**The Project Report on**

# Nids-Cnnlstm Network Intrusion Detection Classification Model Based On Deep Learning

# Submitted To

**Acharya Nagarjuna University**

**A project report submitted in the partial fulfillment of the requirements for the**

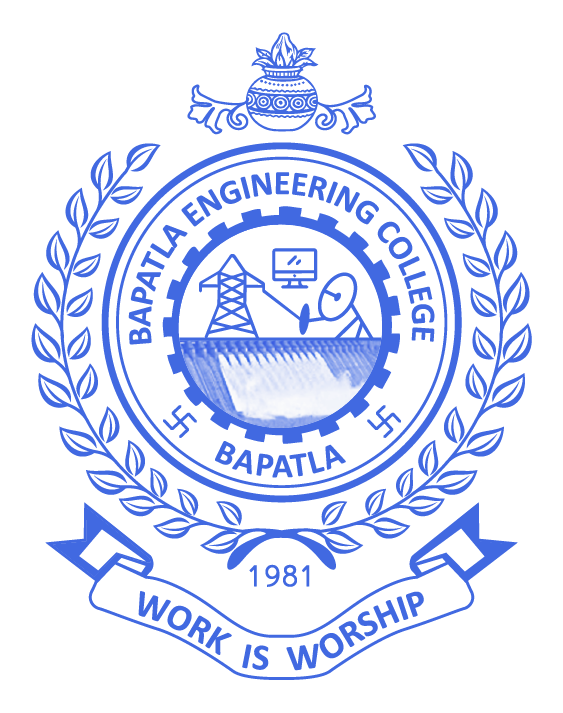
**Award of the Degree of**

**MASTER OF COMPUTER APPLICATIONS**

**Submitted By**

**ARUNURU NAVEEN**

**Reg.No: L20MC23014**

****

**Under The Esteemed Guidance Of**

**Sri N. KIRAN KUMAR M. Tech**

**Assistant Professor**

**DEPARTMENT OF MCA**

**BAPATLA ENGINEERING COLLEGE**

**(Approved by A.I.C.T.E)**

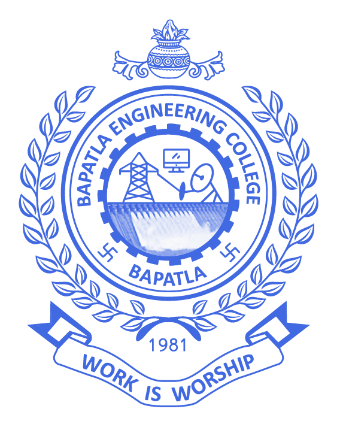
**(Affiliated to ACHARYA NAGARJUNA UNIVERSITY)**

**2019-2021**

**DEPARTMENT OF MCA**

**BAPATLA ENGINEERING COLLEGE**

**BAPATLA-522101**

****

**CERTIFICATE**

This is to certify that this project work entitled “**Secure Cloud Storage based on RLWE Problem”** is the bonafide work carried out by **ARUNURU NAVEEN**, **Reg.No: L20MC23014** submitted in Partial fulfillment of the requirement for the Award of Degree of “**Master of** **Computer Applications**”, during the academic year 2019-2021.

The results submitted in this project have been verified and are found to be satisfactory. The results embodied in this thesis have not been submitted to any other university for the award of the any other degree/diploma.

**Project Guide: Head of the Department:**

**Sri N. Kiran Kumar M. Tech Sri K.N. Prasad M. Tech**

**Assistant Professor Associate Professor & Head**

**Dept. of MCA Dept. of MCA**

**Bapatla Engineering College Bapatla Engineering College**

**Bapatla. Bapatla.**

**Signature of External Examiner**

# ACKNOWLEDGEMENT

I sincerely thank the following distinguished personalities who have given their advice and support for successful completion of this work.

I would like to thank **Dr. V. Damodara Naidu**, Principal, Bapatla Engineering College, Bapatla for providing the online resources and allowing me to carry out this work.

I extended my sincere thanks to **Sri K.N. Prasad**, Head of the Department for extending their cooperation and providing the required resources.

I am deeply indebted to most respect and my project guide **Sri N. Kiran Kumar**, Assistant Professor Dept. of MCA for his valuable and inspiring guidance, comments, suggestions and encouragements.

I extended my sincere thanks to all other teaching and non-teaching staff of the department who helped directly or indirectly for their cooperation and encouragement.

**ARUNURU NAVEEN**

**(L20MC23014)**

**DECLARATION**

This is to declare that the project **“Secure Cloud Storage based on RLWE Problem”** at Bapatla Engineering College has been presented by me during the academic year **2019-2021** in partial fulfillment of the requirements for the **“Master of Computer Application”**.

I also declare that this project is the result of my own efforts and that it has not been submitted to any other universities for the award of degree or diploma.

**ARUNURU NAVEEN**

**(L20MC23014)**

Table of Contents

[ACKNOWLEDGEMENT iii](#_Toc80180563)

[ABSTRACT viii](#_Toc80180564)

[CHAPTER-1 1](#_Toc80180565)

[INTRODUCTION 1](#_Toc80180566)

[1.1 INTRODUCTION 1](#_Toc80180567)

[CHAPTER- 2 5](#_Toc80180568)

[LITERATURE SURVEY 5](#_Toc80180569)

[2.1 LITERATURE REVIEW 5](#_Toc80180570)

[CHAPTER-3 7](#_Toc80180571)

[THEORETICAL BACKGROUND 7](#_Toc80180572)

[3.1 INTRODUCTION: 7](#_Toc80180573)

[3.2 INTRODUCTION TO PYTHON 12](#_Toc80180574)

[3.3 BENFITS OF PYTHON 21](#_Toc80180575)

[CHAPTER-4 30](#_Toc80180576)

[SYSTEM ANALYSIS 30](#_Toc80180577)

[4.1 EXISTING SYSTEM: 30](#_Toc80180578)

[4.1.1 DISADVANTAGES OF EXISTING SYSTEM: 30](#_Toc80180579)

[4.2 PROPOSED SYSTEM: 30](#_Toc80180580)

[4.2.1 ADVANTAGES OF PROPOSED SYSTEM: 31](#_Toc80180581)

[CHAPTER- 5 32](#_Toc80180582)

[SYSTEM DESIGN 32](#_Toc80180583)

[5.1 INTRODUCTION 32](#_Toc80180584)

[5.2 MODULES 32](#_Toc80180585)

[5.2.1 DATASET: 32](#_Toc80180586)

[5.2.2 PREPROCESSING: 32](#_Toc80180587)

[5.2.3 GRAPHS: 32](#_Toc80180588)

[5.2.4 PREDICTION: 32](#_Toc80180589)

[5.3 SYSTEM ARCHITECTURE 33](#_Toc80180590)

[5.4 UML DAIGRAMS 34](#_Toc80180591)

[5.4.1 CONSTRUCTION OF USE CASE DIAGRAMS: 37](#_Toc80180592)

[5.4.2 SEQUENCE DIAGRAMS: 40](#_Toc80180593)

[5.4.3 CLASS DIAGRAM: 42](#_Toc80180594)

[5.4.4 ACTIVITY DIAGRAM: 42](#_Toc80180595)

[CHAPTER-6 44](#_Toc80180596)

[SYSTEM REQUIREMENTS 44](#_Toc80180597)

[6.1 SYSTEM REQUIREMENTS 44](#_Toc80180598)

[6.1.1 HARDWARE REQUIREMENTS: 44](#_Toc80180599)

[6.1.2 SOFTWARE REQUIREMENTS: 44](#_Toc80180600)

[CHAPTER-7 45](#_Toc80180601)

[SYSTEM IMPLEMENTATION 45](#_Toc80180602)

[7.1 INPUT AND OUTPUT DESIGNS 45](#_Toc80180603)

[7.1.1 LOGICAL DESIGN 45](#_Toc80180604)

[7.1.2 PHYSICAL DESIGN 45](#_Toc80180605)

[7.2 INPUT & OUTPUT REPRESENTATION 46](#_Toc80180606)

[7.2.1 INPUT DESIGN 46](#_Toc80180607)

[7.2.2 OBJECTIVES 47](#_Toc80180608)

[7.2.3 OUTPUT DESIGN 47](#_Toc80180609)

[CHAPTER-8 69](#_Toc80180610)

[SYSTEM TESTING 69](#_Toc80180611)

[8.1 INTRODUCTION: 69](#_Toc80180612)

[8.2 LEVELS OF TESTING 69](#_Toc80180613)

[8.2.1 BLACK BOX TESTING 70](#_Toc80180614)

[8.2.2 WHITE BOX TESTING 72](#_Toc80180615)

[CHAPTER-9 74](#_Toc80180616)

[OUTPUT SCREENS 74](#_Toc80180617)

[CONCLUSION 75](#_Toc80180618)

[REFERENCES 76](#_Toc80180619)

**List of Figures**

**Name of the figure Pg.no**

[Figure 3. 1 Structure of cloud computing 7](#_Toc80179715)

[Figure 3. 2 Characteristics of cloud computing 9](#_Toc80179716)

[Figure 3. 3 Structure of service models 10](#_Toc80179717)

[Figure 5. 1 System Architecture 33](#_Toc80179735)

[Figure 5. 2 Use Case Diagram 39](#_Toc80179736)

[Figure 5. 3 Sequence diagram 41](#_Toc80179737)

[Figure 5. 4 Class Diagram 42](#_Toc80179738)

[Figure 5. 5 Activity Diagram 43](#_Toc80179739)

# ABSTRACT

Intrusion detection is the core topic of network security, and the intrusion detection algorithm based on deep learning has become a research hotspot in network security. In this paper, a network intrusion detection classification model (NIDS-CNNLSTM) based on deep learning is constructed for the wireless sensing scenario of the Industrial Internet of Things (IIoT) to effectively distinguish and identify network traffic data and ensure the security of the equipment and operation of the IIoT. NIDS-CNNLSTM combines the powerful learning ability of long short-term memory neural networks in time series data, learns and classifies the features selected by the convolutional neural network, and verifies the applicability based on binary classification and multi-classification scenarios. The model is trained using KDD CUP99, NSL\_KDD, and UNSW\_NB15 classic datasets. The verification accuracy and training loss on the three datasets all show good convergence and level, and the accuracy rate is high when classifying various types of traffic. The overall performance of NIDS-CNNLSTM has been significantly improved compared with the models proposed in previous studies. The effectiveness shows a high detection rate and classification accuracy and a low false alarm rate through experimental results. It is more suitable for large-scale and multi-scenario network data in the IIoT..

.

.

.

**PROBLEM STATEMENT:**

* With the rapid development and wide application of 5G, IoT, Cloud Computing, and other technologies, network scale, and real-time traffic become more complex and massive, cyber-attacks have also become complex and diverse, bringing significant challenges to cyberspace security. As the second line of defense behind the firewall, the Network Intrusion Detection System (NIDS) needs to accurately identify malicious network attacks, provide real-time monitoring and dynamic protection measures, and formulate strategies.
* .

**OBJECTIVE:**

* In real cyberspace, normal activities occupy the dominant position, so most traffic data are normal traffic; only a few are malicious cyber-attacks, resulting in a high imbalance of categories. In the highly imbalanced and redundant network In real cyberspace, normal activities occupy the dominant position, so most traffic data are normal traffic; only a few are malicious cyber-attacks, resulting in a high imbalance of categories. In the highly imbalanced and redundant network.

# CHAPTER-1

# INTRODUCTION

## Introduction

With the development and improvement of Internet technology, the Internet is providing various convenient services for people. However, we are also facing various security threats. Network viruses, eavesdropping and malicious attacks are on the rise, causing network security to become the focus of attention of the society and government departments. Fortunately, these problems can be well solved via intrusion detection. Intrusion detection plays an important part in ensuring network information security. However, with the explosive growth of Internet business, traffic types in the network are increasing day by day, and network behavior characteristics are becoming increasingly complex, which brings great challenges to intrusion detection [1], [2]. How to identify various malicious network traffics, especially unexpected malicious network traffics, is a key problem that cannot be avoided.

