Network Intrusion Detection System using Classification Techniques in Machine Learning

Pratik Chudasma

10509092

Dissertation submitted in partial fulfilment of the requirements for the degree of

[MSc. Data Analytics]

at Dublin Business School

Supervisor: Dr. Shahram Azizi Sazi

May 2020

Declaration

I declare that this dissertation that I have submitted to Dublin Business School for the award of [MSc. Data Analytics] is the result of my own investigations, except where otherwise stated, where it is clearly acknowledged by references. Furthermore, this work has not been submitted for any other degree.

Signed: Pratik Chudasma
Student Number: 10509092
Date: May 2020

Acknowledgement

Firstly I would like to thank God for giving me all the strength and confidence that I have got to pursue my dreams to come out to a different country and achieve my goals.

I would like to express my deepest gratitude to my supervisor Dr. Shahram Azizi Sazi for his guidance, encouragement, and gracious support throughout the course of my work.

Finally, I dedicate my work to my parents who motivated me to pursue Master's degree and who always supported me through prays, financial and moral support, especially during this situation of COVID-19 and difficulties.

Abstract

Worldwide, the Internet is connected across the country. There are threats of network attacks in this Internet environment. The risk of integrity and confidentiality has also increased with the density of information and global reach. Security breach has become too easy. In these days the improvement of the network security is therefore highlighted. Protection of the Network allow the unintended interference to some form to network and avoid it. It consists of software for network intrusion detection that track the network. NIDS is positioned in the network in a strategic location to track traffic inside the network from source to destination apps.

The machine would optimally screen both inbound and outbound traffic, but that would create a congestion that would hinder the system's overall pace. Finally, these methods include machine learning algorithms that render the device flexible and deliver reliable performance. Intrusion activities leave evidence in the auditing data, so it is possible to learn and distinguish the pattern of ordinary and malicious activities with machine learning algorithms.

Machine training techniques can learn normal, anomalous patterns from training data, and create classifiers for computer system attacks. In the area of intrusion detection for our research works, machine learning methods, such as logistic regression, Naive Bayes, K-Nearest Neighbor and Decision Trees were used. The research provides a predictive computational approach to optimize intrusion detection in the Network Traffic Data along with implementing different methodologies for the evaluation of the best accuracy from the Classification and Deep - Learning Algorithms.

A new intrusion intrusion detection system for smart networks has been developed using a two-stage distinction (Anomaly-Misuse) and a deep methodology for learning. As detection methods to identify traffic disruption that could be attacked, the Decision Tree, logistic regression, KNN and Naive Bayes were used and Deep-learning used an ANN method that would recognize the attacks as they exist. The analysis has used the complete 42 dimensional features of the training data set. The findings indicate that the high accuracy values are 100% with 0.10 recall levels at stage 1 of the appraisal, and 99.5% with 0.99 at stage 2 of the validation. Experimental findings indicate that the design of the decision tree contributes to high precision in contrast with other algorithm.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A Intrusion Detection System (IDS) is a device or a software based application that can be used to detect any malignant or privacy related activities present in the network. Any kind of illegal activity is directly reported to the administrator of the system or it's been sent directly to the central system using a system information and event management (SIEM) system. Any form of secure network should provide proper data privacy, data integrity and also assurance against denial of service (DOS) attack. The primary aim of the Intrusion Detection is to correctly identify any illegitimate use or exploitation of computer system.
Intrusion Detection System (IDS) is based on the assumption that the system can detect the behavioral difference a intruder's activities as compared to legitimate user so that their unauthorized actions can be detected.

1.1 Intrusion Detection Category

IDS may be identified based on the location of the detection or by the type of detection.
There are mainly two types of IDS which ranges from single computer to large networks. Based on location of the detection, IDS is classified into two types such as Network Intrusion Detection Systems (NIDS) and Host-Based Intrusion Detection Systems (HIDS)

1.1.1 Network Intrusion Detection Systems (NIDS)

Network Intrusion Detection Systems (NIDS) are placed at very specific strategic points within the network to monitor traffic to and from all the devices connected across the network.
NIDS performs analysis of all the currently passing traffic present on the entire subnet and matches it with the already known traffic generated by the previous attacks. Once the unmatched behaviour is spotted, an alert is directly sent to the network administrator.
Also, NIDS can be classified into two types such as on-line and off-line NIDS. On-line NIDS deals with the real time network data while the off-line NIDS looks after the stored data to check and decide whether it is a attack or not.
NIDS can be used a an stand-alone network system or can be combined with other technologies to increase the overall detection and prediction rates. One such ex-

ample is using Artificial Neural Network (ANN) based Intrusion Detection System which is capable of handling large volume of data in a very self organizing structure which in results helps in better recognition of different intrusion patterns. As seen below in the figure 1.1 we can look at the basic components present in the NIDS Architecture.(Conrad, Misenar, and Feldman 2017)

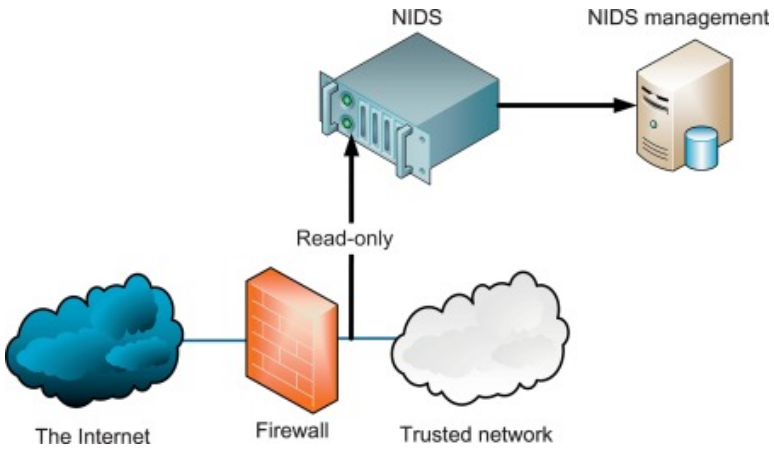


Figure 1.1: NIDS Architecture (Conrad, Misenar, and Feldman 2017)

1.1.1.1 NIDS Architecture

A network-based intrusion detection system (NIDS) detects any vicious traffic on a present network. Typically, NIDS requires special access to the system network in order to inspect the network which includes uni-cast traffic as well. NIDS are passive devices that do not interfere with the traffic they monitor 1.1 shows a typical NIDS architecture. The NIDS sniffs the internal interface of the firewall in read-only mode and sends alerts to a NIDS Management server via a different (ie, read/write) network interface.(Conrad, Misenar, and Feldman 2017)

1.1.2 Host-Based Intrusion Detection Systems (HIDS)

Host-Based Intrusion Detection Systems (HIDS) are systems that typically resides at service points rather than in network points like Network Intrusion Detection System (NIDS). HIDS are widely implemented intrusion systems and are installed on the server and are more focused on analyzing the specific operating system and functionality that resides on HIDS host. HIDS are often condemning in detecting internal attacks directed towards an organization's servers such as DNS, mail and web servers.HIDS can detect a variety of potential attack situations such as file permission changes and improperly formed client-server requests

1.2 Classification

Our aim of the project is to use machine learning and its different types of techniques to perform network intrusion detection. For this project we'll be using Classification Technique based on Machine Learning.
Classification is a supervised machine learning approach in which the computer program learns from the provided input and then uses that data to classify the information in different form.
Classification is a process of sorting the given data into number of different classes. It can be performed on both structured as well as unstructured data.

In Machine Learning, Classification Techniques are further divided into different types of models as shown in the figure 1.2 as well as some of them listed below :-

1. Logistic Regression
2. Naive Bayes Classifier
3. Nearest Neighbor
4. Support Vector Machines
5. Decision Trees
6. Boosted Trees
7. Random Forest
8. Neural Networks

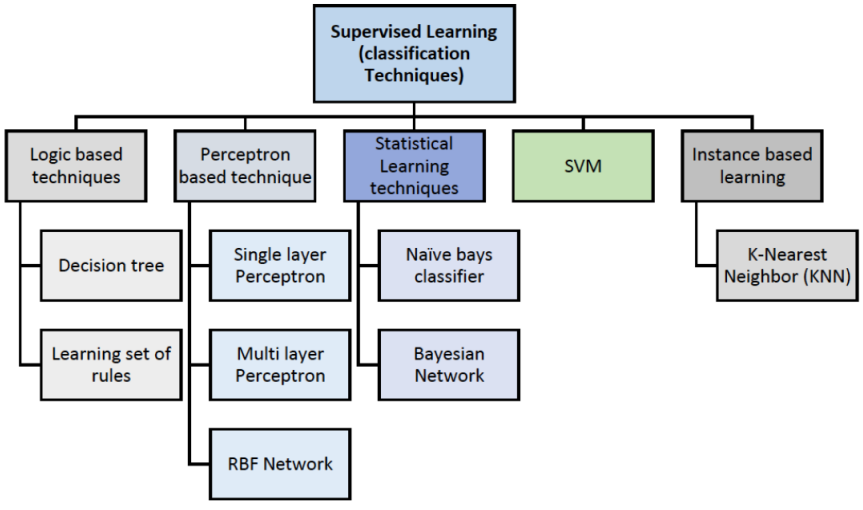


Figure 1.2: Supervised Learning Classification Technique (Soofi and Awan 2017)

Therefore, in order to perform Network Intrusion Detection using classification techniques we will be using different classification models and possibly try to find which model provide us with the highest accuracy and precision. In simple terms, Classification is a techniques which is used to assort data into different classes.

Each model in Classification serves a different purpose in various use cases. For example Logistic Regression can be used where there are only two possible outcomes such as either yes or no, whereas K-Nearest Neighbor (KNN) is used in making prediction by calculating any possible common factors between input sample and training instances.

1.3 Research Objective and challenges

An Intrusion Detection System are widely used in many organizations to overcome the problem of unauthorized use or attack over the system through network traffic. An effective IDS can be built by either of two types such as Network Intrusion Detection system (NIDS) and Host-Based Intrusion Detection System (HIDS). Both the types of systems work differently based on the given environment. An IDS should be effective to avoid types of attacks such as DOS, Probe, U2L,R2L, these attacks usually occurs on the network traffic data.
There are four major challenges faced while building a Intrusion Detection System which are listed as (Team 2017)

1. Ensuring an effective deployment

    - To attain a high level of threat visibility, organisations must ensure that intrusion detection technology is correctly installed and optimised to the working environment.

2. Managing the high volume of alerts

    - HIDS and NIDS typically utilise a combination of signature and anomaly-based detection techniques. This means alerts are generated when a sensor either detects activity that matches a known attack pattern, or flags traffic that falls outside a list of normal behaviours.

3. Understanding and investigating alerts

    - IDS alerts consist of base-level security information which, when viewed in isolation, may mean very little. Investigating IDS alerts can be very time and resource-intensive, requiring supplementary information from other systems to help determine whether an alarm is serious. Specialist skills are essential to interpret system outputs and many organisations lack the dedicated security experts capable of performing this crucial function.

4. Knowing how to respond to threats

    - A common problem for organisations that implement IDS is that they lack an appropriate incident response capability. Identifying a problem is half the battle, knowing how to respond appropriately and having the resources in place to do so is equally important.

1.4 Research Question

- How can machine learning and classification techniques be adequately enforced to detect Network Intrusion System ?
- What are the opportunities and challenges in the field of classification and machine learning in predicting more effective model ?
- To accurately utilize large data-set and apply classification algorithms to identify the model with highest accuracy.
- Identify the best performing classification technique and algorithm.
- How can we use this technology to successfully implement on real time data traffic.

Chapter 2

Literature Review

Literature Review plays a vital role in any research for providing possible relevant information. Literature Review can be stated as an overall summary of the existing articles which are published in the related field of work.

2.1 Literature Review - Introduction

2.1.1 Network Intrusion Detection (Mukherjee, Heberlein, and Levitt 1994)

- According to the research done by Biswanath Mukherjee, L. Todd Heberlein, and Karl N. Levitt it states that Intrusion Detection is a new approach in the field of security in pre-existing computers and other data networks.
- In their research they concluded that Intrusion Detection Systems were based on host-audit-trail and network traffic analysis and their goal was to detect attacks possibly in real-time.
- A number of prototype IDS's were built and their overall concept was quite promising.
- There were number of limitations in prevention based approach for network security some of them listed below as:-
  - It was impossible to build a useful system which was precisely secure. Due to some design flaws in the system large number of components cannot be excluded.
  - Crypto-based systems was not able to prevent attack against lost or stolen keys, or hacked passwords.
  - Lastly, the system was not secure enough from the insiders misusing their privileges.

Few illustrations of intrusion that are involved with systems administrators are as follows:-

  - The user's data can be altered and manipulated through unauthorized access to their files.
  - Unauthorized use of computing resources.

- The research also defines and gives us a brief idea about different types of Intrusion Detection System along with their advantages and disadvantages.

