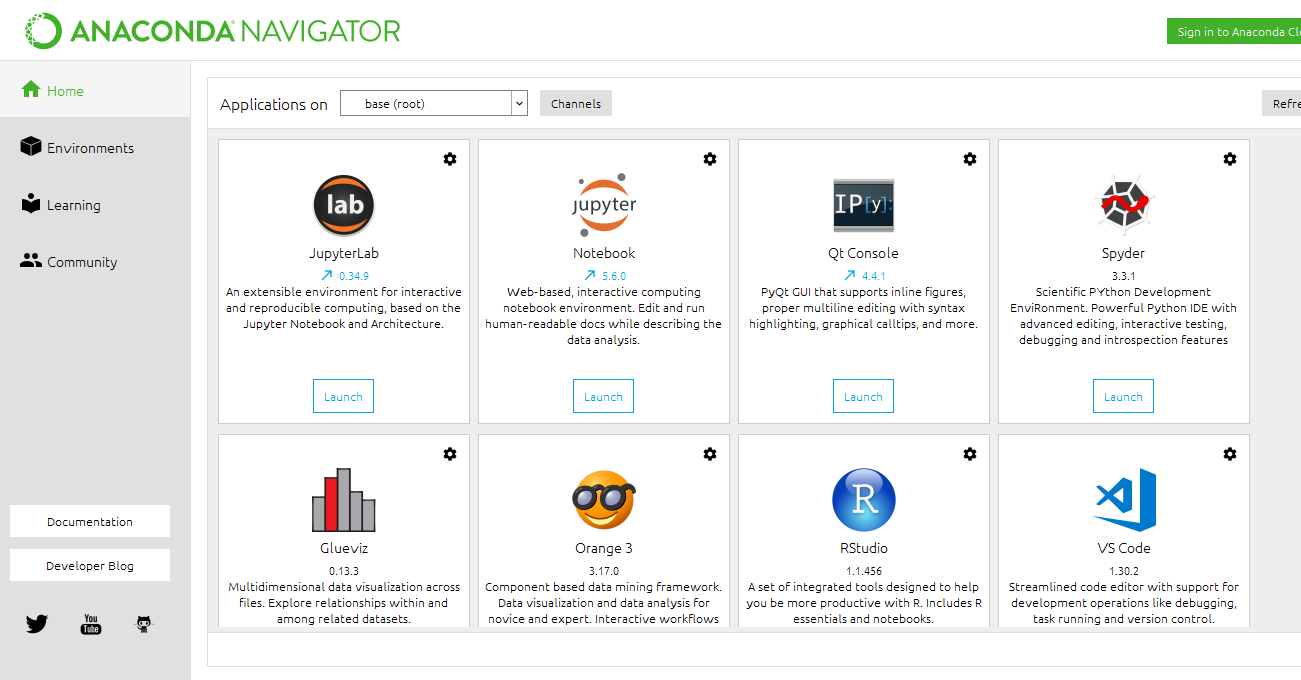
****

Fig: Open the anaconda navigator

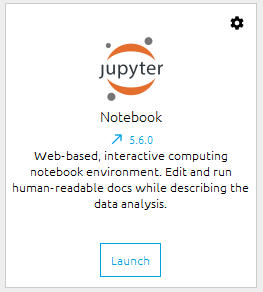
****

Fig: Launch the jupyter notebook platform

****

Fig: Open the correspondent result folder

**Performance measurements of ML algorithm:**

**DoS Attack Prediction:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Parameters** | **LR** | **DT** | **RF** | **SVC** | **KNN** | **NB** |
| **Precision** | 1 | 1 | 1 | 0.97 | 1 | 0.96 |
| **Recall** | 1 | 1 | 1 | 1 | 1 | 1 |
| **F1-Score** | 1 | 1 | 1 | 0.99 | 1 | 0.98 |
| **Sensitivity** | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |
| **Specificity** | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |
| **Accuracy (%)** | 99.94 | 99.96 | 99.97 | 99.55 | 99.97 | 99.26 |

The highest accuracy for DoS attack is Random Forest algorithm

**R2L Attack Prediction:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Parameters** | **LR** | **DT** | **RF** | **SVC** | **KNN** | **NB** |
| **Precision** | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 1 |
| **Recall** | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.98 |
| **F1-Score** | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |
| **Sensitivity** | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.97 |
| **Specificity** | 0.69 | 0.70 | 0.67 | 0.67 | 0.65 | 0.97 |
| **Accuracy (%)** | 98.68 | 98.61 | 98.6 | 98.67 | 98.61 | 97.55 |

