

MACHINE LEARNING BASED NETWORK INTRUSION DETECTION SYSTEM

submitted by

Himangshu Bauri

Class Roll No: 002210503009

Exam Roll: MCA244009

Master of Computer Application

under the supervision of

Prof. (Dr.) Anupam Sinha

in the

Department of Computer Science and Engineering

Faculty of Engineering and Technology

Jadavpur University

2024

Faculty of Engineering and Technology
Jadavpur University

CERTIFICATE OF RECOMMENDATION

To Whom It May Concern

I hereby recommend that the project report entitled "MACHINE LEARNING BASED NETWORK INTRUSION DETECTION SYSTEM" has been carried out by Himangshu Bauri (Reg. No.: 163601 of 2022-23, Roll No.: 002210503009, Exam Roll: MCA244009), under my guidance and supervision and be accepted in partial fulfilment of the requirement for the degree of Master of COMPUTER APPLICATION in DEPARTMENT of COMPUTER SCIENCE and ENGINEERING, JADAVPUR UNIVERSITY during the academic year 2023-2024.

Prof. (Dr.) Anupam Sinha
Project Supervisor
Dept. of Computer Science and
Engineering
Jadavpur University

Prof. (Dr.) Nandini Mukhopadhyay
Head of the Department
Dept. of Computer Science & Engineering
Jadavpur University

Prof. Dipak Laha
Dean, Faculty Council
of Engineering & Technology
Jadavpur University, Kolkata-032

Faculty of Engineering and Technology
Jadavpur University

CERTIFICATE OF APPROVAL

This is to certify that the project report entitled “**MACHINE LEARNING BASED NETWORK INTRUSION DETECTION SYSTEM**” is a bonafide record of work carried out by HIMANGSHU BAURI, in fulfillment of the requirements for the award of the degree of Master of Computer Application in the Department of Computer Science and Engineering, Jadavpur University. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein but approve the project report only for the purpose for which it has been submitted.

Signature of Examiner 1

Date :

Signature of Examiner 2

Date :

Faculty of Engineering and Technology
Jadavpur University

**DECLARATION OF COMPLIANCE WITH
ACADEMIC ETHICS**

I hereby declare that this project report entitled "MACHINE LEARNING BASED NETWORK INTRUSION DETECTION SYSTEM" is an outcome of my genuine and independent work, as part of my Master of Computer Application program.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, all materials and results that are not original to this work have been fully cited and referenced.

Himangshu Bauri

Registration No.: 163601 of 2022-23

Examination Roll No.: MCA244009

Thesis Title :- Machine Learning Based Network Intrusion Detection System

Signature with Date

ACKNOWLEDGEMENT

With my most sincere and gratitude, I would like to thank Prof. Anupam Sinha, Department of Computer Science & Engineering, my supervisor, for his overwhelming support throughout the duration of the project. His motivation always gave me the required inputs and momentum to continue with my work, without which the project work would not have taken its current shape. His valuable suggestion and numerous discussions have always inspired new ways of thinking. I feel deeply honored that I got this opportunity to work under him.

I would like to express my sincere thanks to all my teachers for providing sound knowledge base and cooperation.

I would like to thank all the faculty members of the Department of Computer Science & Engineering of Jadavpur University for their continuous support.

Last, but not the least, I would like to thank my batch mates for staying by side when I need them.

Himangshu Bauri

Registration No: 163601 of 2022-2023

Exam Roll No.: MCA244009

Department of Computer Science & Engineering,
Jadavpur University

ABSTRACT

The rapid expansion and extensive usage of the internet have increased as more people connect to the network, amplifying the need for network security. Intrusion Detection Systems (IDS) are cybersecurity tools that oversee the status of software and hardware in a network environment. These systems are designed to analyze and safeguard the system while forecasting its behavior. IDS continue to grapple with enhancing detection accuracy, which is addressed by utilizing Machine Learning (ML) algorithms to identify intrusions. In this project, various machine learning algorithms are employed to predict different types of network attacks using a dataset that encompasses a range of network intrusions. The IDS not only forecasts malicious activities but also identifies the specific type of network under threat.

Keywords: Intrusion Detection, cybersecurity, accuracy, machine learning, intrusions, network attack.

CONTENTS

		Page No.
	Certificate Of Recommendation	I
	Certificate Of Approval	II
	Declaration Of Originality And Compliance Of Academic Ethics	III
	Acknowledgement	IV
	Abstract	V
	Contents	VI
Chapter-1	INTRODUCTION	1
	1.1 Overview	1
	1.2 Motivation	3
	1.3 Problem Statement	5
	1.4 Organisation of the project work	6
Chapter-2	BACKGROUND	7
	2.1 Importance of Network Security	7
	2.2 Intrusion Detection System (IDS)	8
	2.3 Machine Learning	11
	2.4 K-Nearest Neighbor Classifier	13
	2.5 Decision Tree	15
	2.6 Support Vector Machine	20
Chapter-3	Literary Review	25
	3.1 Survey of Intrusion detection methods	25
	3.2 Literary Review on Machine learning techniques	26
Chapter-4	Methodology	29
	4.1 Architecture	29

	4.2	Method and Materials	30
		4.2.1 Data Preprocessing	32
		4.2.1.1 One Hot Encoding	33
		4.2.1.2 Feature Selection	34
		4.2.1.3 Standarding Data using StandardScaler	37
	4.4	Experiments and Discussion	38
		4.3.1 KNN	38
		4.3.2 Decision Tree	39
		4.3.3 Support Vector Machine	39
Chapter-5	Results		41
	5.1	Results of Binary Classification	41
		5.1.1 Performance results	41
	5.2	Results of Multiclass Classification	43
		5.2.1 Performance results	
Chapter-6	Conclusion		46
	6.1	Conclusion	46

Introduction

1.1 Overview

An Intrusion Detection System is used to detect all types of malicious network traffic and computer usage that can't be detected by a conventional firewall. This includes network attacks against vulnerable services, data driven attacks on applications, host based attacks such as privilege escalation, unauthorized logins and access to sensitive files, and malware (viruses, trojan horses, and worms). Today, political and commercial entities are increasingly engaging in sophisticated cyber-warfare to damage, disrupt, or censor information content in computer networks. In designing network protocols, there is a need to ensure reliability against intrusions of powerful attackers that can even control a fraction of parties in the network. The controlled parties can launch both passive (e.g., eavesdropping, nonparticipation) and active attacks (e.g., jamming, message dropping, corruption, and forging). Intrusion detection is the process of dynamically monitoring events occurring in a computer system or network, analyzing them for signs of possible incidents and often interdicting unauthorized access. This is typically accomplished by automatically collecting information from a variety of

systems and network sources, and then analyzing the information for possible security problems. The Project concept is to achieve the new method for extracting the information from the NSL-KDD Dataset that will help to automatically detect the attack like Dos, probe, R2L, U2R. How such attacks are Detected by KNN, Decision tree, SVM module and provide the security from any anomaly data which will slow your system. and provide security to the serve.

The decision tree (DT) classifier is a supervised ML algorithm that will be utilized for the classification task of Normal or Attack behavior based on the 41-input feature. The resulting DT algorithm develops a decision-making process based on a tree-like model with nodes or branches. The max depth of the decision tree can be defined beforehand. A decision tree is already an explainable machine learning algorithm through visualizations of the resulting trees.

The K-Nearest Neighbors (KNN) classifier is a supervised machine learning algorithm used for classifying network behavior as normal or attack based on the 39-input features. KNN works by finding the k-nearest data points to a given sample and classifying it based on the majority class of those neighbors. KNN's simplicity and straightforward implementation make it a popular choice for intrusion detection, but it can be sensitive to the choice of k and the distance metric.

The Support Vector Machine (SVM) classifier is another supervised machine learning method applied to categorize network activity as normal or attack. SVM creates a

hyperplane that separates different classes of data, optimizing the margin between them. It excels at handling high-dimensional data and can use various kernel functions for greater flexibility in decision boundaries. While SVM can be computationally intensive, it offers robust performance and is effective for detecting network intrusions.

1.2 Motivation

The increasing complexity and frequency of cyber threats present significant challenges for network security professionals. This final year project, titled " MACHINE LEARNING BASED NETWORK INTRUSION DETECTION SYSTEM" seeks to address these challenges by exploring the application of machine learning techniques for detecting intrusions in networks.

The project investigates three well-established supervised machine learning algorithms: K-Nearest Neighbors (KNN), Decision Trees, and Support Vector Machines (SVM), for classifying network behavior as either Normal or Attack based on a 41-input feature set. K-Nearest Neighbors classifies behavior by comparing each instance to its k-nearest neighbors in the dataset and categorizing it based on majority vote. While it offers potential in identifying patterns, KNN may face challenges with high-dimensional data or noisy inputs.

Decision Trees create a decision-making model using a tree-like structure of nodes and branches, facilitating the classification of network activity. This algorithm's explainability

allows for clear visualization of decisions, and its depth can be controlled for optimal performance.

Support Vector Machines effectively handle high-dimensional data by finding an optimal hyperplane that separates different classes in the feature space. The choice of kernel function and other parameters can be adjusted to optimize performance and efficiency.

By integrating these machine learning approaches in intrusion detection systems, this project aims to improve network security by providing robust tools for analyzing, detecting, and mitigating emerging cyber threats. The outcomes of this study may contribute valuable insights and advancements to the field of cybersecurity.