In fact, network traffic can be divided into two categories (normal traffics and malicious traffics). Furthermore, network traffic can also be divided into five categories: Normal, DoS (Denial of Service attacks), R2L (Root to Local attacks), U2R (User to Root attack) and Probe (Probing attacks). Hence, intrusion detection can be considered as a classification problem. By improving the performance of classifiers in effectively identifying malicious traffics, intrusion detection accuracy can be largely improved. Machine learning methods [3]–[8] have been widely used in intrusion detection to identify malicious traffic. However, these methods belong to shallow learning and often emphasize feature engineering and selection. They have difficulty in features selection and cannot effectively solve the massive intrusion data classification problem, which leads to low recognition accuracy and high false alarm rate. In recent years, intrusion detection methods based on deep learning have been proposed successively. In [9], the authors propose a mal-ware traffic classification method based on convolutional

neural network with traffic data as image. This method does not need manual design features, and directly takes the original traffic as the input data to the classifier. In [10], the authors provide an analysis of the viability of Recurrent Neural Networks (RNN) to detect the behavior of network traffic by modeling it as a sequence of states that change over time. In [11], the authors verify the performance of Long ShortTerm memory (LSTM) network in classifying intrusion traffics. Experimental results show that LSTM can learn all the attack classes hidden in the training data. All the above methods treat the entire network traffic as a whole consisting of a sequence of traffic bytes. They don’t make full use of domain knowledge of network traffics. For example, CNN converts continuous network traffic into images for processing, which is equivalent to treating traffics as independent and ignore the internal relations of network traffics. Firstly, network traffic is a hierarchical structure. Specifically, network traffic is a traffic unit composed of multiple data packets. Data packet is a traffic unit composed of multiple bytes. Secondly, traffic features in the same and different packets are significantly different. Sequential features between different packets need to be extracted independently. In other words, not all traffic features are equally important for traffic classification in the process of extracting features on a certain network traffic.

.

**Problem Identification & Objectives**

Aviation is one of the popular means of transportation used by huge number of people every year. The availability of aircraft in good condition and capable of flying is used to gauge the efficiency of this industry. This availability factor decreases due to unanticipated breakdowns that produce a difficulty with the aircraft's operation, resulting in several delays and financial losses to airlines. As a result, we are using predictive maintenance to avoid such scenarios by making forecasts about prospective problems that could disrupt routine aircraft operation.

1. To predict when an aircraft engine will fail in the future, so that maintenance can be planned in advance.

2. Based on the aircraft engine operation and failure events history, to predict when an in-service engine will fail.

3. To check the accuracy of results using Machine learning algorithms when compared to the deep learning methods

# CHAPTER- 2

# literature survey

## 2.1 literature review

In the research of network intrusion detection based on machine learning, scholars mainly distinguish normal network traffic from abnormal network traffic by dimensionality reduction, clustering, and classification, to realize the identity fiction of malicious attacks [10], [11]. Pervez proposed a new method for feature selection and classification merging of multi-class NSL-KDD Cup99 dataset using Support Vector Machine (SVM) and discussed the classification accuracy of classifiers under different dimension features [12]. Shiraz studied some new technologies to improve CANN intrusion detection methods’ classification performance and evaluated their performance on the NSL-KDD Cup99 dataset [13]. He used the K Farthest Neighbor (KFN) and the K Nearest Neighbor (KNN) to classify the data and used the Second Nearest Neighbor (SNN) of the data when the nearest and farthest neighbors have the same class label. The result shows the CANN detection rate and reduces the failure the alert rate is improved or provides the same performance. Bhattacharya proposed a machine learning model based on hybrid Principal Component Analysis (PCA)-Firefly [14]. The dataset used was the open dataset collected from Kaggle. Firstly, the model performs one key coding for transforming the IDS dataset, then uses the hybrid PCA-Firefly algorithm to reduce the dimension, and the XGBoost algorithm classifies the reduced dataset. In recent years, with the powerful ability of automatic feature extraction, deep learning has made remarkable achievements in the fields of Computer Vision (CV), Autonomous driving(AD), Natural Language Processing(NLP). Many scholars apply deep learning to intrusion detection for traffic classification, which has become a hot spot of current research. The method of deep learning is to mine the potential characteristics of high-dimensional data through a training model and transform network traffic anomaly detection into classification problem [15]. Through a large number of sample data training, adaptive learning between normal network traffic and abnormal network traffic effectively enhances real-time intrusion processing. Torres et al. [16] first converted network traffic characteristics into a series of characters and then used Recurrent Neural Network (RNN) to learn their temporal characteristics, which were further used to detect malicious network traffic. Wang et al. [17] proposed a malicious software traffic classification algorithm based on Convolutional Neural Network(CNN). By mapping the traffic characteristics to pixels, the network traffic image is generated, and the image is used as the input of the CNN to realize traffic classification. Staudemeyer and Shamsinejad [13] proposed an intrusion detection algorithm based on Long Short-Term Memory (LSTM), which detects DoS attacks and probe attacks with unique time series in the KDD Cup99 dataset. Kwon et al. [18] has carried out relevant research on the deep learning model, focusing on data simplification, dimension reduction, classification, and other technologies, and proposes a Fully Convolutional Network (FCN) model. By comparing with the traditional machine learning technology, it is proved that the FCN model is useful for network traffic analysis. Tama et al. [19] proposed an anomaly-based IDS based on a two-stage meta-classifier, which uses a hybrid feature selection method to obtain accurate feature representations.

# Chapter-3

# Theoretical background

## 3.1 Introduction:

## 3.2 Introduction to PYTHON

**Python Technology**

Python technology is both a programming language and a platform.

**The python Programming Language**

THE PYTHON PROGRAMMING LANGUAGE IS A HIGH-LEVEL LANGUAGE THAT CAN BE CHARACTERIZED BY ALL OF THE FOLLOWING BUZZWORDS:

* + - Simple
    - Architecture neutral
    - Object oriented
    - Portable
    - Distributed
    - High performance
    - Interpreted
    - Multithreaded
    - Robust
    - Dynamic
    - Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Python programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called Python byte codes —the platform-independent codes interpreted by the interpreter on the Python platform. The interpreter parses and runs each Python byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.

FEATURES OF MACHINE LEARNING

• It is nothing but automating the Automation.

• Getting computers to program themselves.

• Writing Software is bottleneck.

• Machine leaning models involves machines learning from data without the help of humans or any kind of human intervention.

• Machine Learning is the science of making of making the computers learn and act like humans by feeding data and information without being explicitly programmed.

• Machine Learning is totally different from traditionally programming, here data and output is given to the computer and in return it gives us the program which provides solution to the various problems. Below is the figure.

**Traditional Programming vs Machine Learning**

• Machine Learning is a combination of Algorithms, Datasets, and Programs.

• There are Many Algorithms in Machine Learning through which we will provide us the exact solution in predicting the disease of the patients.

• How Does Machine Learning Works?

• Solution to the above question is Machine learning works by taking in data, finding relationships within that data and then giving the output.

**Machine Learning Model**

• There are various applications in which machine learning is implemented such as Web search, computing biology, finance, e-commerce, space exploration, robotics, social networks, debugging and much more.

• There are 3 types of machine learning supervised, unsupervised, and reinforcement.

**BENEFITS OF PYTHON**

• Presence of Third-Party Modules

• Extensive Support Libraries

• Open Source and Community Development

• Learning Ease and Support Available

• User-friendly Data Structures

• Productivity and Speed

• Highly Extensible and Easily Readable Language.

**Python**

Python is high level language and it is also integrated version of the program. Python is an object-oriented approach and its main aim to help programmers to write the code clearly, logical code for small and large scale of project.

Pytrhon is dynamically typed and garbage collected it also support multiple programming and it is both procedure and object oriented and also functional programming. And structural programming also supported. It has many built in function it also supports filter, map and reduce function. All the machine learning algorithm and the libraries are being supported by the python programming language. Python also support list, dict, sets and other generators. Python code can be run in different platform such as anaconda, PyCharm etc.

The main goal of this programing language is as follows:

• Python is simple, object-oriented programming language.

• The language and implementation should provide support for software engineering principles such as strong type library preset for different machine learning algorithm, and all other algorithm in simple manner.

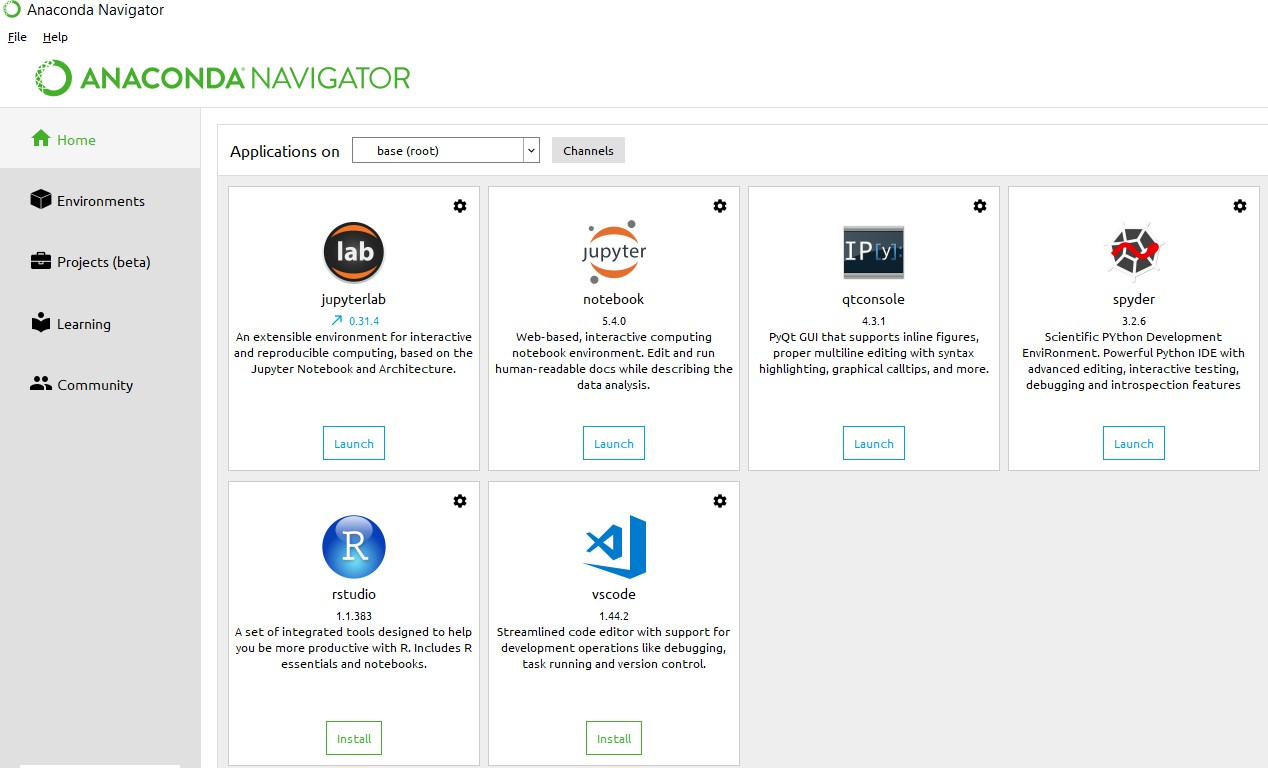
• Coding will be smooth in python and the data analysis can be easily done in python.

This is so much so to the point where we now have modules and APIs at our disposal, and you can engage in machine learning very easily without almost any knowledge at all of how it works. With the defaults from Scikit-learn, you can get 90-95% accuracy on many tasks right out of the gate. Machine learning is a lot like a car, you do not need to know much about how it works in order to get an incredible amount of utility from it.

Despite the apparent age and maturity of machine learning, I would say there's no better time than now to learn it, since you can actually use it. Machines are quite powerful, the one you are working on can probably do most of this series quickly. Data is also very plentiful lately.

**Anaconda**

Anaconda is free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine Learning applications, Large- scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. It is developed and maintained by Anaconda, Inc. The distribution incudes data-science packages suitable for Windows, Linux, and macOS. Packaged versions are required and are managed by the package management system anaconda. This package manager was spun out as a separate open-source package as it ended up being useful on its own and for other things than Python. There is also a small, bootstrap version of Anaconda called Miniconda, which includes only conda, Python, the packages they depends on, and a small number of other packages.



**Anaconda Console**

**Jupyter notebook**

Jupiter Notebook or so called IPython Notebook is an interactive web based computational mean for starting with Jupiter Notebook documents. The term notebook itself is a huge entity to represent the integration with different entity sets. JSON is the main document form from the same for the execution which follows the brief on the schema and the input and output means. It has high integration with several language set and has various flexibilities with the choices. The extension used for the same is “.ipynb” which runs in this platform. It’s an open-source software package with interactive communication means. It has it’s open standards for the same. It’s an open community best for budding programmers . The flexibility of the same is phenomenon and splendidly done the configuration and integration of the same is simplest and easy on hold so that no prior distortion is generated and the efficiency of the same is measured through out any system of choice.

It’s the best software sets that been used across cross for designing and developing of the products and support wide help support. Not only to that, it provides scalability in the code and the deployment of the same. Various Language can be changed and the project can be undertaken on the same. The created notebook files can be shared and stored in various means for further utilization. It supports cultivated and interactive output sets. Easily crossed over for graphing, plotting and visualizing of the elements. Data Integration of the same is to it’s best. The integration of big data and it can process chunks of values in an approx. time which gives a better performance and the higher computational means. Various works on data like cleaning, cleansing, transforming modeling and visualizing can be done by the same

Machine learning is the ability that gives the computer to learn without being explicitly programmed. There are two types of machine learning:

Supervised Learning: supervised learning is the learning of the labelled data. It is the types of machine learning that maps the input and output based on the examples input-output pairs. In supervised learning each training data having pairs of input and desired outputs values. Supervised learning algorithm analyzes the training data and produces a function which can be used for mapping of new data.

