TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

Kathmandu Engineering College

Department of Computer Engineering

MAJOR PROJECT REPORT

ON

NETWORK INTRUSION DETECTION SYSTEM USING VARIOUS MACHINE LEARNING ALGORITHMS

[Code No: CT 755]

By:

Pranjal Bhatta - 39395

Saransh Bhatta - 39418

Sudhanshu Joshi - 39428

Yubraj Sigdel - 39438

Kathmandu, Nepal

BAISAKH, 2080

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

Kathmandu Engineering College

Department of Computer Engineering

NETWORK INTRUSION DETECTION SYSTEM USING VARIOUS MACHINE LEARNING ALGORITHMS

[Code No: CT 755]

PROJECT REPORT SUBMITTED TO

THE DEPARTMENT OF COMPUTER ENGINEERING

IN FULFILLMENT OF THE REQUIREMENTS FOR

THE BACHELOR OF ENGINEERING

By:

Pranjal Bhatta - 39395

Saransh Bhatta - 39418

Sudhanshu Joshi - 39428

Yubraj Sigdel - 39438

Kathmandu, Nepal

BAISAKH, 2080

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

Kathmandu Engineering College

Department of Computer Engineering

CERTIFICATE

The undersigned certify that they have read and recommended to the Department of Computer Engineering, a major project work entitled "Network Intrusion Detection System Using Various Machine Learning Algorithms" submitted by Pranjal Bhatta- 39395, Saransh Bhatta- 39418, Sudhanshu Joshi- 39428 and Yubraj Sigdel- 39438 in fulfillment of the requirements for the degree of Bachelor of Engineering.

Associate Prof. Dr. Nanda Bikram Adhikari

(External Examiner)

Pulchowk Campus, Lalitpur

Assoc. Prof. Sudeep Shakya

(Project Supervisor)

Head of Department

Department of Computer Engineering

Kathmandu Engineering College

Assoc. Prof. Sharad Chandra Joshi

(Project Coordinator)

Department of Computer Engineering

Kathmandu Engineering College

ACKNOWLEDGEDMENT

We would like to express our heartfelt gratitude to the **Department of Computer Engineering,** Kathmandu Engineering College for providing us such a great opportunity. To undertake this project. This project will help us develop our skills related to programming for future reference. We express our sincere gratitude to **Associate Prof. Sudeep Shakya,** the Head of Computer Department as well as our project supervisor,for the necessary templates and documents which has helped us a lot in the completion of our project, and **Associate Prof. Kunjan Amatya,** the Deputy Head of Computer Department for his indispensable guidance and support to harness our skills. We acknowledge the help and support provided by the project coordinator **Associate Prof. Sharad Chandra Joshi.**

We are thankful to our teachers, friends and seniors for providing valuable ideas and pointing out our improvement areas. Finally, we would give our sincere thanks to all those who were involved in any way in our project and are grateful for their contribution.

ABSTRACT

The rapid advances in the internet and communication fields have resulted in a huge increase in the network size and the corresponding data. With the advancement of internet over the years, the number of attacks over the internet have been increasing. An intrusion detection system (IDS) is one such tool that prevents the network from possible intrusions by inspecting the network traffic, to ensure its confidentiality, integrity, and availability. Machine Learning algorithms are applied in IDS in order to identify and classify security threats. The system takes in data in the form of a .pcap file, converts it to equivalent KDD csv file and fed to the machine learning algorithm. This report explores and evaluates various machine learning algorithms such as Random forest, k-nearest neighbor, Support vector machine, Decision tree, Logistic regression, Gradient boosting, Adaboost classifier using NSL-KDD dataset. Experimental results are demonstrated that include: accuracy, precision, f1 score and recall.

Keywords: *intrusion*, *confidentiality*, *Adaboost, NSL-KDD*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
CART	Classification And Regression Tree
CPU	Central Processing Unit
CSV	Comma Separated Value
DL	Deep Learning
DNS	Domain Name System
DoS	Denial of Service
DT	Decision Tree
EDA	Exploratory Data Analysis
GPU	Graphical Processing Unit
IoT	Internet of Things
KNN	K-Nearest Neighbor
ML	Machine Learning
MVC	Model View Controller
NSL-KDD	Network Socket Layer-Knowledge Discovery of Dataset
PCAP	Packet Capture
R2L	Root-to-local
SVM	Support Vector Machine
U2R	User-to-root
WEB-APP	Web Application

CHAPTER 1: INTRODUCTION

As the world advances further into digital technology, various IoT integrated devices have become ubiquitous. The increase in network traffic means network security is critical for safe, secure and private flow of data. An intrusion detection and classification system can be used for alerting and informing any respective user about harm and danger that may happen to the network. The system monitors a network and searches for any anomalies or deviations from normality. Any traffic that is deemed to be potentially dangerous may be flagged and sent to a human user for further review. This form of abnormality detection lends itself well to machine learning and machine learning concepts. Hence, An intrusion detection and classification system built using machine learning can prove to be highly effective and precise.

1.1 BACKGROUND THEORY

The system is designed with machine learning in mind, as it can help with determining anomalies. Anything that is found to be a deviation from expected behavior is considered as an anomaly. As such, the system is trained based on what is considered normal and reports deviations from that as alerts. Anomaly detection approaches have to grapple with some previously defined problems such as: false positives, finding datasets, and the attacker learning how to bypass the system.

Fundamentally, Random Forest/Gradient Boosting or any other machine-learning algorithm tends to be much better at finding similarities than dis-similarities. Therefore, it is important to properly classify what traffic would be considered “normal” and what is “abnormal". However, network traffic often exhibits much more diversity than that can be easily modeled for, which can cause issues when looking for outliers or anomalies as they can easily just be another “normal” traffic that wasn’t accounted for during modeling. The major way to deal with this issue is aggregation. While highly variable over small-to-medium time intervals, traffic properties tend to have greater stability when observed over longer time periods (hours to days, sometimes weeks). For example, in most networks time-of-day and day-of-week effects exhibit reliable patterns: if during today’s lunch break, the traffic volume is

twice as large as during the corresponding time slots last week, that likely reflects something unusual occurring [1].

We can identify “normal” traffic by way of feature selection. Few major features are selected for comparison which, for example, may include “Record total duration”, “Source to Destination transaction bytes”, “Source bits per second”, “Destination bits per second”, “No of connections from the same source or destination address” and so on. These features and more are selected to be trained and compared to determine the nature of any incoming traffic.

Hence, by taking a large enough dataset that contains real network traffic from as large an environment as possible; and ideally multiple of these from different networks, we can properly model and train a system that reasonably detects and classifies any unusual or abnormal traffic in a network.

1.2 DATASET DESCRIPTION

We applied various machine learning techniques to the NSL-KDD dataset that was taken from Kaggle**.** NSL-KDD is a dataset suggested to solve some of the inherent problems of the KDD'99 data set which are mentioned in [2]. It does not include redundant records in the train set, so the classifiers will not be biased towards more frequent records. There is no duplicate records in the proposed test sets; therefore, the performance of the learners are not biased by the methods which have better detection rates on the frequent records. This dataset is 41 featured dataset with the attack divided into four classes; DoS, Probing, user-to-root (U2R), and root-to-local (R2L).

Denial of Service (DoS): is an attack in which the attacker make some computing or memory resource too busy or too full to handle legitimate requests or denies legitimate user access to a machine.

Probing attack: is an attempt to gather information about a network of computers for the apparent purpose of circumventing its security controls.

User to Root(U2R): is a class of exploit in which the attacker starts out with access to a normal user account on the system and is able to exploit some vulnerability to gain root access to the system.

Root to Local(R2L): occurs when an attacker who has the ability to send packets to a machine over a network but who does not have an account on that machine exploits some vulnerability to gain local access as a user of that machine.

The number of records in the train and test sets are reasonable, which makes it affordable to run the experiments on the complete set without the need to randomly select a small portion.

NSL-KDD features can be classified into three categories:

1. Basic features: this category encapsulates all the attributes that can be extracted from TCP/IP connection.

2. Traffic features: this category includes features that are computed with respect to a window interval and is divided into two groups:

a) **"same host" features:** examines only the connections in past 2 seconds that have same destination host as current connection and calculate statistics related to protocol behavior, service, etc.

b) **"same service" features:** examine only the connection in past 2 seconds that have same service as current connection.

3. Content features: unlike most of the DoS and Probing attacks, the R2L and U2R attacks don't have any intrusion frequent sequential patterns. This is because the DoS and Probing attacks involve many connections to some host(s) in a very short period of time; however the R2L and U2R attacks are embedded in the data portions of the packets, and normally involves only a single connection. To detect these kinds of attacks, we need some features to be able to look for suspicious behavior in the data portion, e.g., number of failed login attempts [18].

```
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   duration                     494020 non-null  int64
 1   protocol_type                494020 non-null  object
 2   service                      494020 non-null  object
 3   flag                         494020 non-null  object
 4   src_bytes                    494020 non-null  int64
 5   dst_bytes                    494020 non-null  int64
 6   land                         494020 non-null  int64
 7   wrong_fragment               494020 non-null  int64
 8   urgent                       494020 non-null  int64
 9   hot                          494020 non-null  int64
 10  num_failed_logins            494020 non-null  int64
 11  logged_in                    494020 non-null  int64
 12  lnum_compromised             494020 non-null  int64
 13  lroot_shell                  494020 non-null  int64
 14  lsu_attempted                494020 non-null  int64
 15  lnum_root                    494020 non-null  int64
 16  lnum_file_creations          494020 non-null  int64
 17  lnum_shells                  494020 non-null  int64
 18  lnum_access_files            494020 non-null  int64
 19  lnum_outbound_cmds           494020 non-null  int64
 20  is_host_login                494020 non-null  int64
 21  is_guest_login               494020 non-null  int64
 22  count                        494020 non-null  int64
 23  srv_count                    494020 non-null  int64
 24  serror_rate                  494020 non-null  float64
 25  srv_serror_rate              494020 non-null  float64
 26  rerror_rate                  494020 non-null  float64
 27  srv_rerror_rate              494020 non-null  float64
 28  same_srv_rate                494020 non-null  float64
 29  diff_srv_rate                494020 non-null  float64
 30  srv_diff_host_rate           494020 non-null  float64
 31  dst_host_count               494020 non-null  int64
 32  dst_host_srv_count           494020 non-null  int64
 33  dst_host_same_srv_rate       494020 non-null  float64
 34  dst_host_diff_srv_rate       494020 non-null  float64
 35  dst_host_same_src_port_rate  494020 non-null  float64
 36  dst_host_srv_diff_host_rate  494020 non-null  float64
 37  dst_host_serror_rate         494020 non-null  float64
 38  dst_host_srv_serror_rate     494020 non-null  float64
 39  dst_host_rerror_rate         494020 non-null  float64
 40  dst_host_srv_rerror_rate     494020 non-null  float64
 41  label                        494020 non-null  object
dtypes: float64(15), int64(23), object(4)
```

Figure 1.1: List of Features

1.3 PROBLEM STATEMENTS

We live in a time where the majority of our information is linked with the internet. Keeping these information safe from unauthorized persons is very important. The amount of data stolen or misused from the internet is absurd. Data breachers are using advanced tools and techniques to get access of the devices where the data is stored. Almost all of the data breaches are done through the internet. So having a good intrusion detection system in the network is a good option to secure data and personal information.