1.3 Problem Statement:

This project aims to enhance network intrusion detection systems by leveraging machine learning algorithms such as K-Nearest Neighbors (KNN), Decision Trees, and Support Vector Machines (SVM). The investigation seeks to improve the accuracy, robustness, and adaptability of intrusion detection systems in the face of increasingly sophisticated cyber threats.

1. Evaluate the efficacy of K-Nearest Neighbors (KNN) in classifying network traffic based on the 41-input feature set, focusing on its ability to balance between accuracy and computational efficiency.
2. Design and implement a decision tree-based IDS architecture, optimizing the model's depth and structure for clear, interpretable decision-making while effectively detecting network anomalies.
3. Analyze the potential of Support Vector Machines (SVM) to identify network threats by determining the best hyperplane separating normal and attack behavior in high-dimensional feature spaces

1.4 Organisation of the Project work

Chapter 1 provides an introduction to network security, emphasizing its importance in protecting data and resources from unauthorized access and cyber threats.

Chapter 2 reviews the concept of Network Intrusion Detection Systems (NIDS), detailing their role in monitoring network traffic for suspicious activities and potential threats.

Chapter 3 briefly describes all basic theoretical concepts related to network security, NIDS, and three machine learning algorithms: K-Nearest Neighbors (KNN), Decision Tree, and Support Vector Machine (SVM).

Chapter 4 describes the preprocessing steps and methodologies used for data preparation, including standardization and feature selection, which are essential for effective machine learning model performance.

Chapter 5 discusses the experiments conducted, the performance of the three machine learning models in binary and multi-class classifications, and presents the results and comparisons of their accuracy, precision, recall, and F1-scores.

Chapter 6 concludes the research, summarizing the findings, discussing the strengths and limitations of the approaches, and suggesting potential future work in the field of network intrusion detection using machine learning models.

Background

2.1 Importance of Network Security

. Network security is crucial for several reasons. It protects sensitive data such as personal information, financial details, and business secrets from unauthorized access or breaches. By securing networks, organizations can prevent cyber attacks like malware, ransomware, and phishing, safeguarding systems and data. Effective network security ensures smooth business operations and minimizes downtime, contributing to business continuity. Additionally, many industries have strict data protection and security regulations that organizations must comply with, and robust network security helps maintain compliance and avoid legal issues. Strong network security also preserves customer trust and corporate reputation, enhancing loyalty and confidence in the organization.

2.2 Intrusion Detection System (IDS)

Intrusion detection systems are methods and applications created to assess and protect computers, networks, programs and data from attacks, unauthorized access, illegal read/write and deletion/corruption. These can be divided into host and network based depending on where and how it is deployed. Intrusion detection systems should not be mistaken for other security measures that may exist in a network or system such as firewalls and antiviruses. Intrusion detection systems work in combination to these as they are more closely related to detecting and raising alarms. Intrusions can happen both internally or externally due to these some system setups operate a hybrid intrusion detection setup. Both host-based and network-based intrusion systems detect intrusions using one of two methods or some times a combination of both, these are:

Host Based:

This kind of intrusion detection system is primarily used to monitor the internal environment of a host; resources, filesystem and applications of the host. A host based system monitors both the state and behavior and state of its host. Due to this it can detect unusual behavior of programs and files (M. Gupta, 2015).

Even though host based intrusion detection systems were primarily created for internal monitoring there are some variations of the host based intrusion detection system that can also detect network intrusions (A. P. Singh, 2016).

Network Based:

The main difference between the network based and the host based is how it is deployed. It exists as part of the network to detect intrusion attempts on the network. The network based intrusion detection system analyzes inbound and outbound traffic for patterns outside the expected behavior (J. G. Noraini, 2011). It is good to note that a network based intrusion detection system does not replace a firewall in a system.

Types of Intrusion Detection Approach:

Intrusion detection systems approach the problem of detection in two basic ways regardless of the type, host based or network based. The approach is either signature based or anomaly based.

Signature Based:

Signature based systems essentially work on the premise that the attack has occurred before and we have a 'signature' of the attack, in simple terms a fairly detailed idea about the structure and content of malformed packets. This information can be

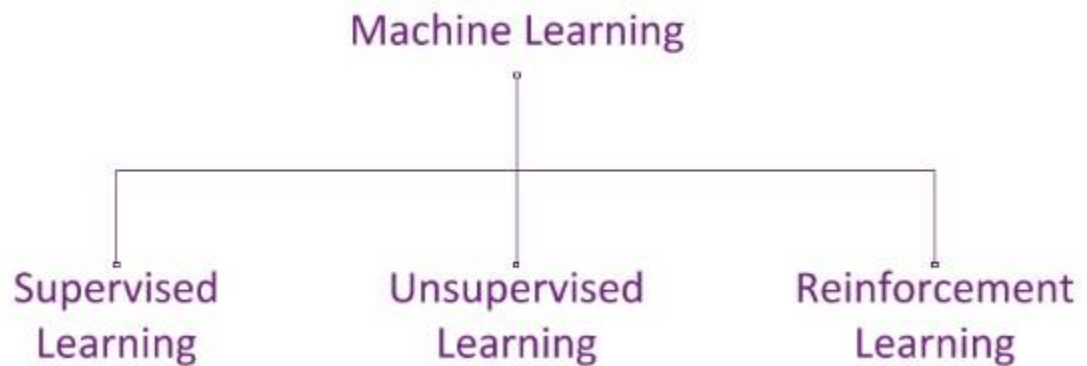
encoded in the form of a 'signature' and stored in a database of all known attack signatures.

Anomaly Based:

Anomaly based detection searches through networks for abnormal behavior, it suspects any network behavior that deviates from the regular network operating baseline as an anomaly. This method is preferred under certain conditions to the signature based system because it has a much higher chance of detecting new (zero day) attacks.

2.3 Machine Learning

Machine learning is a branch of artificial intelligence (AI) that focuses on the development of algorithms and statistical models that enable computers to perform tasks without explicit instructions, relying on patterns and inference instead. In essence, it enables computers to learn from data and improve their performance over time.



Supervised Learning :

Supervised learning algorithms are trained using labeled examples, such as an input where the desired output is known. For example, a piece of equipment could have data points labeled either "F" (failed) or "R" (runs). The learning algorithm receives a set of inputs along with the corresponding correct outputs, and the algorithm learns by comparing its actual output with correct outputs to find errors. It then modifies the model accordingly. Through methods like classification, regression, prediction, and gradient boosting, supervised learning uses patterns to predict the values of the label on additional unlabeled data[1].

Unsupervised Learning :

Unsupervised learning is used against data that has no historical labels. The system is not told the "right answer." The algorithm must figure out what is being shown. The goal is to explore the data and find some structure within. Or it can find the main attributes that separate customer segments from each other[2].

Reinforcement Learning :

Reinforcement learning (RL) is a type of machine learning paradigm where an agent learns to make decisions by interacting with an environment. Unlike supervised learning, where the algorithm is trained on labeled data, and

unsupervised learning, where the algorithm learns from unlabeled data, reinforcement learning learns through trial and error, receiving feedback in the form of rewards or penalties based on its actions.

Supervised Learning

2.4 K-Nearest Neighbor classifier:

The K-Nearest (KNN) is a simple and accurate machine learning algorithm that classifies instances based on the nearest training instance in a feature space. It can be used for both regression and classification. KNN is an algorithm that makes no assumptions about the structure of data and distribution which means it is a nonparametric algorithm. This is an advantage because real life data scarcely obeys theoretical rules[4].

KNN works by classifying or predicting based on a fixed number (K) of training instances closest to the input instance. This means that for a chosen value of K, an input instance would be classified or predicted to belong to the same class as the closest number of K instances nearest to it

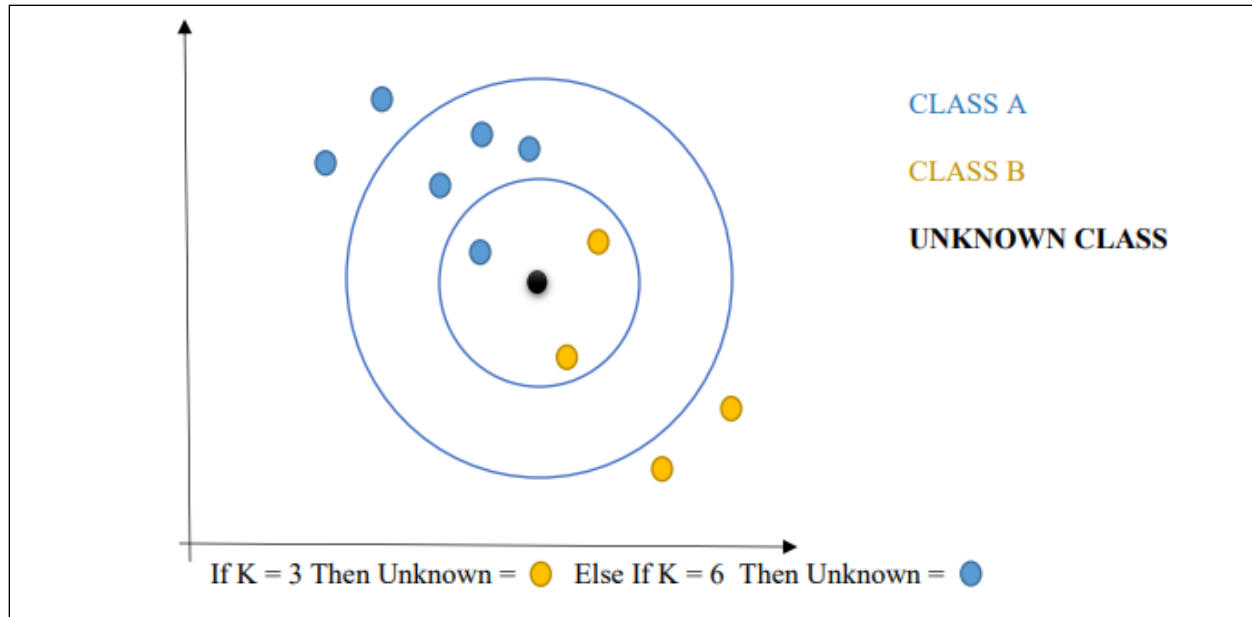


Figure – 2.1

The distance from the unknown to the nearest neighbors is measured using different methods. The most popular one is Euclidean distance, other popular methods include Manhattan distance, Hamming distance, Minkowski distance.