Fig 2.1 Supervised Learning The output to solve the supervised learning algorithm are as:

• Determine the types of data, before doing anything else the user should understand which types of data set is to be used for training the data.

• Gathered the training data sets either in form of human experts or from measurements.

• Determine the feature of inputs from the learned data and depends on the inputs it changed into feature vector; number of features should not be large but should contains enough information to accurately predict the outputs.

• Check the learned function and the learned algorithm for example we use support vector machines or decisions tree.

• Complete the design and run the trained data sets.

• Analyzed the output and verify the data sets to get the accurate outputs.

Unsupervised Learning:

Unsupervised learning is a type of machine learning that helps in finding the previously unknown patterns in the data set without any known labels. It is known as self- organization and allows modelling probability densities of given inputs.

Fig 2.2 unsupervised Learning Some of the algorithm used in unsupervised learning are:

• Clustering

• Anomaly detection

• Neural networks

• Approach for learning latent variable models

• Non labelled data

Semi Supervised Machine Learning algorithm: It’s like the middle man which have some labeled data and some unlabeled which can be prosed by the both the structured and unsupervised learning.

The algorithms have been compared based upon the parameters: Size of the dataset and Number of technical indicators used. Accuracy and F-measure values have been computed for each algorithm. Long term model has been used to compute the accuracy and F-measure.

Reinforcement Learning: This type of learning is used to reinforce or strengthen the network based on critic information. That is, a network being trained under reinforcement learning, receives some feedback from the environment. However, the feedback is evaluative and not instructive as in the case of supervised learning. Based on this feedback, the network performs the adjustments of the weights to obtain better critic information in future.

This learning process is similar to supervised learning but we might have very less information. The following figure gives the block diagram of reinforcement learning:

**import numpy as np**

* NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.
* At the core of the NumPy package, is the ndarray object. This encapsulates n-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:
  + NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an ndarray will create a new array and delete the original.
  + The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.
  + NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python’s built-in sequences.
  + A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today’s scientific/mathematical Python-based software, just knowing how to use Python’s built-in sequence types is insufficient - one also needs to know how to use NumPy arrays.

**import time**

This module provides various time-related functions. For related functionality, see also the datetime and calendar modules.

Although this module is always available, not all functions are available on all platforms. Most of the functions defined in this module call platform C library functions with the same name. It may sometimes be helpful to consult the platform documentation, because the semantics of these functions varies among platforms.

An explanation of some terminology and conventions is in order.

The epoch is the point where the time starts, and is platform dependent. For Unix, the epoch is January 1, 1970, 00:00:00 (UTC). To find out what the epoch is on a given platform, look at time.gmtime(0).

The term seconds since the epoch refers to the total number of elapsed seconds since the epoch, typically excluding leap seconds. Leap seconds are excluded from this total on all POSIX-compliant platforms.

The functions in this module may not handle dates and times before the epoch or far in the future. The cut-off point in the future is determined by the C library; for 32-bit systems, it is typically in 2038.

Function strptime() can parse 2-digit years when given %y format code. When 2-digit years are parsed, they are converted according to the POSIX and ISO C standards: values 69–99 are mapped to 1969–1999, and values 0–68 are mapped to 2000–2068.

UTC is Coordinated Universal Time (formerly known as Greenwich Mean Time, or GMT). The acronym UTC is not a mistake but a compromise between English and French.

DST is Daylight Saving Time, an adjustment of the timezone by (usually) one hour during part of the year. DST rules are magic (determined by local law) and can change from year to year. The C library has a table containing the local rules (often it is read from a system file for flexibility) and is the only source of True Wisdom in this respect.

The precision of the various real-time functions may be less than suggested by the units in which their value or argument is expressed. E.g. on most Unix systems, the clock “ticks” only 50 or 100 times a second.

On the other hand, the precision of time() and sleep() is better than their Unix equivalents: times are expressed as floating point numbers, time() returns the most accurate time available (using Unix gettimeofday() where available), and sleep() will accept a time with a nonzero fraction (Unix select() is used to implement this, where available).

The time value as returned by gmtime(), localtime(), and strptime(), and accepted by asctime(), mktime() and strftime(), is a sequence of 9 integers. The return values of gmtime(), localtime(), and strptime() also offer attribute names for individual fields.

See struct\_time for a description of these objects.

Changed in version 3.3: The struct\_time type was extended to provide the tm\_gmtoff and tm\_zone attributes when platform supports corresponding struct tm members.

Changed in version 3.6: The struct\_time attributes tm\_gmtoff and tm\_zone are now available on all platforms.

**import os**

This module provides a portable way of using operating system dependent functionality. If you just want to read or write a file see open(), if you want to manipulate paths, see the os.path module, and if you want to read all the lines in all the files on the command line see the fileinput module. For creating temporary files and directories see the tempfile module, and for high-level file and directory handling see the shutil module.

Notes on the availability of these functions:

The design of all built-in operating system dependent modules of Python is such that as long as the same functionality is available, it uses the same interface; for example, the function os.stat(path) returns stat information about path in the same format (which happens to have originated with the POSIX interface).

Extensions peculiar to a particular operating system are also available through the os module, but using them is of course a threat to portability.

All functions accepting path or file names accept both bytes and string objects, and result in an object of the same type, if a path or file name is returned.

On VxWorks, os.popen, os.fork, os.execv and os.spawn\*p\* are not supported.

# 

# Chapter-4

# System analysis

## 4.1 EXISTING SYSTEM:

* In existing methods analysis of the viability of Logistic regression to detect the behaviour of network traffic by modelling it as a sequence of states that change over time.
* In existing methods verify the performance of Machine learning model based network in classifying intrusion traffics.

### 4.1.1 DISADVANTAGES OF EXISTING SYSTEM:

* All the above methods treat the entire network traffic as a whole consisting of a sequence of traffic bytes. They don’t make full use of domain knowledge of network traffics.
* Existing methods treats traffics as independent and ignore the internal relations of network traffics.
* .

## 4.2 PROPOSED SYSTEM:

* We use the classic NSL-KDD and the up-to-date benchmark datasets and conduct detailed analysis and data cleaning. (2) proposed an AMI intrusion detection model based on the cross-layer feature fusion of CNN and LSTM to obtain comprehensive features with multi-domain characteristics based on the NSL\_KDD datasets.
* We propose an end-to-end deep learning model with ml models that is composed of logistic regression and attention mechanism. CNN can well solve the problem of Software Defined Networks and provide a new research method for Early Warning Proactive System
* We compare the performance of ML Modes with traditional deep learning methods, the model can extract information from each packet. By making full use of the structure information of network traffic, the logistic regression model can capture features more comprehensively. 4) We evaluate our proposed network with a real NSL-KDD dataset. The experimental results show that the performance of algorithm is better than the traditional methods

.

### 4.2.1 ADVANTAGES OF PROPOSED SYSTEM:

* This method is used to analyse the important degree of packet vectors to obtain fine-grained features which are more salient for malicious traffic detection.
* At the output layer, the features generated by attention mechanism are then imported into a fully connected layer for feature fusion, which obtains the key features that accurately characterize network traffic behaviour.

# CHAPTER- 5

# SYSTEM design

## 5.1 introduction

System Design Introduction:

The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces.

## 5.2 modules

### 5.2.1 Data COLLECTION:

There are three symbolic data types in NSL-KDD data features: protocol type, flag and service. We use one-hot encoder mapping these features into binary vectors. One-Hot Processing: NSL-KDD dataset is processed by one-hot method to transform symbolic features into numerical features. For example, the second feature of the NSL-KDD data sample is protocol type. The protocol type has three values: tcp, udp, and icmp. One-hot method is processed into a binary code that can be recognized by a computer, where tcp is [1, 0, 0], udp is [0, 1, 0], and icmp is [0, 0, 1]

### 5.2.2 Pre-processing:

When the dataset is extracted, part of the data contains some noisy data, duplicate values, missing values, infinity values, etc. due to extraction errors or input errors. Therefore, we first perform data preprocessing. The main work is as follows. (1) Duplicate values: delete the sample’s duplicate value, only keep one valid data. (2) Outliers: in the sample data, the sample size of missing values(Not a Number, NaN) and Infinite values(Inf) is small, so we delete this. (3) Features delete and transform: In CSE-CIC-IDS2018, we delete features such as ‘‘Timestamp’’, ‘‘Destination Address’’, ‘‘Source Address’’, ‘‘Source Port’’, etc. If features ‘‘Init Bwd Win Byts’’ and features ‘‘Init Fwd Win Byts’’ have a value of −1, we add two check dimensions. The mark of −1 is 1. Otherwise, it is 0. In NSL-KDD, we use the One Hot encoder to complete this conversion. For example, ‘‘TCP’’, ‘‘UDP’’ and ‘‘ICMP’’ are functions of three protocol types. After OneHot encoding, they become binary vectors (1, 0, 0), (0, 1, 0), (0, 0, 1). The protocol type function can be divided into three categories, including 11 categories for flag function and 70 categories for service function. Therefore, the 41 dimensions initial feature vector becomes 122 dimensions. (4) Numerical standardization: In order to eliminate the dimensional influence between indicators and accelerate the gradient descent and model convergence, the data is standardized, that is, the method of obtaining Z-Score, so that the average value of each feature becomes 0 and the standard deviation becomes 1, converted to a standard normal distribution, which is related to the overall sample distribution, and each sample point can have an impact on standardization. The standardization formula is as follows, u is the mean of each feature, s is the standard deviation of each feature, and x 0 i is the element corresponding to each column’s features.

### 5.2.3 Train-Test Split and Model FITTING:

Now, we divide our dataset into training and testing data. Our objective for doing this split is to assess the performance of our model on unseen data and to determine how well our model has generalized on training data. This is followed by a model fitting which is an essential step in the model building process.

**Model Evaluation and Predictions:**

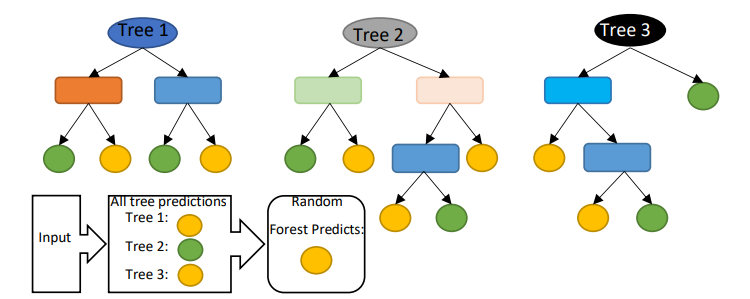
This is the final step, in which we assess how well our model has performed on testing data using certain scoring metrics, I have used 'accuracy score' to evaluate my model. First, we create a model instance, this is followed by fitting the training data on the model using a fit method and then we will use the predict method to make predictions on x\_test or the testing data, these predictions will be stored in a variable called y\_test\_hat. For model evaluation, we will feed the y\_test and y\_test\_hat into the accuracy\_score function and store it in a variable called test\_accuracy, a variable that will hold the testing accuracy of our model. We followed these steps for a variety of classification algorithm models and obtained corresponding test accuracy scores.

**Algorithms**

**RANDOM FOREST**

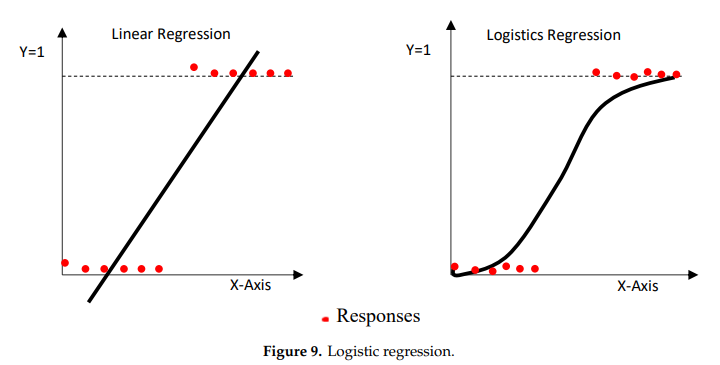
The Random Forest classification model is made up of several decision trees. In simple terms, it combines the results from numerous decision trees to reach a single result. The main difference between decision trees and random forests is that decision trees consider all the possible feature splits, however, random forests will only select a subset of those features.