Looking into the past, there have been many such data breaches into the local network and that into big companies. This shows us that even successful giant companies are in threat of intruders. If we detect the intruder in the network and inform the user about the intrusion then the user will have the chance to secure their data. Using machine learning algorithms relevant to the implementation of the system, we can reliably detect such attempts of intrusion.

1.4 OBJECTIVE

The objective of our project is:

- To create a system that detects and classifies intrusion in a network using packet capture (.pcap) file.
- Providing a tool to generate report for network engineers for investigation regarding intrusion attempt.

1.5 SCOPE AND APPLICATIONS

With the advent of novel innovations and policies such as e-governance, digital banking and so on. Protection and security of a network is vital to any organization, entity or overarching structure that wishes for its data and system to be reliable and secure. A system that carries the heavy load of protecting the provided network from unwanted traffic and malicious attacks will be of utmost importance.

The system will be applicable for entities of any size that wish to add an additional layer in their network security infrastructure. Moreover, the system can also be used for stress testing purposes of existing security framework and infrastructure.

Hence, the application of the system is as follows:

- Provide an additional layer of security for a network.
- It can be used to help analyze the quantity and types of attacks. Organizations can use this information to change their security systems or implement more effective controls.

CHAPTER 2: LITERATURE REVIEW

Network intrusion detection has been investigated extensively. As a result, to gain a better understanding of the field, different methodologies must be classified and evaluated. Because Machine Learning (ML) and Deep Learning (DL) are continuously growing research topics, we look at several models and the techniques they have used in these two fields for less than a decade.

Mustafa et al. [3] proposed a beta mixture model for anomaly detection. The author computed a normal network profile, calculated the deviations from that network profile, and considered that as an anomaly. The UNSW NB15 dataset was used for performance evaluation using external evaluation metrics. This technique slightly improved the accuracy of the UNSW-NB15 dataset. In another study [4], the authors use ANNs for signature-based intrusion detection in IoT devices. They tested the approach on various malicious and heterogeneous data, achieving an average accuracy of 84% with an average false-positive rate of 8%.

Benjamin et al. [5] proposed an ensemble-based intrusion detection technique against application layer protocols (MQTT, HTTP, and DNS), as shown in Figure 2.1. The authors generated the new statistical flow features from the protocols based on their potential properties. The experimental analysis showed that statistically generated features are related to normal and suspected activities. Ensemble-based Adaboost was developed using naive Bayes, decision trees, and artificial neural network (ANN). NIMS botnet was used to detect and evaluate ensemble learning.

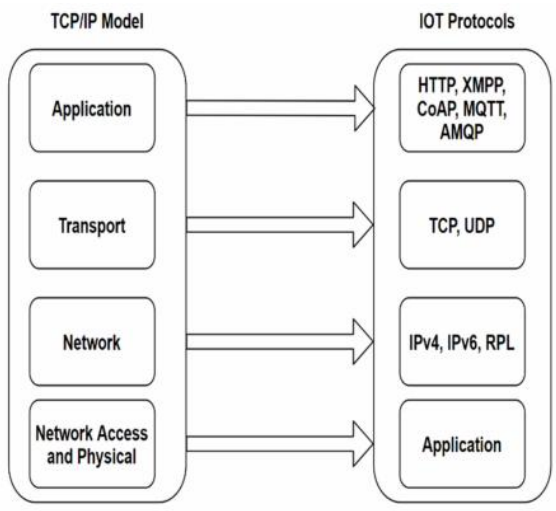


Figure 2.1: TCP/IP Model and IoT Protocol

Wei et al.[6] proposed an Adaboost-based network intrusion detection system based on the principle of weak learners. They used naive Bayes as weak learners and boosted the performance utilizing the weight update ability of the Adaboost. Similarly, Yuan et al.[7] proposed a semi-supervised tri-Adaboost based methodology for network intrusion detection. They suggested using three distinct weak Adaboost-based classifiers for training their model. The dimension of the feature is reduced using the chi-square methodology.

Giacinto and Roli [8] developed a network intrusion system using an automated design of classifier systems. Intrusions were detected based on the voting rule method. The experimental results showed that hybrid classification techniques are effective if each classifier performs well. Redundant features have little contribution to NIDS. Srilatha and Johnson [9] focused on examining and finding potential data features for intrusion detection systems. They aimed to design an optimized and computationally effective NIDS. Srilatha proposed a hybrid technique involving ensemble-based classifiers. PCA was used to reduce input dimensions, but has not achieved the expected results in the NIDS domain. MIT Lincoln laboratory prepared the dataset for the evaluation of the proposed solution.

A study by [10] gives an overview of Machine Learning and Deep Learning approaches in cybersecurity, focusing on network intrusion detection. They look at methods like k-Nearest Neighbor, Support Vector Machines, Deep Belief Networks, Decision Trees Recurrent Neural Networks, and Convolutional Neural Networks. They take notice of three issues in this overview: The scarcity of benchmark datasets and the non-uniformity of evaluation metrics, which makes comparison difficult, and finally, the lack of focus on algorithm efficiency. They also point out several trends in intrusion detection research, including hybrid model research, deep learning's prospects and problems, the expanding number of papers evaluating different algorithms and their applicability, and the potential for new benchmark datasets.

Several anomaly detections approaches are considered by [11]. Misuse-based algorithms like Support Vector Machines and rule-based approaches are among their anomaly detection techniques. They also explore IDS datasets and evaluate anomaly detection algorithms based on computational complexity, output format, and attack

priority. However, because this analysis appears to be limited to DARPA/KDD Cup attacks, it has limited application to more current datasets.

A study of the application of machine learning to several domains of networking, including network security, is presented in [12]. They look at 36 techniques to evaluate misuse-based, anomaly-based, Deep and Reinforcement Learning-based, and hybrid intrusion detection using the KDD Cup 1999 (a dataset developed during a competition called the Third International Knowledge Discovery and Data Mining Tools Competition), and NSL-KDD (Network Security Laboratory Knowledge Discovery and Data Mining) datasets. They also discovered that there is a lack of recent datasets, real-world anomaly based detection systems, insufficient real-time implementations, and a general lack of systems that meet other unique requirements. Finally, they offer a broader view of networking and point to the need for real-world data rather than generated datasets, as well as uniform evaluation metrics to make comparisons easier.

The researchers [13], investigate the use of machine learning and data mining to identify intrusions. Decision Trees, Clustering, Bayesian Networks, Artificial Neural Networks, Association Rules, Ensemble Learning, Evolutionary Computation, Hidden Markov models, Naive Bayes, Sequential Pattern Mining, Inductive Learning, and Support Vector Machines are some of the methodologies available.

From the above literature review, we observed that that previous research uses Adaboost mainly for the weight update property and does not consider the importance of discriminative features to distinguish the threats from normal network activity.

Thus, we believe in using a well-optimized (NSL-KDD) dataset selecting the most discriminative features, and utilizing Adaboost as a weight update model. At the same time, the decision tree is a primary classifier. Decision trees performs exceptionally well on network data. The weight update in the algorithm is done by Adaboost, which shows promising results when used in combination with the decision tree.

CHAPTER 3: METHODOLOGY

3.1 PROCESS MODEL

The process model we have used is Incremental process model.The incremental process model is also known as successive version model. First, a simple working system implementing only the basic features is built and is delivered to the users. Then many successive iterations are implemented and delivered to the users until the desired system is realized. It involves both development and maintainance. The product is finished when all the requirements are satisfied. This model combines the elements of the waterfall model with iterative philosophy of prototyping.We will use an incremental model for the development of our project. Our main focus at the beginning was based on the core objective of our product. Development process revolves around the idea of establishing a user friendly and effective product for users.