Mathematically they can be represented by:

$$\text{Euclidean distance} = d(p, q) = d(q, p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \dots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^n (q_i - p_i)^2}$$

$$\text{Manhattan distance} = d_1(p, q) = ||p - q||_1 = \sum_{i=1}^n |p_i - q_i|$$

$$\text{Hamming distance} = d_{ij} \sum_{k=1}^p |x_{ik} - x_{jk}|$$

$$\text{Minkowski distance} = \sum_{i=1}^n (|x_i - y_i|^p)^{\frac{1}{p}}$$

[5]Each distance type is ideal under certain use cases. The Manhattan distance is best suited for KNN problems where the features are of different types. The Euclidean distance is suited to the opposite case where features are of the same type. The value of 'k' is a very important part of the KNN algorithm because of prediction accuracy. Selecting a value for 'k' is a task that should not be taken for granted. Selecting small values for k will more than likely result in lower accuracy if the algorithm is training from a noisy dataset, and if the value for 'k' is too high the model may overfit resulting in low accuracy. An odd number 'k' should also be chosen if the number of classes are even to prevent ties in the majority vote[7].

2.5 Decision Tree :

A decision tree is a supervised machine learning algorithm used for both classification and regression tasks. It is a predictive modeling tool that learns a series of hierarchical decision rules based on the features of the data (Wikipedia).

In[7] DT is represented as a tree-like layout with a tree branch indicating internal nodes to describe a test of a certain feature, reflecting the outcome of the test. The leaves of the tree refer to the nodes as a class. It is a non-parametric classifier and supervised learning system that uses conditional control statements to represent data characteristics as a tree shown in below Figure-2.2.

The goal of the DT classification technique is to anticipate a response and then follow the decisions down the tree from the root node to the leaf node, where the answer is stored.

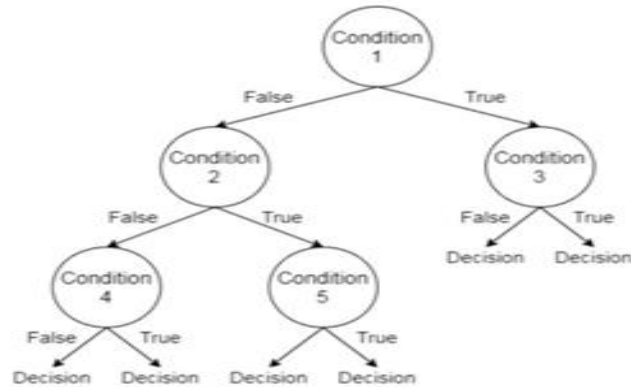


Figure – 2.2

Decision Tree Algorithm:

In the primary purpose of employing a decision tree is to use simple decision rules derived from the training data and develop a training model that can be used to forecast the target variable class or value.

The terminology related to decision trees are:

a.Root Node: Root Node represents the whole population or sample and is split into two or more homogenous groups.[10]

b. Splitting:The division of a node into two or more sub-nodes.

c. Decision Node: The point at which a sub-node is split into further sub-nodes.

d. Pruning: Removal of unwanted branches from sub-nodes of a decision node. It reduces the size of the decision trees.

e. Branch / Sub-tree: A subsection of the entire tree.

f. Parent & Child Node:

“A parent node of sub-nodes is a node that has been split into sub-nodes, whereas subnodes are the child of a parent node”[10]

Below figure-2.3 shows an example of a DT, where each internal node splits down the tree from the root node into two or more sub-nodes based on the values of an input attribute discrete function. In each decision node a single attribute is partitioned according to the attribute values. Therefore, each terminal node (B and C), also known as leaf nodes, are allocated to a single class that represents the best suitable goal values or responses. This is a recursive procedure that is repeated for each subtree rooted at the new node[11].

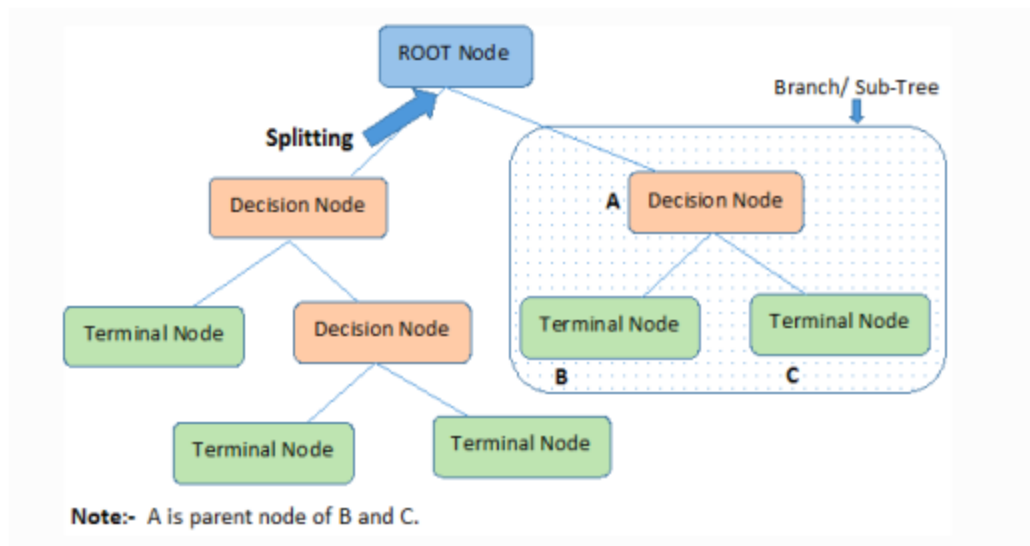


Figure-2.3

Training a Decision Tree:

DTs are built by recursively splitting training samples using features from the data that are used in a specific task. This means that making the number of splits and strategic split criteria will heavily affects the accuracy of the tree. Below figure shows a basic idea of a DT algorithm that works with the given dataset, also known as "training set," used to train the model as well as to evaluate the performance of measurement e.g., accuracy, precision and recall. This process will be repeated recursively for all child nodes until one of the conditions is met. However, the decision criteria are different for classification and regression trees. Nominal responses come from classification trees, while numeric

responses come from regression trees. In this project, we will be focusing on using classification trees.

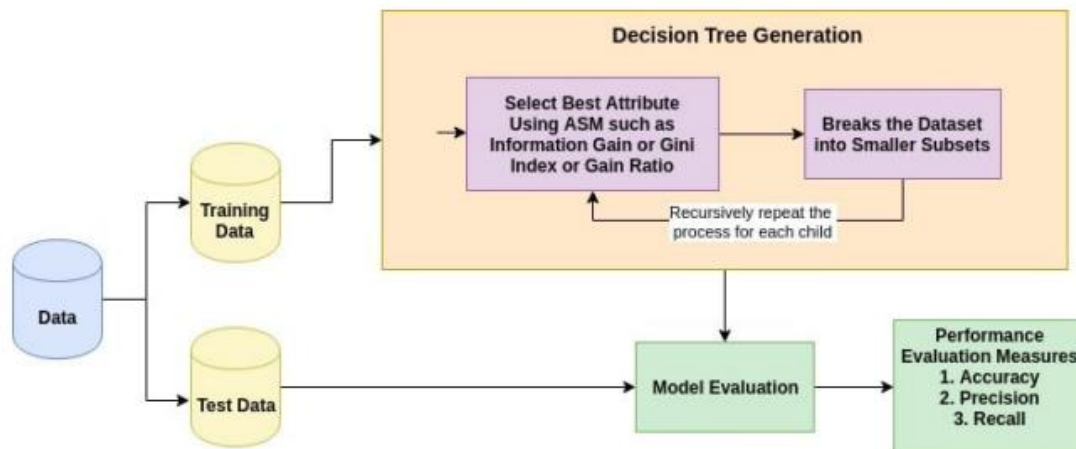


Figure-2.4

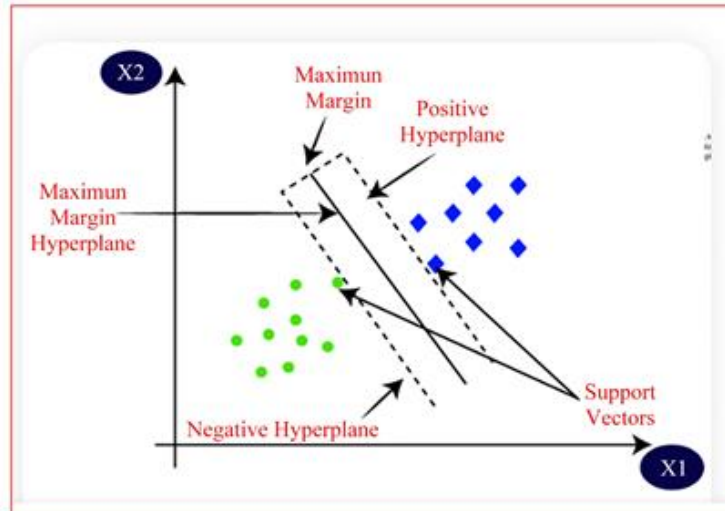
DECISION TREE IMPLEMENTATION:

1. Benefits of Decision Tree:

DT is one of the most popular algorithms in ML. The benefits of DT are that it can quickly learn from a dataset by studying information about the system crucial features that reveal malicious activity. It uses a non-parametric approach that is not dependent on probability distribution assumptions and is distribution-free. Moreover, it can promote to check on any advancement of attack signatures and different activities that have occurred while perceiving the data patterns. As a result, the value of various security frameworks is increased by examining the layout of intrusion detection information.