The highest accuracy for R2L attack is Logistic Regression algorithm

**U2R Attack Prediction:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Parameters** | **LR** | **DT** | **RF** | **SVC** | **KNN** | **NB** |
| **Precision** | 1 | 1 | 1 | 1 | 1 | 1 |
| **Recall** | 1 | 1 | 1 | 1 | 1 | 1 |
| **F1-Score** | 1 | 1 | 1 | 1 | 1 | 1 |
| **Sensitivity** | 1 | 1 | 1 | 1 | 1 | 1 |
| **Specificity** | 1 | 0 | 0 | 0 | 0 | 1 |
| **Accuracy (%)** | 100 | 99.99 | 99.99 | 99.99 | 99.99 | 100 |

The highest accuracy for U2R attack is Logistic Regression and Naive Bayes algorithms

**Probe Attack Prediction:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Parameters** | **LR** | **DT** | **RF** | **SVC** | **KNN** | **NB** |
| **Precision** | 1 | 1 | 1 | 1 | 1 | 1 |
| **Recall** | 1 | 1 | 1 | 1 | 1 | 0.99 |
| **F1-Score** | 1 | 1 | 1 | 1 | 1 | 1 |
| **Sensitivity** | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |
| **Specificity** | 0.98 | 0.98 | 0.97 | 0.90 | 0.98 | 1 |
| **Accuracy (%)** | 99.89 | 99.9 | 99.90 | 99.82 | 99.90 | 99.11 |

The highest accuracy for Probe attack is Random Forest and KNN algorithms

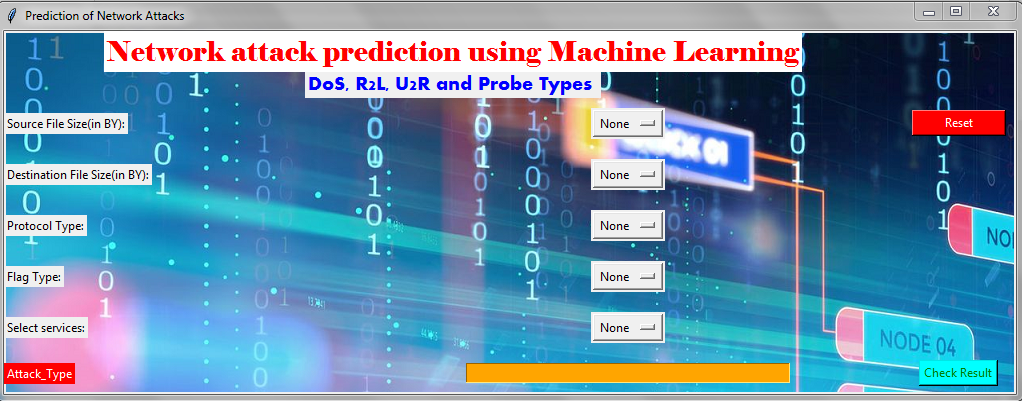
**Overall Network Attack Prediction:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Parameters** | **LR** | **DT** | **RF** | **SVC** | **KNN** | **NB** |
| **Precision** | 0.85 | 0.93 | 0.93 | 0.89 | 0.93 | 0.84 |
| **Recall** | 1 | 0.95 | 0.95 | 0.94 | 0.94 | 1 |
| **F1-Score** | 0.92 | 0.94 | 0.94 | 0.92 | 0.93 | 0.91 |
| **Sensitivity** | 0.99 | 0.94 | 0.94 | 0.94 | 0.94 | 0.99 |
| **Specificity** | 0.97 | 0.98 | 0.98 | 0.98 | 0.98 | 0.97 |
| **Accuracy (%)** | 97.78 | 98.4 | 98.44 | 97.80 | 98.32 | 97.61 |