RF was developed by Breiman, L. [60]. This is an ensemble learning algorithm made up of several DT classifiers, and the output category is determined collectively by these individual trees. When the number of trees in the forest increases, the fallacy in generalization error for forests converges. There are also important benefits of the RF. For example, it can manage high-dimensional data without choosing a feature; trees are independent of each other during the training process, and implementation is fairly simple; however, the training speed is generally fast and, at the same time, the generalization functionality is good enough [4]. Random forest algorithm for machine learning has tree predictions, and based on tree predictions, the RF provides random forest predictions [61]. The RF model is visualized in Figur



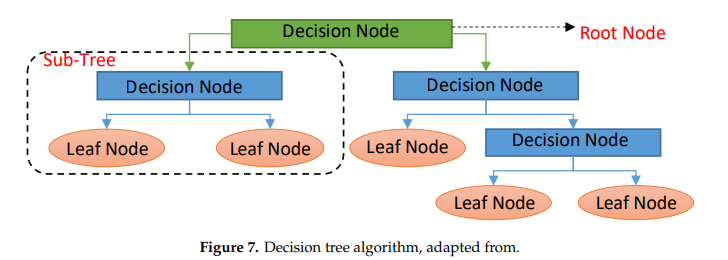
**Logistic Regression (LR):**

reported a study on forecasting the downtime of a printing machine based on real time predictions of imminent failures. In their study, they utilized unstructured historical machine data to train the ML classification algorithms including RF, XGBoost, and LR in predicting the machine failures. Various metrics were analyzed to determine the goodness of fit of the models. These metrics include empirical cross-entropy, area under the receiver operating characteristic curve (AUC), receiver operating characteristic curve itself (ROC), precision-recall curve (PRC), number of false positives (FP), true positives (TP), false negatives (FN), and true negatives (TN) at various decision thresholds, and calibration curves of the estimated probabilities. Based on the results obtained, in terms of ROC, all the algorithms performed significantly better and almost similar. But in terms of decision thresholds, RF and XGBoost perform better than LR. Using a given set of independent variables, linear regression is used to estimate the continuous dependent variations. However, using a given set of independent variables, logistic regression is used to estimate the categorical contingent variations [68]. Graph of the linear regression model and logistics regression model are shown in Figure 9.

****

Decision Tree (DT):

Decision Tree is a network system composed primarily of nodes and branches, and nodes comprising root nodes and intermediate nodes. The intermediate nodes are used to represent a feature, and the leaf nodes are used to represent a class label [52]. DT can be used for feature selection [57]. DT algorithm is presented in Figure



DT classifiers have gained considerable popularity in a number of areas, such as character identification, medical diagnosis, and voice recognition. More notably, the DT model has the potential to decompose a complicated decision-making mechanism into a series of simplified decisions by recursively splitting covariate space into subspaces, thereby offering a solution that is sensitive to interpretation

#### Support Vector Machine

A support vector machine (SVM) is a type of supervised machine learning classification algorithm which outputs an optimal hyperplane that categorizes new examples given labeled training data [[15](#_bookmark72)]. SVMs were introduced initially in 1960s and were later refined in 1990s. However, it is only now that they are becoming very popular, owing to their ability to achieve outstanding results.

Simple SVM: In case of linearly separable data in two dimensions, as shown in Figure [2.6](#_bookmark5), a typical machine learning algorithm tries to find a boundary that divides the data in such a way that the misclassification error can be minimized. If you closely look at Figure [2.6](#_bookmark5), there can be several boundaries that correctly divide the data points. The two dashed lines as well as one solid line classify the data correctly.

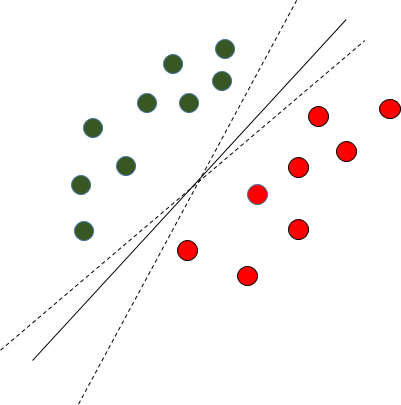


Fig. 2.6 Multiple Decision Boundaries

SVM differs from the other classification algorithms in the way that it chooses the decision boundary that maximizes the distance from the nearest data points of all the classes. An SVM doesn’t merely find a decision boundary; it finds the most optimal decision boundary. The most optimal decision boundary is the one which has maximum margin from the nearest points of all the classes. The nearest points from the decision boundary that maximize the distance between the decision boundary and the points are called support vectors as seen in Figure¬[2.7](#_bookmark6). The decision boundary in case of support vector machines is called the maximum margin classifier, or the maximum margin hyper plane.

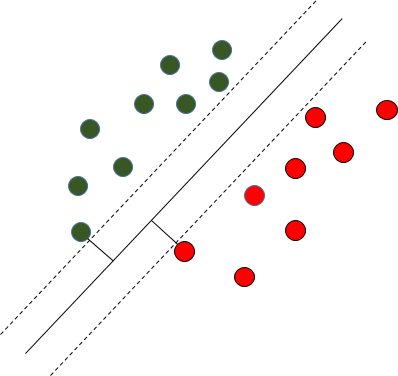


Fig. 2.7 Decision Boundary with Support Vectors

Kernel SVM: In the previous two figures Figure [2.6](#_bookmark5) and Figure [2.7](#_bookmark6) it was shown how the simple SVM algorithm can be used to find decision boundary for linearly separable data. However, in the case of non-linearly separable data, such as the one shown in Figure [2.8](#_bookmark7), a straight line cannot be used as a decision boundary.

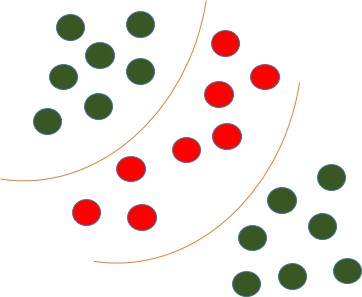


Fig. 2.8 Non-linearly Separable Data

In case of non-linearly separable data, the simple SVM algorithm cannot be used. Rather, a modified version of SVM, called Kernel SVM, is used. Basically, the kernel SVM projects the non-linearly separable data lower dimensions to linearly separable data in higher dimen- sions in such a way that data points belonging to different classes are allocated to different dimensions.

#### Random Forest

Random forest is a tree-based algorithm which involves building several trees (decision trees), then combining their output to improve generalization ability of the model. The method of combining trees is known as an ensemble method. Ensembling is nothing but a combination of weak learners (individual trees) to produce a strong learner.

Definition: A random forest is a classifier consisting of a collection of tree structured classifiers *h*(*x,* Θ*k*)*, k* = 1*, ...* where the Θ*k* are independent identically distributed (*i.i.d*) random vectors and each tree casts a unit vote for the most popular class at input [[4](#_bookmark61)].

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

* + - * Pick N random records from the dataset.
      * Build a decision tree based on these N records.
      * Choose the number of trees you want in your algorithm and repeat steps (i) and (ii).
      * In case of a classification problem, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote.

Figure [2.1](#_bookmark0) shows different trees labelling the class differently. What ensemble does is take the mode (maximum occurring class) of the output produced by n different trees to create a better model. To say it in simple words: Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

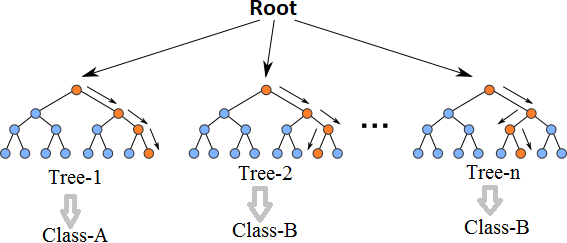


Fig. 2.1 Multiple decision trees [[12](#_bookmark69)]

Even though decision trees are pretty intuitive and easier to understand, they can be very noisy. Few changes in the data can lead to different splits and completely different models. The instability of the tree makes it unrealistic as a prediction model by itself. A single decision tree is insufficient and generally overfits the data, that is it can capture the structure of the in-sample data very well, but it tends to work poorly out-of-sample. In the context of statistics, decisions trees have low bias (as it can fit the data well) but high variances (the predictions are noisy).

Understanding the working principle of decision trees is imperative in the understanding of Random Forest Algorithm. The most popular algorithm for decision trees is ID3 algorithm. It finds the best attributes/features that best classifies the target attribute. One of the most commonly used way to figure out the best attribute is by calculating Information Gain which is, in turn, calculated using another property called Entropy.

The calculation of entropy of a system is done as follows:

*c*

*Entropy*(*S*) = ∑ *pilog*2 *pi* (2.1)

−

*i*=1

Here, c is the total number of classes or attributes and *pi* is number of examples belonging to the *ith* class. Information gain is simply the expected reduction in entropy caused by partitioning all our examples according to a given attribute. Mathematically, it is defined as:

*Gain*(*S, A*) ≡ *Entropy*(*S*) − ∑ |*Sv*| *Entropy*(*Sv*) (2.2)

*v*∈*Values*(*A*) |*S*|

S refers to the entire set of examples that we have. A is the attribute we want to partition or split. |S| is the number of examples and |*Sv*| is the number of examples for the current value of attribute A. The attribute with the highest information gain sits at the root node, and the tree is first split based on that attribute.

#### XGBoost

XGBoost is another ensemble learning method. As it is almost never sufficient to reply upon the results of just one model, it combines the predictive powers of multiple learners to reach a conclusion. The base learners are weak learners in which the bias is high, and the predictive power is just slightly better than random guessing. But each of these weak learners add some vital information for prediction, resulting in a strong learner by effectively combining these weak learners. The final strong learner brings down both the bias and the variance.

The tree ensemble model consists of a set of classification and regression trees (CART). Figure [2.2](#_bookmark1) shows a simple example of a CART that classifies whether someone will like an

app or not. The original figure from [[5](#_bookmark62)] had been modified to paint a better picture of our dataset.

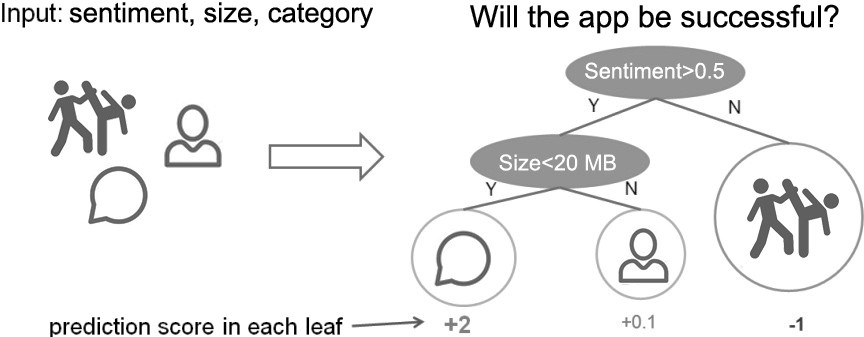


Fig. 2.2 CART Model Representation

Suppose, the many app categories available are classified into different leaves and as- signed a score on the corresponding leaf. Unlike decision trees, in which the leaf only contains decision values, in CART, a real score is associated with each of the leaves, which gives a better interpretation.

The task of training the model involves finding the best parameters *θ* that best fit the

training data *xi* and labels *yi*. This is done via the objective function which measures how well the model fits the training data. Objective functions are composed of two parts: training loss and regularization term which can be denoted by:

*ob j*(*θ* ) = *L*(*θ* ) + Ω(*θ* ) (2.3)

where *L* is the training loss function, and Ω is the regularization term. The regularization term controls the complexity of the model, helping to avoid overfitting.

While trees are built in a parallel manner in bagging, boosting builds trees sequentially such that each subsequent tree aims to reduce the errors of the previous tree. Figure [2.3](#_bookmark2) perfectly illustrates the concept. Due to each tree learning from its predecessors and updating the residual errors (difference between an observed y-value and the corresponding predicted y-value), the tree that grows next in the sequence will always learn from an updated version of the residuals. This is known as an additive strategy where what has already been learned is fixed, and a new tree is added one at a time.

The boosting process in its absolute basic can be broken down into the following steps [[22](#_bookmark79)]:

* + - * Fit a model to the data: *F*1(*x*) = *y*
      * Fit a model to the residuals: *h*1(*x*) = *y* − *F*1(*x*)

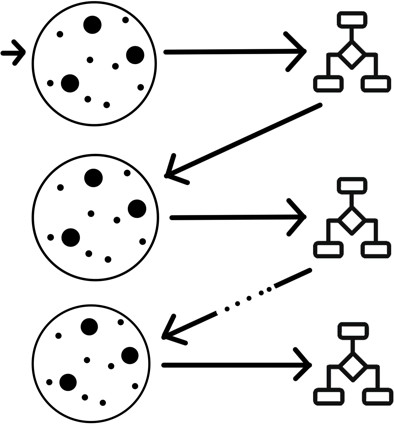


Fig. 2.3 Sequential Tree Structure

* Create a new model: *F*2(*x*) = *F*1(*x*) + *h*1(*x*)

By creating more models that correct the errors of the previous models, this can be generalized to:

*F*(*x*) = *F*1(*X* ) → *F*2(*x*) = *F*1(*x*) + *h*1(*x*)*. . . ..* → *FM*(*x*) = *FM*−1(*x*) + *hM*−1(*x*)*.* (2.4)

At each step, the residual would also need to be calculated: *hm*(*x*) = *y* − *Fm*(*x*) where *hm*(*x*) can be any model, but in our case, it is a tree-based learner. With this in mind, suppose that instead of training *h*0 on the residuals of *F*0, we train *h*0 on the gradient of the loss function, *L*(*y, F*0(*x*)) with respect to the prediction values produced by *Fm*(*x*). With samples in *hm* grouped into leaves, an average gradient can be calculated and then scaled by some factor, *γ*, such that *Fm* + *γhm* minimizes the loss function for the samples in each leaf. In practice, a different factor is chosen for each leaf. For iteration m = 1 to M:

* Calculate the gradient of L at the point *sm*−1
* “Step” in the direction of greatest descent (the negative gradient) with step size *γ*. That is, *sm* = *sm*−1 − *γL*(*sm*−1). If *γ* is small and *M* is sufficiently large, *sM* will be the location of *L* ‘s minimum value.