2.1.2 Classification Techniques in Machine Learning: Applications and Issues (Soofi and Awan 2017)

- Aized Amin Soofi* and Arshad Awan took the concept of classification and gave us a detailed information about different type of classification technique that are used in machine learning.
- In the research various classification techniques have been in discussed in detail along with their useful applications. An overview of different classification technique with their applicable applications are detailed in table 2.1.

Table 2.1: Applications of Classification Technique

Classification Techniques	Applications
ID3	predicting student performance land capability classification tolerance related knowledge acquisition computer crime forensics fraud detection application
C4.5	Decision making of loan application by debtor Predicting Software Defects Thrombosis collagen diseases Electricity price prediction coal logistics customer analysis Selecting Question Pools
Bayesian Network	automatic and interactive mode for Image Segmentation traffic incident detection signature verification efficient patrolling of nurses examine dental pain telecommunication and internet networks
K- Nearest neighbor	Microarray data classification Phoneme Prediction Face recognition Agarwood oil quality grading Classification of nuclear receptors and their subfamilies Short-term traffic flow forecasting Plant Leaf Recognition
SVM	Scene classification Predict corporate financial distress Induction motors fault diagnosis Analog circuit fault diagnosis enterprise market competition

Along with the applications, few of the issues regarding process of the algorithms were also discussed in the paper. Some of the solutions were also provided related to the issue caused in the classification approach. Some of them are listed in table 2.2

Table 2.2: Classification Techniques Issue and Solutions

Classification Approach	**Issue**	**Solution/Technique**
Decision tree (ID3 and C4.5)	multi valued attributes Complex information entropy and attribute with more values Noisy data classification	Algorithm by combining ID3 and association function(AF) modification to the attribute selection methods, pre pruning strategy and rainforest approach Enhanced algorithm with Taylor formula Credal-C4.5 tree
Bayesian Network	Attributes conditional density estimation Inference (large domain discrete and continuous variables) Multi-dimensional data	Gaussian kernel function decision-tree structured conditional probability greedy learning algorithm
K nearest neighbor	space requirement time requirement KNN scaling over multimedia dataset	Prototype selection feature selection and extraction methods finding R-Tree index multimedia KNN query processing system
SVM	controlling the false positive rate low sparse SVM classifier multi-label classification	Risk Area SVM (RA-SVM) Cluster Support Vector Machine (CLSVM) fuzzy SVMs (FSVMs)

2.1.3 Network Traffic Classification Techniques and Comparative Analysis Using Machine Learning Algorithms (Shafiq et al. 2016)

- The paper states the theoretical concepts about different machine learning algorithms based on network intrusion detection anomaly.
- Machine Learning technique was based on labelled datasets. In this approach, a machine learning classifier is trained as input for the system and then using trained sample prediction form classifier, unknown classes are classified.
- Network Traffic Classification was the first step to identify and correctly analyze various types of applications flowing in a network.
- Using this technique, it helped Internet Service Providers (ISP) or network operators to manage and upgrade overall performance of the network.
- There were multiple techniques available to classify the internet traffic such as Port Based, Pay Load Based and Machine Learning out of which Machine Learning was the most popular and useful.
- In this paper they discussed in a sequential format about network traffic classification techniques and real time internet data was developed using network traffic capture tool.
- Feature Extraction tools were used to extract features from four machine learning algorithms such as Support Vector Machine (SVM), C4.5 Decision Tree, Naive Bayes and Bayes Net Classifiers.

Table 2.3: Accuracy Results and Comparison of MLA

Classifiers	**Accuracy (%)**	**T Time (Second)**
C4.5	78.9189	0
SVM	74.0541	.0.3
BayesNet	68.1081	0.01
Naive Bayes	71.8919	0.01

- Table 2.3 show the accuracy results of each machine learning algorithms with their total time required to run each model.
- The experimental analysis showed that C405 Decision Tree classifier gave very good accuracy results as compared to other classifiers.
- For comparative analysis of these four algorithms, they captured WWW, DNS, FTP, P3P and TELNET application for traffic interval of 1 minute using Wire Shark Tool and feature extraction using Netmate Tool.

2.1.4 Practical real-time intrusion detection using machine learning approaches (Sangkatsanee, Wattanapongsakorn, and Charnsripinyo 2011)

- The increasing prevalence of system attacks are a well-known issue that can affect the availability, confidentiality and integrity of critical information for both individuals and businesses(Sangkatsanee, Wattanapongsakorn, and Charnsripinyo 2011).
- The study here propose an intrusion detection method that used a supervised machine learning technique in real time. Our technique is quick and efficient and can be applied to a number of machine learning techniques (Sangkatsanee, Wattanapongsakorn, and Charnsripinyo 2011).
- They have used various efficient machine learning algorithms to analyze the efficiency of IDS approach. Among the various methods, hypotheses showed that the Decision tree approach is capable of surpassing other techniques.
- With the above result,furthermore using the decision tree algorithm they developed a Real Time Intrusion Detection System that's helps in classifying the attack data among the normal online network data.
- Further they also detected network data using the information gain system which was our feature selection criteria by identifying the 12 important features of network data
- As the detection rate reached above 98 percent in 2 seconds which helped the RT-IDS to distinguished between the normal network actions among the Probe and Denial of service attacks.
- They also created a new post-processing technique to reduce the false alarm rate and improve the efficiency and precision of the intrusion detection system (Sangkatsanee, Wattanapongsakorn, and Charnsripinyo 2011).
- In the end, they also evaluated the IDS model by testing the detection speed, CPU usage, memory consumption, and detection accuracy.
- The research work had several contributions as follows:-
  - Presented an uncomplicated IDS model which required no human expert to identify the attack and also work on pre-existing systems.
  - Proposed 12 essential features to identify which were related to DOS or other attack types.
  - Created a Real-Time model which was able to identify attack as well as classify them into three groups such as Normal, Denial of Service (DOS) and Port Scanning (Probe).
  - Developed a post-processing procedure to reduce false alarm rate.

2.1.5 Using Genetic Algorithm for Network Intrusion Detection (W. Li 2004)

- This paper describes a technique of applying Genetic Algorithm (GA) to network Intrusion Detection Systems (IDSs) (W. Li 2004)
- A brief overview of the Intrusion Detection System, genetic algorithm, and related detection techniques is presented (W. Li 2004)
- This implementation includes both temporal and spatial information of different network connections in the encoding.
- Genetic Algorithm is part of computational models based on the concept of principals of evolution and natural selection.
- Random chromosomes are selected from the population at the start of the genetic algorithm process. The set of chromosomes during a stage of evolution is called as population These chromosomes are representations of the problem to be solved.

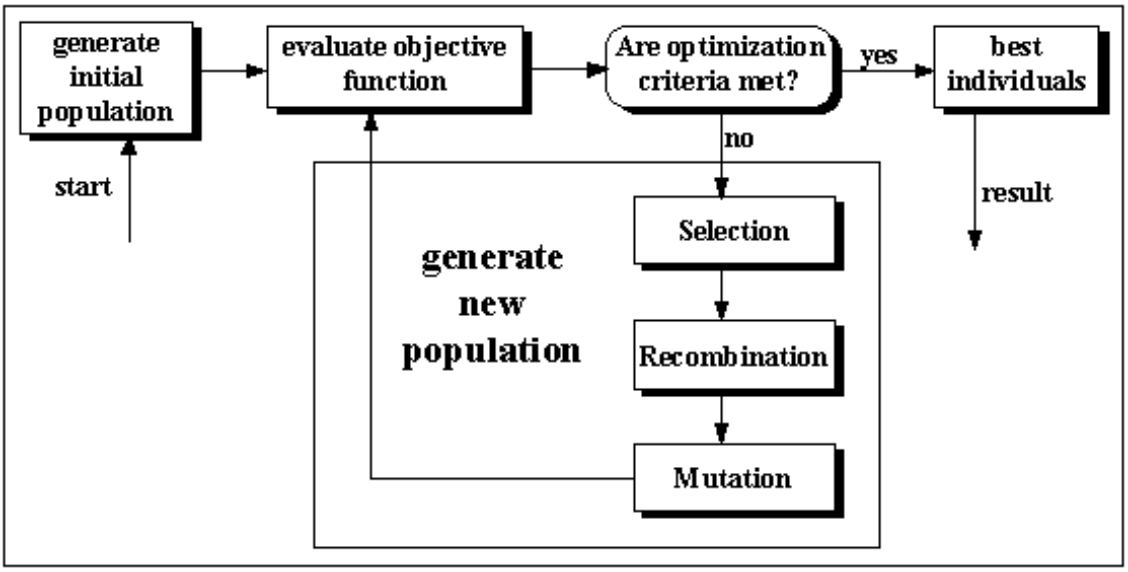


Figure 2.1: Structure of Genetic Algorithm

- Using Genetic Algorithm for NIDS helps in proper identification of complex divergent performance.
- The research work is based on the concepts of TCP/IP network protocols along with encoding the network connection data into the IDS.

2.1.6 Survey of Classification Techniques in Data Mining (Phyu 2009)

- Classification is a Data Mining (Machine Learning) technique that is used to predict category for data instances.
- Major Classification algorithms are explained in brief such as Decision Tree, Bayesian Networks, K-Nearest Neighbor (KNN), Genetic Algorithms, Fuzzy Logic Technique and Case-Based Reasoning.
- This survey research was conducted to provide a extensive review of different classification techniques in Machine Learning.
- Different analysis tools were used to conclude any matching patterns or find relationships between large datasets. This tools include mathematical algorithms, machine learning algorithms and statistical models.
- Classification techniques was capable of processing very large datasets as compared to regression and its been applied to various applications due to its popularity.
- From the research done on paper is was concluded that Decision Tree and Bayesian Network were the best algorithm while working on different machine learning techniques.

2.1.7 A two-stage hybrid classification technique for network intrusion detection system (Hussain, Lalmuanawma, and Chhakchhuak 2016)

- Based on the research work in this paper it is stated that Conventional Network Intrusion Detection System (NIDS) often uses individual classification techniques which results in failure to serve better detection rate.
- In this paper, a new two-stage hybrid classification method has been proposed by using Support Vector Machine (SVM) and Artificial Neural Network (ANN).
- In the first stage SVM will be used as an anomaly detector while in second stage ANN used as misuse detector.
- The main objective of considering this method was to increase classification accuracy with less probability of false alarm.
- The First stage detects any possible intrusion or abnormal activities through anomaly. On the other hand, second stage analyses the attack and classifies the type of attack into four different classes such as Denial of Service (DOS), User to Root (U2R), Remote to Local (R2L) and Probe (Hussain, Lalmuanawma, and Chhakchhuak 2016)

- Based on the results obtained through simulation process it was concluded that proposed hybrid classification method was much more effective and also outperformed the traditional conventional method of individually performing Support Vector Machine (SVM) and Artificial Neural Network (ANN) algorithms.
- The proposed technique showed much more reliable results of detecting intrusion activities over the network information.
- All the simulations ran were based on the NSL-KDD datasets which were the updated version of the typical KDD99 version dataset.

2.1.8 Intelligent Intrusion Detection Systems using Artificial Neural Networks (Shenfield, Day, and Ayesh 2018)

- This paper presents a rather new approach for malicious network traffic attacks using Artificial Neural Network (ANN) based Intrusion Detection System (IDS).
- The experiment uses typical network traffic benign data such as images, dynamic library files, log files, music files, etc. and also malicious data such as shell code files and vulnerability repository.
- This proposed architecture of ANN was able to easily distinguish between benign and malicious network traffic more accurately.
- An average accuracy of over 98 percent was achievable with the proposed ANN architecture. Also, average area under the Receiver Operator Characteristic (ROC) curve of 0.98 and in 10-fold cross-validation with an average false positive rate of less than 2 percent was achieved.
- This results showed that the proposed ANN classification technique was accurate, precise and robust.
- The proposed model was also able to enhance the expediency of intrusion detection system applied on both conventional network traffic and for network traffic for cyber-physical systems as smart-grids.
- The research method currently works on an offline approach for detecting shellcode patterns while an online and real-time network intrusion detection system is and ongoing project that can integrate with the existing system.

2.1.9 An Intelligent Intrusion Detection System for Mobile Ad-Hoc Networks Using Classification Techniques (Ganapathy, Yogesh, and Kannan 2011)

- This paper proposes a multi-level approach of classification technique for effective intrusion detection in Mobile AD-Hoc Networks.
- The proposed concept uses combination of tree classifier which uses a labelled training data and an Enhanced Multiclass SVM algorithm for binary classification.
- For the procedure, data reduction process was carried out using attribute selection technique from KDD 99 cup data set.
- From the tests conducted through this experiments, improvement was observed on both high detection rates as well as low false alarms rates.
- In this research work, SVM was combined with Decision Tree in order to design a multiclass SVM which would be capable of classifying the attacks in four different classes such as Probing, Denial of Service (DOS), User to Root (U2R) and Root to Location (R2L) (Ganapathy, Yogesh, and Kannan 2011)
- The main advantage of this algorithm was to provide effective preprocessing algorithm along with classification approach to attack network intrusion attack on Mobile Ad hoc Networks (MANETs) which would eventually improve overall accuracy and reduce training as well as testing time.

