A MINI-PROJECT (2) REPORT ON

INTRUSION DETECTION SYSTEM USING MACHINE LEARNING TECHNIQUES

Submitted in partial fulfillment of requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

Submitted by:

K. VAMSI MOHAN REDDY	(19091A05H3)
A. SAI THANU SREE	(19091A05C7)
L. VASAVI	(19091A05H4)

Under the Guidance of
Dr. N. MADHUSUDHANA REDDY M. Tech, Ph.D.
Professor, Dept. of CSE

(ESTD-1995)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RAJEEV GANDHI MEMORIAL COLLEGE OF ENGINEERING & TECHNOLOGY
(AUTONOMOUS)
AFFILIATED TO J.N.T UNIVERSITY ANANTAPUR. ACCREDITED BY NBA (TIER-1) & NAAC OF UGC. NEW DELHI, WITH A+ GRADE
RECOGNIZED UGC-DDU KAUSHAL KENDRA
NANDYAL-518501, (Estd-1995)

YEAR: 2022-2023

Rajeev Gandhi Memorial College of Engineering &Technology
(AUTONOMOUS)
AFFILIATED TO J.N.T UNIVERSITY ANANTAPUR. ACCREDITED BY NBA (TIER-1) &
NAAC OF UGC. NEW DELHI, WITH A+ GRADE
RECOGNIZED UGC-DDU KAUSHAL KENDRA
NANDYAL-518501, (Estd-1995)

(ESTD – 1995)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that **K. VAMSI MOHAN REDDY** (*19091A05H3*), **A. SAI THANU SREE** (*19091A05C7*) and **L. VASAVI** (*19091A05H4*) of IV- B. Tech I- semester, have carried out the mini-project (2) work entitled "**INTRUSION DETECTION SYSTEM USING MACHINE LEARNING TECHNIQUES**" under the supervision and guidance of **Dr. N. MADHUSUDHANA REDDY,** Professor, CSE Department, in partial fulfillment of the requirements for the award of Degree of **Bachelor of Technology** in **Computer Science and Engineering** from **Rajeev Gandhi Memorial College of Engineering & Technology (Autonomous),** Nandyal is a bonafied record of the work done by them during 2022-2023.

Project Guide	**Head of the Department**
Dr. N. Madhusudhana Reddy **M.Tech, Ph.D.**	**Dr. K. Subba Reddy** **M.Tech, Ph.D.**
Professor, Dept. of CSE	**Professor, Dept. of CSE**

Place: Nandyal
Date:

External Examiner

Candidate's Declaration

We hereby declare that that the work done in this project entitled "**INTRUSION DETECTION SYSTEM USING MACHINE LEARNING TECHNIQUES**" submitted towards completion of mini-project (2) in IV Year I Semester of B. Tech (CSE) at the **Rajeev Gandhi Memorial College of Engineering & Technology**, Nandyal. It is an authentic record of our original work done under the esteemed guidance of **Dr. N. Madhusudhana Reddy**, Professor, Department of **COMPUTER SCIENCE AND ENGINEERING**, RGMCET, Nandyal.

We have not submitted the matter embodied in this report for the award of any other Degree in any other institutions.

By

K. Vamsi Mohan Reddy	(19091A05H3)
A. Sai Thanu Sree	(19091A05C7)
L. Vasavi	(19091A05H4)

Dept. of CSE,

RGMCET.

Place: Nandyal

Date:

ACKNOWLEDGEMENT

We manifest our heartier thankfulness pertaining to your contentment over our project guide **Dr. N. Madhusudhana Reddy** garu, Professor of Computer Science Engineering Department, with whose adroit concomitance the excellence has been exemplified in bringing out this project to work with artistry.

We express our gratitude to **Dr. K. Subba Reddy** garu, Head of the Department of Computer Science Engineering department, all teaching and non-teaching staff of the Computer Science Engineering Department of Rajeev Gandhi memorial College of Engineering and Technology for providing continuous encouragement and cooperation at various steps of our project.