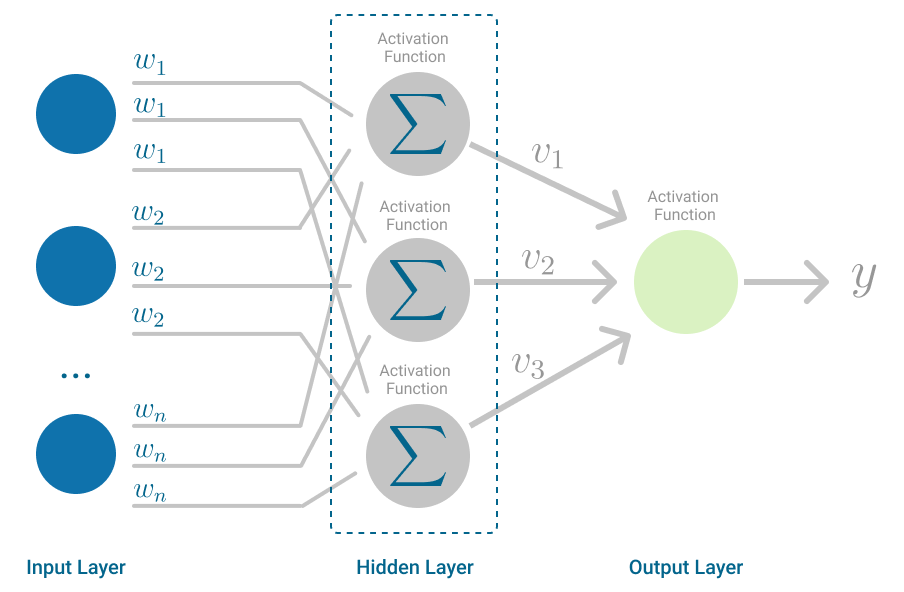
Most of these are true for all previous gradient boosting algorithms that came before XGBoost, but what really separates it from the others is [[22](#_bookmark79)]:

* Regularization: XGBoost can penalize complex models through both L1 and L2 regularization which helps prevent over-fitting.
  + Handling sparse data: Missing values or data processing steps like one-hot encoding can make data sparse. XGBoost incorporates a sparsity-aware split finding algorithm that can handle different types of sparsity patterns in the data.
  + Weighted quantile sketch: Most existing tree based algorithms can find the split points when the data points are of equal weights (using quantile sketch algorithm). However, they can not handle weighted data. XGBoost has a distributed weighted quantile sketch algorithm that can effectively handle weighted data.
  + Block structure for parallel learning: For faster computing, XGBoost can utilize multiple cores on the CPU. Unlike other algorithms, this enables the data layout to be reused by subsequent iterations, instead of computing it again.
  + Cache awareness: In XGBoost, non-continuous memory access is required to get the gradient statistics by row index. Hence, XGBoost has been designed to make optimal use of hardware.
  + Out-of-core computing: This feature optimizes the available disk space and maximizes its usage when handling huge datasets that do not fit into memory

**Multilayer Perception :**

The**Multilayer Perceptron** was developed to tackle this limitation. It is a neural network where the mapping between inputs and output is non-linear.

A Multilayer Perceptron has input and output layers, and one or more **hidden layers** with many neurons stacked together. And while in the Perceptron the neuron must have an activation function that imposes a threshold, like ReLU or sigmoid, neurons in a Multilayer Perceptron can use any arbitrary activation function.



Multilayer Perceptron. (Image by author)

Multilayer Perceptron falls under the category of [feedforward algorithms](https://en.wikipedia.org/wiki/Feedforward_neural_network), because inputs are combined with the initial weights in a weighted sum and subjected to the activation function, just like in the Perceptron. But the difference is that each linear combination is propagated to the next layer.

Each layer is feeding the next one with the result of their computation, their internal representation of the data. This goes all the way through the hidden layers to the output layer.

But it has more to it.

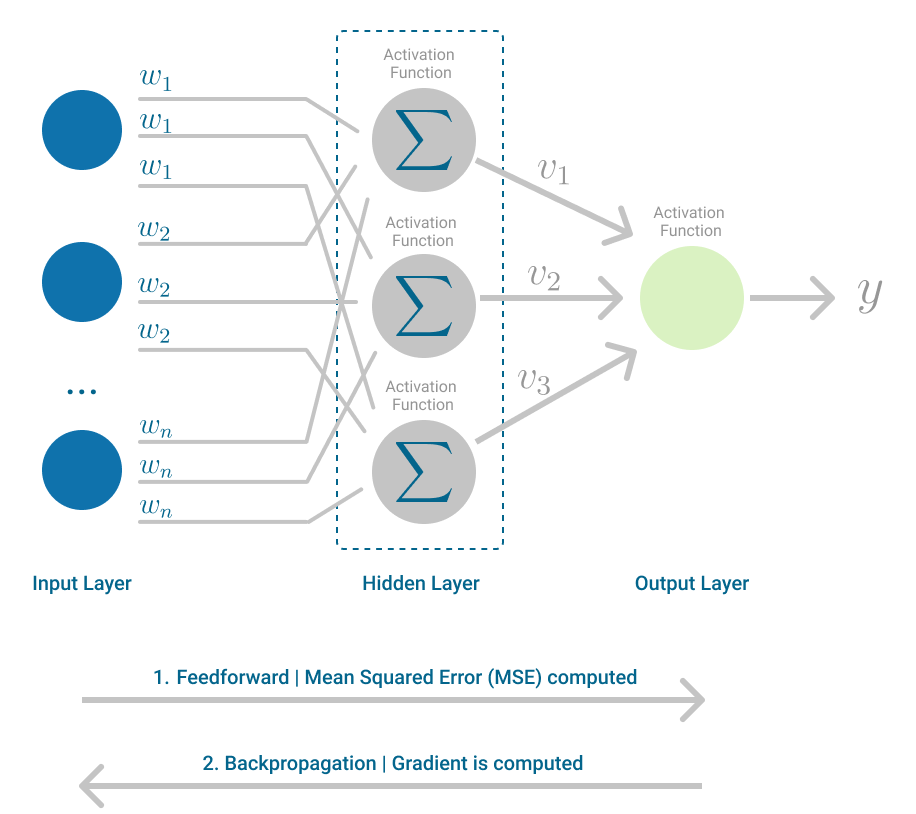
If the algorithm only computed the weighted sums in each neuron, propagated results to the output layer, and stopped there, it wouldn’t be able to learn the weights that minimize the cost function. If the algorithm only computed one iteration, there would be no actual learning.

This is where [**Backpropagation**](https://en.wikipedia.org/wiki/Backpropagation)[7] comes into play.

# Backpropagation

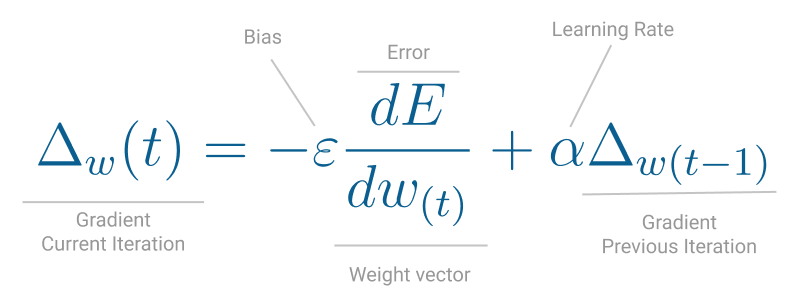
Backpropagation is the learning mechanism that allows the Multilayer Perceptron to iteratively adjust the weights in the network, with the goal of minimizing the cost function.

There is one hard requirement for backpropagation to work properly. The function that combines inputs and weights in a neuron, for instance the weighted sum, and the threshold function, for instance ReLU, must be differentiable. These functions must have a **bounded derivative**, because [Gradient Descent](https://towardsdatascience.com/stochastic-gradient-descent-explained-in-real-life-predicting-your-pizzas-cooking-time-b7639d5e6a32) is typically the optimization function used in MultiLayer Perceptron.



Multilayer Perceptron, highlighting the Feedforward and Backpropagation steps. (Image by author)

In each iteration, after the weighted sums are forwarded through all layers, the gradient of the **Mean Squared Error** is computed across all input and output pairs. Then, to propagate it back, the weights of the first hidden layer are updated with the value of the gradient. That’s how the weights are propagated back to the starting point of the neural network!



One iteration of Gradient Descent. (Image by author)

This process keeps going until gradient for each input-output pair has converged, meaning the newly computed gradient hasn’t changed more than a specified convergence threshold, compared to the previous iteration.

## 5.3 system architecture

A system architecture or systems architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system. Organized in a way that supports reasoning about the structures and behaviors of the system.

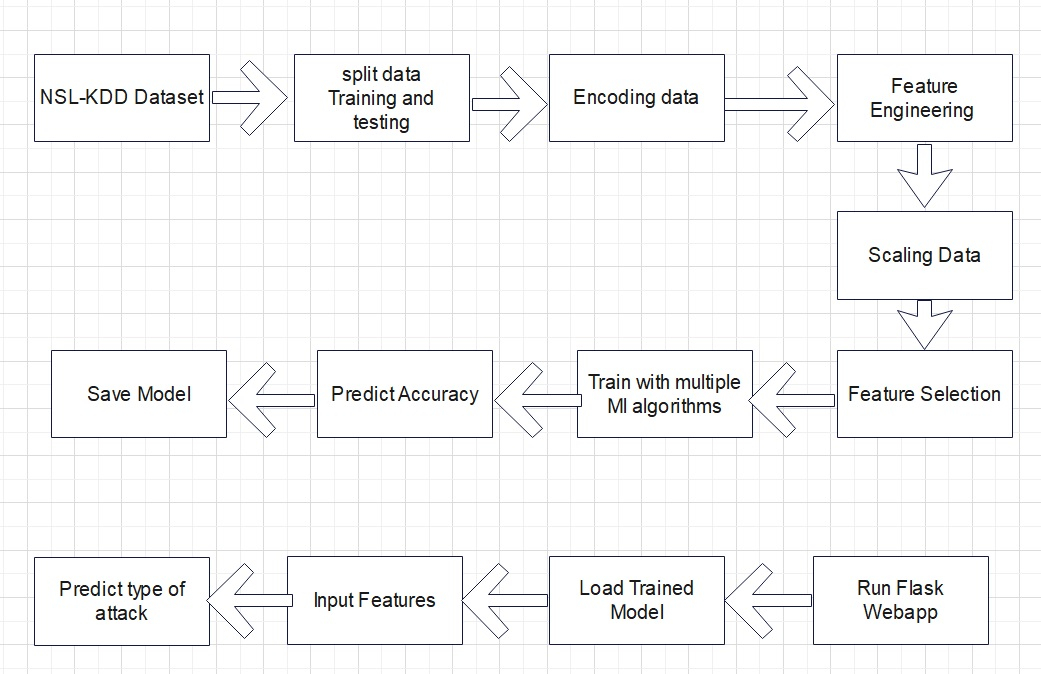


Figure 5. 1 System Architecture

3-Tier Architecture:

The three-tier software architecture (a three-layer architecture) emerged in the 1990s to overcome the limitations of the two-tier architecture. The third tier (middle tier server) is between the user interface (client) and the data management (server) components. This middle tier provides process management where business logic and rules are executed and can accommodate hundreds of users (as compared to only 100 users with the two tier architecture) by providing functions such as queuing, application execution, and database staging.

The three tier architecture is used when an effective distributed client/server design is needed that provides (when compared to the two tier) increased performance, flexibility, maintainability, reusability, and scalability, while hiding the complexity of distributed processing from the user. These characteristics have made three layer architectures a popular choice for Internet applications and net-centric information systems.

**Advantages of Three-Tier:**

* Separates functionality from presentation.
* Clear separation – better understanding.
* Changes limited to well define components.
* Can be running on WWW.
* Effective network performance.