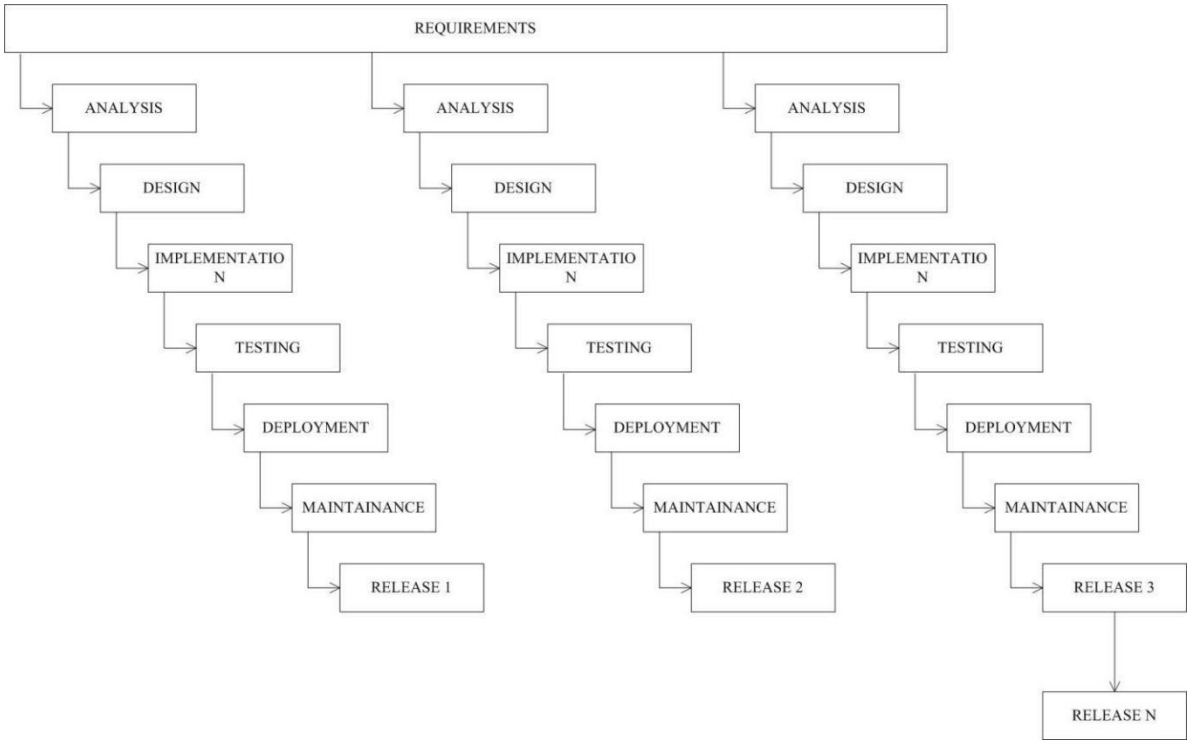


Figure 3.1: Incremental Process Model

3.2 BLOCK DIAGRAM

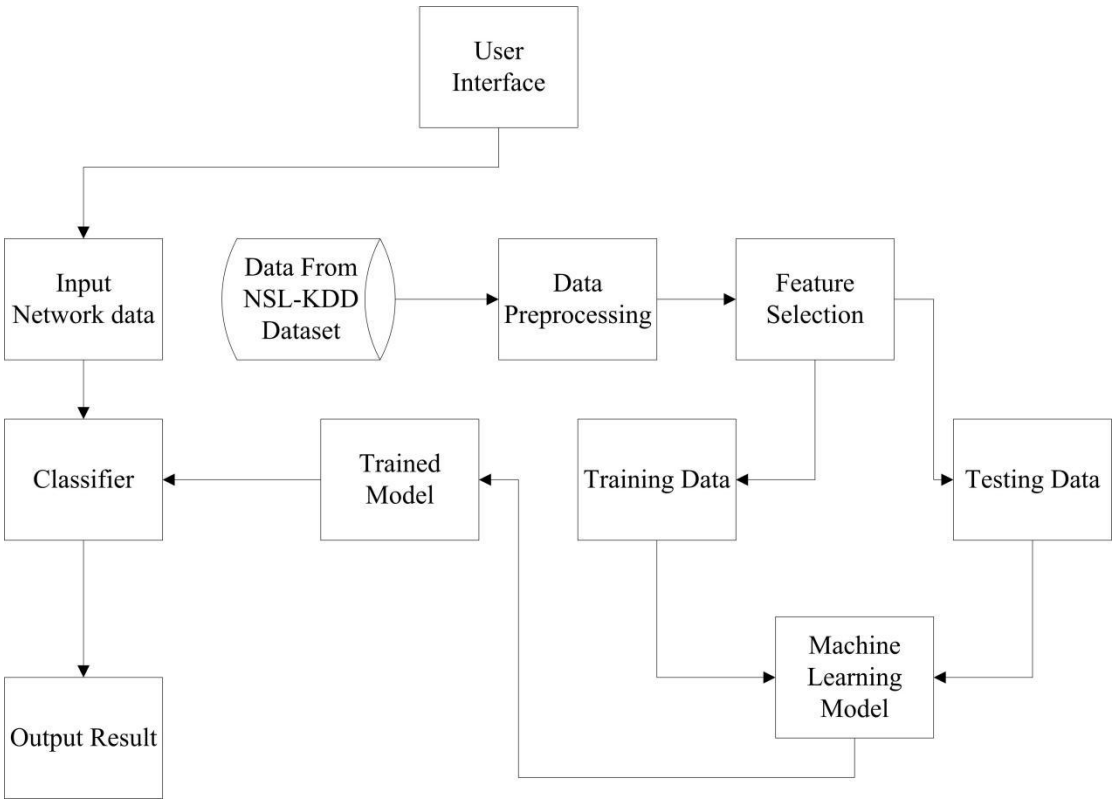


Figure 3.2 System Block Diagram

3.3 ALGORITHM

3.3.1 MACHINE LEARNING

Machine learning is an application of AI that enables systems to learn and improve from experience without being explicitly programmed. It focuses on developing computer programs that can access data and use it to learn for themselves. Similar to how the human brain gains knowledge and understanding, machine learning relies on input, such as training data or knowledge graphs, to understand entities, domains and the connections between them.

ML has proven valuable because it can solve problems at a speed and scale that cannot be duplicated by the human mind alone. With massive amounts of computational ability behind a single task or multiple specific tasks, machines can be trained to identify patterns in and relationships between input data and automate routine processes.

Some of the machine learning techniques that are to be used in our project are described below:

3.3.2 K-NEAREST NEIGHBOR

K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. However, it is mainly used for classification predictive problems in industry. The following two properties would define KNN well:-

I. Lazy Learning Algorithm - KNN is a lazy learning algorithm because it does not have a specialized training phase and used all the data for training while classification.

II. Non-parametric learning algorithm - KNN is also a non-parametric learning algorithm because it doesn't assume anything about the underlying data.

K-nearest neighbors (KNN) algorithm uses 'feature similarity' to predict the values of new data points which further means that the new data point will be assigned a value based on how closely it matches the points in the training set.

3.3.3 DECISION TREE

Decision tree is a type of supervised learning algorithm that is mostly used in classification problems. It works for both categorical and continuous input and output variables. In this technique, we split sample into two or more homogeneous sets (or sub-populations) based on most significant splitter / differentiator in input variables. In decision tree internal node represents a test on the attribute, branch depicts the outcome and leaf represents decision made after computing attribute.

The general motive of using Decision Tree is to create a training model which can be used to predict class or a value of target variables by learning decision rules inferred from prior data (training data).

The understanding level of Decision Tree algorithm is so easy compared with other classification algorithms. The Decision Tree algorithm tries to solve the problem, by using tree representation. Each internal node of the tree corresponds to an attribute, and each leaf node corresponds to a class label.

3.3.4 RANDOM FOREST

A random forest is a machine learning technique that's used to solve regression and classification problems. It utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems. A random forest algorithm consists of many decision trees. The 'forest' generated by the random forest algorithm is trained through bagging or bootstrap aggregating. Bagging is an ensemble meta-algorithm that improves the accuracy of machine learning algorithms. It establishes the outcome based on the predictions of the decision trees. It predicts by taking the average or mean of the output from various trees. Increasing the number of trees increases the precision of the outcome.

3.3.5 SUPPORT VECTOR MACHINE

Support Vector Machine (SVM) is a type of machine learning algorithm used for classification and regression analysis. It was introduced by Vapnik and Cortes in 1995 and has since become a popular algorithm due to its ability to handle high-dimensional data, perform well in both linear and nonlinear classification, and work with both labeled and unlabeled data.

The basic idea behind SVM is to find the hyperplane that best separates the data points into different classes. A hyperplane is a line, plane or hyperplane in high-dimensional space that divides the data into two classes. SVM seeks to find the hyperplane that maximizes the margin between the classes, which is the distance between the hyperplane and the closest data points of each class. This allows for a greater degree of generalization to new, unseen data.

To perform classification using SVM, the algorithm first maps the data points into a high-dimensional space where a linear hyperplane can be used to separate the data. This is done using a kernel function, which maps the data into a higher dimensional space without actually computing the coordinates of the data in that space. The kernel function allows for a more efficient computation and a more flexible classification boundary.

SVM has been used in a variety of applications, including text classification, image recognition, and bioinformatics. One notable application of SVM is in detecting fraudulent transactions, where it has been shown to perform well compared to other machine learning algorithms[14].

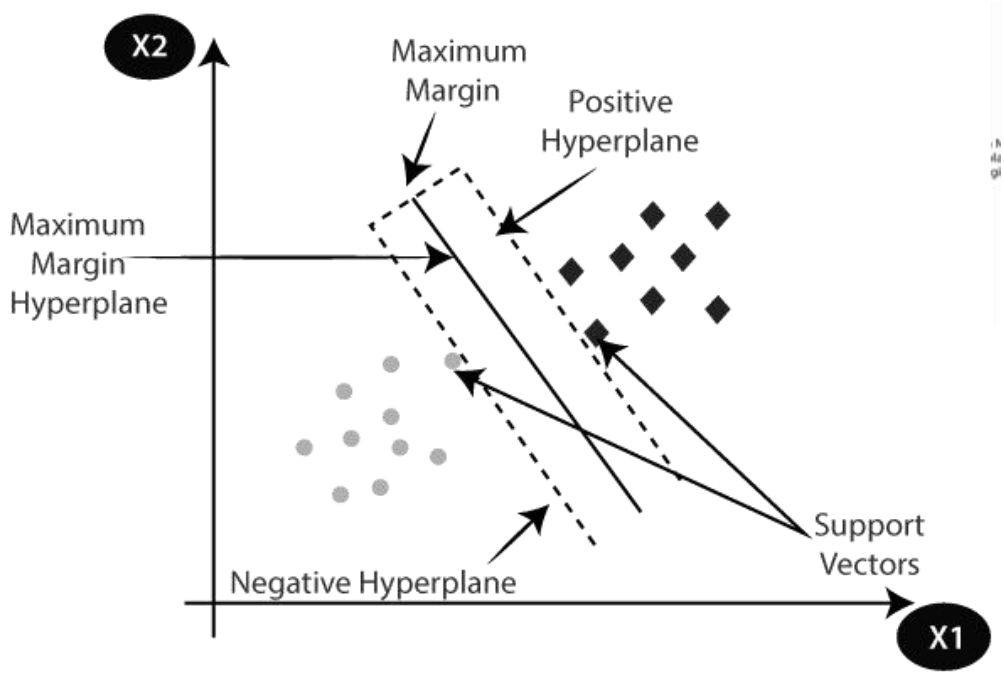


Figure 3.3: Support Vector Machine Algorithm

3.3.6 LOGISTIC REGRESSION

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables. It predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems. In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).

Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets. It can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification [15].