2.6 Support Vector Machines:

Support Vector Machine (SVM) is a relatively simple Supervised Machine Learning Algorithm used for classification and/or regression. It is more preferred for classification but is sometimes very useful for regression as well. Basically, SVM finds a hyper-plane that creates a boundary between the types of data. In 2-dimensional space, this hyper-plane is nothing but a line. In SVM, we plot each data item in the dataset in an N-dimensional space, where N is the number of features/attributes in the data. Next, find the optimal hyperplane to separate the data. So by this, we must have understood that inherently, SVM can only perform binary classification (i.e., choose between two classes). However, there are various techniques to use for multi-class problems.



A simple illustration of the Support Vector Machine (SVM) algorithm in 2-dimensions.

Figure-2.5

How Does It Work:

In the base form, linear separation, SVM tries to find a line that maximizes the separation between a two-class data set of 2-dimensional space points. To generalize, the objective is to find a hyperplane that maximizes the separation of the data points to their potential classes in an n-dimensional space. The data points with the minimum distance to the hyperplane (closest points) are called *Support Vectors*[12].

In the image below, the Support Vectors are the 3 points (2 blue and 1 green) laying on the scattered lines, and the separation hyperplane is the solid red line:

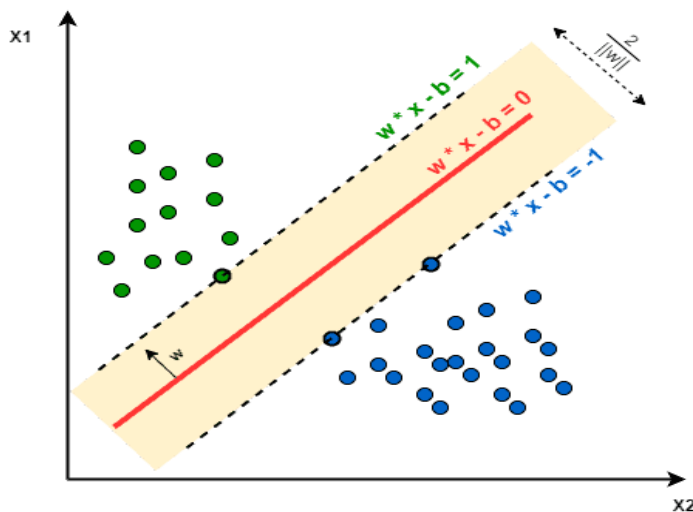


Figure-2.6

The computations of data points separation depend on a kernel function. There are different kernel functions: Linear, Polynomial, Gaussian, Radial Basis Function (RBF), and Sigmoid. Simply put, these functions determine the smoothness and efficiency of class separation, and playing around with their hyperparameters may lead to overfitting or underfitting.

Polynomial Kernel Function: $K(\mathbf{x}_i, \mathbf{x}) = (\mathbf{x}_i^\top \mathbf{x} + 1)^q$

- The polynomial kernel function computes the dot product of the feature vectors $\mathbf{x}_i$ and $\mathbf{x}$, raises it to the power of q , and adds 1.
- q is the degree of the polynomial.

Gaussian Kernel Function (Radial Basis Function, RBF): $K(\mathbf{x}_i, \mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}\|^2}{2\sigma^2}\right)$

- The Gaussian kernel function computes the similarity between two feature vectors $\mathbf{x}_i$ and $\mathbf{x}$ based on the Euclidean distance.
- σ is a parameter that controls the width of the Gaussian.

Sigmoid Kernel Function: $K(\mathbf{x}_i, \mathbf{x}) = \tanh(v(\mathbf{x}_i^\top \mathbf{x}) + c)$

- The sigmoid kernel function computes the hyperbolic tangent of a linear combination of the feature vectors $\mathbf{x}_i$ and $\mathbf{x}$.
- v and c are parameters that control the scaling and shifting of the hyperbolic tangent function.

Multiclass Classification Using SVM :

In its most simple type, SVM doesn't support multiclass classification natively. It supports binary classification and separating data points into two classes. For multiclass classification, the same principle is utilized after breaking down the multiclassification problem into multiple binary classification problems.

The idea is to map data points to high dimensional space to gain mutual linear separation between every two classes. This is called a **One-to-One approach**, which breaks down the multiclass problem into multiple binary classification problems. A binary classifier per each pair of classes[12].

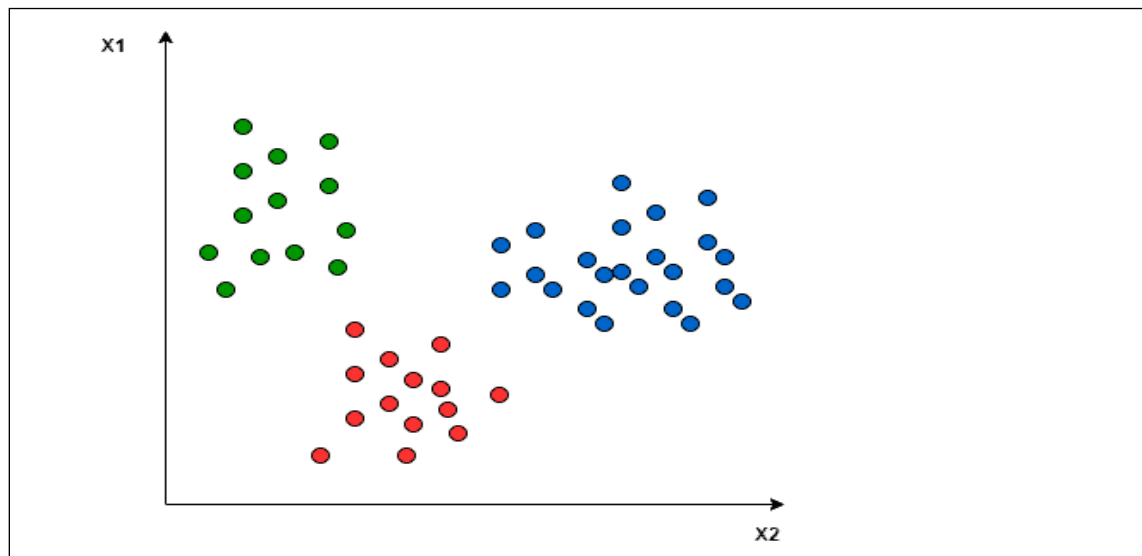
Another approach one can use is **One-to-Rest**. In that approach, the breakdown is set to a binary classifier per each class.

A single SVM does binary classification and can differentiate between two classes. So that, according to the two breakdown approaches, to classify data points from m classes data set[12]:

In the One-to-Rest approach, the classifier can use m SVMs. Each SVM would predict membership in one of the m classes.

In the One-to-One approach, the classifier can use $m(m-1)/2$ SVMs.