## 5.4 UML DAIGRAMS

Global Use Case Diagrams:

Identification of actors:

Actor: Actor represents the role a user plays with respect to the system. An actor interacts with, but has no control over the use cases.

Graphical representation:



<<Actor name>>

An actor is someone or something that:

Interacts with or uses the system.

* Provides input to and receives information from the system.
* Is external to the system and has no control over the use cases.

Actors are discovered by examining:

* Who directly uses the system?
* Who is responsible for maintaining the system?
* External hardware used by the system.
* Other systems that need to interact with the system.

Questions to identify actors:

* + Who is using the system? Or, who is affected by the system? Or, which groups need help from the system to perform a task?
  + Who affects the system? Or, which user groups are needed by the system to perform its functions? These functions can be both main functions and secondary functions such as administration.
  + Which external hardware or systems (if any) use the system to perform tasks?
  + What problems does this application solve (that is, for whom)?
  + And, finally, how do users use the system (use case)? What are they doing with the system?

The actors identified in this system are:

1. System Administrator
2. Customer
3. Customer Care

Identification of use cases:

Use case: A use case can be described as a specific way of using the system from a user’s (actor’s) perspective.

Graphical representation:



A more detailed description might characterize a use case as:

* Pattern of behavior the system exhibits
* A sequence of related transactions performed by an actor and the system
* Delivering something of value to the actor

Use cases provide a means to:

* capture system requirements
* communicate with the end users and domain experts
* test the system

Use cases are best discovered by examining the actors and defining what the actor will be able to do with the system.

Guide lines for identifying use cases:

* For each actor, find the tasks and functions that the actor should be able to perform or that the system needs the actor to perform. The use case should represent a course of events that leads to clear goal
* Name the use cases.
* Describe the use cases briefly by applying terms with which the user is familiar.

This makes the description less ambiguous

Questions to identify use cases:

* What are the tasks of each actor?
* Will any actor create, store, change, remove or read information in the system?
* What use case will store, change, remove or read this information?
* Will any actor need to inform the system about sudden external changes?
* Does any actor need to inform about certain occurrences in the system?
* What usecases will support and maintains the system?

**1.2 Flow of Events**

A flow of events is a sequence of transactions (or events) performed by the system. They typically contain very detailed information, written in terms of what the system should do, not how the system accomplishes the task. Flow of events are created as separate files or documents in your favorite text editor and then attached or linked to a use case using the Files tab of a model element.

A flow of events should include:

* When and how the use case starts and ends
* Use case/actor interactions
* Data needed by the use case
* Normal sequence of events for the use case
* Alternate or exceptional flows

### 5.4.1 Construction of Use case diagrams:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

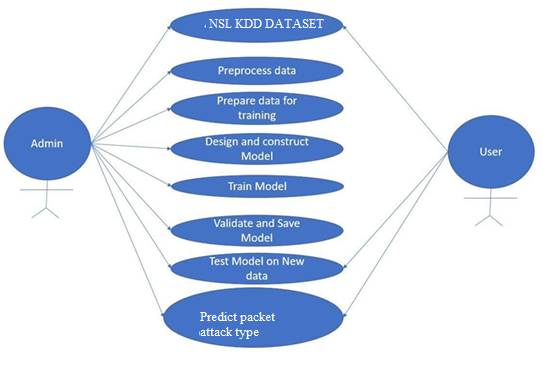


Figure 5. 2 Use Case Diagram

### 5.4.2 SEQUENCE DIAGRAMS:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

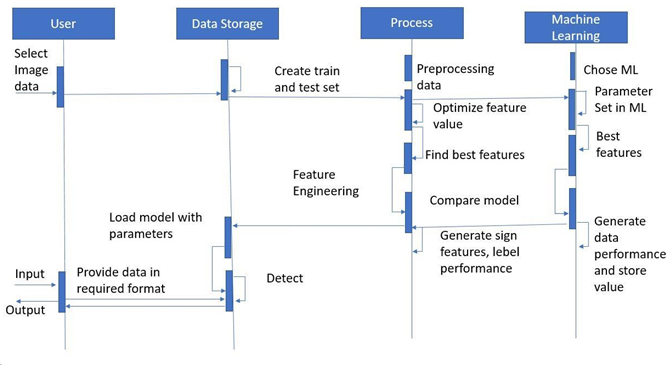


Figure 5. 3 Sequence diagram

### 5.4.3 CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

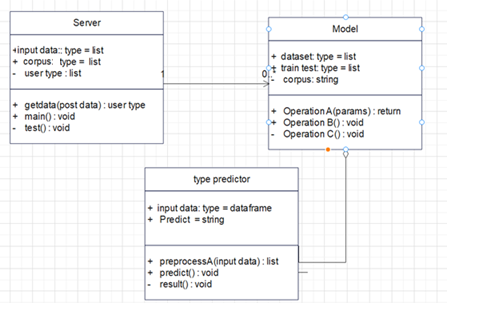


Figure 5. 4 Class Diagram

### 5.4.4 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

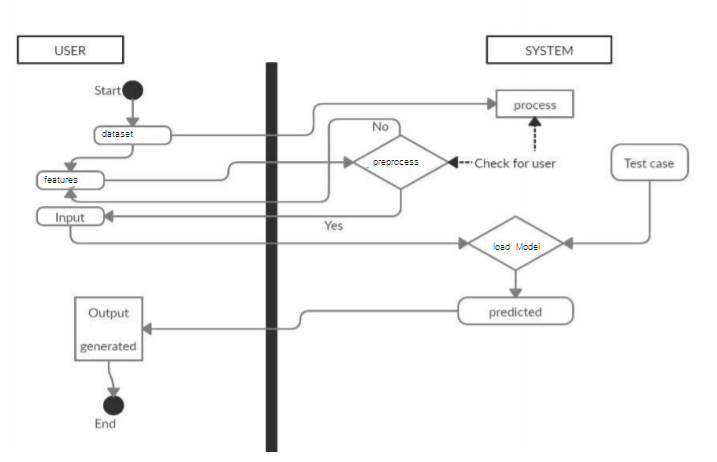
****

Figure 5. 5 Activity Diagram

# 

# CHAPTER-6

# system requirements

## 6.1 SYSTEM REQUIREMENTS

### 6.1.1 HARDWARE REQUIREMENTS:

* System : Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz
* Hard Disk : 1 TB.
* Input Devices : Keyboard, Mouse
* Ram : 4 GB.

### 6.1.2 SOFTWARE REQUIREMENTS:

* Operating system : Windows XP/7/10.
* Coding Language : Python
* Tool : Anaconda
* Interface : OPENCV

# Chapter-7

# System implementation

To conduct studies and analyses of an operational and technological nature, and To promote the exchange and development of methods and tools for operational analysis as applied to defense problems.

## 7.1 input and output designs

### 7.1.1 Logical design

The logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. This is often conducted via modeling, using an over-abstract (and sometimes graphical) model of the actual system. In the context of systems design are included. Logical design includes ER Diagrams i.e. Entity Relationship Diagrams

### 7.1.2 Physical design

The physical design relates to the actual input and output processes of the system. This is laid down in terms of how data is input into a system, how it is verified / authenticated, how it is processed, and how it is displayed as output. In Physical design, following requirements about the system are decided.

1. Input requirement,
2. Output requirements,
3. Storage requirements,
4. Processing Requirements,
5. System control and backup or recovery.

Put another way, the physical portion of systems design can generally be broken down into three sub-tasks:

1. User Interface Design
2. Data Design
3. Process Design

User Interface Design is concerned with how users add information to the system and with how the system presents information back to them. Data Design is concerned with how the data is represented and stored within the system. Finally, Process Design is concerned with how data moves through the system, and with how and where it is validated, secured and/or transformed as it flows into, through and out of the system. At the end of the systems design phase, documentation describing the three sub-tasks is produced and made available for use in the next phase.

Physical design, in this context, does not refer to the tangible physical design of an information system. To use an analogy, a personal computer's physical design involves input via a keyboard, processing within the CPU, and output via a monitor, printer, etc. It would not concern the actual layout of the tangible hardware, which for a PC would be a monitor, CPU, motherboard, hard drive, modems, video/graphics cards, USB slots, etc. It involves a detailed design of a user and a product database structure processor and a control processor. The H/S personal specification is developed for the proposed system.

## 7.2 INPUT & OUTPUT REPRESENTATION

### 7.2.1 Input Design

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

* What data should be given as input?
* How the data should be arranged or coded?
* The dialog to guide the operating personnel in providing input.
* Methods for preparing input validations and steps to follow when error occur.

### 7.2.2 Objectives

Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

### Output Design

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

* 1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
  2. Select methods for presenting information.
  3. Create document, report, or other formats that contain information produced by the system.

**Code**

import pandas as pd

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

import numpy as np

from sklearn.utils import shuffle

train=pd.read\_csv("multiData.csv")

#train=pd.read\_csv("faultData.csv")

train.head()

train = shuffle(train)

train.columns

y\_train=train['label2']

#y\_train=train['label1']

train=train.drop(['label2'],axis=1)

#train=train.drop(['label1'],axis=1)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(train, y\_train, test\_size=0.05, random\_state=42)

X\_train.info()

from sklearn.ensemble import AdaBoostClassifier

from sklearn.metrics import accuracy\_score

ada= AdaBoostClassifier()

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

pred=ada.predict(X\_test)

print("Adaboost Accuracy Score with Test Data")

accuracy\_score(y\_test, pred)\*100

ada.score(X\_train,y\_train)\*100

# import pickle

# filename = 'rf\_model.sav'

# pickle.dump(DecisionTree, open(filename, 'wb'))

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

DecisionTree= DecisionTreeClassifier()

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

pred=DecisionTree.predict(X\_test)

print(pred)

print(X\_test[0:5])

print("DecisionTreeClassifier Accuracy Score with Test Data")

accuracy\_score(y\_test, pred)\*100

DecisionTree.score(X\_train,y\_train)

from sklearn.linear\_model import LogisticRegression

logreg= LogisticRegression()

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

pred=logreg.predict(X\_test)

print("LogisticRegression Accuracy Score with Test Data")

accuracy\_score(y\_test, pred)

logreg.score(X\_train,y\_train)

from sklearn.ensemble import RandomForestClassifier

Decisionforest= RandomForestClassifier()

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

pred=Decisionforest.predict(X\_test)

print(pred)

print(X\_test[0:20])

print("RandomForestClassifier Accuracy Score with Test Data")

accuracy\_score(y\_test, pred)\*100

Decisionforest.score(X\_train,y\_train)

# Chapter-8

# System testing

## 8.1 INTRODUCTION:

Testing is the debugging program is one of the most critical aspects of the computer programming triggers, without programming that works, the system would never produce an output of which it was designed. Testing is best performed when user development is asked to assist in identifying all errors and bugs. The sample data are used for testing. It is not quantity but quality of the data used the matters of testing. Testing is aimed at ensuring that the system was accurately an efficiently before live operation commands.

Testing objectives:

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say, testing is a process of executing a program with intent of finding an error.

1. A successful test is one that uncovers an as yet undiscovered error.
2. A good test case is one that has probability of finding an error, if it exists.
3. The test is inadequate to detect possibly present errors.
4. The software more or less confirms to the quality and reliable standards.

## 8.2 Levels of Testing

Code testing:

This examines the logic of the program. For example, the logic for updating various sample data and with the sample files and directories were tested and verified.

Specification Testing:

Executing this specification starting what the program should do and how it should performed under various conditions. Test cases for various situation and combination of conditions in all the modules are tested.

Unit testing:

In the unit testing we test each module individually and integrate with the overall system. Unit testing focuses verification efforts on the smallest unit of software design in the module. This is also known as module testing. The module of the system is tested separately. This testing is carried out during programming stage itself. In the testing step each module is found to work satisfactorily as regard to expected output from the module. There are some validation checks for fields also. For example the validation check is done for varying the user input given by the user which validity of the data entered. It is very easy to find error debut the system.

Each Module can be tested using the following two Strategies:

1. Black Box Testing
2. White Box Testing

### 8.2.1 BLACK BOX TESTING

What is Black Box Testing?

Black box testing is a software testing techniques in which functionality of the software under test (SUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on the software requirements and specifications.

In Black Box Testing we just focus on inputs and output of the software system without bothering about internal knowledge of the software program.



The above Black Box can be any software system you want to test. For example : an operating system like Windows, a website like Google ,a database like Oracle or even your own custom application. Under Black Box Testing , you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

Black box testing - Steps

Here are the generic steps followed to carry out any type of Black Box Testing.

* Initially requirements and specifications of the system are examined.
* Tester chooses valid inputs (positive test scenario) to check whether SUT processes them correctly. Also some invalid inputs (negative test scenario) are chosen to verify that the SUT is able to detect them.
* Tester determines expected outputs for all those inputs.
* Software tester constructs test cases with the selected inputs.
* The test cases are executed.
* Software tester compares the actual outputs with the expected outputs.
* Defects if any are fixed and re-tested.