2.1.10 A Deep Learning Approach for Network Intrusion Detection System (Javaid et al. 2016)

- A Network Intrusion Detection System (NIDS) assists the administrator to detect any network security breaches in their management system (Javaid et al. 2016)
- Despite all of these, there were many challenges faced while developing a formative and effective NIDS system for impulsive attacks.
- Therefore, to overcome all the issues, a deep learning based approach was developed for efficient and formidable NIDS structure.
- For building this model,a combination of models were used such as Self-Taught Learning (STL), a deep learning based technique on NSL-KDD a benchmark dataset for Network Intrusion (Javaid et al. 2016)
- The results were compared on the grounds of accuracy, precision, recall and f-measure values.

- NIDS are categorized into two classes such as:- (Javaid et al. 2016)
    1. Signature (misuse) based NIDS (SNIDS)
    2. Anomaly Detection based NIDS (ADNIDS).
- In SNIDS, signature attacks are pre-installed in the NIDS. A pattern matching sequence is conducted for the network traffic against the installed signatures on the system to detect intrusion. On the other side, ADNIDS classifies a network intrusion when it detects a irregularity in the normal traffic patterns.
- SNIDS is useful in detecting known attacks and shows results with high accuracy with less false alarms. Whereas, ADNIDS is used to detect relatively unknown and new form of attacks.

2.1.11 HIDE: a Hierarchical Network Intrusion Detection System Using Statistical Preprocessing and Neural Network Classification (Zhang et al. 2001)

- This paper introduces a new approach with Hierarchical Intrusion Detection (HIDE) system which uses statistical preprocessing and neural network classification for detecting network based attacks.
- For the proposed research five different types of neural network classifiers were used such as:-
    1. Perceptron,
    2. Backpropagation (BP),
    3. Perceptron-backpropagation-hybrid (PBH),
    4. Fuzzy ARTMAP, and
    5. Radial-based Function
- The results indicated that Backpropagation (BP) and Perceptron-backpropagation-hybrid (PBH) were the best among five for more efficient classification of our data.
- A Stress-Test was also conducted on the entire system to conclude that HIDE was a reliable technique in detecting flooding attacks with very low attack intensity as much as only five to ten percent.

2.1.12 Network Intrusion Detection using Naïve Bayes (Panda and Patra 2007)

- The paper uses Naive Bayes classification algorithm for anomaly based Network Intrusion Detection System.
- The tests were ran on KDD'99 dataset which displayed good results in intrusion detection system.
- The dataset was under the sponsorship of Defence Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory (AFRL), MIT Lincoln Laboratory has collected and distributed the datasets for the evaluation of network intrusion detection systems.
- DARPA is one of the most popular dataset that is used to test and evaluate large number of Intrusion Detection System. Also, KDD'99 dataset is the subset of DARPA dataset.
- KDD'99 dataset was effective in certain area when compared directly with back propagation neural network approach.
- The experiment was carried out on only 10 percent of KDD'99 dataset.
- The performance of the system was evaluated on the basis of detection rate and false positive rate.
- It was noticed that the proposed system performed better in the field of false positive rate, computational time and was cost effective as well.

2.1.13 Random Forest Modeling for Network Intrusion Detection System (Farnaaz and Jabbar 2016)

- Based on the paper, Intrusion Detection System (IDS) attempts to identify and notify the activities of users as normal or anomaly. IDS is a nonlinear and complicated problem that deals with network traffic data.
- In this research, Intrusion Detection System was developed using Random Forest Classifier. Random Forest is an ensemble classifier and performs very well as compared to other traditional classifiers that are used for classification of attacks.
- The main aim of Intrusion Detection system was to classify various types of attacks such as DOS, Probe, U2R, R2L.
- To evaluate the performance of the model, NSL-KDD dataset was used and the results showed and efficient model with low false alarm rate and high detection rate.
- While constructing individual trees in random forest, randomization was applied to select the best node to split.
- Feature Selection method was carried out and was classified into three categories such as
    1. filter method
    2. wrapper method
    3. embedded method
- For future work, to further improve accuracy applying evolutionary computation as a feature selection measure was proposed.

2.2 Literature Review - Classification Technique

In Machine Learning, Classification is defined as an technique which can be used to identify and group number of observations into different classes. This is done on the basis of the training data set containing instances or observations. For example, diagnosis can be given to a patient based upon their observed health conditions like sex, age, previous health issues, etc. Classification can also be defined as a pattern recognition.
For our research, we will mainly focus upon Supervised Learning in Classification Technique. In the field of machine learning, Classification is considered as an instance of Supervised learning.
Classification is a supervised machine learning approach in which the computer system learns from the input data available and then uses that information to classify new instances or observations.
There are number of algorithms in Classification Technique as seen in figure 1.2 below (Educba 2020)

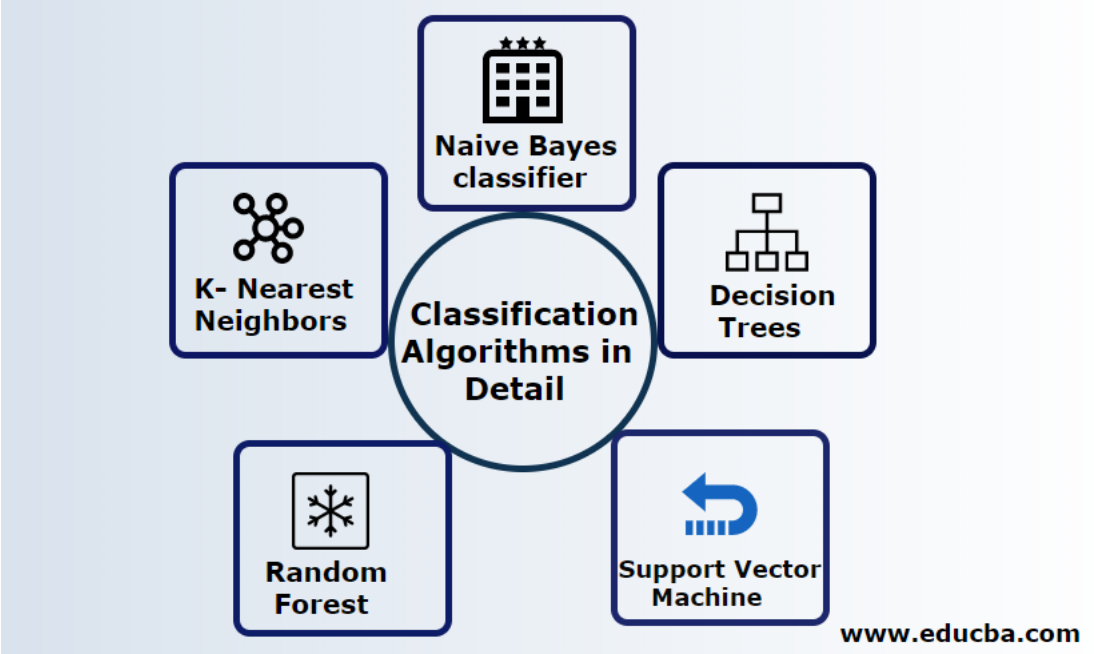


Figure 2.2: Classification Algorithms (Educba 2020)

2.2.1 Naive Bayes Classifier

- Naive Bayes Classifier is a Bayes' theorem based algorithm, it is the simplest instance of a probabilistic classifier. It is one of the statistical classifications and require less amount of training data to estimate the number of parameters.
- Naive Bayes is considered as one of the fastest classifier as its highly scalable and can easily handle discrete as well as continuous data.
- This algorithm is used to make predictions in real-time.
- Naive Bayes is a type of supervised learning algorithm that uses a maximum likelihood method of parameter estimation. (Sangkatsanee, Wattanapongsakorn, and Charnsripinyo 2011)
- There are different type of Naive Bayes Classifiers such as Bernoulli Naive Bayes, Gaussian Naive, Multinomial Naive Bayes.
- Bayesian Classification equation is given as below 2.3

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

Figure 2.3: Bayesian Classification (Educba 2020)

where A,B are events and P(A—B) is the Posterior Probabilities.

- If A & B are independent of each other then,

  $P(A,B)=P(A).P(B)$

- Naive Bayes predictions are independent even though they are used for making recommendation systems. They are widely used in many real-time applications and also in document classification. Naive Bayes can be build using python library.
- Naive Bayes performs very well over the other classification problems, including text categorization, collaborative filtering, email filtering and medical diagnosis. (Bouckaert 2004)
- Naive Bayes require very less computational power hence it can run on any system. It can work accurately while handling large datasets therefore it is also assumed in multi-class prediction problems.
- The main drawback of this algorithms is that they have features that are independent of each other and also the model will assign zero probability.

2.2.2 Decision Tree

- A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements. (Wikipedia 2020a)
- Decision Tree is a supervised machine learning algorithm which is used to perform classification and prediction on a data set.
- A typical decision tree starts with a single node and during processing it generates number of nodes representing all the possible outcomes based on the previous root node. These serves as separate branches of a tree therefore, it is visualised as a tree structure.
- Each of the branches further extends and leads to additional nodes as their outcomes. This process continues until a definite outcome is obtained.
- Decision tree is a top-down approach as the root nodes extends downwards in the form of branches. It has a structure of the flow chart and can handle high dimensional data as well.
- The final outcome of the root node is directly linked to the input variable.
- There are three types of nodes in Decision Tree such as:-
    1. Decision Nodes - represented by squares.
    2. Chance Nodes - represented by circles.
    3. End Nodes - represented by triangles.
- There are some important nomenclature related to Decision Tree such as (Singh 2020a)
    1. **Root Node:**
       It represents the main root of the tree which divides into two or more homogeneous groups.
    2. **Splitting:**
       It is the process of splitting a single node into two or more sub nodes.
    3. **Decision Node:**
       When a sub node extends further into sub nodes it is called as a Decision node.
    4. **Terminal/Leaf Node:**
       The nodes which don't split further and have reached their final level is known as a terminal or leaf node.
    5. **Sub-Tree/Branch:**
       A sub-section generated from the entire decision tree is known as a Sub-Tree or a Branch
    6. **Parent and child Node:**
       The root nodes which are divided into sub-nodes is termed as a parent node of sub-nodes where the sub-nodes are the child of the parent node and called as child node.

- The nomenclature of basic Decision Tree is shown in figure 2.4

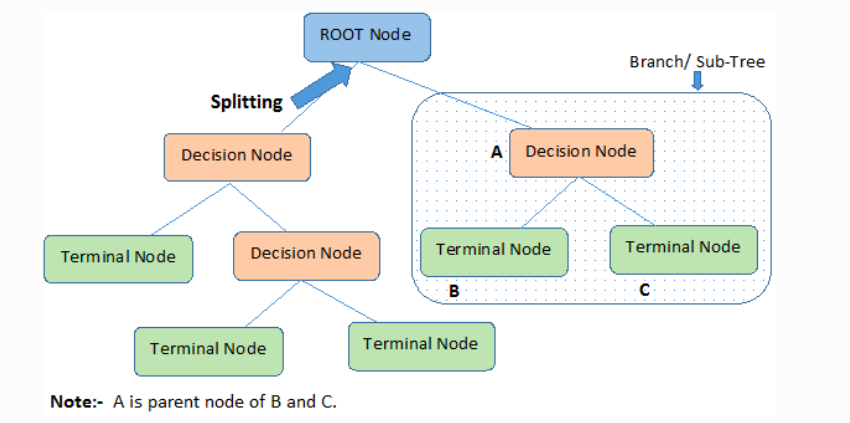


Figure 2.4: Decision Tree Algorithm (Singh 2020a)

- The basic algorithm for decision tree is a top-down recursive divide-and-conquer approach.
  The algorithm is summarized as following (Phyu 2009):

  1. create a root node N;
  2. if all the sample are of same class C then
  3. return N as a terminal node labeled with the class C;
  4. if attribute list is empty then
  5. return N as a terminal node and labelled as the most common class in sample;
  6. select test attribute,the attribute with highest information gain in entire attribute list;
  7. label node N with test attribute for each known value a_i of test attribute;
  8. create a new branch from node N for the given condition of test attribute a_i
  9. let s_i be the sample set for the test attribute a_i
  10. if s_i is empty then attach a terminal labelled node with the most common class in sample set;
  11. else attach the node returned by Generate _Decision_Tree (s_i , attribute-list_test attribute)

- The tree's accuracy depends upon the strategy of making splits for decision. The algorithm selection is also based upon the type of target variables such as: (Singh 2020a)

  – ID3 - extension of D3
  – C4.5 - successor of ID3
  – CART - Classification and Regression Tree
  – CHAID - Chi-square automatic interaction detection Performs multi-level splits when computing classification trees
  – MARS - Multivariate Adaptive Regression Splines