Involuntarily, we are perspicuous to divulge our sincere gratefulness to our Principal, **Dr. T. Jaya Chandra Prasad** garu, who has been observed posing valiance in abundance towards our individuality to acknowledge our project work tangentially.

At the outset we thank our honourable Chairman **Dr. M. Santhi Ramudu** garu, for providing us with exceptional faculty and moral support throughout the course.

Finally, we extend our sincere thanks to all the **Staff Members** of CSE Department who have cooperated and encouraged us in making our project successful.

Whatever one does, whatever one achieves, the first credit goes to the **Parents** be it not for their love and affection, nothing would have been responsible. We see in every good that happens to us their love and blessings.

BY

K. Vamsi Mohan Reddy	(19091A05H3)
A. Sai Thanu Sree	(19091A05C7)
L. Vasavi	(19091A05H4)

ABSTRACT

The rapid development and massive usage of internet is increased as the number of people connecting to the network. The need to secure the network also increased. Intrusion Detection System (IDS) is a cyber security technique, monitors the state of software and hardware running in the network. An IDS is used to analyse, protect the system, and predict the behavior of the system.

Intrusion Detection System faces the challenges in improving the detection accuracy. The main aim of the project is to solve this problem using various Machine Learning algorithms which can predict the type of network attack. The dataset is used to predict intrusions contains variety of intrusions in the network. The intrusion detector system will not only predict the malicious activity, it also points out the type of network.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Introduction:

The goal of intrusion detection is to monitor the network attacks to detect anomalous behavior and misuse in network. Intrusion detection concept was introduced in early 1980's after the evolution of internet. There was a sudden rise in reputation and incorporation in security infrastructure. Since then, several events in IDS technology have advanced intrusion detection to its current state. James Anderson's wrote a paper for a government organization and imported an approach that audit trails contained important information that could be valuable in tracking misuse and understanding of user behavior.

The detection in audit data and its importance led to terrific improvements in the sub systems of every operating system. IDS and Host Based Intrusion Detection System (HIDS) were first defined in 1983, SRI International and Dorothy Denning began working on a government project that launched a new effort into intrusion detection system development. Around 1990's the revenues are generated, and intrusion detection market has been raised. After a year, Cisco recognized the priority for network intrusion detection and purchased the Wheel Group for attaining the security solutions. The government actions like Federal Intrusion Detection Networks (FID Net) were designed under Presidential Decision Directive 63 is also adding impulse to the IDS.

1.2 Objectives:

The main objective is to detect intrusions with suitable algorithms and to compare the accuracies of the machine learning and deep learning algorithms. The objective of work is to analyse various types of attacks on a network system and improve the detection accuracy rate with new type of dataset by machine learning algorithm and deep learning algorithm. Machine learning algorithms are used to solve known problems with familiar datasets. NSL-KDD dataset is used to build a network intrusion detector, a predictive model can detect such intrusions and distinguish between intrusions that are anomaly or normal connections using SMOTE (Synthetic Minority Oversampling Technique) combined with XGBOOST (Extreme Gradient Boosting) and LSTM (Long Short Term Memory) algorithms in the domain of machine learning and deep learning.

2. SYSTEM ANALYSIS

2.1 Existing system:

In real cyberspace, malicious cyber-attacks are resulting in a high imbalance of data. Cyber-attacks can hide in a large amount of normal traffic. The concept of Difficult Set Sampling Technique (DSSTE) algorithm is used to tackle the class imbalance problem in network traffic.

This method reduces the imbalance and makes the classification model learn difficult samples.