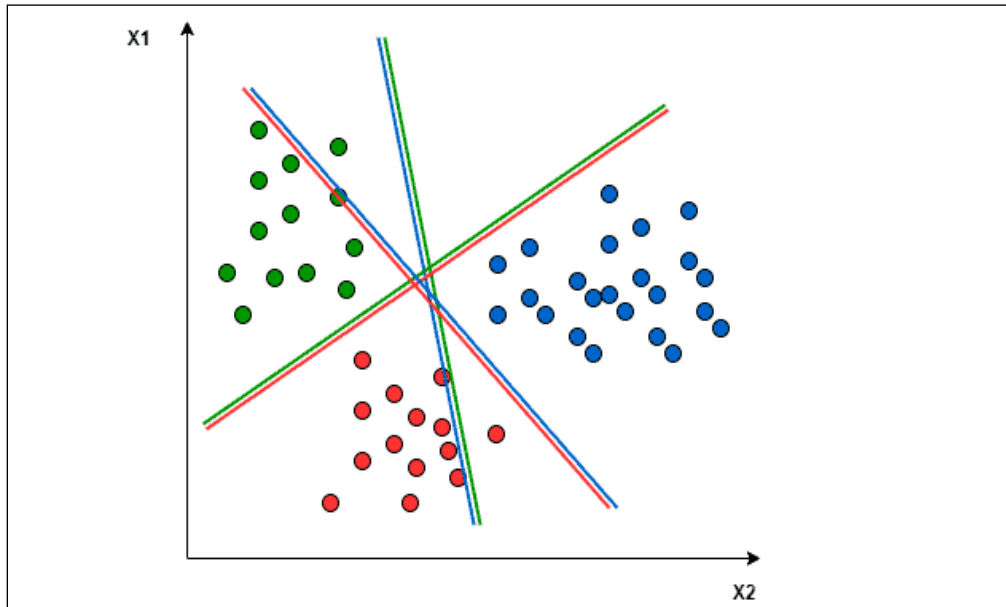
Let's take an example of 3 classes classification problem; green, red, and blue, as the following image:



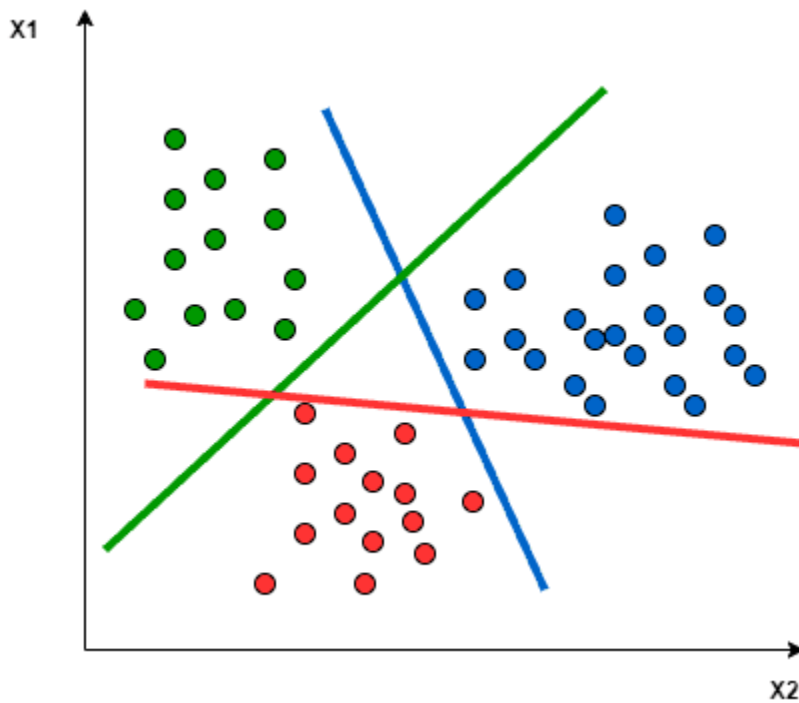
Applying the two approaches to this data set results in the followings:

In the **One-to-One** approach, we need a hyperplane to separate between every two classes, neglecting the points of the third class. This means the separation takes into

account only the points of the two classes in the current split. For example, the red-blue line tries to maximize the separation only between blue and red points. It has nothing to do with green points:



In the **One-to-Rest** approach, we need a hyperplane to separate between a class and all others at once. This means the separation takes all points into account, dividing them into two groups; a group for the class points and a group for all other points. For example, the green line tries to maximize the separation between green points and all other points at once:



Literature Review

A wealth of literature covers various aspects of intrusion detection. In this section, we present existing related works.

3.1 Survey of intrusion detection methods :

Most of the related research focuses on intrusion detection methods. Bhuyan et al. (2013) briefly describe and compare a large number of network anomaly detection methods and systems.

Ahmed et al. (2016) analyzed anomaly detection methods and the complexity of machine learning/data mining (ML/DM) algorithms. Milenkoski et al. (2015) evaluated common practices for intrusion detection systems by analyzing the existing standard evaluation parameters, including workloads and metrics. Buczak and Guven (2015) discussed machine learning and data mining methods for network analysis to support intrusion detection.

3.2 Literary Review on Machine learning techniques:

Gaikwad et al.[13], [14] in their paper, make use of the Bagging Ensemble method to implement an IDS. They used Partial Decision Tree as a base classifier due to its simplicity. Next, they used a Genetic algorithm for feature selection. The proposed

intrusion detection system is evaluated in terms of classification accuracy, true positives, false-positive, and model building time. The outcomes revealed that the introduced system scored the highest classification accuracy of 99.71% using cross-validation. This model exhibits higher classification accuracy than all classifiers except the C4.5 classifier on the test dataset.

Belavagi et al.[15] in his paper, assembled an exact model using Logistic Regression, Gaussian Naive Bayes, RF, and SVM. The outcomes show that the RF algorithm outperforms the other algorithms on classifying whether the data is normal or an attack, based on precision, recall, and accuracy. It has 99% accuracy.

Aburomman et al.[16] implemented an ensemble algorithm merging Linear Discriminant Analysis(LDA), Principle Component Analysis(PCA) with SVM. This method produced better accuracy in comparison to a single feature extraction method. A hybrid model consisted of SVM and Genetic Algorithm is presented by Atefi et al.[16]. The results showed that when the hybrid model was used, the precision-fitted in around 98% and performed

In [18] the authors proposed a novel feature representation approach, called the cluster center and nearest neighbor(CANN) approach. This methodology estimated and added two distances, the first dependent on the distance between each data sample and its cluster center. The subsequent distance is between the

data and its nearest neighbor in the same cluster. This new and one-dimensional distance is utilized to address each data sample for intrusion detection by a K-NN classifier. The outcomes show that the CANN classifier performs better than k-NN and SVM in classification accuracy, detection rates, and false alarms. It provides high computational efficiency for the time of classifier training and testing.

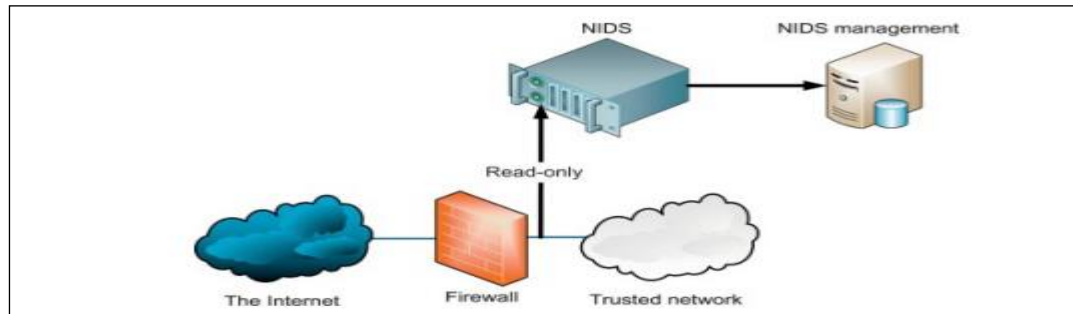
Support Vector Machines (SVM) is based on intuitive geometric principles that linearly separates the training data to obtain the minimum expected risk for the application, as per Vapnik in 1995 [19]. As all problems cannot be separated linearly, SVM use some kernel functions like polynomial or radial basis function in order to transform the linear algorithms to nonlinear ones via a map into feature spaces. In this paper, we use polynomial kernel to obtain the support vectors along the surface of this function which further can be used by SVM for necessary classification on data.

[20]Dr. S.Vijayarani¹ and Ms. Maria Sylviaa [2015] proposed, 'Intrusion Detection System- A Study' This main objective of this papers to provide a complete study about the definition of intrusion detection, history, life cycle, types of intrusion detection methods, types of attacks, techniques and different tools, research needs, applications and challenges.

Methodology

This chapter provides a brief information to the design approach and the datasets used in building an intelligent model of an IDS. The key features required for solutions are adaptive, lightweight, multi-layer, distributive and ability to learn from the past. We designed an innovative architecture IDS that would reduce the size of the datasets for the classifier. We have selected the NSL-KDD Intrusion Detection Dataset for the experiments and proposed an intelligent solution which satisfies the key requirements of the solutions. Choosing NSL-KDD provides insightful analysis using various machine learning algorithms for intrusion detection. Initial process of this data analysis went through EDA (Exploratory Data Analysis) which includes pre-processing the data and creating meaningful visualizations to understand data better. Later, the data was prepared using hot-encoding for class variables to categorize data into 5 groups as Normal, Dos, Probe, R2L and U2R. Which was later helpful to understand any future instance in machine learning modeling process.

4.1 Architecture



4.2 Method and Materials:

In this section, the methodology of the research is discussed. According to the literature studies, there is a critical need for the creation of effective machine learning models for identifying attacks in datasets. The dataset NSL-KDD was analyzed and trained using three Machine Learning algorithms Decision Tree(DT), and Support Vector Machine (SVM).