- There are also number of criteria while selecting attributes in building a decision tree such as:
  - Entropy (Sehra 2018)
    - A decision tree is built in top-down sequence from a root node to terminal nodes and involves partitioning the data into subsets that contains similar values.
    - ID3 uses entropy to calculate the homogeneity of a sample.
    - If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has entropy of one.
    - Entropy of a single attribute using the frequency table can be given as:
      $E(S) = \sum_{i=1}^{c} -p_i log_2 p_i$ (Jain 2017)
    - Similarly, Entropy of a two attributes can be given as:
      $E(T, X) = \sum_{c\epsilon,X} P(c)E(c)$ (Jain 2017)
  - Information Gain
    - Information Gain is commonly used while building a decision tree from the available dataset.
    - Highest value of information gain is selected from the variable in order to minimize entropy.
    - We can say that information gain is inversely proportional to entropy.
    - Information Gain is calculated using two variable hence it can be derived from entropy. Therefore Information Gain can be given as :
      $Gain(T, X) = Entropy(T) - Entropy(T, X)$ (Jain 2017)
      In a simpler way, we can conclude that:

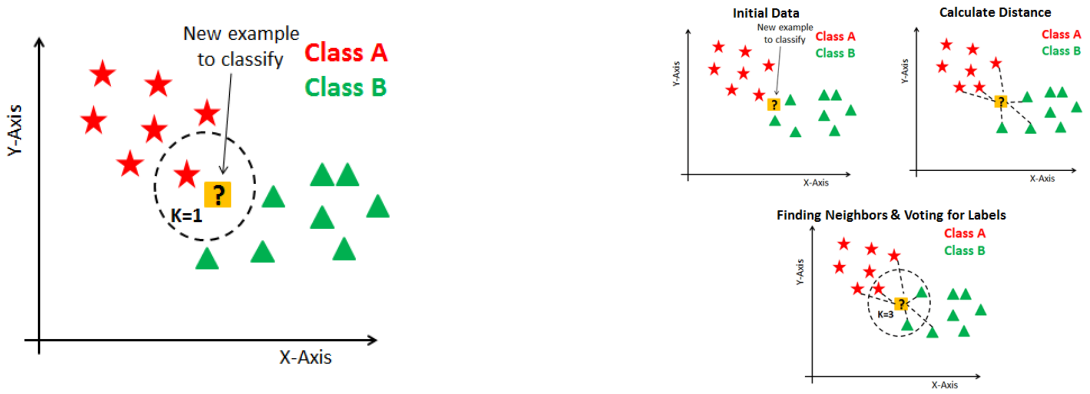
      $InformationGain = Entropy(before) - \sum_{j=1}^{K} Entropy(j, after)$ (Singh 2020a)
  - Gain Ratio (Singh 2020a)
    - Information gain is biased towards choosing attributes with a large number of values as root nodes.
    - It means it prefers the attribute with a large number of distinct values.
    - Gain Ratio can be expressed as:

      $GainRatio = [\frac{InformationGain}{SplitInfo}] = [\frac{Entropy(before) - \sum_{j=1}^{K} Entropy(j,after)}{\sum_{j=1}^{K} w_j log_2 w_j}]$ (Singh 2020a)

2.2.3 K-Nearest Neighbor (KNN)

- K-Nearest Neighbor (KNN) is relatively easy classification technique. KNN classifies new observations into their appropriate categories by searching for closest instances in the training data set.
- Closest instances are defined in terms of distance metric known as Euclidean Distance.
- The Euclidean Distance between any two points such as X_1=$(x_{11}, x_{12}, x_{13}, ..., x_{1n})$ and X_2=$(x_{21}, x_{22}, x_{23}, ..., x_{2n})$, is

  $dist(X_1, X_2) = \sqrt{\sum_{i=1}^{n}(x_{1i} - x_{2i})^2}$ (Adetunmbi et al. 2008)
- KNN comprises of training as well as testing phases just like any other supervised learning technique.
- Flow of KNN Algorithm can be stated as below (Harrison 2018):
  1. Load the data
  2. Initialize K to any chosen number of neighbors
  3. For each example in data
     (a) Calculate the distance between current example and query example from the data
     (b) Add the index and the distance of the example to an ordered collection
  4. Sort the ordered collection of distances and indices in ascending order of their distances
  5. Pick the first K entries from the ascending arranged collection
  6. Get the labels of the selected K entries
  7. If classification, return the mode of the K labels
  8. If regression, return the mean of the K labels
- Working of KNN Algorithm (Navlani 2018):
  - In KNN, K is the number of nearest neighbors. Value of K is generally a odd number if the number of classes is 2.
  - In simple case, when K=1, then the algorithm is known as the nearest neighbor algorithm.



(a) KNN Initialization (b) Finding Neighbors

Figure 2.5: K-Nearest Neighbor Modelling (Navlani 2018)

- The following figures 2.5 and 2.6 shows the highlights about how the algorithm works.

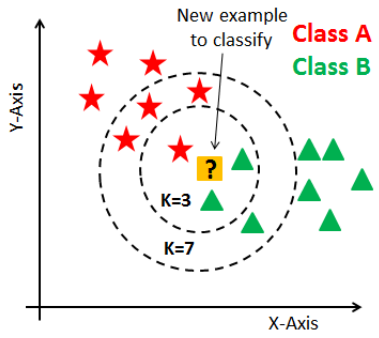


Figure 2.6: KNN Classification (Navlani 2018)

2.2.4 Logistic Regression

- Logistic Regression is also called as a Predictive Learning Model. It is a statistical method for analyzing data which contains one or more independent variables that are used to determine an output. (Sidana 2017)
- The outcome is measured in which there are only two possible outcomes.
- The goal of Logistic Regression is to find the best model and show relationship between a dependent variables and a set of independent variables.
- Logistic Regression is better than few of the binary classification algorithms such as K-Nearest Neighbor algorithm since it uses quantitatively factors to perform classification technique.
- In basic form, Logistic Regression uses a logistic function that models a binary dependent variable.(Wikipedia 2020c)
- Logistic Function transforms the output using the logistic sigmoid function to return a probability value which then can be mapped to two or more discrete classes
- Logistic Regression can help to predict a class for example whether a student has passed or failed in exam.
- There are total three types of Logistic Regression such as :-
    1. Binary (Pass/Fail)
    2. Multi (Cats, Dogs, Sheep)
    3. Ordinal (Low, Medium, High)
- We use sigmoid function to map the predicted values to probability in Logistic Regression.The function helps in mapping any real value into another value anywhere between 0 and 1. Also in machine learning, we use sigmoid to map predictions to probabilities. This map is also called as Logistic Function Curve (ML-Cheatsheet 2017)

- The Sigmoid Function can be given as :-

  $S(z) = \frac{1}{1+e^{-z}}$ (ML-Cheatsheet 2017)

- The Logistic Function Curve is shown in figure 2.7

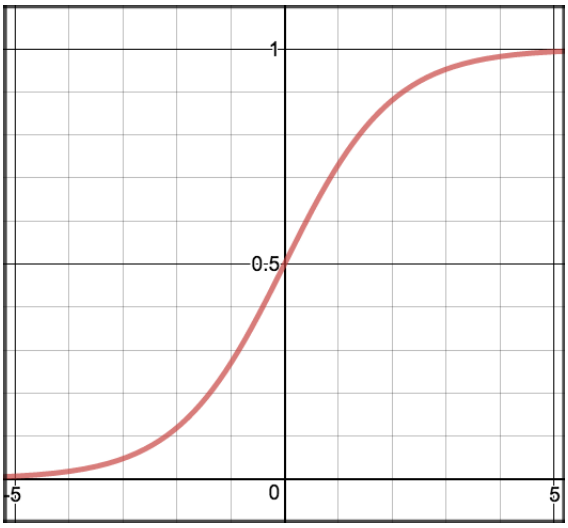


Figure 2.7: Logistic Function Curve (ML-Cheatsheet 2017)

- The Logistic Regression Equation can be given as (Swamy 2019)

  $log(\frac{p_i}{1-p_i}) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 ..... \beta_k X_k$

  Or else the equation can also be given as

  $P = \frac{e^{a+bx}}{1+e^{a+bx}}$

2.2.5 Support Vector Machine

- Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification as well as regression problems.
- SVM's are more commonly used for classification problems rather than regression.
- SVM are based on the idea of searching for a hyperplane that divides a dataset precisely into two classes as shown in figure 2.8

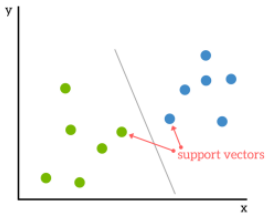


Figure 2.8: Support Vector (Bambrick 2016)

- Support Vectors are the data points that are nearest to the hyperplane. In simple terms, a hyperplane is a line that divides data points linearly and classifies a set of data.
- In order to correctly classify the data we need to have a clear 3D view of the hyperplane.
  A simple comparison of 2D and 3D hyperplane is shown in figure 2.9

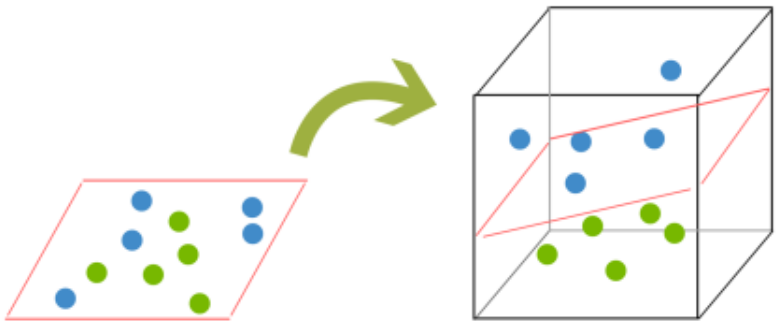


Figure 2.9: 2D to 3D Hyperplane (Bambrick 2016)

- There are number of Pros and Cons of SVM such as (Bambrick 2017)
    1. Pros:
        (a) Works really well when it has clear margin of separation
        (b) Very effective in high dimensional spaces
        (c) Effective in cases where number of dimensions are greater than the number of samples
        (d) It is memory efficient as it uses subset of training points in the decision function also called as support vectors
    2. Cons:
        (a) Since the training time is much long therefore it cannot be used for working on large datasets
        (b) It doesn't perform well when the dataset has more noise i.e. if classes are overlapping
        (c) SVM doesn't directly provide probability estimates, these are calculated using an expensive five-fold cross-validation

2.2.6 Random Forest

- Random Forest is an ensemble learning method that is used for classification and regression as well as other related problems. It is operated by constructing a collection of decision trees at training time and outputting the mode of classes or mean prediction of the classes of the individual tree.(Singh 2020b)
- Random Forest is an ensemble model made of many decision trees using bootstrapping, random subsets of features, and average voting to make predictions. This is an example of a bagging ensemble (Will_Koehrsen 2018)
- If the model is over fitting in Decision tree for their training dataset then Random Forest is used to overcome this problem. Basically, Random Forest is built up of multiple decision trees.
- The model uses two key concepts in the process such as (Will_Koehrsen 2018)
  1. Random sampling of training data points when building trees
  2. Random subsets of features considered when splitting nodes
- Random forest ensures that the behavior of each individual tree is not too correlated with the behavior of any other tree in the model by using the following two methods (Singh 2020b)
  1. Bagging or Bootstrap Aggregation
  2. Random Feature Selection
- For classification problems, Random Forest's mathematical equation can be given as
  $Gini = 1 - \sum_{i=1}^{c}(p_i)^2$

  Whereas, for regression it is given as
  $MSE = \frac{1}{N}\sum_{i=1}^{N}(f_i - y_i)^2$
- Pros and cons (Schott 2019)
  - Pros:
    - Used for regression and classification problems, making it a diverse model
    - Prevents over fitting of data
    - Fast to train with test data
  - Cons
    - Slow in creating predictions once model is made
    - Must beware of outliers and holes in the data

2.2.7 SMOTE

- SMOTE stands for Synthetic Minority Oversampling Technique. Smote is a statistical technique for increasing the number of cases in our dataset in a balanced way.
- The module works by generating new instances from existing minority cases that you supply as input. This implementation of SMOTE does not change the number of majority cases.
- It aims to balance class distribution by randomly increasing minority class examples by replicating them. It generates the virtual training records by linear interpolation for the minority class.
- These synthetic training records are generated by randomly selecting one or more of the k-nearest neighbors for each example in the minority class. After the oversampling process, the data is reconstructed and several classification models can be applied for the processed data.
- For each instance x_i in minority class, SMOTE searches its k nearest neighbors and one neighbor is randomly selected as x'. Then a random number between [0,1] δ is generated. The new artificial sample x_{new} is created as

  $x_{new} = x_i + (x' - x_i) * \delta$ (Zheng, Cai, and Y. Li 2016)

- Advantages:
  1. Mitigates the problem of over fitting caused by random oversampling as synthetic examples are generated rather than replication of instances
  2. No loss of useful information
- Disadvantages:
  1. While generating synthetic examples SMOTE does not take into consideration neighboring examples from other classes. This can result in increase in overlapping of classes and can introduce additional noise
  2. SMOTE is not very effective for high dimensional data

2.3 Deep Learning Approach

- Deep Learning (DL) is a sub-field of machine learning concerned with algorithms inspired by the structure and function of the brain called Artificial Neural Networks (ANN). (Brownlee 2019)
- As said above, Deep learning algorithm would perform a task repeatedly, each time modifying it a little to improve its outcome every time. We refer to 'deep learning' because the neural networks have various (deep) layers that enable learning. (Marr 2018)
- Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. (LeCun, Bengio, and Hinton 2015)
- There are total two types of Neural Networks in Deep Learning such as (Wikipedia 2020b)
    1. Artificial Neural Networks (ANN)
    2. Deep Neural Networks

2.3.1 Artificial Neural Networks (ANN)

- An Artificial Neural Network is an information processing technique. It works similarly how a human brain processes the information. ANN includes a large number of connected processing unit that work in conjunction to process information. (Sharma 2017)
- Artificial Neural Network (ANN) is typically organized in number of layers. Layers are made up of number of interconnected nodes which contains an 'activation function'.
- A neural network contain total 3 layers:(Sharma 2017)
    1. **Input Layer**
       The purpose of input layer is to receive input values of the explanatory attributes for each observation.
       Usually, the number of input layers are equal to the number of explanatory variables.
       The nodes of the input layer are passive i.e. they don't change the data.
    2. **Hidden Layer**
       The hidden layers gives transformation to the input values inside the network. It connects with outgoing arcs to output nodes or to other hidden layers.
       The actual processing is done via a system of weighted 'connections' is done in hidden layer.
    3. **Output Layer**
       The hidden layers are linked to the output layers. Output layer receives connections from hidden layers or from input layer.
       The active nodes of the output layer combine and change the data to produce the output values.