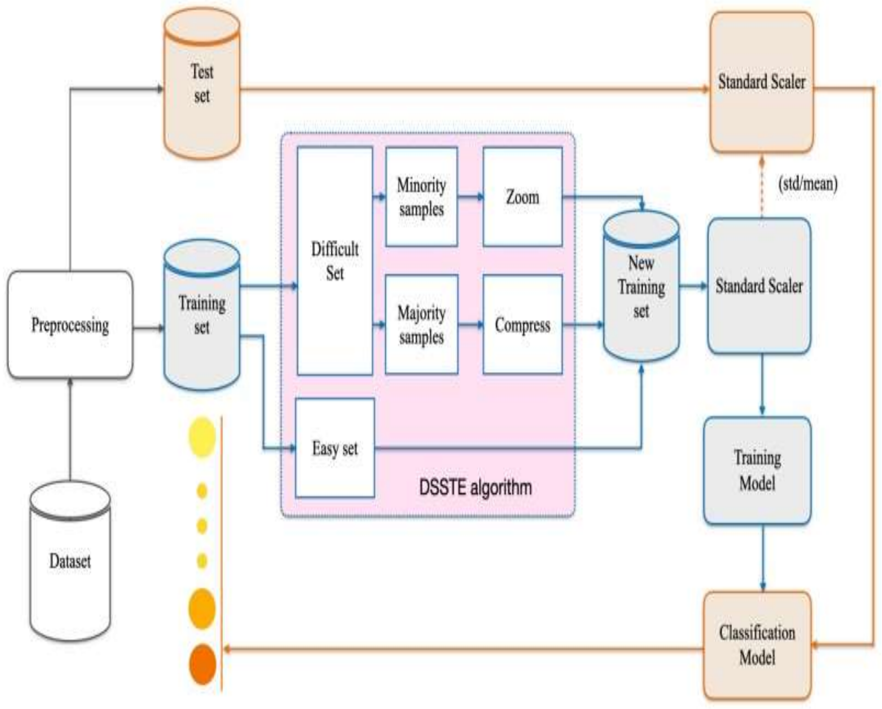


Figure.2.1 Framework of network intrusion detection existing model.

The Difficult Set Sampling Technique(DSSTE) algorithm is used to compress the majority samples and augment the number of minority samples in difficult samples, reducing imbalance in the training set that the intrusion detection system can achieve better classification accuracy.

The Existing intrusion detection is shown in Figure 2.1. Data pre-processing is first performed in the intrusion detection structure, including duplicate, outlier, and missing value processing. Then, partitioning the test set , training set and the training set processed for data balancing uses the proposed DSSTE algorithm. Before modelling, to increase the speed of the convergence, Standard Scaler is used to standardize the data and digitize the sample labels. Finally, the training set is used to train the classification model, and then the model is evaluated by the test set.

2.1.1 Disadvantages of Existing system:

- By using Difficult Set Sampling technique there is a chance of overfitting of data as like in random over sampling.
- This technique balances the data by replicating the minority samples.
- In this case dataset is prone to overfitting as same information is copied.

2.2 Motivation to the Problem:

The purpose of the project is to build an Intrusion Detection System to detect network intrusions using machine learning algorithms XG-Boost and LSTM and sampling technique used is SMOTE. The objective is to predict network attack with better accuracy and to find suitable algorithm to detect intrusions.

2.3 Proposed system:

It aims to classify the type of network attack using machine learning and deep learning algorithms. The dataset is Over Sampled using SMOTE(Synthetic Minority Over Sampling Technique) and the algorithms XG-Boost(Extreme Gradient Boosting) classifier , LSTM(Long Term Short Memory) are used for classification.

2.3.1 Advantages of Introduced System:

- High accuracy on detection rate.
- Comparison of various algorithms can be done easily.

3. FEASIBILITY STUDY

Feasibility study is an important phase in the software development process. Preliminary investigation examines project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Solution and Economical feasibility for adding new modules and debugging old running system.