The highest accuracy for overall Network attack is Random Forest algorithm

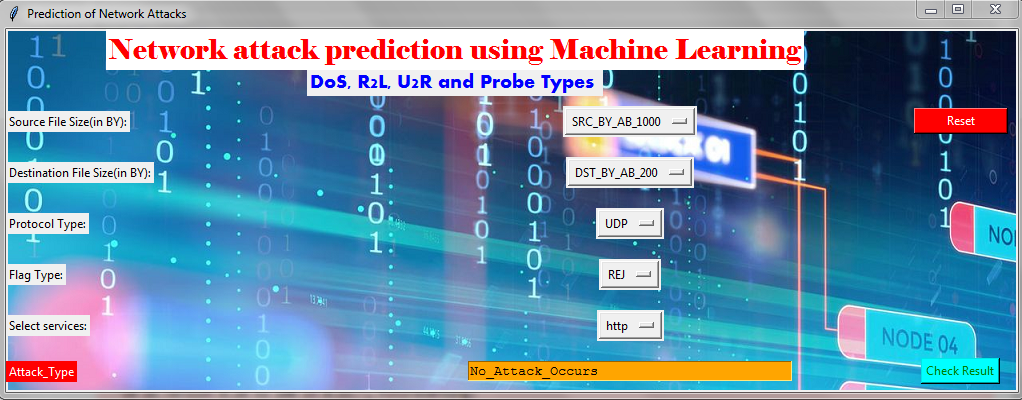
**OUTPUT SCREENSHOTS:**

**INPUT:**

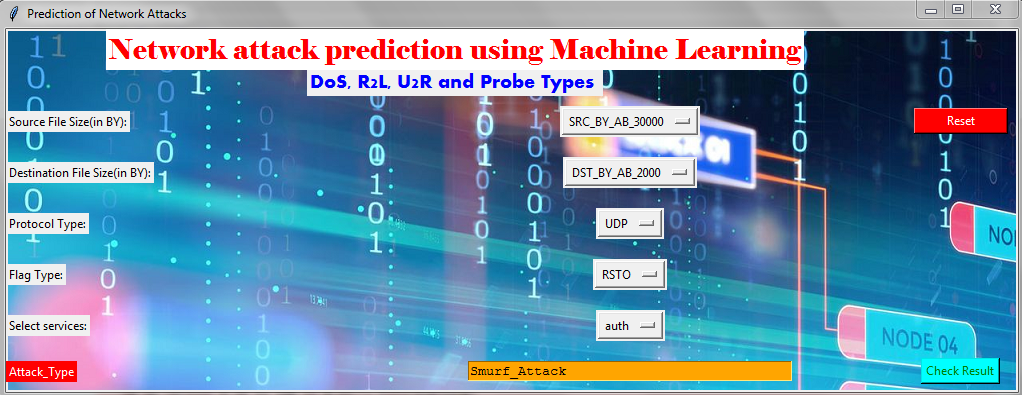
****

**OUTPUT:**

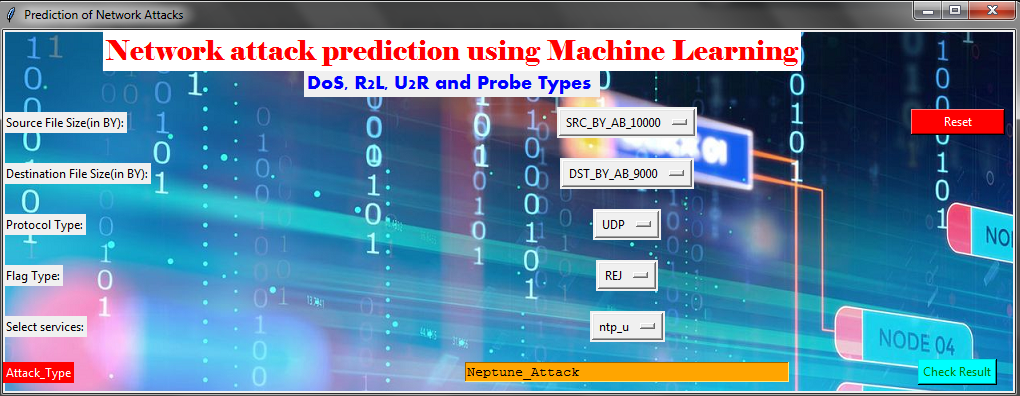
Test-01:



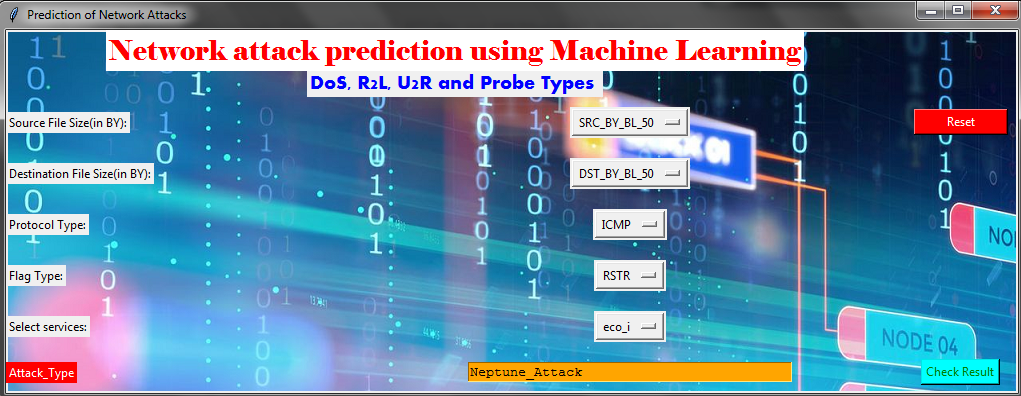
Test-02:



Test-03:



Test-04:

****

**CONCLUSION AND FUTURE WORK**

**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 random forest algorithm (98.44%) by comparing each algorithm with type of all network attacks for future prediction results by finding best connections. This brings some of the following insights about diagnose the network attack of each new connection. To presented a prediction model with the aid of artificial intelligence to improve over human accuracy and provide with the scope of early detection. It can be inferred from this model that, area analysis and use of machine learning technique is useful in developing prediction models that can helps to network sectors reduce the long process of diagnosis and eradicate any human error.

**Future Work**

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