- The structure of deep learning is described in figure 2.10 Deep learning methods which are inspired by the structure depth of human brain learn from lower level characteristic to higher levels concept.

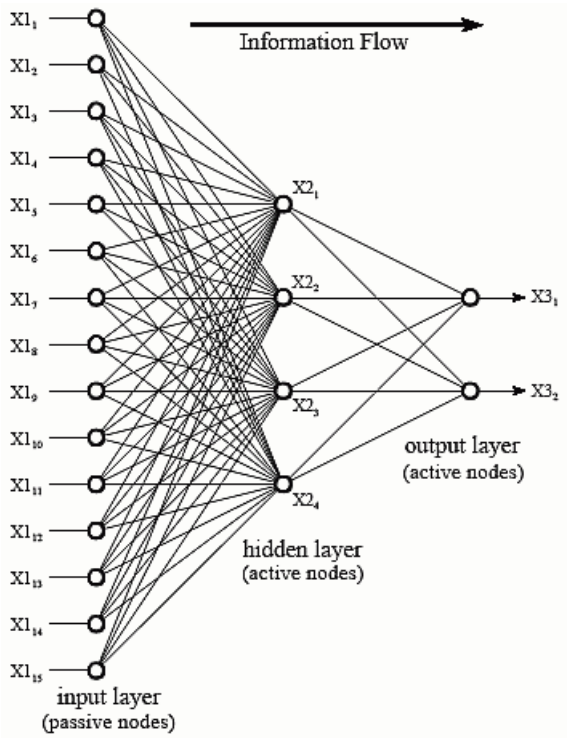


Figure 2.10: Structure of Deep Learning (Dong and Wang 2016)

- Backpropagation is one of the most commonly used supervised artificial neural network algorithm. Backpropagation Figure 2.11 aims to train the network to achieve a balance between the ability to respond correctly to the input patterns that are supplied for training the network and the ability to give reasonable responses to input that is similar to that used in training. (Hussain, Lalmuanawma, and Chhakchhuak 2016)

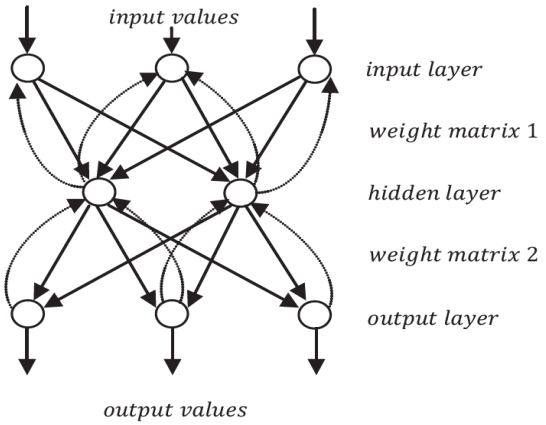


Figure 2.11: Backpropagation Neural Network (Hussain, Lalmuanawma, and Chhakchhuak 2016)

- The training of a network by backpropagation involves three stages: The feed-forward of the input training pattern, the calculation and backpropagation of the associated error and the tuning of the weights so that the forward pass produces an output vector for a given input vector based on the current state of the network weights. (Hussain, Lalmuanawma, and Chhakchhuak 2016)
- Framework of Artificial Neural Network (Srivastava 2014)

    1. Assign random weights to all the linkages to start the algorithm
    2. Using the inputs and the (Input to Hidden node) linkages find the activation rate of Hidden Nodes
    3. Using the activation rate of hidden node and linkages to output, find the activation rate of output nodes
    4. Find the error rate at the output node and recalibrate all the linkages between hidden nodes and output nodes
    5. Using the weights and error found at output node, cascade down the error to hidden nodes
    6. Recalibrate the weights between hidden node and the input node
    7. Repeat the process till the convergence criterion is met
    8. Using the final linkage weights score the activation rate of the output nodes

Chapter 3

Research Methodology

Methodology and Research Method

There are number of steps involved in performing a machine learning technique based on classification method using a dataset.
The basic steps in involved in performing a machine learning process are shown in figure 3.1

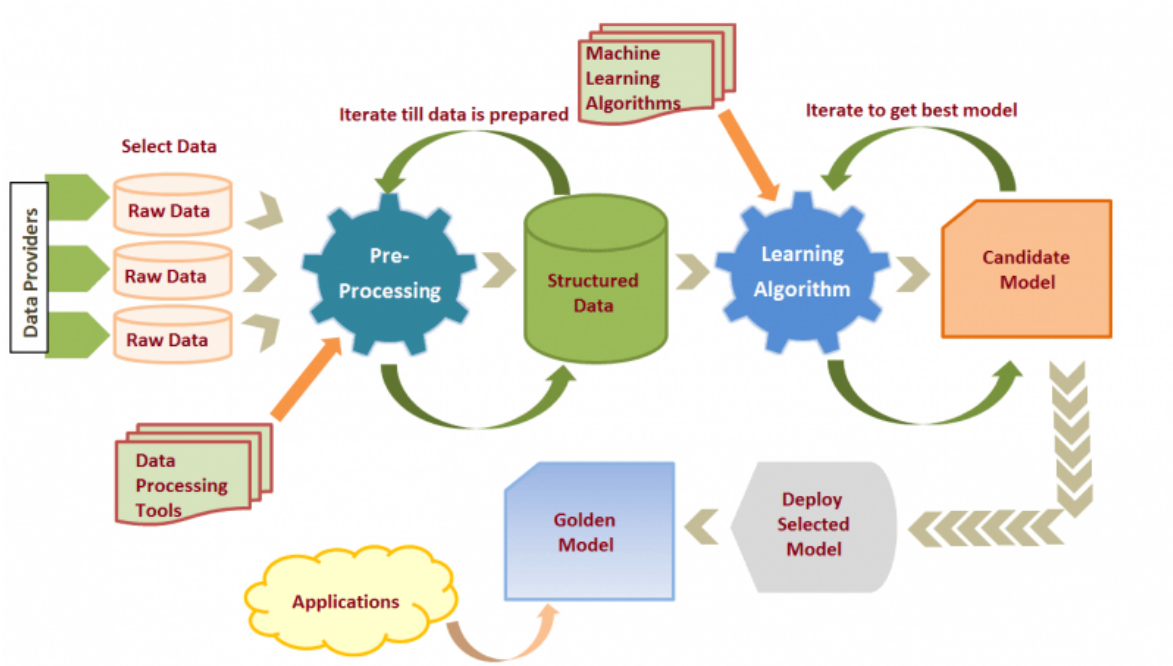


Figure 3.1: Steps of Machine Learning (Mittal 2017)

The steps involved in building a machine learning model are stated as below:

3.1 Data Description

- The main aim of this research is to make a comparison between number of classification techniques and apply on the dataset to predict which model gives us the highest accuracy and precision with less false positive rate.
- For this research on Network Intrusion Detection System (NIDS) we have selected a dataset based on intrusion system.
- The dataset is created and audited which contains wide range of intrusion simulated in a military network environment to acquire raw TCP/IP dump data for a network simulating a US AIR Force LAN network.
- A connection in a network is a sequence of TCP packets which starts and ends at some particular time duration during which the data flows from a source IP address to the target IP address where each connection is labelled as either normal or as an attack with exactly one specific attack type.
- The selected dataset has been divided into two sets, one for training while other for testing purpose. Each dataset contains total 42 and 41 features respectively where each serves a different purpose in processing.
- The dataset classifies the amount of attacks into four different categories such as (Hussain, Lalmuanawma, and Chhakchhuak 2016)

    1. **Denial of Service (DOS):**
       A DoS attack is a type of attack in which the intruder restricts access to the authorized person into the system or network.
    2. **Remote to Local (R2L):**
       It is a type of attack that aims at gaining access to a local account from another host or network.
    3. **User to Root (U2R):**
       These attacks are exploitation in which the intruder starts off on the system with a limited user account or normal user privileges and attempts to abuse vulnerabilities in the system to gain root access.
    4. **Probe:**
       Probe is an attack in which the hacker scans a machine or a network to gather information or find known vulnerabilities

3.2 Data Preprocessing

In the field of machine learning, Data Preprocessing is the first step to be carried out before initializing the process. Data Preprocessing is basically used to transform and convert raw and unfiltered data into a much more suitable and understandable format.
Typically, real world data might be incomplete, inconsistent, inaccurate, unstructured form and might also have some errors and missing values. To overcome all this, data preprocessing is used. It helps in clean, format and organize the raw data and make it ready to use in building machine learning model.

Data Preprocessing cannot be conducted into a single process hence it is distributed among several steps.
The steps involved in Data Preprocessing are as below:(Dey 2018)

1. **Import Libraries**

    - Before we start any programming, our first step is importing various libraries required to build a model.
    - A library is primarily a collection of modules that can be called and used. Libraries is a collection of functions and methods which allows us to perform many actions without actually writing any code.

```
In [1]:
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from math import * # module math
import matplotlib.pyplot as plt # visualization
from PIL import Image
import seaborn as sns # visualization
import itertools
import io
import plotly.offline as py # visualization
py.init_notebook_mode(connected=True) # visualization
import plotly.graph_objs as go # visualization
from plotly.subplots import make_subplots
import plotly.figure_factory as ff # visualization
import warnings
warnings.filterwarnings("ignore")

%matplotlib inline
```

Figure 3.2: Importing Libraries for Python

    - From the figure 3.2, we have installed number of libraries such as *numpy*, *pandas*, *math*, *etc.* This libraries plays a vital role in performing any program. For example, pandas is used for data processing, numpy is used to perform linear algebra while *plotly* is used to make visualization in the program.

2. **Import the Dataset**

- A lot of datasets are in the form of CSV formats. CSV is a comma-separated-values file which allows data to be stored in tabular format.
- For our research we have selected dataset based on Network Intrusion Detection system. As we will be performing many classification techniques on the dataset therefore we have divided our dataset into two subsets for training as well as testing the data.

```
In [3]: Training = pd.read_csv(r"C:\Users\HP\Downloads\Dissertation Files\Train_data.csv")
        Testing = pd.read_csv(r"C:\Users\HP\Downloads\Dissertation Files\Test_data.csv")

In [4]: print(Training.head(5))

        print("Training data has {} rows & {} columns".format(Training.shape[0],Training.shape[1]))
```

```
   duration protocol_type   service flag  src_bytes  dst_bytes  land  \
0         0           tcp  ftp_data   SF        491          0     0
1         0           udp     other   SF        146          0     0
2         0           tcp   private   S0          0          0     0
3         0           tcp      http   SF        232       8153     0
4         0           tcp      http   SF        199        420     0

   wrong_fragment  urgent  hot  num_failed_logins  logged_in  num_compromised  \
0               0       0    0                  0          0                0
1               0       0    0                  0          0                0
2               0       0    0                  0          0                0
3               0       0    0                  0          1                0
4               0       0    0                  0          1                0

   root_shell  su_attempted  num_root  num_file_creations  num_shells  \
0           0             0         0                   0           0
1           0             0         0                   0           0
2           0             0         0                   0           0
3           0             0         0                   0           0
4           0             0         0                   0           0
```

Figure 3.3: Importing Datasets

- As seen in the figure 3.3, we have imported two different datasets and labelled them as **Training** and **Testing** and read it using a method called *read_csv*
- To check if the dataset has been correctly imported, we will use the *print* command to check the dataset and display overview of columns from the dataset.