Feasibility study should be performed based on various criteria and parameters. The various feasibility studies are:

- Technical Feasibility
- Solution Feasibility
- Economic Feasibility

3.1 Technical Feasibility:

To check the technical feasibilities. the technical requirements of the system Since machine learning algorithms are based on pure mathematics. There is very little requirement for any professional software. Most of the tools are open source. We can run this software in any software in any system without any software requirements which makes them highly portable.

3.2 Solution Feasibility:

To check the level of acceptance of the system by the user. This includes the process of providing solution efficiently. The user must not feel threatened by the system. The objective is to predict network attack with better accuracy and to find suitable algorithm to detect intrusions.

3.3 Economic Feasibility:

Since the project is Machine Learning based, the cost spent in executing this project would not demand cost for software's and related products, as most of the products are open source and free to us. Our project must be minimal cost.

4. SYSTEM REQUIREMENT SPECIFICATION

4.1 Requirement analysis

Requirements Analysis is the process of defining the expectations of the users for an application that is to be built or modified. It involves all the tasks that are conducted to identify the need of different stakeholders.

Requirement Analysis is a software engineering task that bridges the gap between system level software allocation and software design. Requirement Analysis is to specify software function and performance indicates software interface with other system elements.

Requirement Analysis of the system starts with the specification given by the user. This user-required specification could be formal or informal. In formal method, the user clearly states the purpose of the software. This acts as the good basis for the software engineers for the requirement analysis. In informal method purpose and the outputs are not clearly specified by the user. The responsibility is on the software engineer to get the purpose and outputs by interacting more the user.

4.2 Software Requirements Specification (SRS)

A software requirements specification, a requirements specification for a software system is complete description of the behavior of a system to be developed and may include a set of use cases that describe interactions the user will have with the software. In addition, it also contains non-functional requirements. Non-functional requirements impose constraints on the design or implementation.

The software requirements specification document enlists all necessary requirements that are required for the project development. To derive the requirements, we need to have clear and through understanding of the products to be developed. This is prepared after detailed communications with the project team and customer.

A Software Requirements Specification minimizes the time and effort required by developers to achieve desired goals and minimizes the development cost. A good SRS defines how an application will interact with system hardware, other programs, and human users in a wide variety of real-world situations.

4.3 Functional Requirements:

In software engineering a functional requirement defines as a function of system or its components where a function is described as a specification of behavior between output and input. It provides the following actions.

Technical Issues:

Many software projects have failed due to an incomplete or inaccurate analysis process, especially technical issues. It is a very key step in process.

Risk Analysis:

Project Risk Analysis is for the estimates of known accuracy and risk on capital investment projects. The main challenge is to determine how to model and visualize the complex relationships between normal data and data affected by attacks, define and monitor the risk impacts, analyse the probability of risk occurrence, mitigate the negative impact of risks, and monitor the course of the project with risks and uncertainties.

Performance Requirements:

The project has the following performance requirements

- The prime requirement is that no error condition causes a project to exit abruptly.
- Any error occurred in any process should return an understandable error message.
- The response should be fast and accurate, the analysis should not be confused at any point of time about action that is happening.

The system performance should be adequate.

4.4 Non-Functional Requirements:

Reliability:

The project is guaranteed to provide reliable results for every test dataset. The system shall operate with expected potency. The accuracy of chosen classifiers should be adequate and should be comparable with accuracy of other classifiers.

Usability:

The IDS developed can be used in many public and hybrid networks to detect intrusions that occur in them. The classifiers used should have good accuracy and provides results in reliable manner.

Scalability:

The need for scalability has been a driver for much of the technology innovations of the past few years. The system should perform with same accuracy even when performed on large test data sets. This adaption makes system scalable towards any kind of datasets.

Maintainability:

Maintainability is the ability to make changes to the product over time. The system should be maintainable to be consistent in performance. For example, it should be able to cope with another better classifiers in future to meet required performance and accuracy.