Types of Black Box Testing

There are many types of Black Box Testing but following are the prominent ones -

* Functional testing – This black box testing type is related to functional requirements of a system; it is done by software testers.
* Non-functional testing – This type of black box testing is not related to testing of a specific functionality, but non-functional requirements  such as performance, scalability, usability.
* Regression testing – Regression testing is done  after code fixes , upgrades or any other system maintenance to check the new code has not affected the existing code.

### 8.2.2 WHITE BOX TESTING

White Box Testing is the testing of a software solution's internal coding and infrastructure. It focuses primarily on strengthening security, the flow of inputs and outputs through the application, and improving design and usability.White box testing is also known as clear, open, structural, and glass box testing.

It is one of two parts of the "box testing" approach of software testing. Its counter-part, blackbox testing, involves testing from an external or end-user type perspective. On the other hand, Whitebox testing is based on the inner workings of an application and revolves around internal testing. The term "whitebox" was used because of the see-through box concept. The clear box or whitebox name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings. Likewise, the "black box" in "black box testing" symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested

WHAT DO YOU VERIFY IN WHITE BOX TESTING?

White box testing involves the testing of the software code for the following:

* Internal security holes
* Broken or poorly structured paths in the coding processes
* The flow of specific inputs through the code
* Expected output
* The functionality of conditional loops
* Testing of each statement, object and function on an individual basis

The testing can be done at system, integration and unit levels of software development. One of the basic goals of whitebox testing is to verify a working flow for an application. It involves testing a series of predefined inputs against expected or desired outputs so that when a specific input does not result in the expected output, you have encountered a bug.

HOW DO YOU PERFORM WHITE BOX TESTING?

  To give you a simplified explanation of white box testing, we have divided it into **two basic steps**. This is what testers do when testing an application using the white box testing technique:

**STEP 1) UNDERSTAND THE SOURCE CODE**

The first thing a tester will often do is learn and understand the source code of the application. Since white box testing involves the testing of the inner workings of an application, the tester must be very knowledgeable in the programming languages used in the applications they are testing. Also, the testing person must be highly aware of secure coding practices. Security is often one of the primary objectives of testing software. The tester should be able to find security issues and prevent attacks from hackers and naive users who might inject malicious code into the application either knowingly or unknowingly.

**Step 2) CREATE TEST CASES AND EXECUTE**

The second basic step to white box testing involves testing the application’s source code for proper flow and structure. One way is by writing more code to test the application’s source code. The tester will develop little tests for each process or series of processes in the application. This  method requires that the tester must have intimate knowledge of the code and is often done by the developer. Other methods include manual testing, trial and error testing and the use of testing tools as we will explain further on in this article.

Unit testing:

|  |  |
| --- | --- |
| Sl # Test Case : ­ | UTC­1 |
| Name of Test: ­ | Load dataset |
| Items being tested: ­ | Dataset features and labels are displayed or not |
| Sample Input: ­ | Dataset csv file |
| Expected output: ­ | All features and labels should be displayed |
| Actual output: ­ | Total data is displayed |
| **Remarks: ­** | **Pass.** |

|  |  |
| --- | --- |
| Sl # Test Case : ­ | UTC­2 |
| Name of Test: ­ | Split data |
| Items being tested: ­ | Data is divided in to train and test set |
| Sample Input: ­ | Test and train size |
| Expected output: ­ | Dataset is divided in to 2 parts |
| Actual output: ­ | Based on given test size data is divided and stored in train and test sets |
| Remarks: ­ | pass |

**Integration Testing:**

Integration testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing. Integration testing is defined as the testing of combined parts of an application to determine if they function correctly. It occurs after unit testing and before validation testing. Integration testing can be done in two ways: Bottom­up integration testing and Top­down integration testing.

* + 1. **Bottom­up Integration**

This testing begins with unit testing, followed by tests of progressively higher­level combinations of units called modules or builds.

* + 1. **Top­down Integration**

In this testing, the highest­level modules are tested first and progressively, lower­level modules are tested thereafter.

In a comprehensive software development environment, bottom­up testing is usually done first, followed by top­down testing. The process concludes with multiple tests of the complete application, preferably in scenarios designed to mimic actual situations. Table 6.5 shows the test cases for integration testing and their results

|  |  |
| --- | --- |
| Sl # Test Case : ­ | ITC­1 |
| Name of Test: ­ | Train Model |
| Item being tested: ­ | Model fit is performed |
| Sample Input: ­ | Train x and train y |
| Expected output: ­ | Fit is performed |
| Actual output: ­ | Training is done and accuracy is displayed |
| Remarks: ­ | Pass. |

|  |  |
| --- | --- |
| Sl # Test Case : ­ | ITC­2 |
| Name of Test: ­ | Accuracy calculation |
| Item being tested: ­ | If accuracy of each algorithm is calculated |
| Sample Input: ­ | Test x and test y |
| Expected output: ­ | Accuracy of each algorithm |
| Actual output: ­ | Accuracy of each model |
| Remarks: ­ | Pass. |

**System testing**:

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black­box testing, and as such, should require no knowledge of the inner design of the code or logic. System testing is important because of the following reasons:

System testing is the first step in the Software Development Life Cycle, where the application is tested as a whole.

The application is tested thoroughly to verify that it meets the functional and technical specifications.

The application is tested in an environment that is very close to the production environment where the application will be deployed.

System testing enables us to test, verify, and validate both the business requirements as well as the application architecture.

System Testing is shown in below tables

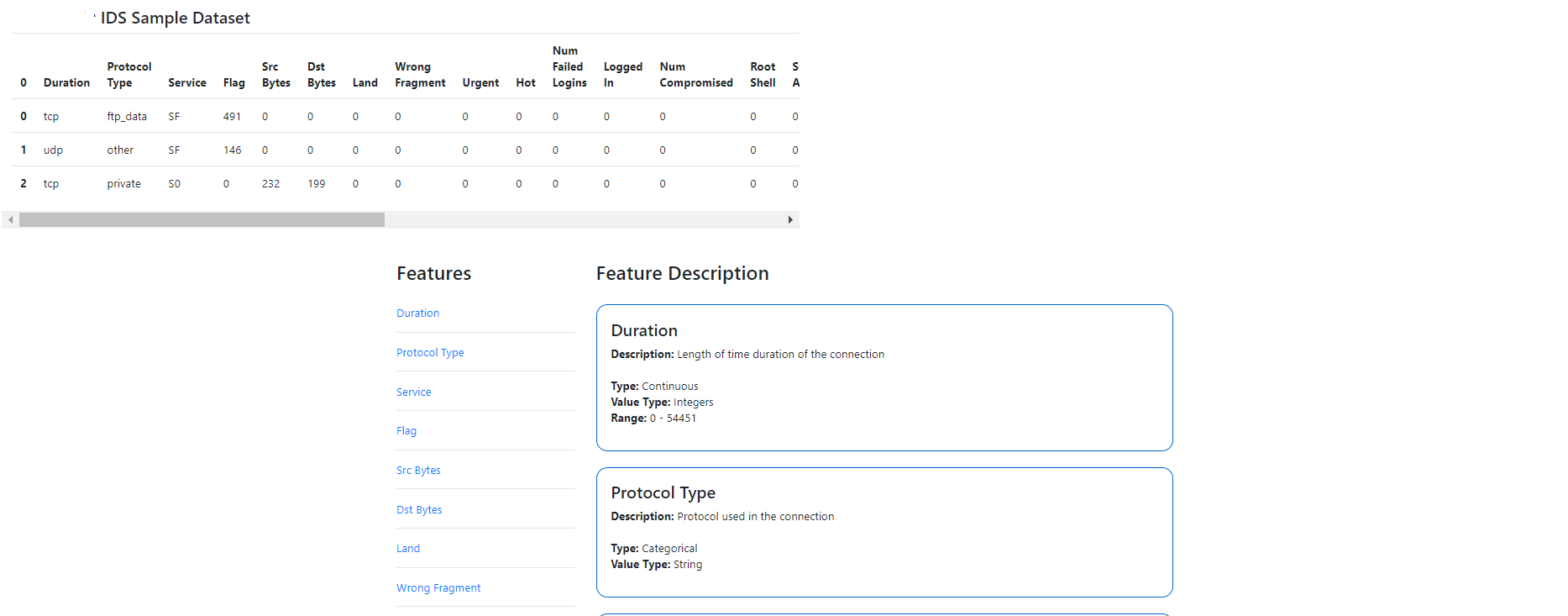
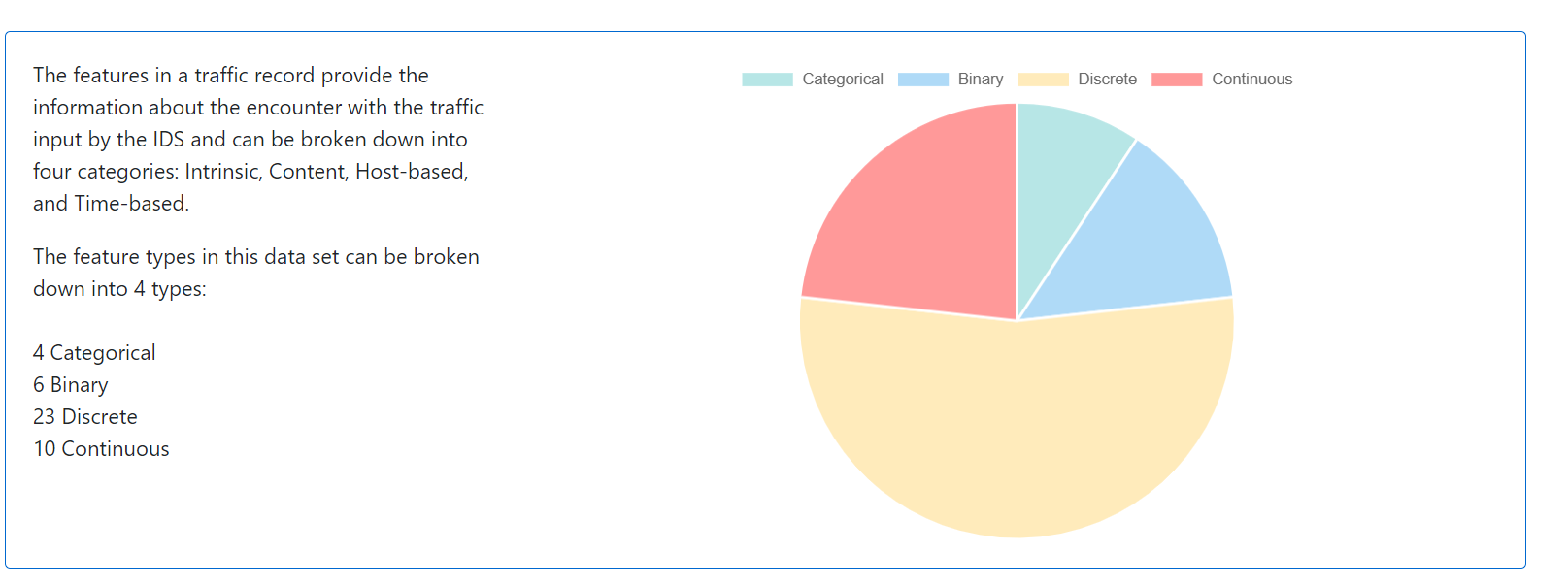
|  |  |
| --- | --- |
| Sl # Test Case : ­ | STC­1 |
| Name of Test: ­ | System testing in various versions of OS |
| Item being tested: ­ | OS compatibility. |
| Sample Input: ­ | Execute the program in windows XP/ Windows­7/8 |
| Expected output: ­ | Performance is better in windows­7 |
| Actual output: ­ | Same as expected output, performance is better in windows­7 |
| Remarks: ­ | Pass |