3.3.7 BOOSTING

Boosting is an ensemble modeling technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using weak models in series. Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models are added.It combines several weak learners into strong learners [16]. There are two most popular boosting algorithms:

3.3.7.1 GRADIENT BOOSTING

Gradient Boosting is a powerful boosting algorithm that combines several weak learners into strong learners, in which each new model is trained to minimize the loss function such as mean squared error or cross-entropy of the previous model using gradient descent. In each iteration, the algorithm computes the gradient of the loss function with respect to the predictions of the current ensemble and then trains a new weak model to minimize this gradient. The predictions of the new model are then added to the ensemble, and the process is repeated until a stopping criterion is met. I In contrast to AdaBoost, the weights of the training instances are not tweaked, instead, each predictor is trained using the residual errors of the predecessor as labels. There is a technique called the Gradient Boosted Trees whose base learner is CART (Classification and Regression Trees). [16].

3.3.7.2 ADABOOST

AdaBoost was the first really successful boosting algorithm developed for the purpose of binary classification. AdaBoost is short for Adaptive Boosting and is a very popular boosting technique that combines multiple “weak classifiers” into a single “strong classifier” [17].

Algorithm:

1. Initialize the dataset and assign equal weight to each of data point.
2. Provide this as input to the model and identify the wrongly classified data points.
3. Increase the weight of the wrongly classified data points.