4.5 Software Requirements:

• Operating System	:	**Windows 10/11**
• IDE	:	**Google Colab**
• Programming Language	:	**PYTHON 3.X**
• Libraries	:	**NumPy, Pandas, matplotlib, seaborn, sklearn, SciPy**

4.6 Hardware requirements:

• Processor Type	:	**intel i3 or advanced**
• RAM	:	**4GB DD2 RAM or more**
• Hard disk	:	**500 GB**

5. SYSTEM DESIGN

Design is a meaningful engineering representation of something that is to be built. It is the most crucial phase in the developments of a system. Software design is a process through which the requirements are translated into a representation of software. The logical system design arrived at because of systems analysis is converted into physical system design.

5.1 Modules:

A module is defined as the unique and addressable components of the software which can be solved and modified independently without disturbing (or affecting in very small amount) other modules of the software. Thus, very software design should follow modularity. The process of breaking down a software into multiple independent modules where each module is developed separately is called Modularization.

5.1.1 Data Preprocessing:

The dataset is explored, the categorical attributes are converted into numeric attributes by Label Encoding. The class label is explored, the attack types with negligible values are replaced with the label as ‘others.

normal = 0, neptune = 1, others = 2

5.1.2 Sampling Technique:

SMOTE is an oversampling technique where the synthetic samples are generated for the minority class. This algorithm helps to overcome the overfitting problem posed by random oversampling.

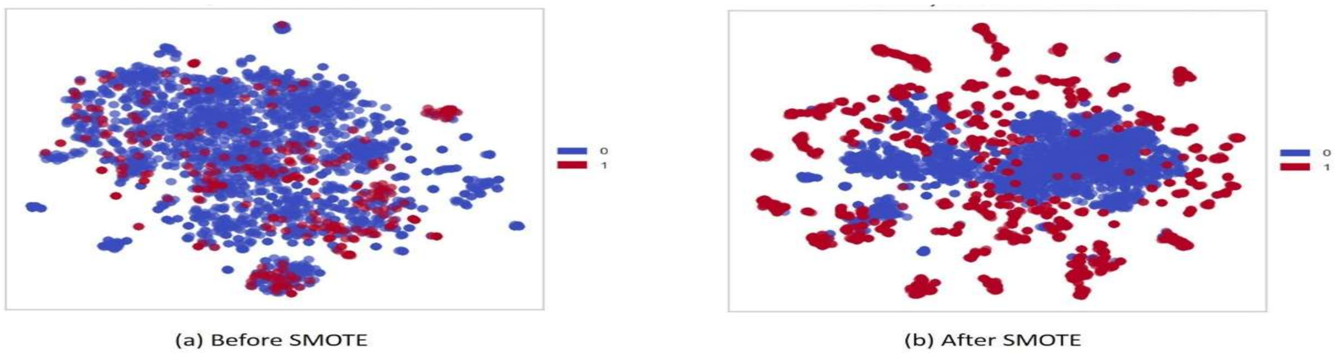


Fig 5.1: Synthetic samples of SMOTE

5.1.3 Modelling :

XG-Boost:

- XG-Boost is a scalable and highly accurate implementation of gradient boosting, being built largely for energizing machine learning model performance and computational speed.
- It improves the accuracy of intrusion detection; the detection method becomes more effective and more accurate.

LSTM:

- Long Short-Term Memory (LSTM) could remember values over arbitrary intervals. It is a suitable method to classify and predict known and unknown intrusions.
- It is designed to avoid the long-term dependency issue; LSTM can remember data for long periods.