3. **Checking Missing Values**

- Sometimes we may find dataset which have missing values or data in the dataset. There might be many reason for this such as programming error or the data might be lost transferring manually from legacy dataset.
- To fill the missing values we can simply delete the entire row or replace the missing value with some sensible data.

```
In [6]:
def dataoveriew(df, message):
    print(f'{message}:\n')
    print("Rows:", df.shape[0])
    print("\nNumber of features:", df.shape[1])
    print("\nFeatures:")
    print(Training.columns.tolist())
    print("\nMissing values:", df.isnull().sum().values.sum())
    print("\nUnique values:")
    print(df.nunique())
```

```
In [7]:
dataoveriew(Training, 'Overiew of the Training dataset')
```

```
Overiew of the Training dataset:

Rows: 25192

Number of features: 42

Features:
['duration', 'protocol_type', 'service', 'flag', 'src_bytes', 'dst_bytes', 'land', 'wrong_fragment', 'urgent', 'hot', 'num_f
ailed_logins', '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_hos
t_srv_count', 'dst_host_same_srv_rate', 'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate', 'dst_host_srv_diff_host_rat
e', 'dst_host_serror_rate', 'dst_host_srv_serror_rate', 'dst_host_rerror_rate', 'dst_host_srv_rerror_rate', 'class']

Missing values: 0

Unique values:
duration                     758
protocol_type                  3
service                       66
flag                          11
src_bytes                   1665
```

Figure 3.4: Finding Missing Values

- We have selected the **Training** dataset and checked if there's any missing value in the data. To check missing values we have elected to print overview of the entire Training dataset which shows us the results as total number of rows and number of columns i.e. number of features. and also if any missing values which in our case is 0 as we can see from figure 3.4

4. **Encoding categorical data**

- Sometimes our data can be in qualitative form, i.e. we have texts as in our data. Our computer system cannot understand data in text format and therefore cannot process it.
- Since our models and systems are based on mathematical equations and calculations therefore it is necessary to convert categorical data into numerical data format.
- We have converted the data into numerical format using *sklearn* library and used it *LabelEncoder* to conduct the process.
  To further enhance and simplify our data, we have assigned default value of mean as 0 and variance as 1 to all the numerical attributes in our dataset as we can see in figure 3.5

In [12]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

# extract numerical attributes and scale it to have zero mean and unit variance
cols = Training.select_dtypes(include=['float64','int64']).columns
sc_train = scaler.fit_transform(Training.select_dtypes(include=['float64','int64']))
sc_test = scaler.fit_transform(Testing.select_dtypes(include=['float64','int64']))

# turn the result back to a dataframe
sc_traindf = pd.DataFrame(sc_train, columns = cols)
sc_testdf = pd.DataFrame(sc_test, columns = cols)
```

In [13]:
```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()

# extract categorical attributes from both training and test sets
cattrain = Training.select_dtypes(include=['object']).copy()
cattest = Testing.select_dtypes(include=['object']).copy()

# encode the categorical attributes
traincat = cattrain.apply(encoder.fit_transform)
testcat = cattest.apply(encoder.fit_transform)

# separate target column from encoded data
enctrain = traincat.drop(['class'], axis=1)
cat_Ytrain = traincat[['class']].copy()
```

Figure 3.5: Encoding categorical data

5. **Splitting the Dataset into Training set and Test Set**

   - After converting the data we need to split the dataset into two sets as a Training set and a Testing set
   - We will train our machine learning model on our training set. Our learning model will try to understand any correlations in our training set and then test the model on our testing set to check how accurately it can predict.
   - Generally, the dataset is split into 80% and 20% for training set and testing set respectively.

```
In [18]: from sklearn.model_selection import train_test_split

         X_train,X_test,Y_train,Y_test = train_test_split(Training_x,Training_y,train_size=0.70, random_state=2)
```

Figure 3.6: Splitting the Dataset

   - We can do this by importing *train_test_split* from *model_selection* library of scikit as seen in figure 3.6
   - While splitting a dataset, we will create 4 sets as:
     - X_train - Training part of the matrix of features
     - X_test - Test part of the matrix of features
     - Y_train - Training part of the dependent variables associated with the X train sets
     - Y_test - Test part of the dependent variables associated with the X test sets

6. **Feature Scaling**

   - In machine learning, Feature Scaling means scaling the features to the same scale.
   - Normalization scale features between 0 and 1, also retaining their proportional range to each other.
   - Standardization scales features to have a mean(u) of 0 and standard deviation (a) of 1

     $$X' = \frac{x-u}{a}$$

```
In [12]: from sklearn.preprocessing import StandardScaler
         scaler = StandardScaler()

         # extract numerical attributes and scale it to have zero mean and unit variance
         cols = Training.select_dtypes(include=['float64','int64']).columns
         sc_train = scaler.fit_transform(Training.select_dtypes(include=['float64','int64']))
         sc_test = scaler.fit_transform(Testing.select_dtypes(include=['float64','int64']))

         # turn the result back to a dataframe
         sc_traindf = pd.DataFrame(sc_train, columns = cols)
         sc_testdf = pd.DataFrame(sc_test, columns = cols)
```

Figure 3.7: Feature Scaler

   - We can do this by importing *StandardScaler* from library of scikit as seen in figure 3.7. Also, we have previously stated few details regarding Feature Scaler above section 4

3.3 Tools used in Machine Learning

Every program needs some kind of tools for processing whether it is for classification or for regression or any other technique. Similarly, in our project, to carry out various classification technique s in machine learning we also require certain tools.
Tools are nothing but software_applications and hardware support that required to accomplish and perform classification algorithms.
Tools required for building a classification model in machine learning are as follows:

3.3.1 Jupyter Notebook

- JupyterLab is a web-based interactive development environment for Jupyter notebooks, code, and data. 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.(Jupyter 2020)

- Jupyter Notebook is built off of IPython, an interactive way of running Python code in the terminal using the REPL model (Read-Eval-Print-Loop) Jupyter Notebook can be installed either by using Anaconda or by Pip.

3.3.2 Python

- To start building our model, we need to select a programming language on which we can start our coding. For classification technique we have used **Python** as our primary programming language.

- Python is an interpreted, high-level, general-purpose programming language. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.(Wikipedia 2020d) Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management.

- The language's core philosophy is summarized in the document The Zen of Python, which includes aphorisms such as: (Wikipedia 2020e)

    1. Beautiful is better than ugly.
    2. Explicit is better than implicit.
    3. Simple is better than complex.
    4. Complex is better than complicated.
    5. Readability counts.

- Python was designed for readability, and has some similarities to the English language with influence from mathematics. Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.(W3Schools 2020)

- Features of Python Programming Language (Solutions 2017)
    1. Readable and Maintainable Code
    2. Multiple Programming Paradigms
    3. Compatible with Major Platforms and Systems
    4. Robust Standard Library
    5. Many Open Source Frameworks and Tools
    6. Simplify Complex Software Development
    7. Adopt Test Driven Development
- There are wide array of application on which Python can be used. Some of them are listed below (W3Schools 2020)
    1. Used on a server to create web applications.
    2. Used alongside softwares to create workflows.
    3. Python can easily connect to database systems. It can also read and modify files.
    4. Python can be used to handle big data and perform complex mathematics.
    5. Python can be used for rapid prototyping, or for production-ready software development.

Chapter 4

Implementation and Results

4.1 Data Preprocessing

- 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.
- Typically, real world data might be incomplete, inconsistent, inaccurate, unstructured form and might also have some errors and missing values. To overcome all this, data preprocessing is used. It helps in clean, format and organize the raw data and make it ready to use in building machine learning model.
- There are number of steps that are collectively involved in conducting Data Preprocessing. Steps involved in Data Preprocessing are as follows:

  1. Importing Libraries
  2. Importing Dataset
  3. Checking any missing values
  4. Encoding Categorical Data
  5. Splitting the Dataset into Training set and Test Set
  6. Feature Scaling

  All the steps in Data Preprocessing are already been explained in detail in section 3.2 of the previous chapter.

4.2 NIDS Implementation

- As discussed earlier in the previous sections, the dataset contains various types of variables with different values. We will first pre-process the dataset before applying different classification techniques on the data.
- All the nominal data attributes are converted into discrete attributes for the processing. In addition, there is one attribute, *num_outbound_cmds*, in the dataset whose value is always 0 for all the records in the training and test data. (Javaid et al. 2016)
- In the implementation process, both the labelled and unlabelled data for feature learning and classifier training are obtained from the same source. Figure 4.1 shows the steps involved in NIDS Implementation

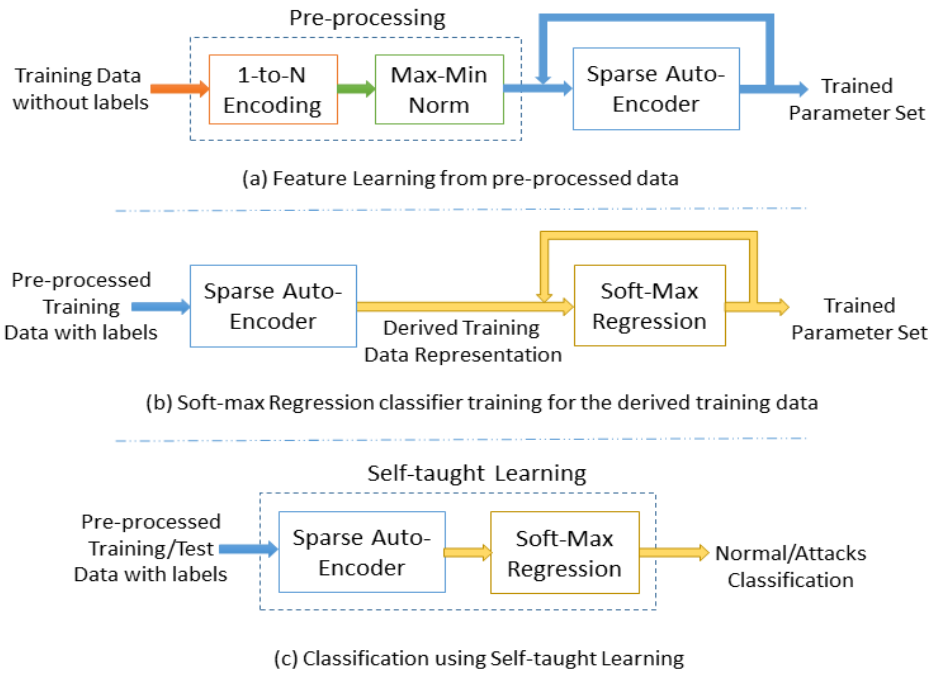


Figure 4.1: Steps in NIDS Implementation using STL

4.3 Accuracy Metrics

- We evaluate the performance of our models based on the Accuracy Metrics and is also known as a Confusion Matrix.
- With the help of Confusion matrix we can easily find out our model's Accuracy, Precision, Recall and F-Measure.(Javaid et al. 2016)

Table 4.1: Confusion Matrix

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

- All the above metrics are given as:
  1. **Accuracy**
     - It is defined as the percentage of correctly classifying the records over the total number of records.
       $Accuracy = \frac{TP+TN}{(TP+FP+FN+TN)}$
  2. **Precision (P)**
     - Defined as the % ratio of the number of True Positives (TP) records divided by the sum of True Positives (TP) and False Positives (FP) classified records.
       $P = \frac{TP}{(TP+FP)} * 100\%$
  3. **Recall (R)**
     - Defined as the % ratio of number of True Positives records divided by the sum of True Positives and False Negatives (FN) classified records.
       $R = \frac{TP}{(TP+FN)} * 100\%$
  4. **F-Measure (F)**
     - Defined as the harmonic mean of Precision and Recall and represents a balance between them.
       $F = \frac{2.P.R}{(P+R)}$

4.4 Feature Selection

- Feature Selection is the process where we can automatically or manually select features from the dataset which can contribute the most for our prediction variable or output in which we are interested in.
- We have done feature selection using Random Forest Classifier. We imported $RandomForestClassifier$ library form Scikit.
- Random Forest is used in conjunction with Feature selection because it is a tree based strategy which ranks and selected the features based on the purity of the node which in our case is $SourceBytes$ stated as src_bytes as seen in figure 4.2.

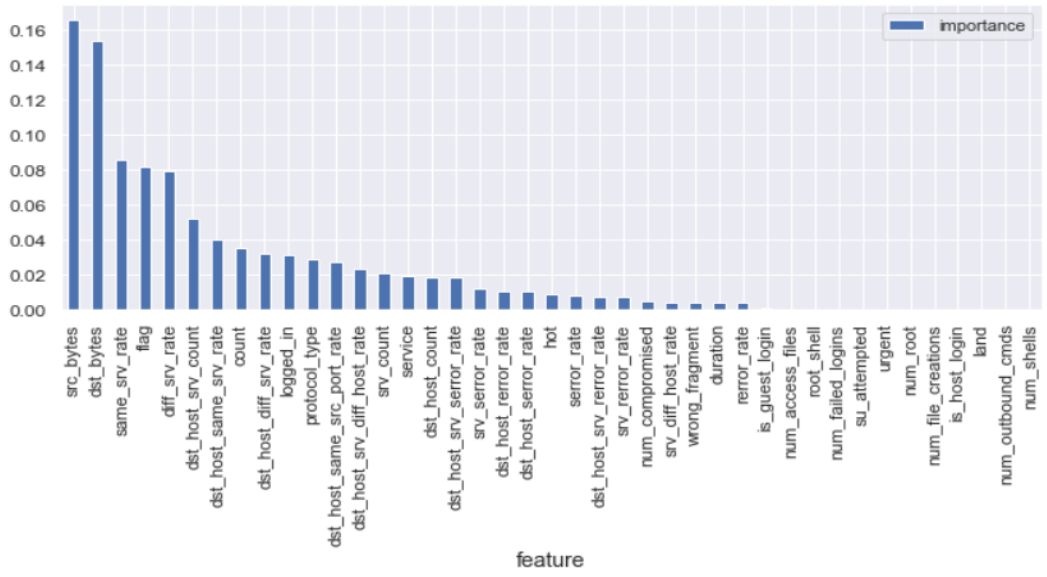


Figure 4.2: Feature Selection

4.5 Recursive Feature Elimination

- Recursive Feature Elimination (RFE) is a feature selection method that fits into a model and is used to eliminate all the non essential features that are present or until the specified number of features are reached in the dataset.