Dataset:

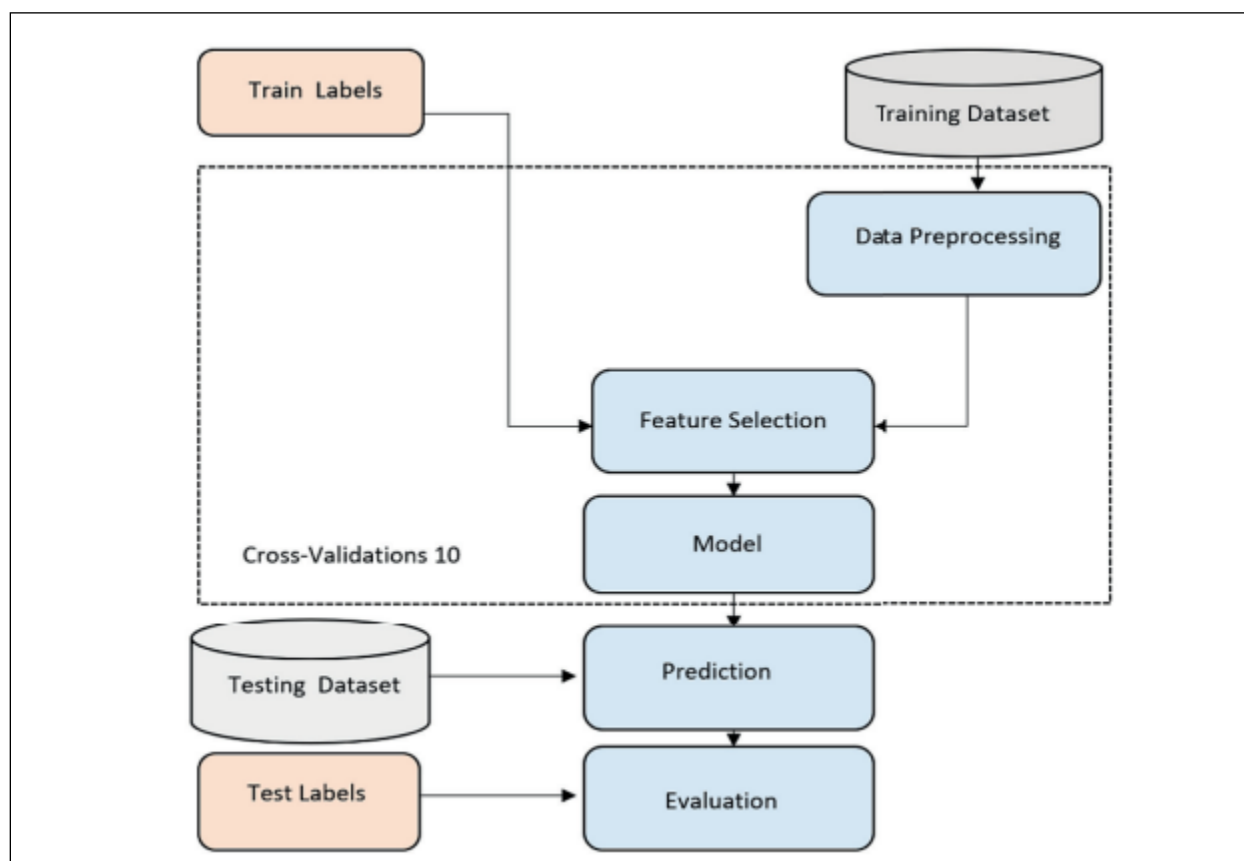
NSL-KDD is a condensed version of the original KDD dataset that was acquired from the Canadian Institute for Cybersecurity . It has the same features as KDD. Each record has 41 features and one class attribute. Each connection is classified as either an attack or a normal connection. NSL-KDD has a total of 39 attacks, each of which is classified into one of four categories: DOS, R2L, U2R, and Probing. For building our models, we used 25,192 instances as training. Next, these trained models were evaluated and tested using 11851 instances. Finally, the rest of the dataset was used for validation.

1. DOS: denial-of-service, which means preventing authorized users' access to a service, such as syn flooding.

2. R2L: This refers to breaking into a remote machine to get access to the victim's machine, such as guessing passwords.

3. U2R: When a normal account is used to log into a victim machine and tries to gain root privilege, using a technique such as a buffer overflow.

4. Probing: examining and scanning the victim's machine for vulnerabilities in terms of learning more about it, such as port scanning.

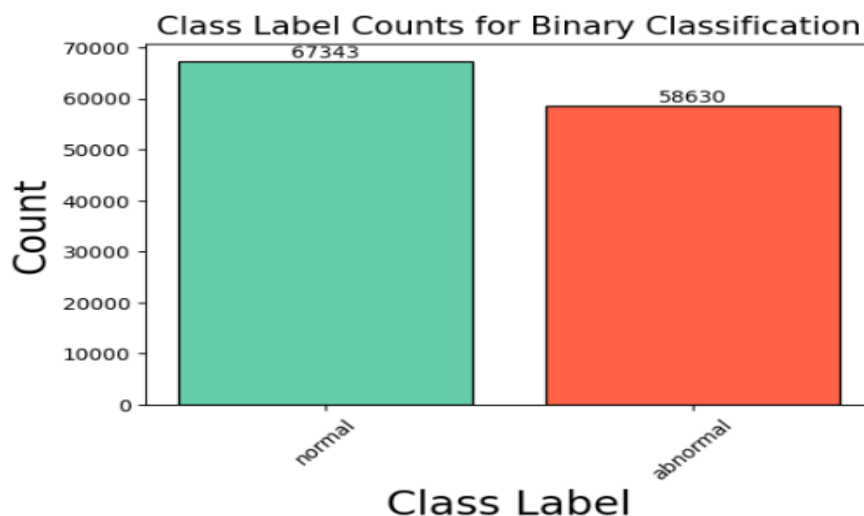


Flowchart sequence of steps for build IDS models

4.2.1 Data pre-processing:

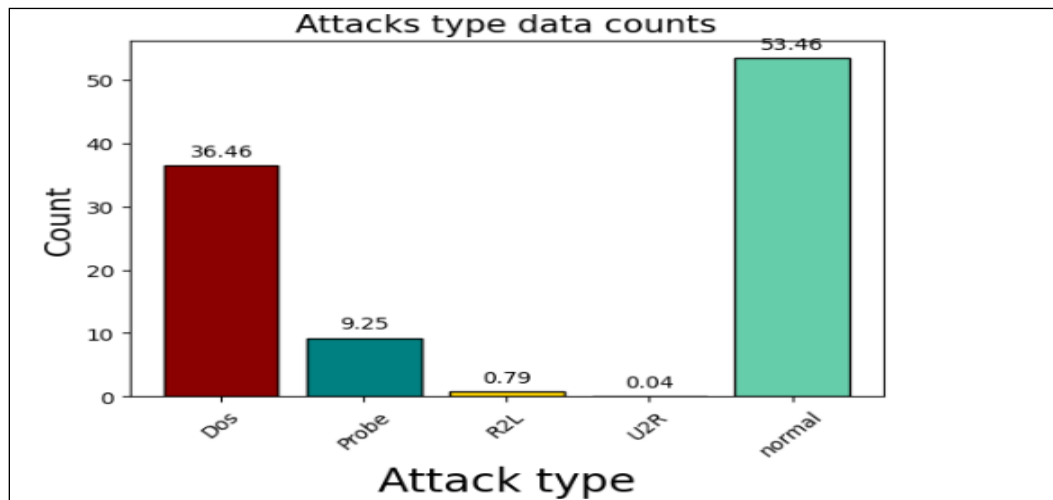
Pre-processing the data is a very important step in preparing the data to be fed into the algorithm. The goal of data preparation is to eliminate ambiguity in the dataset and provide IDS with accurate data. It unifies feature selection and normalization. Many symbolic attributes in the dataset, such as flags and protocol types, have nominal values. These values must be converted to numeric values for the dataset to perform better. Multi-class classification problems (4 attack classes and normal classes) and binary classification problems (normal or attack) have been transformed using discretized datasets.

Labels of dataset for binary class:



We mapped all attack instances as abnormal and normal instances as normal, transforming the NSL-KDD dataset into a binary classification task. This methodology outlines a systematic approach to preprocessing, feature selection, model training, and evaluation, leading to the development of an effective NIDS capable of distinguishing between normal and malicious network traffic.

Labels of dataset for multiclass



we mapped individual attack types to broader categories to facilitate a more structured and organized analysis of attack patterns and behaviors within the dataset. Specifically, attacks such as 'land', 'neptune', 'smurf', 'pod', 'back', and 'teardrop' were classified under the "Dos" category, while 'portsweep', 'ipsweep', 'satan', and 'nmap' were categorized as "Probe". Additionally, attacks like 'buffer_overflow', 'loadmodule', 'perl', and 'rootkit' were grouped into the "U2R" category. Any instance labeled as "normal" was directly mapped to the "normal" category, while all other types of attacks were classified under the "R2L" category.

4.2.1.1 One-Hot Encoding for Train and Test Datasets:

One-hot encoding is a critical preprocessing step in machine learning projects, particularly when dealing with categorical variables. By converting categorical data into a numerical format, we enable machine learning algorithms to effectively utilize these

features for model training and prediction. In this phase of our project, we performed one-hot encoding on both the training and test datasets (`train_df` and `test_df`).

Here's a concise overview of the process:

1. **One-Hot Encoding for Training Data:** We applied one-hot encoding to the entire training dataset using the `pd.get_dummies()` function. This transformation converted categorical variables into binary-encoded columns, representing each category as a separate binary feature.
2. **One-Hot Encoding for Test Data:** Similarly, we performed one-hot encoding on the test dataset (`test_df`) using the same method. It's essential to apply the same encoding scheme to both training and test datasets to ensure consistency in feature representation.

4.2.1.2 Feature selection:

It has been observed that feature selection methods have enhanced the quality of prediction model by eliminating the irrelevant features and utilized only important features while building the prediction models. The feature selection is an essential data pre-processing stage in data mining. Correlation is a popular and successful strategy for identifying the most closely linked characteristics in any dataset; it defines the strength

of the relationship between features, based on the assumption that features are conditionally independent given the class. A good feature subset contains features that are highly correlated (predictive of) the class yet uncorrelated and not predictive of one another.