4. If (got required result)

    a) Go to step 5

    else

    b) Goto step 2

5. End

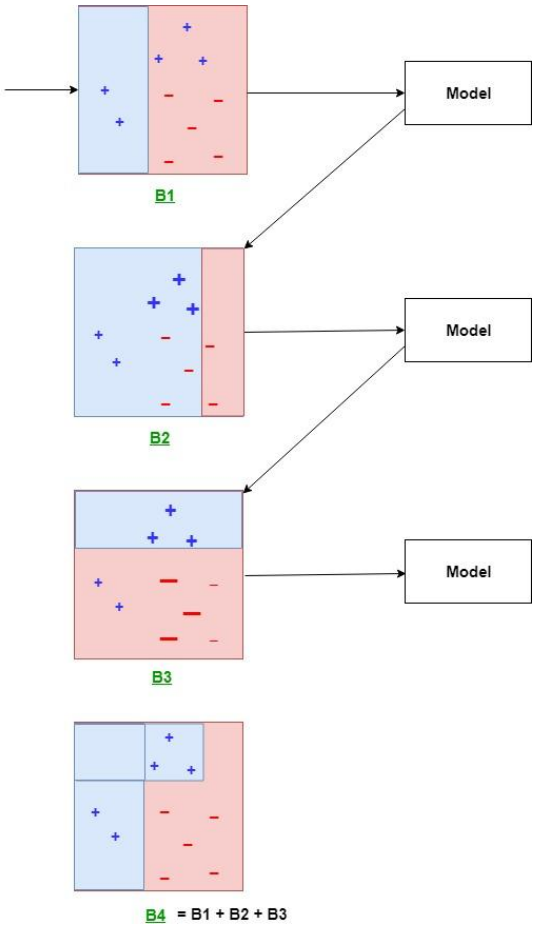


Figure 3.4: Training model using AdaBoost

3.4 FLOWCHART

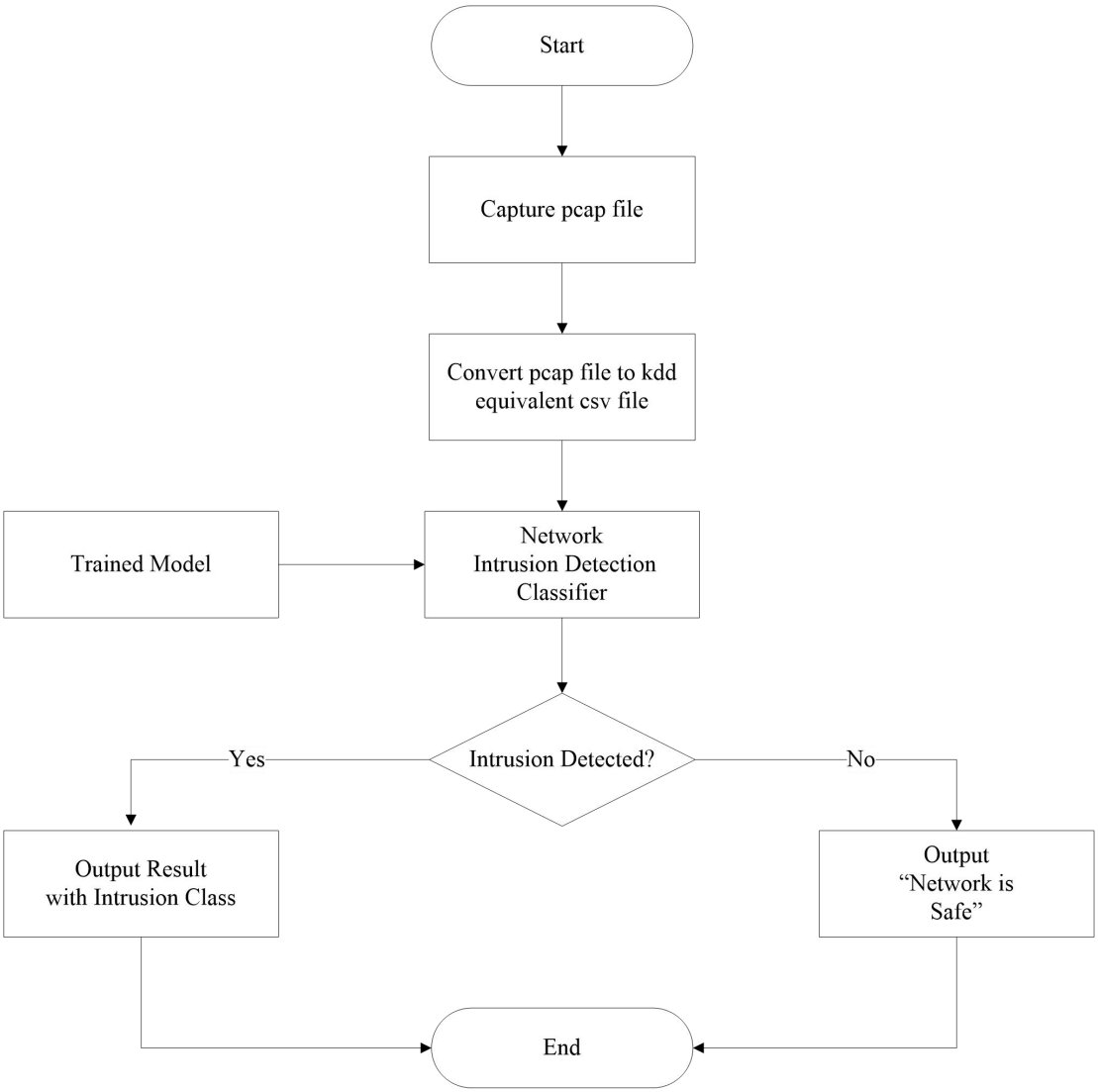


Figure 3.5: System Flow Chart

3.5 NECESSARY UML DIAGRAMS

3.5.1 UML DATA FLOW DIAGRAM

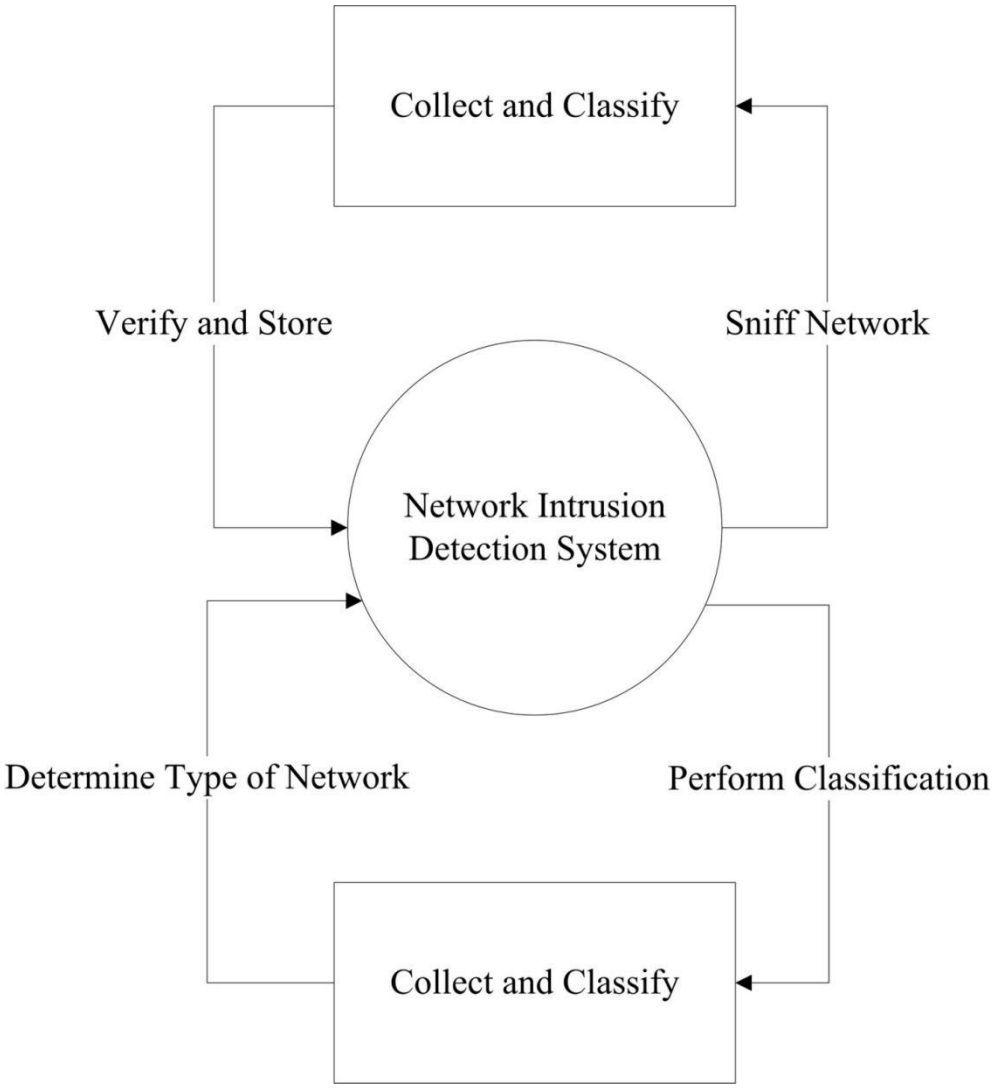


Figure 3.6: UML Data Flow Diagram Level-0

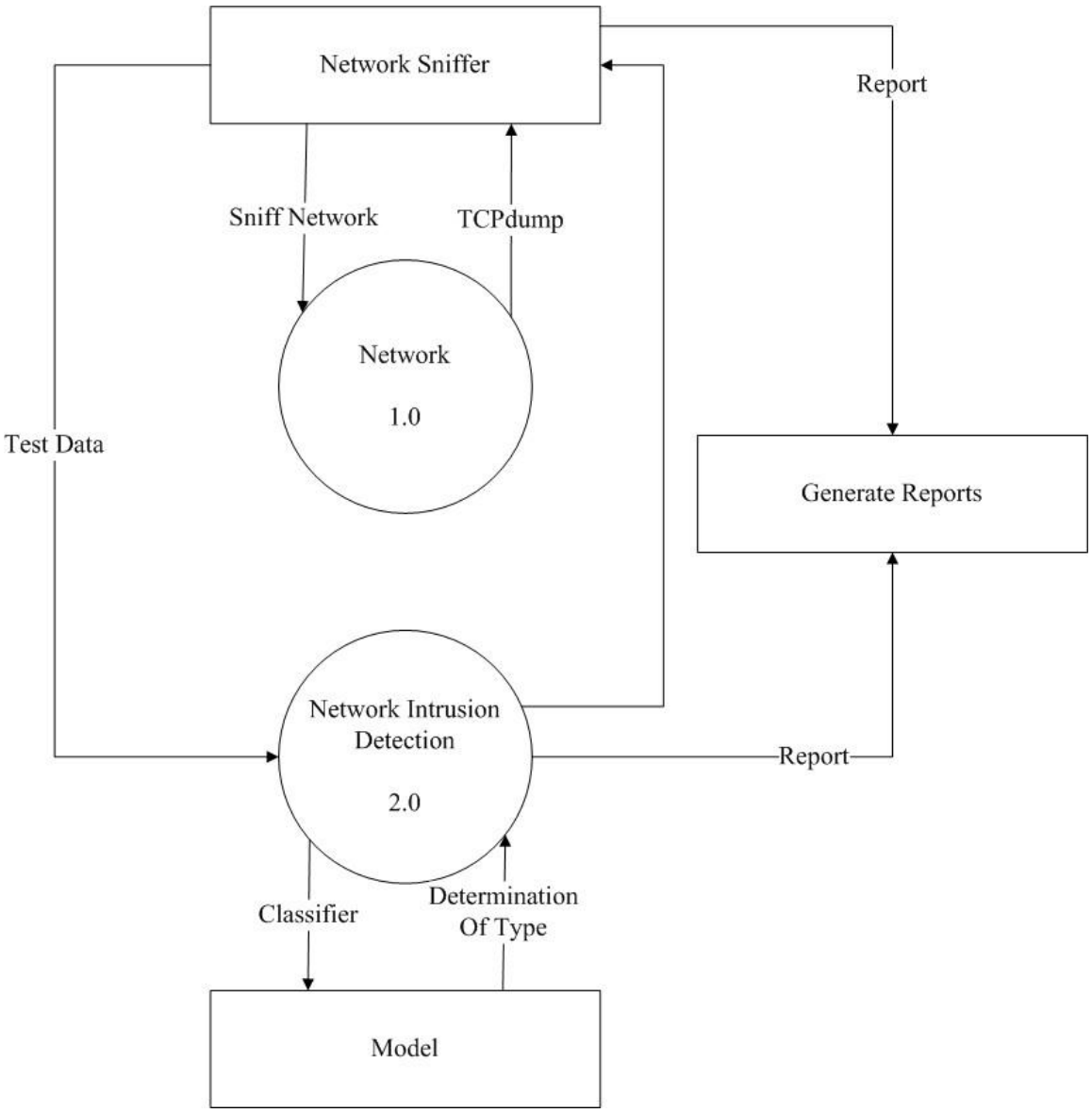


Figure 3.7: UML Data Flow Diagram Level-1

3.5.2 UML USE CASE DIAGRAM

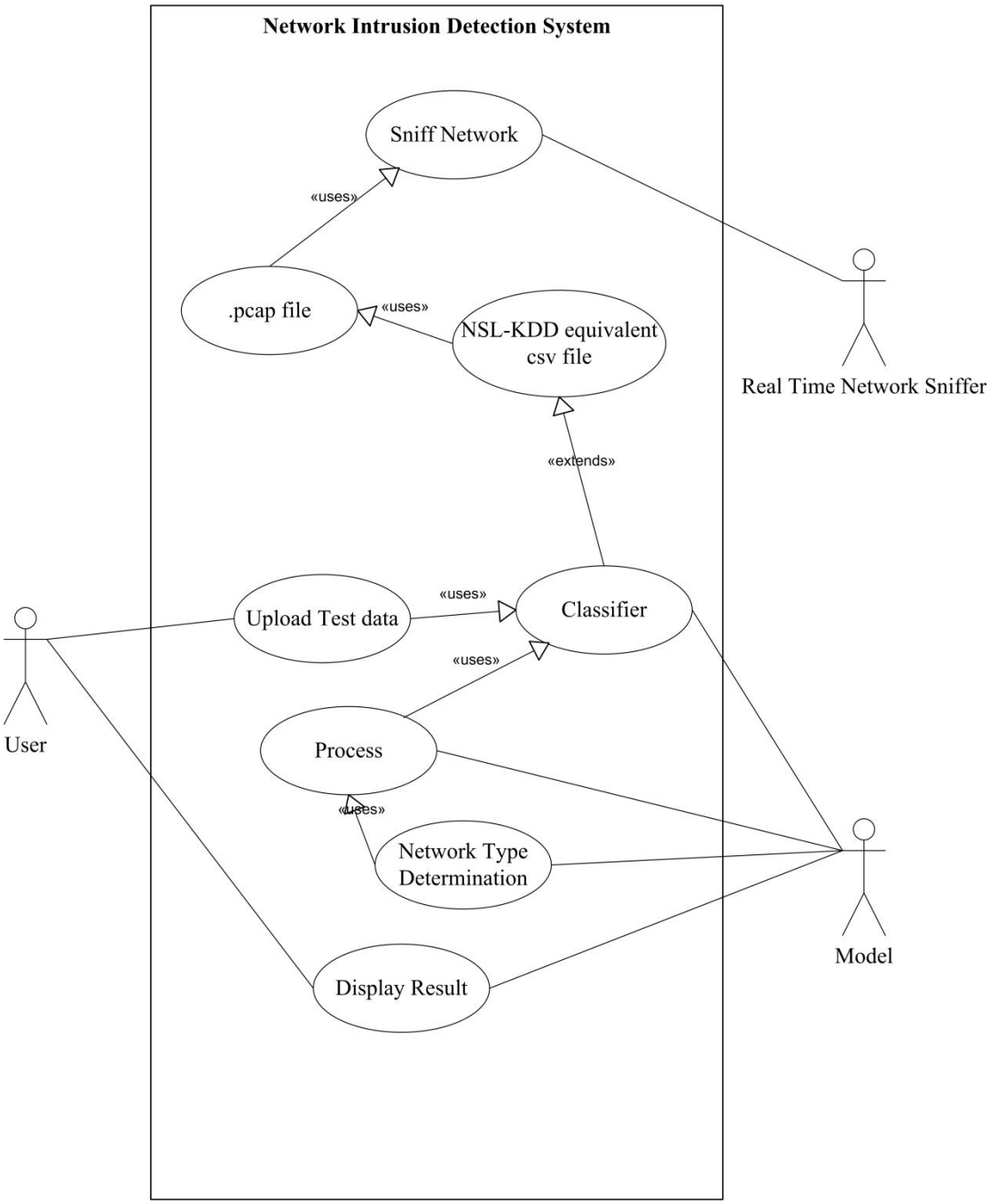


Figure 3.8: UML Use Case Diagram

Use Case Scenario Name: Network Intrusion Detection System
Actors: User, Real Time Network Sniffer, Model
Purpose: To capture network packets and detect intrusion based on network packets.
Overview: The real time network sniffer sniffs network packet and save in .pcap file. The .pcap file is then converted in NSL-KDD equivalent csv file. User can input the test data to the classifier, or classifier can take input as NSL-KDD equivalent csv file, then classifier performs certain processes and model determines the network types. The model then displays the output result and uses sees the result via webpage's user interface.