```
In [17]:
from sklearn.feature_selection import RFE
import itertools
rfc = RandomForestClassifier()

# create the RFE model and select 10 attributes
rfe = RFE(rfc, n_features_to_select=15)
rfe = rfe.fit(Training_x, Training_y)

# summarize the selection of the attributes
feature_map = [(i, v) for i, v in itertools.zip_longest(rfe.get_support(), Training_x.columns)]
selected_features = [v for i, v in feature_map if i==True]

selected_features

Out[17]: ['src_bytes',
 'dst_bytes',
 'logged_in',
 'count',
 'srv_count',
 'same_srv_rate',
 'diff_srv_rate',
 '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',
 'protocol_type',
 'service',
 'flag']
```

Figure 4.3: Recursive Feature Elimination

- From the figure 4.3, we have selected few of the features such as *src_bytes*, *dst_bytes*, *count*, *service*, *flag*, etc.

4.6 Fitting Models

- Model fitting is a measure of how well a machine learning model generalizes to similar data to that on which it is trained. The model which is well-fitted generally produces more accurate results.

In [19]:
```
from sklearn.svm import SVC
from sklearn.naive_bayes import BernoulliNB
from sklearn import tree
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression

# Train KNeighborsClassifier Model
KNN_Classifier = KNeighborsClassifier(n_jobs=-1)
KNN_Classifier.fit(X_train, Y_train);

# Train LogisticRegression Model
LGR_Classifier = LogisticRegression(n_jobs=-1, random_state=0)
LGR_Classifier.fit(X_train, Y_train);

# Train Gaussian Naive Baye Model
BNB_Classifier = BernoulliNB()
BNB_Classifier.fit(X_train, Y_train)

# Train Decision Tree Model
DTC_Classifier = tree.DecisionTreeClassifier(criterion='entropy', random_state=0)
DTC_Classifier.fit(X_train, Y_train)
```

Out[19]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=0, splitter='best')
```

Figure 4.4: Model Fitting

- Model fitting is essential part of machine learning. If our model doesn't fit our dataset correctly then the outcome preoduces might not be accurate and can't rely on the outcome to be useful enough for prediction.

4.7 Model Evaluation

Model Evaluation is an integral part of model development process in machine learning. It helps to find the best model that represents our dataset and also how well the chosen model will work in near future.
For our dataset we have selected four classification techniques such as:

4.7.1 Decision Tree

```
============================== Decision Tree Classifier Model Evaluation ==============================

Cross Validation Mean Score:
 0.9956333771928133

Model Accuracy:
 1.0

Confusion matrix:
 [[8245    0]
 [   0 9389]]

Classification report:
              precision    recall  f1-score   support

     anomaly       1.00      1.00      1.00      8245
      normal       1.00      1.00      1.00      9389

    accuracy                           1.00     17634
   macro avg       1.00      1.00      1.00     17634
weighted avg       1.00      1.00      1.00     17634
```

Figure 4.5: Model Evaluation - Decision Tree

Based on the results produced using Decision Tree in Model Evaluation, the model showed an accuracy of 100% and with Cross Validation Mean Score of 0.99. All the accuracy metrics such as Accuracy, Precision, Recall and F-1 score were highest among all other classification techniques used in the procedure.

4.7.2 K-Nearest Neighbor Classifier

```
============================== KNeighborsClassifier Model Evaluation ==============================

Cross Validation Mean Score:
 0.9914370153431007

Model Accuracy:
 0.9937620505840989

Confusion matrix:
 [[8168   77]
 [  33 9356]]

Classification report:
              precision    recall  f1-score   support

     anomaly       1.00      0.99      0.99      8245
      normal       0.99      1.00      0.99      9389

    accuracy                           0.99     17634
   macro avg       0.99      0.99      0.99     17634
weighted avg       0.99      0.99      0.99     17634
```

Figure 4.6: Model Evaluation - K-Nearest Neighbor Classifier

Performing K-Nearest Neighbor (KNN) showed result with model accuracy of 99.37% along with Cross Validation Mean Score of 0.99. The results were also high in terms of anomaly precision and normal network recall.

4.7.3 Logistic Regression

```
============================== LogisticRegression Model Evaluation ==============================

Cross Validation Mean Score:
 0.9537821405741347

Model Accuracy:
 0.9549166383123512

Confusion matrix:
 [[7763  482]
 [ 313 9076]]

Classification report:
              precision    recall  f1-score   support

     anomaly       0.96      0.94      0.95      8245
      normal       0.95      0.97      0.96      9389

    accuracy                           0.95     17634
   macro avg       0.96      0.95      0.95     17634
weighted avg       0.96      0.95      0.95     17634
```

Figure 4.7: Model Evaluation - Logistic Regression

Results of Logistic Regression was achieved with model accuracy of 95.49% and Cross Validation Mean Score of 0.95

4.7.4 Naive Bayes

```
============================== Naive Baye Classifier Model Evaluation ==============================

Cross Validation Mean Score:
 0.9071666840303904

Model Accuracy:
 0.9071679709651809

Confusion matrix:
 [[7000 1245]
 [ 392 8997]]

Classification report:
               precision    recall  f1-score   support

      anomaly       0.95      0.85      0.90      8245
       normal       0.88      0.96      0.92      9389

    accuracy                           0.91     17634
   macro avg       0.91      0.90      0.91     17634
weighted avg       0.91      0.91      0.91     17634
```

Figure 4.8: Model Evaluation - Naive Bayes

Naive Bayes results had the least model accuracy and cross validation mean score. Model Accuracy of 90.71% and Cross Validation Mean Score of 0.90 was obtained.

4.8 Model Validation

In machine learning, Model Validation is referred as a process in which a trained model is evaluated with the testing data set. The testing data set is a separate part of the same dataset from which he training dataset is derived.
Model Validation is defined as the set of processes and activities that are calculated to verify that the models that are executed are performing as expected with their business motive.
Similar to model evaluation, We have performed Model Validation as well on all the four previous models.

4.8.1 Decision Tree

```
============================== Decision Tree Classifier Model Test Results ==============================

Model Accuracy:
 0.9949722148716592

Confusion matrix:
 [[3482   16]
 [  22 4038]]

Classification report:
              precision    recall  f1-score   support

     anomaly       0.99      1.00      0.99      3498
      normal       1.00      0.99      1.00      4060

    accuracy                           0.99      7558
   macro avg       0.99      1.00      0.99      7558
weighted avg       0.99      0.99      0.99      7558
```

Figure 4.9: Model Validation - Decision Tree

Based on the results obtained from model evaluation as well as model validation, in both the case Decision Tree model had the best model accuracy. In model validation, Decision Tree has the model accuracy of 99.49%

4.8.2 K-Nearest Neighbor Classifier

```
============================== KNeighborsClassifier Model Test Results ==============================

Model Accuracy:
 0.9916644614977508

Confusion matrix:
 [[3458   40]
 [  23 4037]]

Classification report:
              precision    recall  f1-score   support

     anomaly       0.99      0.99      0.99      3498
      normal       0.99      0.99      0.99      4060

    accuracy                           0.99      7558
   macro avg       0.99      0.99      0.99      7558
weighted avg       0.99      0.99      0.99      7558
```

Figure 4.10: Model Validation - K-Nearest Neighbor Classifier

Model Validation results was quite similar to that of Model Evaluation in case of K-Nearest Neighbor Classifier with model accuracy of 99.16%

4.8.3 Logistic Regression

```
============================== LogisticRegression Model Test Results ==============================

Model Accuracy:
 0.9554114845197142

Confusion matrix:
 [[3297  201]
 [ 136 3924]]

Classification report:
              precision    recall  f1-score   support

     anomaly       0.96      0.94      0.95      3498
      normal       0.95      0.97      0.96      4060

    accuracy                           0.96      7558
   macro avg       0.96      0.95      0.96      7558
weighted avg       0.96      0.96      0.96      7558
```

Figure 4.11: Model Validation - Logistic Regression

Results of Logistic Regression was same in case of Model Validation as well as Model Evaluation with model accuracy of 95%

4.8.4 Naive Bayes

```
============================== Naive Baye Classifier Model Test Results ==============================

Model Accuracy:
 0.906721354855782

Confusion matrix:
 [[2981  517]
 [ 188 3872]]

Classification report:
               precision    recall  f1-score   support

      anomaly       0.94      0.85      0.89      3498
       normal       0.88      0.95      0.92      4060

     accuracy                           0.91      7558
    macro avg       0.91      0.90      0.91      7558
 weighted avg       0.91      0.91      0.91      7558
```

Figure 4.12: Model Validation - Naive Bayes

In both the cases, Naive Bayes had the least accuracy where in both model validation and model evaluation has model accuracy of 90%. Naive Bayes has the least model accuracy as compared to other classification techniques.

4.9 Deep Learning

For Deep Leaning approach we have used Artificial Neural Network (ANN) algorithm to check the prediction of our dataset in Network Intrusion Detection.
In performing ANN, Data Preprocessing method is similar to rest of the classification technique which are explained in detail in section 3.2 The next steps involved in ANN and their results are stated as below:

1. **Importing Keras**

    - Keras is a neural network library used in python.
    - We will define the input layer, hidden layer and output layer as seen in figure 4.13 for performing ANN.

In [9]:
```
from sklearn.model_selection import train_test_split

X_train,X_test,Y_train,Y_test = train_test_split(train_x,train_y,train_size=0.70, random_state=2)

#Fitting Models
# Importing the Keras libraries and packages
import keras
from keras.models import Sequential
from keras.layers import Dense

# Initialising the ANN
classifier = Sequential()

# Adding the input layer and the first hidden layer
classifier.add(Dense(units = 8, kernel_initializer = 'uniform', activation = 'relu', input_dim = 15))

# Adding the second hidden layer
classifier.add(Dense(units = 8, kernel_initializer = 'uniform', activation = 'relu'))

# Adding the output layer
classifier.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'sigmoid'))

# Compiling the ANN
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

# Fitting the ANN to the Training set
classifier.fit(X_train, Y_train, batch_size = 10, epochs = 30)
```

Figure 4.13: Importing Keras

```
WARNING:tensorflow:From C:\Users\HP\Anaconda3\lib\site-packages\tensorflow\python\ops\resource_variable_ops.py:435: colocate
_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From C:\Users\HP\Anaconda3\lib\site-packages\tensorflow\python\ops\math_ops.py:3066: to_int32 (from tenso
rflow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/30
17634/17634 [==============================] - 4s 243us/step - loss: 0.2168 - accuracy: 0.9190
Epoch 2/30
17634/17634 [==============================] - 3s 183us/step - loss: 0.1460 - accuracy: 0.9494
Epoch 3/30
17634/17634 [==============================] - 3s 196us/step - loss: 0.1375 - accuracy: 0.9532
Epoch 4/30
17634/17634 [==============================] - 3s 193us/step - loss: 0.1320 - accuracy: 0.9570
Epoch 5/30
17634/17634 [==============================] - 3s 186us/step - loss: 0.1271 - accuracy: 0.9578
Epoch 6/30
17634/17634 [==============================] - 3s 198us/step - loss: 0.1226 - accuracy: 0.9613
Epoch 7/30
17634/17634 [==============================] - 3s 181us/step - loss: 0.1196 - accuracy: 0.9624
Epoch 8/30
17634/17634 [==============================] - 3s 180us/step - loss: 0.1164 - accuracy: 0.9627
Epoch 9/30
17634/17634 [==============================] - 3s 181us/step - loss: 0.1140 - accuracy: 0.9639
Epoch 10/30
17634/17634 [==============================] - 3s 180us/step - loss: 0.1112 - accuracy: 0.9648
Epoch 11/30
17634/17634 [==============================] - 3s 186us/step - loss: 0.1102 - accuracy: 0.9659
Epoch 12/30
17634/17634 [==============================] - 3s 186us/step - loss: 0.1080 - accuracy: 0.9657
Epoch 13/30
17634/17634 [==============================] - 3s 183us/step - loss: 0.1057 - accuracy: 0.9666
```