All Features:

#	Feature	#	Feature
1	duration	22	is_guest_login
2	protocol_type	23	Count
3	service	24	srv_count
4	flag	25	serror_rate
5	src_bytes	26	srv_serror_rate
6	dst_bytes	27	error_rate
7	land	28	srv_error_rate
8	wrong_fragment	29	same_srv_rate
9	urgent	30	diff_srv_rate
10	hot	31	srv_diff_host_rate
11	num_failed_logins	32	dst_host_count
12	logged_in	33	dst_host_srv_count
13	num_compromised	34	dst_host_same_srv_rate
14	root_shell	35	dst_host_diff_srv_rate

15	su_attempted	36	dst_host_same_src_port_rate
16	num_root	37	dst_host_srv_diff_host_rate
17	num_file_creations	38	dst_host_serror_rate
18	num_shells	39	dst_host_srv_serror_rate
19	num_access_files	40	dst_host_error_rate
20	num_outbound_cmds	41	dst_host_srv_error_rate
21	is_host_login		

Selecting top 30 highly correlated features with respect to target variable : intrusion code

To identify the features most strongly associated with the target variable , we computed the Pearson correlation coefficients between all pairs of features in the **train and test** DataFrame. By taking the absolute values of these correlations, we were able to assess the strength of relationships without considering the direction. We then isolated the correlations involving target variable and sorted them in descending order. This process highlighted the top 30 features with the highest absolute correlations, providing a clear

indication of which features were most predictive of the target variable. This analysis informed our feature selection and helped refine our model to improve its predictive accuracy.

The Pearson correlation coefficient, denoted as r , measures the strength and direction of the linear relationship between two variables X and Y . The formula for calculating the Pearson correlation coefficient is:

$$r = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^n (X_i - \bar{X})^2} \sqrt{\sum_{i=1}^n (Y_i - \bar{Y})^2}}$$

Where X_i and Y_i are the individual sample points indexed by i , $\bar{X}$ and $\bar{Y}$ are the means of the X and Y values respectively, and n is the number of paired observations.

intrusion_code	1.000000
dst_host_srv_count	0.624241
logged_in	0.576031
flag_SF	0.548643
dst_host_same_srv_rate	0.525829
service_http	0.511499
same_srv_rate	0.510871
service_private	0.398863
dst_host_diff_srv_rate	0.391328
count	0.388358
dst_host_srv_serror_rate	0.384796
dst_host_serror_rate	0.382118
serror_rate	0.381646
srv_serror_rate	0.378669
flag_S0	0.373549
service_eco_i	0.338355
dst_host_same_src_port_rate	0.319096
dst_host_srv_rerror_rate	0.310227
srv_rerror_rate	0.309909
rerror_rate	0.308668
dst_host_rerror_rate	0.293805
protocol_type_icmp	0.292874
diff_srv_rate	0.282314
flag_RST	0.255984
dst_host_srv_diff_host_rate	0.239646
service_domain_u	0.229571
flag_REJ	0.198654
dst_host_count	0.196102
service_smtp	0.189656
protocol_type_udp	0.155004
Name: intrusion_code, dtype: float64	

4.2.1.3 Standardizing Data using StandardScaler:

Standardization plays a pivotal role in preprocessing our data, ensuring that features are uniformly scaled and preventing certain features from disproportionately influencing the learning process. By standardizing our data, we aimed to enhance the convergence speed of gradient-based algorithms and improve the overall stability and performance of our machine learning models.

1. **Conversion to DataFrame:** We converted the input data (NumPy array **X_train**) to a DataFrame (**X_train_df**) using pandas for easier manipulation.
2. **Column Name Reset:** Column names were reset in the DataFrame to maintain consistency, ensuring each feature was appropriately labeled.
3. **Standardization:** A StandardScaler object (**scaler**) was fitted to **X_train_df**, computing mean and standard deviation for each feature.
4. **Transformation:** The fitted scaler was used to transform the training data, standardizing each feature with mean 0 and standard deviation 1.

4.3 Experiments:

Model Training :

After preprocessing the dataset, we trained three different machine learning models: K-Nearest Neighbors (KNN), Decision Tree, and Support Vector Machine (SVM).

4.3.1 K-Nearest Neighbors Classifier:

We used the **KNeighborsClassifier** from the **scikit-learn** library. The KNN classifier was initialized using the default parameters provided by the library, which included

n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, and p=2. The **fit** method was then called on the classifier object with the feature matrix and the target variable. This method computed the necessary information to classify new data points based on their nearest neighbors in the training set.

This approach enabled us to classify both binary and multi-class datasets effectively, providing a robust baseline for further model evaluation and tuning.

4.3.2 Decision Tree Classifier:

We utilized the **DecisionTreeClassifier** from the **scikit-learn** library. The Decision Tree was initialized with default parameters provided by the library, which included

- **criterion='gini'**: Measures the quality of a split.
- **splitter='best'**: Chooses the best split.
- **max_depth=None**: Nodes are expanded until all leaves are pure or contain fewer than **min_samples_split** samples.
- **min_samples_split=2**: Minimum number of samples required to split an internal node.
- **min_samples_leaf=1**: Minimum number of samples required to be at a leaf node.

The model was trained using the **fit** method with the feature matrix and target variable, enabling effective classification of both binary and multi-class datasets.

4.3.3 Support Vector Machine:

We employed the **SVC** (Support Vector Classifier) from the **scikit-learn** library.

The SVM classifier was initialized with default parameters:

- **C=1.0**: Regularization parameter.
- **kernel='rbf'**: Uses the radial basis function kernel.
- **degree=3**: Degree of the polynomial kernel function (if **poly** is used).
- **gamma='scale'**: Kernel coefficient for **rbf**, **poly**, and **sigmoid**.
- **coef0=0.0**: Independent term in kernel function for **poly** and **sigmoid**.

The model was trained using the **fit** method with the feature matrix and target variable, enabling effective classification using Support Vector Machines (SVM).

Results

After training the models, we used a separate test dataset to evaluate their performance and determine their accuracy. This process involves feeding the test dataset into the trained models and comparing the predicted labels with the actual labels in the test dataset. The accuracy of each model is calculated as the percentage of correct predictions over the total number of predictions.

Evaluation metric:

Standard performance metrics for evaluation including accuracy, precision, recall, F1-Score were used. Given that,

True Positive (TP): is the number of samples correctly detected as malware

False Positive (FP): is the number of samples incorrectly detected as malware

True Negative (TN): is the number of samples correctly detected as benign

False Negative (FN): is the number of samples incorrectly detected as benign.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN},$$

Precision (P): is the number of true positives predictions (TP) divided by all true positive predictions (TP) plus the number of false positives (FP).

$$Precision = \frac{TP}{TP + FP}$$

Recall (R): is the number of true positives (TP) divided by the number of true positives plus the number of false negatives (FN).

$$Recall = \frac{TP}{TP + FN},$$

F1 Score: F1 score is the weighted average of precision and recall.

$$F1-Score = 2 * \frac{Recall * Precision}{Recall + Precision}$$

5.1 Results of Binary Classification:

5.1.1 performance result:

	KNN	DECISION TREE	SVM
accuracy	93.83%	92.57%	91.74%

The experimental results reveal that K-Nearest Neighbors (KNN) achieved the highest accuracy of 93.83%, followed by the Decision Tree classifier with an accuracy of 92.57%. Support Vector Machine (SVM) attained an accuracy of 91.75%. These findings suggest that KNN outperforms both Decision Tree and SVM in terms of classification accuracy for binary classification.

Test Result of KNN:

...	precision	recall	f1-score	support
abnormal	0.92	0.97	0.95	2543
normal	0.96	0.90	0.93	1966
accuracy			0.94	4509
macro avg	0.94	0.93	0.94	4509
weighted avg	0.94	0.94	0.94	4509

The above classification report illustrates that for the 'abnormal' class, the precision is 0.92, recall is 0.97, and F1-score is 0.95, with a support of 2543 instances. Conversely, for the 'normal' class, precision stands at 0.96, recall at 0.90, and F1-score at 0.93, with a support of 1966 instances. The overall accuracy of the model is 0.94, suggesting a balanced performance across classes.

Test Result of Decision Tree:

...	precision	recall	f1-score	support
abnormal	0.93	0.94	0.93	2543
normal	0.92	0.91	0.91	1966
accuracy			0.93	4509
macro avg	0.93	0.92	0.92	4509
weighted avg	0.93	0.93	0.93	4509

The precision, recall, and F1-score for the 'abnormal' class are 0.93, 0.94, and 0.93, respectively, with a support of 2543 instances. Conversely, for the 'normal' class, precision, recall, and F1-score stand at 0.92, 0.91, and 0.91, with a support of 1966 instances. The overall accuracy of the model is 0.93, indicating robust performance in distinguishing between 'abnormal' and 'normal' classes.

Test Result of SVM:

...	precision	recall	f1-score	support
abnormal	0.91	0.95	0.93	2543
normal	0.93	0.87	0.90	1966
accuracy			0.92	4509
macro avg	0.92	0.91	0.92	4509
weighted avg	0.92	0.92	0.92	4509

The above classification report indicates a precision of 0.91 and recall of 0.95 for the 'abnormal' class, with an F1-score of 0.93 and support of 2543 instances. Conversely, for the 'normal' class, precision is 0.93, recall is 0.87, and F1-score is 0.90, with a support of 1966 instances. The overall accuracy of the model is 0.92, reflecting its capability to differentiate between 'abnormal' and 'normal' classes with satisfactory precision and recall rates.