3.5.4 UML ACTIVITY DIAGRAM

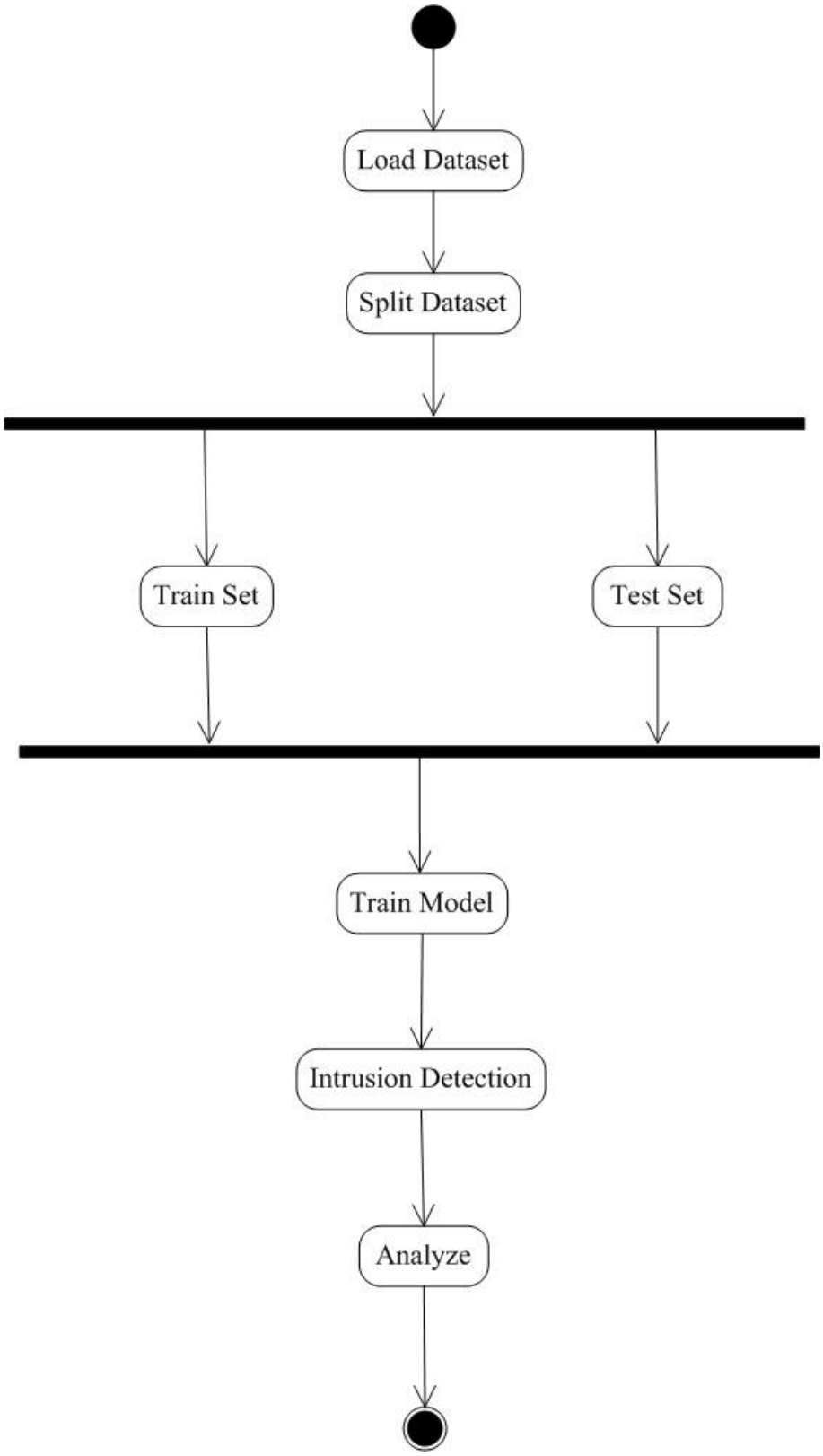


Figure 3.9: UML Activity Diagram

3.5.5 UML SEQUENCE DIAGRAM

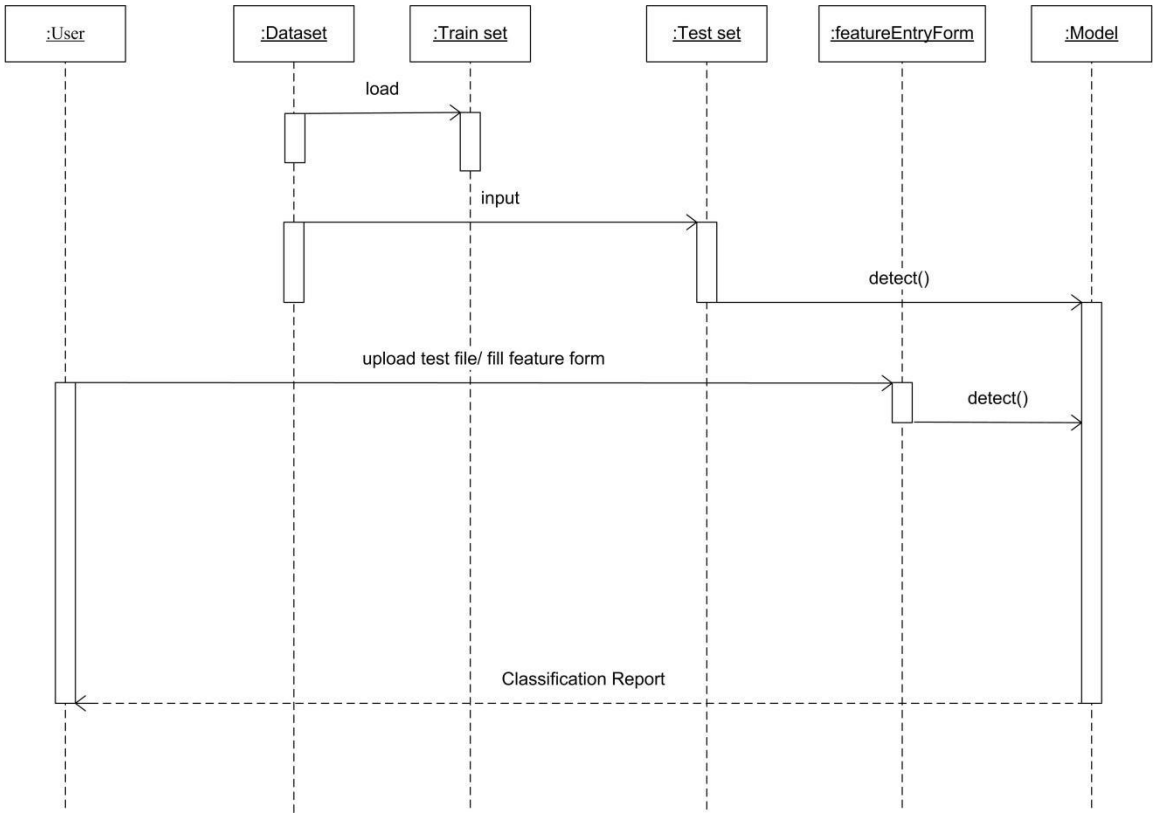


Figure 3.10: UML Sequence Diagram

3.6 TOOLS USED

3.6.1 Python

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages. With availability of various features like: easy to learn, read and maintain, interactive, portable and availability of various libraries, python is widely used.

Here it is used as the main development language. Various Data processing and Data analysis is done using it. It is also used to make machine learning models.

3.6.2 Matplotlib

Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open source alternative to MATLAB. Developers can also use matplotlib's APIs (Application Programming Interfaces) to embed plots in GUI applications. A Python matplotlib script is structured so that a few lines of code are all that is required in most instances to generate a visual data plot.

In our project it is used for the visualization, reporting and presenting results of data. It was also used to plot bar-graph about training and testing accuracy as shown in *Figure 3.9.*

3.6.3 Flask

Flask is a micro web framework for Python used to develop web applications. It is lightweight and easy to use. Flask is based on Werkzeug WSGI toolkit and Jinja2 template engine. It allows for easy routing and handling of HTTP requests, making it an excellent choice for small to medium-sized web applications. Additionally, Flask has a built-in development server and supports extensions to add additional functionality.

As our project is web-app based, flask is used to design web-based user interfaces that allows users to monitor and manage network activities. It is used to implement network intrusion detection system.

3.6.4 Pickle

The pickle module implements binary protocols for serializing and de-serializing a Python object structure. Pickling is the process whereby a Python object hierarchy is converted into a byte stream, and unpickling is the inverse operation, whereby a byte stream is converted back into an object hierarchy

After a model has been trained, it is saved as a ".sav" file so that it may be used at any time and avoid having to train and test the model repeatedly. It is used to save and load that trained file.

3.6.5 pyshark

Python wrapper for tshark, allowing python packet parsing using wireshark dissectors. There are quite a few python packet parsing modules, this one is different because it doesn't actually parse any packets, it simply uses tshark's (wireshark command-line utility) ability to export XML's to use its parsing. This package allows parsing from a capture file or a live capture, using all wireshark dissectors that have been installed.

In our project, it is used for capturing and filtering network packets. It is also used for analyzing network traffic and extracting network metadata.

3.6.7 VsCode

Visual Studio Code (VS Code) is a free and open-source code editor developed by Microsoft that's widely used by programmers for writing and debugging code in different programming languages. It has a rich set of features including syntax highlighting, code folding, auto-completion, debugging, and version control integration. VS Code also has a built-in terminal for executing commands and running scripts, making it more convenient for developers. Additionally, it supports a vast collection of extensions and themes that allow for customization to suit individual preferences. VS Code runs on multiple operating systems and is easy to use, making it a top choice for both beginner and experienced programmers.

It is used as the primary IDE for our project. Various python programs and Flask web framework are coded using this IDE.

3.6.8 Jupyter Notebook

Jupyter Notebook is a web-based interactive computational environment for creating and sharing documents containing live code, equations, visualizations, and explanatory text. It supports various programming languages, including Python, R, and Julia, and allows users to create and run code in a document-style interface. This makes it a popular tool among data scientists and researchers who can use it for data exploration, experimentation, and sharing. Jupyter Notebook provides a flexible environment for documenting code and results, as well as generating reports and presentations. It also supports the integration of external libraries and APIs, making it a powerful tool for data analysis and machine learning.

Jupyter Notebook was primarily used for analysis and processing of dataset and to train various machine learning algorithms. It was used to create a machine learning model in order to use it in near future without having to code and train model again and again.