Figure 4.14: ANN Initialization

2. Calculating Cross Validation Score and Model Accuracy for ANN

```
In [10]:
yhat_train = (classifier.predict(X_train) > 0.5)
yhat_test = (classifier.predict(X_test) > 0.5)

In [11]:
# PREDICTING FOR TEST DATA
pred_ann = classifier.predict(test_df)

In [*]:
from sklearn.model_selection import cross_val_score
from sklearn import metrics
from keras.wrappers.scikit_learn import KerasClassifier

def build_classifier():
    classifier = Sequential()
    classifier.add(Dense(units = 8, kernel_initializer = 'uniform', activation = 'relu', input_dim = 15))
    classifier.add(Dense(units = 8, kernel_initializer = 'uniform', activation = 'relu'))
    classifier.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'sigmoid'))
    classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
    return classifier

classifier = KerasClassifier(build_fn = build_classifier, batch_size = 10, epochs = 30)

scores = cross_val_score(estimator = classifier,X = X_train, y = Y_train, cv = 10, n_jobs = 1)
Epoch 1/30
15870/15870 [==============================] - 4s 266us/step - loss: 0.2174 - accuracy: 0.9237
Epoch 2/30
15870/15870 [==============================] - 4s 233us/step - loss: 0.1326 - accuracy: 0.9548
Epoch 3/30
15870/15870 [==============================] - 3s 207us/step - loss: 0.1193 - accuracy: 0.9591
Epoch 4/30
15870/15870 [==============================] - 3s 194us/step - loss: 0.1099 - accuracy: 0.9622
Epoch 5/30
```

Figure 4.15: Cross Validation Score

3. Results of ANN with Model Evaluation

```
============================== ANN Model Evaluation ==============================

Cross Validation Mean Score:
 0.9740271747112275

Model Accuracy:
 0.9600204151071793

Confusion matrix:
 [[7982  263]
 [ 442 8947]]

Classification report:
              precision    recall  f1-score   support

           0       0.95      0.97      0.96      8245
           1       0.97      0.95      0.96      9389

    accuracy                           0.96     17634
   macro avg       0.96      0.96      0.96     17634
weighted avg       0.96      0.96      0.96     17634
```

Figure 4.16: Model Evaluation of ANN

Based on the ANN results obtained from Model Evaluation, the model had Cross validation Score of 0.97, while Model Accuracy of ANN was 96%

4. Test Results of ANN

```
============================== ANN Model Test Results ==============================

Model Accuracy:
 0.9622916115374438

Confusion matrix:
 [[3391  107]
 [ 178 3882]]

Classification report:
              precision    recall  f1-score   support

           0       0.95      0.97      0.96      3498
           1       0.97      0.96      0.96      4060

    accuracy                           0.96      7558
   macro avg       0.96      0.96      0.96      7558
weighted avg       0.96      0.96      0.96      7558
```

Figure 4.17: Test Results of ANN

The test results of ANN were almost similar to that of results by Model Evaluation with Model Accuracy of 96.22%.

Chapter 5

Conclusion

In this paper, we have presented a practical and efficient Network Intrusion Detection System using classification and deep learning technique which can also be implemented on other machine learning algorithms and can be used on existing systems.

Intrusion detection is a realistic and functional solution in offering a particular definition of defense in our large and current networks (possible uncertainty) Computer and networking programs. Intrusion monitoring systems are based on host-audit-trail and network traffic analysis and are designed to detect threats, preferably in real time.

Table 5.1: Performance summary of Classification Techniques

	Model Evaluation				Model Validation		
	Accuracy	Precision	Recall	Cross Validation Score	Accuracy	Precision	Recall
Decision Tree	100%	100%	100%	99.56	99.49%	100%	99%
K-Nearest Neighbor Classifier	99.37%	99%	100%	99.14	99.16%	99%	99%
Logistic Regression	95.49%	95%	97%	95.37	95.54%	95%	97%
Naive Bayes	90.71%	88%	96%	90.71	90.67%	88%	95%
ANN	96%	97%	96%	94.40	96.22%	97%	96%

In this paper , various popular classification and deep learning techniques for machine learning have been discussed with their basic working mechanisms, strengths and weaknesses. Potential implementations and problems related to their possible approaches have also been highlighted. Classification methods tend to be strong in modeling interactions.In the research study, supervised machine learning systems and rules were applied to intrusion detection data sets to forecast the probability of attack in the network context and the performance of the learning methods were evaluated on the basis of their predictive precision and ease of learning.

A modern Intelligent Network Intrusion Detection Method has been developed using a two-stage (Anomaly-Misuse) classification and deep learning methodology,as it is checked. Decision Tree, logistic regression, KNN and Naive Bayes were used as classification methods to identify traffic disturbances that could be targeted, and Deep learning used an ANN approach that classifies attacks if they occur. The complete 42 dimensional characteristics of the training data collection have been used in the experiment.
In both algorithms (stage-1 & stage-2) separate functions and parameters are evaluated. The results of the evaluation show that the high accuracy rate is 100 percent with a recall rate of 0.10 at stage 1 of the evaluation and the accuracy rate of the decision tree is 99.5 percent with a recall rate of 0.99 at stage 2 of the validation. All the results of the various model are shown in table 5.1. Experimental findings show that Decision tree algorithm results in high accuracy compared to other logistic Regression, KNN and Naive Bayes machine learning classifiers. Similar to the Neural Network based approach, our approach has a higher detection efficiency, less time-consuming and a low cost factor. It does, though, produce a few more false positives.

Bibliography

Adetunmbi, Adebayo O et al. (2008). "Network intrusion detection based on rough set and k-nearest neighbour". In: *International Journal of Computing and ICT Research* 2.1, pp. 60–66.

Bambrick, Noel (2016). *Support Vector Machines: A Simple Explanation*. https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html.

— (2017). *Understanding Support Vector Machine(SVM) algorithm*. https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/.

Bouckaert, Remco R (2004). "Naive bayes classifiers that perform well with continuous variables". In: *Australasian joint conference on artificial intelligence*. Springer, pp. 1089–1094.

Brownlee, Jason (2019). *What is Deep Learning*. https://machinelearningmastery.com/what-is-deep-learning/.

ML-Cheatsheet (2017). *Logistic Regression*. https://ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html.

Conrad, Eric, Seth Misenar, and Joshua Feldman (2017). "Chapter 7 - Domain 7: Security operations". In: *Eleventh Hour CISSP® (Third Edition)*. Ed. by Eric Conrad, Seth Misenar, and Joshua Feldman. Third Edition. Syngress, pp. 145–183. ISBN: 978-0-12-811248-9. DOI: https://doi.org/10.1016/B978-0-12-811248-9.00007-3. URL: http://www.sciencedirect.com/science/article/pii/B9780128112489000073.

Dey, Amitabha (2018). *Data Preprocessing for Machine Learning*. https://medium.com/datadriveninvestor/data-preprocessing-for-machine-learning-188e9eef1d2c.

Dong, Bo and Xue Wang (2016). "Comparison deep learning method to traditional methods using for network intrusion detection". In: *2016 8th IEEE International Conference on Communication Software and Networks (ICCSN)*. IEEE, pp. 581–585.

Educba (2020). *Classification Algorithms*. https://www.educba.com/classification-algorithms/.

Farnaaz, Nabila and MA Jabbar (2016). "Random forest modeling for network intrusion detection system". In: *Procedia Computer Science* 89.1, pp. 213–217.

Ganapathy, S, P Yogesh, and A Kannan (2011). "An intelligent intrusion detection system for mobile ad-hoc networks using classification techniques". In: *International Conference on Power Electronics and Instrumentation Engineering*. Springer, pp. 117–122.

Harrison, Onel (2018). *Machine Learning Basics with the K-Nearest Neighbors Algorithm*. https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761.

Hussain, Jamal, Samuel Lalmuanawma, and Lalrinfela Chhakchhuak (2016). "A two-stage hybrid classification technique for network intrusion detection system". In: *International Journal of Computational Intelligence Systems* 9.5, pp. 863–875.

Jain, Rishabh (2017). *Decision Tree. It begins here*. https://medium.com/@rishabhjain_22692/decision-trees-it-begins-here-93ff54ef134.

Javaid, Ahmad et al. (2016). "A deep learning approach for network intrusion detection system". In: *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, pp. 21–26.

Jupyter (2020). *Jupyter*. https://jupyter.org/.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning". In: *nature* 521.7553, pp. 436–444.

Li, Wei (2004). "Using genetic algorithm for network intrusion detection". In: *Proceedings of the United States department of energy cyber security group* 1, pp. 1–8.

Marr, Bernard (2018). *What Is Deep Learning AI ?* https://www.forbes.com/sites/bernardmarr/2018/10/01/what-is-deep-learning-ai-a-simple-guide-with-8-practical-examples/.

Mittal, Akhil (2017). *Machine Learning Process And Scenarios*. https://elearningindustry.com/machine-learning-process-and-scenarios.

Mukherjee, Biswanath, L Todd Heberlein, and Karl N Levitt (1994). "Network intrusion detection". In: *IEEE network* 8.3, pp. 26–41.

Navlani, Avinash (2018). *KNN Classification using Scikit-learn*. https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn.

Panda, Mrutyunjaya and Manas Ranjan Patra (2007). "Network intrusion detection using naive bayes". In: *International journal of computer science and network security* 7.12, pp. 258–263.

Phyu, Thair Nu (2009). "Survey of classification techniques in data mining". In: *Proceedings of the International MultiConference of Engineers and Computer Scientists*. Vol. 1, pp. 18–20.

Sangkatsanee, Phurivit, Naruemon Wattanapongsakorn, and Chalermpol Charnsripinyo (2011). "Practical real-time intrusion detection using machine learning approaches". In: *Computer Communications* 34.18, pp. 2227–2235.

Schott, Madison (2019). *Random Forest Algorithm for Machine Learning*. https://medium.com/capital-one-tech/random-forest-algorithm-for-machine-learning-c4b2c8cc9feb.

Sehra, Chirag (2018). *Decision Trees Explained Easily*. https://medium.com/@chiragsehra42/decision-trees-explained-easily-28f23241248.

Shafiq, Muhammad et al. (2016). "Network traffic classification techniques and comparative analysis using machine learning algorithms". In: *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*. IEEE, pp. 2451–2455.

Sharma, Sheetal (2017). *Artificial Neural Network (ANN) in Machine Learning*. https://www.datasciencecentral.com/profiles/blogs/artificial-neural-network-ann-in-machine-learning.

Shenfield, Alex, David Day, and Aladdin Ayesh (2018). "Intelligent intrusion detection systems using artificial neural networks". In: *ICT Express* 4.2, pp. 95–99.

Sidana, Mandy (2017). *Intro to types of classification algorithms in Machine Learning*. https://medium.com/@Mandysidana/machine-learning-types-of-classification-9497bd4f2e14.

Singh, Nagesh (2020a). *Decision Tree Algorithm*. https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html.

— (2020b). *Random Forest — A Powerful Ensemble Learning Algorithm*. https://www.kdnuggets.com/2020/01/random-forest-powerful-ensemble-learning-algorithm.html.

Solutions, Mindfire (2017). *Python: 7 Important Reasons Why You Should Use Python*. https://medium.com/@mindfiresolutions.usa/python-7-important-reasons-why-you-should-use-python-5801a98a0d0b.

Soofi, Aized Amin and Arshad Awan (2017). "Classification techniques in machine learning: applications and issues". In: *Journal of Basic and Applied Sciences* 13, pp. 459–465.

Srivastava, Tavish (2014). *How does Artificial Neural Network (ANN) algorithm work? Simplified!* https://www.analyticsvidhya.com/blog/2014/10/ann-work-simplified/.

Swamy, Himanshu (2019). *How to Deploy a Logistic Regression Model in GCP*. https://medium.com/analytics-vidhya/insiders-view-on-logistic-regression-and-how-do-we-deploy-regression-model-in-gcp-as-batch-c62a64563210.

Team, The Redscan (2017). *The key challenges of intrusion detection and how to overcome them*. https://www.redscan.com/news/the-key-challenges-of-intrusion-detection-and-how-to-overcome-them/.

W3Schools (2020). *Python Introduction*. https://www.w3schools.com/python/python_intro.asp.

Wikipedia (2020a). *Decision Tree*. https://en.wikipedia.org/wiki/Decision_tree.

— (2020b). *Deep Learning*. https://en.wikipedia.org/wiki/Deep_learning.

— (2020c). *Logistic Regression*. https://en.wikipedia.org/wiki/Logistic_regression.

— (2020d). *Python (programming language)*. https://en.wikipedia.org/wiki/Python_(programming_language).

— (2020e). *Zen of Python*. https://en.wikipedia.org/wiki/Zen_of_Python.

Will_Koehrsen (2018). *An Implementation and Explanation of the Random Forest in Python*. https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76.

Zhang, Zheng et al. (2001). "HIDE: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification". In: *Proc. IEEE Workshop on Information Assurance and Security*, pp. 85–90.

Zheng, Zhuoyuan, Yunpeng Cai, and Ye Li (2016). "Oversampling method for imbalanced classification". In: *Computing and Informatics* 34.5, pp. 1017–1037.