5.2 Results of Multi Classification:

5.2.1 Performance Results:

Accuracy:

	KNN	DECISION TREE	SVM
accuracy	92.40%	94.11%	89.57%

The experimental results for multi-class classification demonstrate that the decision tree classifier achieved the highest accuracy of 94.11%, followed by K-Nearest Neighbors (KNN) with an accuracy of 92.41%, and Support Vector Machine (SVM) with an accuracy of 89.57%. These findings suggest that the decision tree classifier outperforms both KNN and SVM in accurately classifying instances across multiple classes.

Test Result for KNN:

[bb]					
...		precision	recall	f1-score	support
	0	0.91	0.97	0.94	9711
	1	0.95	0.89	0.92	5741
	2	0.87	0.91	0.89	1106
	3	0.96	0.91	0.93	5949
	4	0.61	0.30	0.40	37
	accuracy			0.93	22544
	macro avg	0.86	0.80	0.82	22544
	weighted avg	0.93	0.93	0.93	22544

The above classification report presents precision, recall, and F1-score for multiple classes. For class 0(Normal), precision is 0.91, recall is 0.97, and F1-score is 0.94 with a support of 9711 instances. Similarly, for class 1(Dos), precision is 0.95, recall is 0.89, and F1-score is 0.92 with a support of 5741 instances. Class 2(Probe) exhibits a precision of 0.87, recall of 0.91, and F1-score of 0.89 with 1106 instances. Class 3(R2L) demonstrates a precision of 0.96, recall of 0.91, and F1-score of 0.93 with 5949 instances. Lastly, class 4(U2R) displays a precision of 0.61, recall of 0.30, and F1-score of 0.40 with 37 instances. The overall accuracy of the model is 0.93, indicating robust performance across multiple classes.

Test Result for DECISION TREE

...		precision	recall	f1-score	support
	0	0.93	0.99	0.96	9711
	1	0.99	0.91	0.95	5741
	2	0.96	0.96	0.96	1106
	3	0.97	0.96	0.96	5949
	4	1.00	0.95	0.97	37
	accuracy			0.96	22544
	macro avg	0.97	0.95	0.96	22544
	weighted avg	0.96	0.96	0.96	22544

The classification report demonstrates high precision, recall, and F1-score across multiple classes. For class 0(Normal), precision is 0.93, recall is 0.99, and F1-score is 0.96 with a support of 9711 instances. Class 1(Dos) exhibits precision of 0.99, recall of 0.91, and F1-score of 0.95 with 5741 instances. Class 2(Probe) displays precision of 0.96, recall of 0.96, and F1-score of 0.96 with 1106 instances. Similarly, class 3(R2L) demonstrates precision of

0.97, recall of 0.96, and F1-score of 0.96 with 5949 instances. Lastly, class 4(U2R) achieves perfect precision, recall, and F1-score of 1.00 with 37 instances. The overall accuracy of the model is 0.96, indicating robust performance across multiple classes.

Test Result for SVM:

...	precision	recall	f1-score	support
0	0.93	0.99	0.96	9711
1	0.99	0.91	0.95	5741
2	0.96	0.96	0.96	1106
3	0.97	0.96	0.96	5949
4	1.00	0.95	0.97	37
accuracy			0.96	22544
macro avg	0.97	0.95	0.96	22544
weighted avg	0.96	0.96	0.96	22544

The classification report demonstrates high precision, recall, and F1-score across multiple classes. For class 0(Normal), precision is 0.93, recall is 0.99, and F1-score is 0.96 with a support of 9711 instances. Class 1(Dos) exhibits precision of 0.99, recall of 0.91, and F1-score of 0.95 with 5741 instances. Class 2(Probe) displays precision of 0.96, recall of 0.96, and F1-score of 0.96 with 1106 instances. Similarly, class 3(R2L) demonstrates precision of 0.97, recall of 0.96, and F1-score of 0.96 with 5949 instances. Lastly, class 4(U2R) achieves perfect precision, recall, and F1-score of 1.00 with 37 instances. The overall accuracy of the model is 0.96, indicating robust performance across multiple classes.

Conclusion

6.1 Conclusion:

In this project, we developed and evaluated machine learning models for network intrusion detection using the NSL-KDD dataset. Our approach involved extensive data preprocessing, including feature selection and one-hot encoding, to ensure the dataset was suitable for model training. We mapped attack types into broader categories for multi-class classification and simplified the dataset for binary classification.

Three different classifiers—K-Nearest Neighbors (KNN), Decision Tree, and Support Vector Machine (SVM)—were trained and tested. The KNN classifier demonstrated strong performance in distinguishing between normal and abnormal network traffic, though it was sensitive to feature scaling and computationally intensive for larger datasets. The Decision Tree classifier provided a clear and interpretable model structure but exhibited overfitting at higher depths. The SVM classifier, with its robustness and ability to handle high-dimensional data, showed competitive performance, although it required careful tuning of hyperparameters.

The overall results indicated that the Decision Tree classifier achieved the highest accuracy, followed closely by KNN and SVM.

References:

- [1] https://www.iraj.in/journal/journal_file/journal_pdf/12-422-1554182717129-135.pdf
- [2] Trivedi, Harsh, Kumar(2015) 'A Brief STUDIES on Machine Learning Techniques.' IJSET - International Journal of Innovative Science, Engineering & Technology, 2(1), 47-48.
- [3] Reinforcement learning
- [4] Ilemona Solomon Atawodi(2019), A MACHINE LEARNING APPROACH TO NETWORK INTRUSION DETECTION SYSTEM USING K NEAREST NEIGHBOR AND RANDOM FOREST, 16.
- [5] José, I. (2018). KNN (K-Nearest Neighbors) #1. Retrieved from Towards Data Science : <https://towardsdatascience.com/knn-k-nearest-neighbors-1-a4707b24bd1d>
- [6] Ilemona Solomon Atawodi(2019), A MACHINE LEARNING APPROACH TO NETWORK INTRUSION DETECTION SYSTEM USING K NEAREST NEIGHBOR AND RANDOM FOREST, 17-18.
- [7] S. C. L. Shilpashree. S, Nayana G Bhat, Sunil Kumar G, "Decision Tree: A Machine Learning for Intrusion Detection," International Journal of Innovative Technology and Exploring Engineering (IJITEE), April 2019 [Online]. Available: <https://www.ijitee.org/wp-content/uploads/papers/v8i6s4/F12340486S419.pdf>
- [8] Singh, Nagesh (2020a). Decision Tree Algorithm. <https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html>.
- [9] M.-S. A. Fares Fourati, Fellow. , "Artificial Intelligence for Satellite Communication: A Review," IEEE, Jan 25, 2021 [Online]. Available: <https://arxiv.org/pdf/2101.10899.pdf>
- [10] N. S. Chauhan. "Decision Tree Algorithm." KDnuggets. February 9, 2022 [Online]. Available: <https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html>
- [11] A. Navlani. "Decision Tree Classification in Python." datacamp. December 29, 2018 [Online]. Available: <https://www.datacamp.com/community/tutorials/decision-treeclassification-python>.
- [12] <https://www.baeldung.com/cs/svm-multiclass-classification#:~:text=In%20its%20most%20simple%20type,into%20multiple%20binary%20classification%20problems>.

- [13] DP Gaikwad and Ravindra C Thool. "Intrusion detection system using bagging with partial decision treebase classifier". In: Procedia Computer Science 49 (2015), pp. 92–98.
- [14] Maryna Krotofil and Dieter Gollmann. "Industrial control systems security: What is happening?" In: 2013 11th IEEE International Conference on Industrial Informatics (INDIN). IEEE. 2013, pp. 670–675.
- [15] Manjula C. Belavagi and Balachandra Muniyal. "Performance Evaluation of Supervised Machine Learning Algorithms for Intrusion Detection". In: Procedia Computer Science 89 (2016). Twelfth International Conference on Communication Networks, ICCN 2016, August 19– 21, 2016, Bangalore, India Twelfth International Conference on Data Mining and Warehousing, ICDMW 2016, August 19-21, 2016, Bangalore, India Twelfth International Conference on Image and Signal Processing, ICISP 2016, August 19-21, 2016, Bangalore, India, pp. 117– 123. issn: 1877-0509.
- [16] A. A. Aburomman and M. Bin Ibne Reaz. "Ensemble of binary SVM classifiers based on PCA and LDA feature extraction for intrusion detection". In: 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC). 2016, pp. 636–640.
- [17] K. Atefi and S. Yahya. "Anomaly detection based on profile signature in network using machine learning technique". In: 2016 IEEE Region 10 Symposium (TENSYPMP). 2016, pp. 71–76.

[18] Wei-Chao Lin. "CANN: An intrusion detection system based on combining cluster centers and nearest neighbors". In: Knowledge-based systems 78 (2015), pp. 13–21.

[19] V. N. Vapnik, The nature of statistical learning theory, Information and Statistics Series, 1995, Springer.

[20] Dr. S.Vijayarani¹ and Ms. Maria Sylvia's "Intrusion detection system-A study" IJSPTM, Vol 4,No1,February 2015