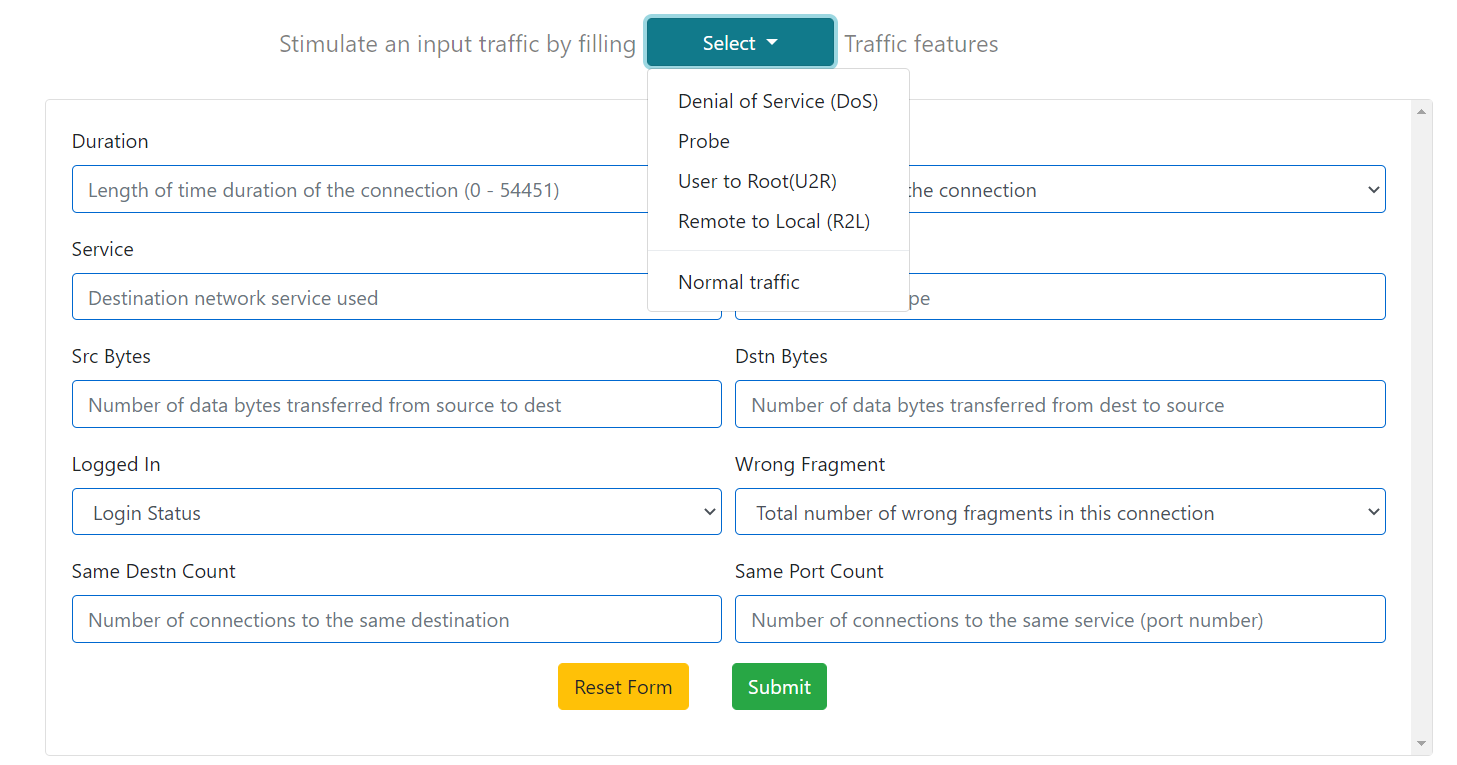
# CHAPTER-9

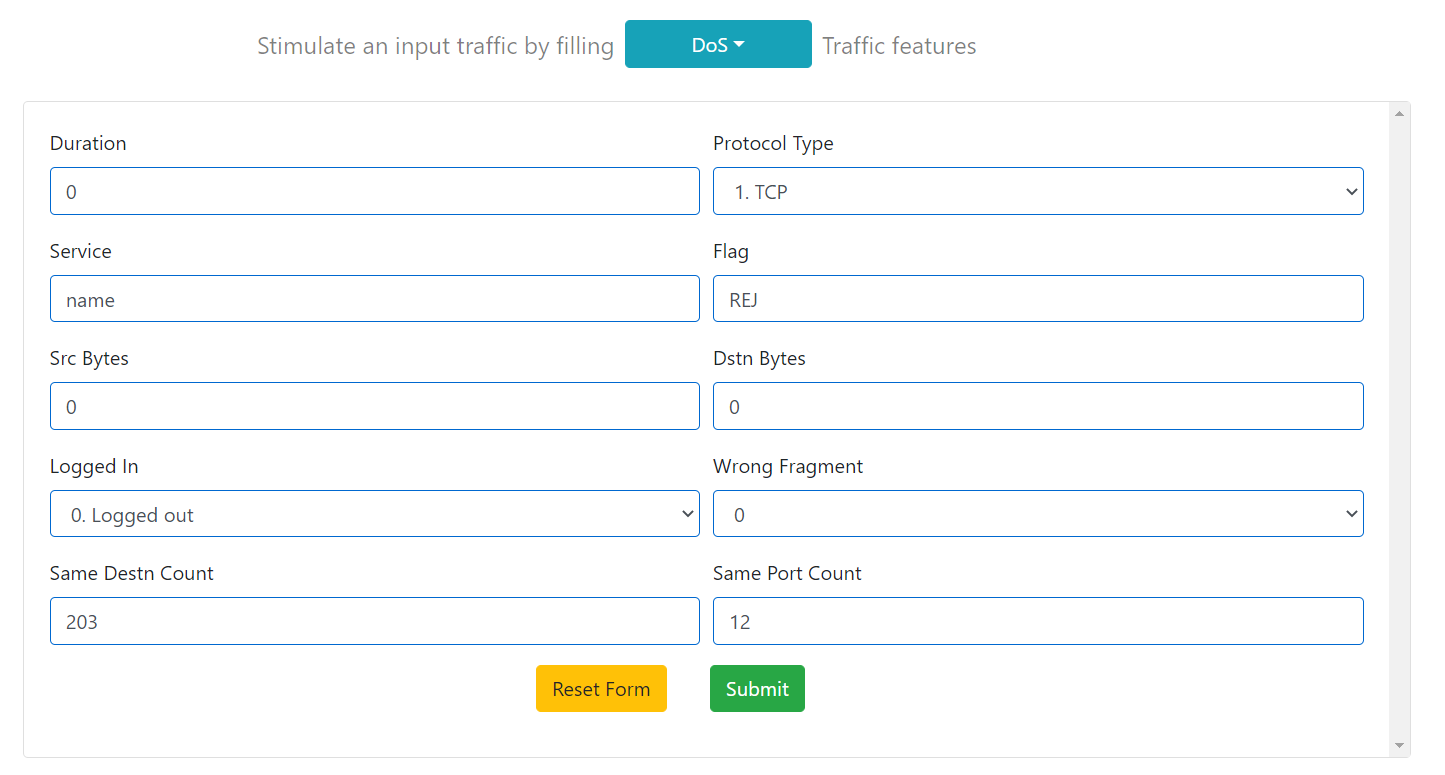
# Output Screens

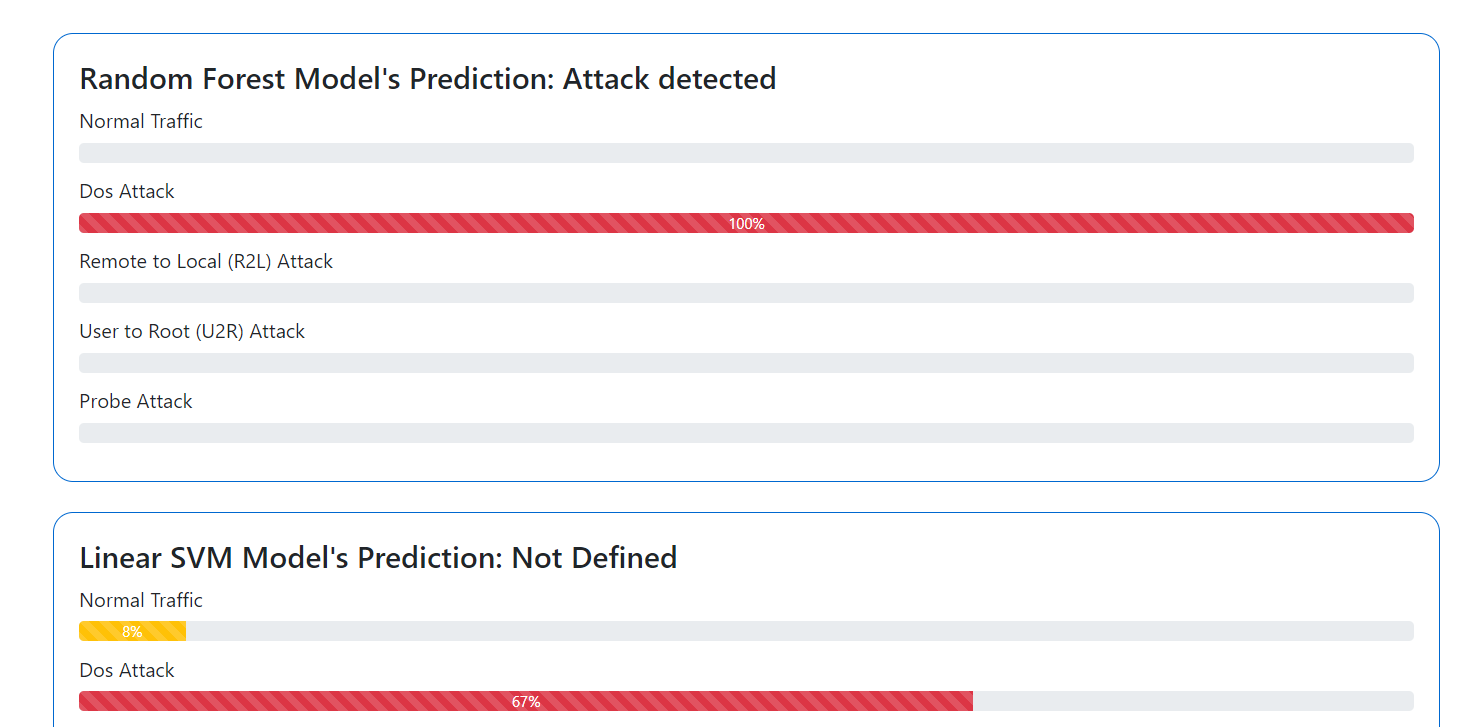
## 9.1 Dataset SCREEN









# CONCLUSION

As network intrusion continues to evolve, the pressure on network intrusion detection is also increasing. In particular, the problems caused by imbalanced network traffic make it difficult for intrusion detection systems to predict the distribution of malicious attacks, making cyberspace security face a considerable threat. This paper proposed a novel Difficult Set Sampling Technique, which enables the classification model to strengthen imbalanced network data learning. A targeted increase in the number of minority samples that need to be learned can reduce the imbalance of network traffic and strengthen the minority’s learning under challenging samples to improve the classification accuracy. We used six classical classification methods in machine learning and deep learning and combined them with other sampling techniques. Experiments show that our method can accurately determine the samples that need to be expanded in the imbalanced network traffic and improve the attack recognition more effectively. In the experiment, we found that deep learning performance is better than machine learning after sampling the imbalanced training set samples through the MLP algorithm. Although the neural networks strengthen data expression, the current public datasets have already extracted the data features in advance, which is more limited for deep learning to learn the preprocessed features and cannot take advantage of its automatic feature extraction. Therefore, in the next step, we plan to directly use the deep learning model for feature extraction and model training on the original network traffic data, performance the advantages of deep learning in feature extraction, reduce the impact of imbalanced data and achieve more accurate classification.

# 

# REFERENCES

D. E. Denning, ‘‘An intrusion-detection model,’’ IEEE Trans. Softw. Eng., vol. SE-13, no. 2, pp. 222–232, Feb. 1987.

[2] N. B. Amor, S. Benferhat, and Z. Elouedi, ‘‘Naive Bayes vs decision trees in intrusion detection systems,’’ in Proc. ACM Symp. Appl. Comput. (SAC), 2004, pp. 420–424.

[3] M. Panda and M. R. Patra, ‘‘Network intrusion detection using Naive Bayes,’’ Int. J. Comput. Sci. Netw. Secur., vol. 7, no. 12, pp. 258–263, 2007.

[4] M. A. M. Hasan, M. Nasser, B. Pal, and S. Ahmad, ‘‘Support vector machine and random forest modeling for intrusion detection system (IDS),’’ J. Intell. Learn. Syst. Appl., vol. 6, no. 1, pp. 45–52, 2014.

[5] N. Japkowicz, ‘‘The class imbalance problem: Significance and strategies,’’ in Proc. Int. Conf. Artif. Intell., vol. 56, 2000, pp. 111–117.

[6] Y. LeCun, Y. Bengio, and G. Hinton, ‘‘Deep learning,’’ Nature, vol. 521, no. 7553, pp. 436–444, 2015.

[7] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, ‘‘Deep learning for visual understanding: A review,’’ Neurocomputing, vol. 187, pp. 27–48, Apr. 2016.

[8] T. Young, D. Hazarika, S. Poria, and E. Cambria, ‘‘Recent trends in deep learning based natural language processing [review article],’’ IEEE Comput. Intell. Mag., vol. 13, no. 3, pp. 55–75, Aug. 2018.

[9] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, ‘‘A deep learning approach to network intrusion detection,’’ IEEE Trans. Emerg. Topics Comput. Intell., vol. 2, no. 1, pp. 41–50, Feb. 2018.

[10] D. A. Cieslak, N. V. Chawla, and A. Striegel, ‘‘Combating imbalance in network intrusion datasets,’’ in Proc. IEEE Int. Conf. Granular Comput., May 2006, pp. 732–737.