3.7 VERIFICATION AND VALIDATION

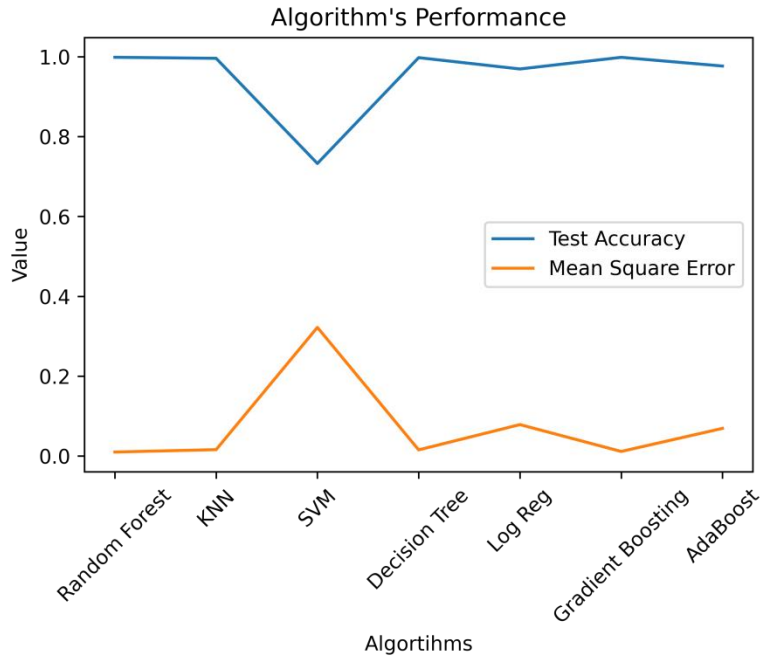


Figure 3.11: Testing Accuracy and Mean Squared Loss

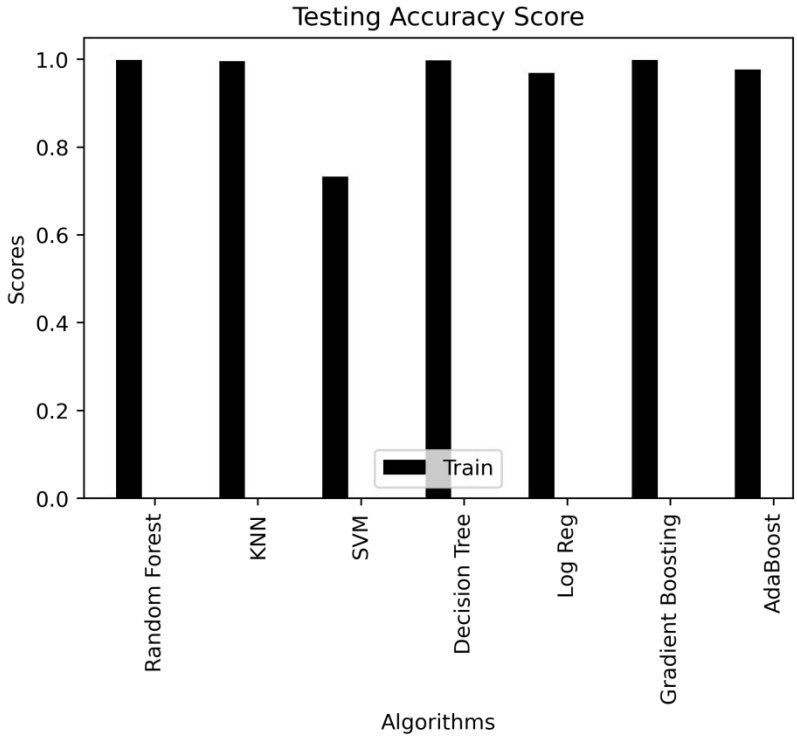


Figure 3.11: Bar Graph of Testing Accuracy Score

Table 3.1: Showing Scores of different Algorithms

	Accuracy (in %)	Precision (in %)	F1-score (in %)	Recall(in %)
Random Forest	**99.87**	**99.86**	**99.86**	**99.87**
K-Nearest Neighbor	99.61	99.58	99.58	99.61
Support Vector Machine	73.27	54.02	62.02	73.27
Decision Tree	99.78	99.78	99.76	99.78
Logistic Regression	94.94	96.16	96.45	96.94
Gradient Boosting	99.86	99.85	99.86	99.86
AdaBoost Classifier	97.68	96.59	97.13	97.68

With analysis of above line graph and bar chart , we get to know that **Random Forest Algorithm** has highest accuracy score, 0.9987, among all of the algorithms along

with least mean squared error, 0.0099, whereas the least performing algorithm on the provided dataset was **Support Vector Machine** with accuracy score 0.7327 and mean squares loss 0.3221.

CHAPTER 4: EPILOGUE

4.1 Result

By training various machine learning models, we have got a best test accuracy of 0.9987 via **Random Forest Algorithm** with lesser mean square error, 0.0099, among all. However, the least performing algorithm in our system was **Support Vector Machine** with test accuracy of 0.7327 and maximum mean square error, 0.3221.

4.2 Conclusion

Hence, this project can allow the following:

- To detect network intrusions at a relatively high accuracy rate
- Allow network engineers support during preliminary investigation
- Allow the general public the ability to analyze their network traffic

4.3 Future Enhancement

- To create a model that detects network intrusion based on real time traffic I.e. sniffing real time network data and detecting the intrusion on it automatically without creating intermediate csv files.

REFERENCES

[1] Sommer, R., & Paxson, V. (2010). Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. *IEEE Symposium on Security and Privacy*.2010

[2] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set," *Submitted to Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, 2009.

[3] Moustafa, N.; Creech, G.; Slay, J. Anomaly Detection System using Beta Mixture Models and Outlier Detection. In Progress in Computing, *Analytics and Networking; Springer*: Singapore, 2018; pp. 125-135.

[4] Zhang, J.; Zulkernine, M. Anomaly based network intrusion detection with unsupervised outlier detection. In Proceedings of the 2006 *IEEE International Conference on Communications*, Istanbul, Turkey, 11–15 June 2006; Volume 5, pp. 2388–2393.

[5] Moustafa, N.; Turnball, B.; Kwang, K. An Ensemble Intrusion Detection Technique Based on Proposed Statistical Flow Features for Protecting Network Traffic of Internet of Things. *IEEE Internet Things J*. 2018, 6, 4815–4830.

[6] W. Li and Q. Li, "Using Naive Bayes with AdaBoost to Enhance Network Anomaly Intrusion Detection," *Third International Conference on Intelligent Networks and Intelligent Systems*, 2010, pp. 486-489.

[7] Yuan, Y.; Huo, L.; Yuan, Y.; Wang, Z. Semi-supervised tri-Adaboost algorithm for network intrusion detection. *Int. J. Distrib. Sens. Netw*. 2019, 15, 1550147719846052.

[8] Giacinto, G.; Roli, F. An approach to the automatic design of multiple classififier systems. *Pattern Recognit. Lett*. 2001, 22, 25–33.

[9] Chebrolu, S.; Abraham, A.; Thomas, J.P. Feature deduction and ensemble design of intrusion detection systems. *Computer Security.* 2005, 24, 295–307.

[10] Xin, Y., Kong, L., Liu, Z., Chen, Y., Li, Y., Zhu, H., Gao, M., Hou, H., Wang, C. Machine Learning and Deep Learning Methods for Cybersecurity. IEEE Access. 2018; 6: 35365–3538

[11] Ahmed, M., Naser Mahmood, A., Hu, J. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*. 2016; 60: 19–31.

[12] Boutaba, R. Salahuddin, M.A., Limam, N., Ayoubi, S., Shahriar, N., Estrada-Solano, F., Caicedo, O.M. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. *Journal of Internet Services and Applications*. 2018; 9(1).

[13] Buczak, A.L., Guven, E.A. Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Communications Surveys & Tutorials*. 2016; 18(2): 1153–1176.

[14] Vapnik, V., & Cortes, C. (1995). Support-vector networks. *Machine learning*, 20(3), 273-297.

[15] Logistic Regression in Machine Learning - Javatpoint [online]

[16] Gradient Boosting in ML - GeeksforGeeks [online]

[17] Boosting in Machine Learning | Boosting and AdaBoost - GeeksforGeeks [online]

[18] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. A Detailed Analysis of the KDD CUP 99 Data Set. *IEEE Symposium on Computational Intelligence in Security and Defense Applications* (CISDA 2009)

SCREENSHOTS

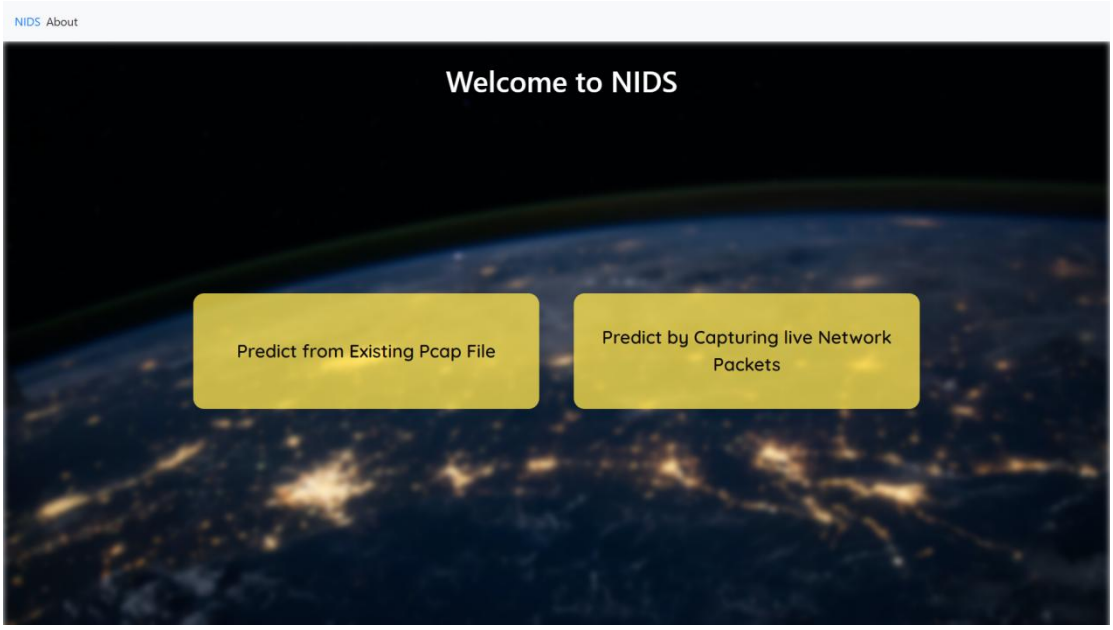


Figure 5.1: Frontend:Landing page with two different interface

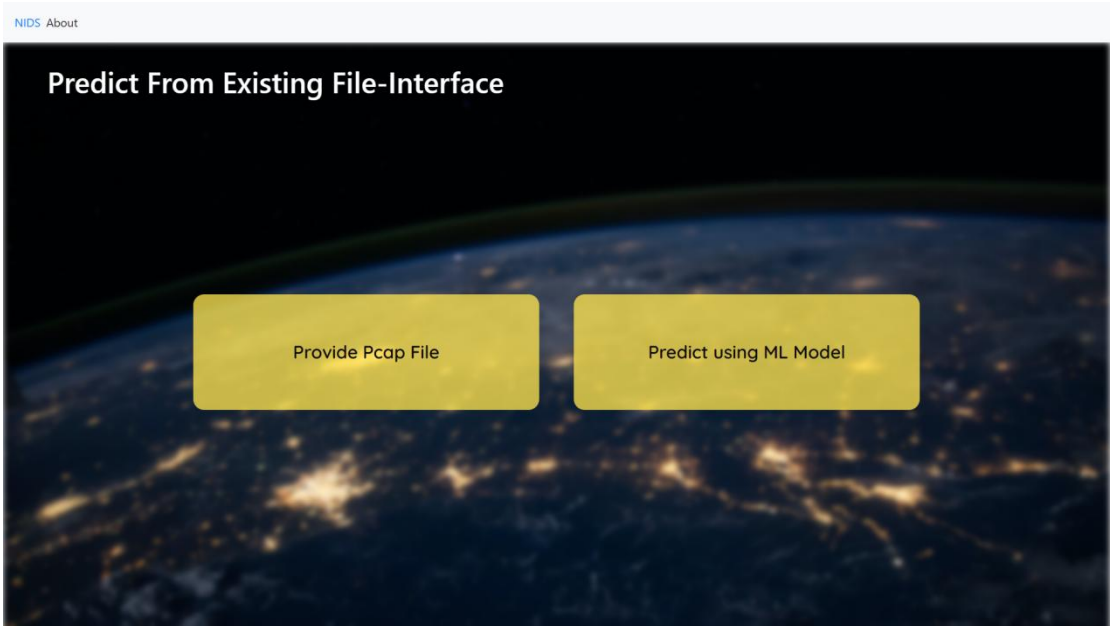


Figure 5.2: Frontend:Interface to predict using pcap file

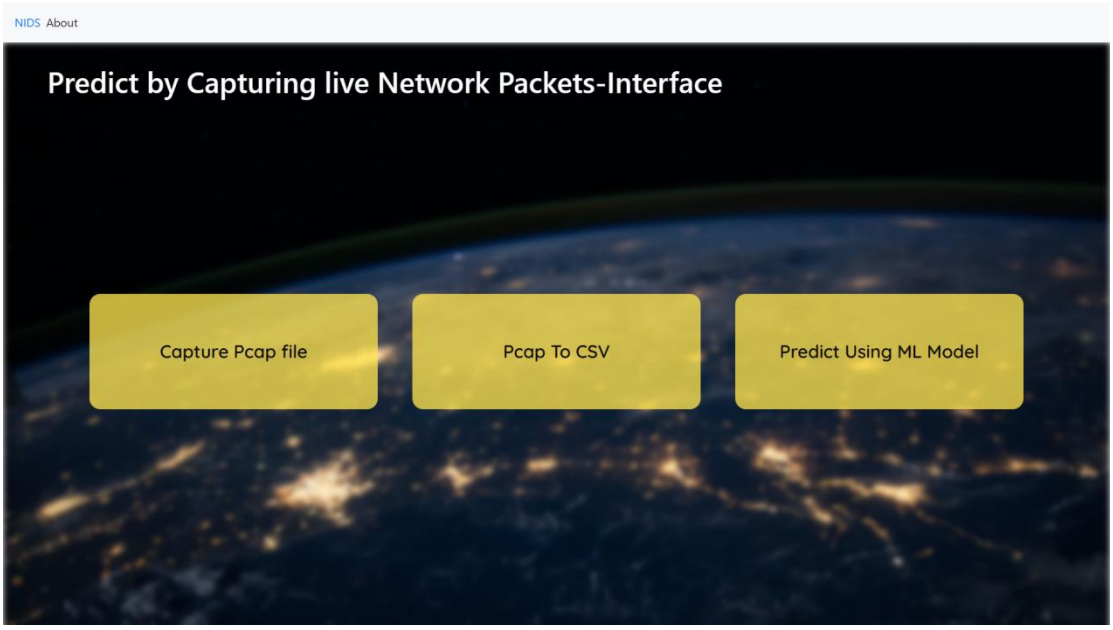


Figure 5.3: Frontend:Interface to predict using live network packets

Figure 5.4: Frontend:Page Showing the live network packets being captured

Figure 5.5:Frontend:Page showing pcap file being converted to KDD

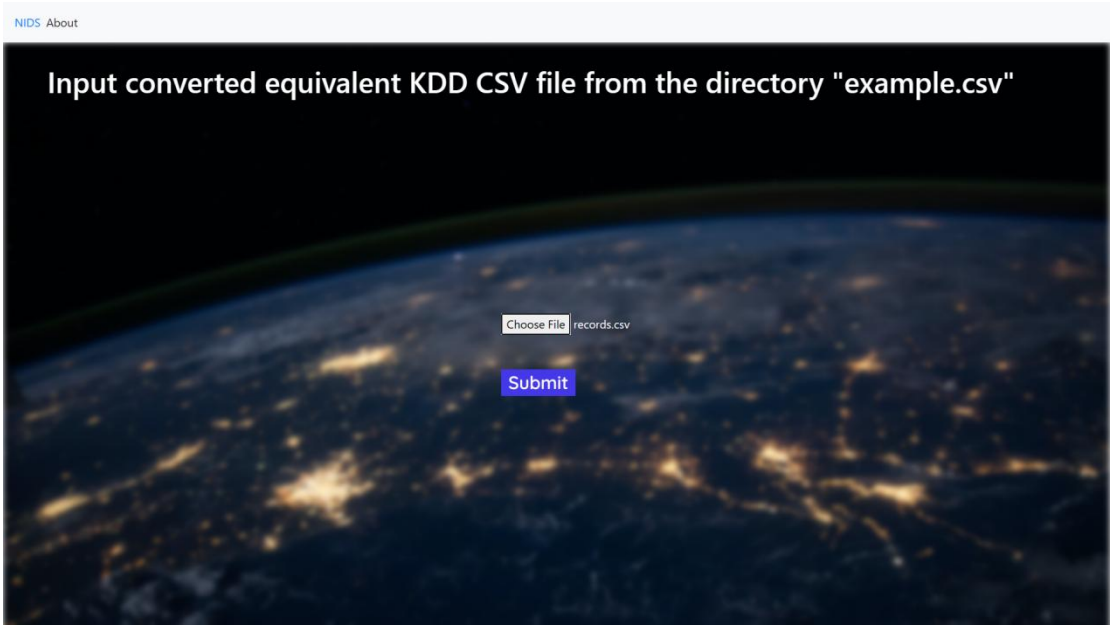


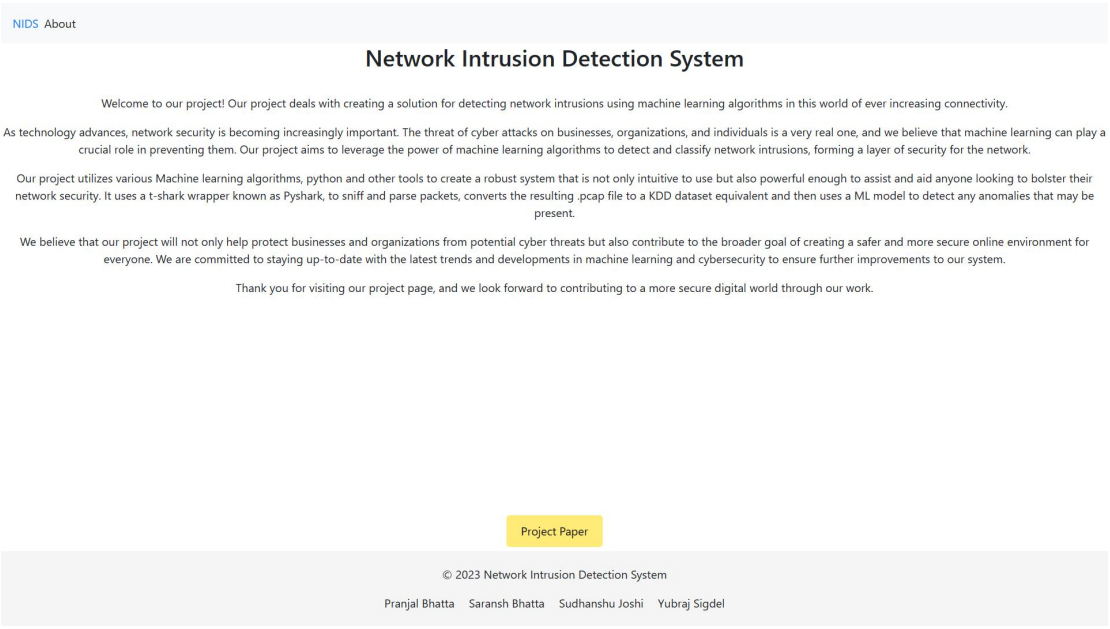
Figure 5.6: Frontend:Form to get csv file to do predictions

Predictions

Row	Prediction
1	Normal
2	Normal
3	Normal
4	Normal
5	Normal
6	Normal
7	Normal
8	U2R
9	Normal
10	Normal
11	Normal
12	Normal
13	Normal
14	Normal
15	Normal
16	Normal
17	Normal
18	Normal
19	Normal
20	U2R

Go back to home page

Figure 5.7: Frontend:Showing Predictions



NIDS About

Network Intrusion Detection System

Welcome to our project! Our project deals with creating a solution for detecting network intrusions using machine learning algorithms in this world of ever increasing connectivity.

As technology advances, network security is becoming increasingly important. The threat of cyber attacks on businesses, organizations, and individuals is a very real one, and we believe that machine learning can play a crucial role in preventing them. Our project aims to leverage the power of machine learning algorithms to detect and classify network intrusions, forming a layer of security for the network.

Our project utilizes various Machine learning algorithms, python and other tools to create a robust system that is not only intuitive to use but also powerful enough to assist and aid anyone looking to bolster their network security. It uses a t-shark wrapper known as Pyshark, to sniff and parse packets, converts the resulting .pcap file to a KDD dataset equivalent and then uses a ML model to detect any anomalies that may be present.

We believe that our project will not only help protect businesses and organizations from potential cyber threats but also contribute to the broader goal of creating a safer and more secure online environment for everyone. We are committed to staying up-to-date with the latest trends and developments in machine learning and cybersecurity to ensure further improvements to our system.

Thank you for visiting our project page, and we look forward to contributing to a more secure digital world through our work.

Project Paper

© 2023 Network Intrusion Detection System

Pranjal Bhatta Saransh Bhatta Sudhanshu Joshi Yubraj Sigdel

Figure 5.8: Frontend:About Page