5.2 System Architecture:

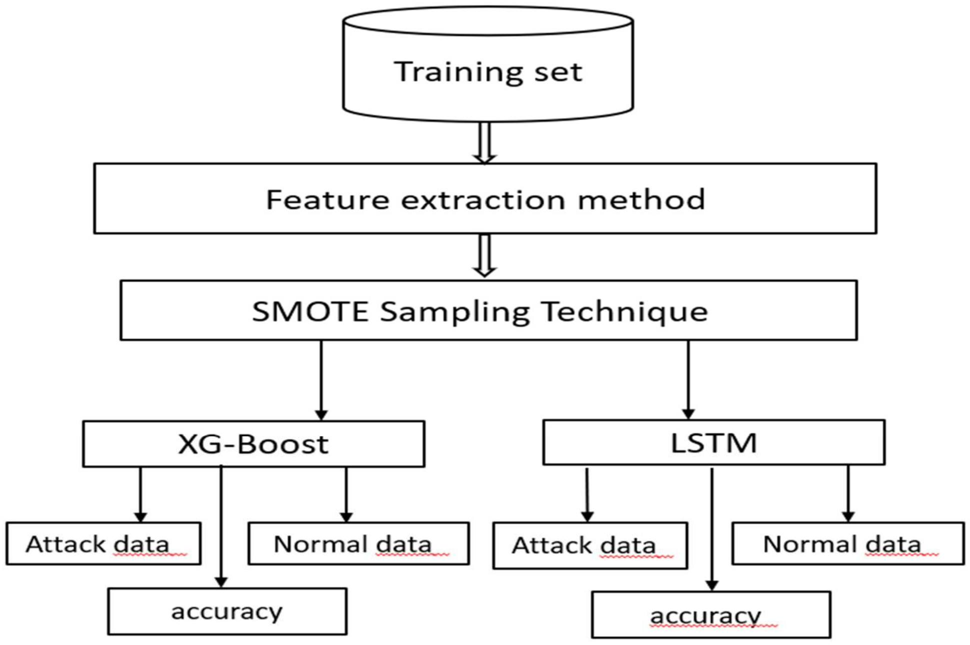


Fig 5.2: System Architecture

The intrusion detection model uses machine learning algorithms such as XGBOOST (Extreme Gradient Boosting) and LSTM (Long Short Term Memory). The performance of the classifier was improved by optimizing the algorithms for similarity and combining it with data imbalance processing techniques.

The architecture of the intrusion detection is as follows:

1. The analysis of the NSL-KDD dataset revealed the imbalance in the samples of the network attack type dataset. This imbalance resulted in higher false detection rates and lower accuracy for detecting smaller-scale samples. Therefore, this system proposes the combination sampling method by combining machine learning and deep learning algorithms with the SMOTE algorithm. This approach can reduce the number of outlier samples, enrich the attribute features of the minority samples, and increase the sampling number of the minority samples to build more balanced sample data of the network environment.

2. To further reduce the computational overhead time and increase the detection performance, it is necessary to convert the non-numerical features in the original dataset digitally and then convert the values to a specific range by normalization. This allowed dataset normalization and feature selection by information gain to filter out unnecessary features.

3. The classification model is trained by inserted the normalized processed dataset into the algorithms used.

4. A classification model with relatively good performance is obtained after training and the results are evaluated by conducting performance tests on the NSL-KDD dataset.

5.3 Introduction to UML:

A UML diagram is a diagram based on the UML (Unified Modelling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts, or classes, in order to better understand, alter, maintain, or document information about the system.

Each UML diagram is designed to let developers and customers view a software system from a different perspective and in varying degrees of abstraction. UML diagrams commonly created in visual modelling tools include.

5.3.1 Class Diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. It describes the attributes and operations of a class and also the constraints imposed on the system

5.3.2 Use case Diagram

A use case diagram at its simplest is a representation of user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other of diagrams as well. The use cases are represented by either circles or ellipses.

5.3.3 Activity Diagram

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join etc.

5.3.4 Sequence Diagram

Sequence diagrams in UML shows how object interact with each other and the order those interactions occur. It's important to note that show the interactions for a particular scenario. The processes are represented vertically, and interactions are show as arrows.

5.4 UML Diagrams:

The **figure 5.3** shows the use case diagram of the system. The system consists of mainly two actors or users interacts with it, they are Admin and Student. The use cases they involved and their interaction with the system are depicted in the below figure.

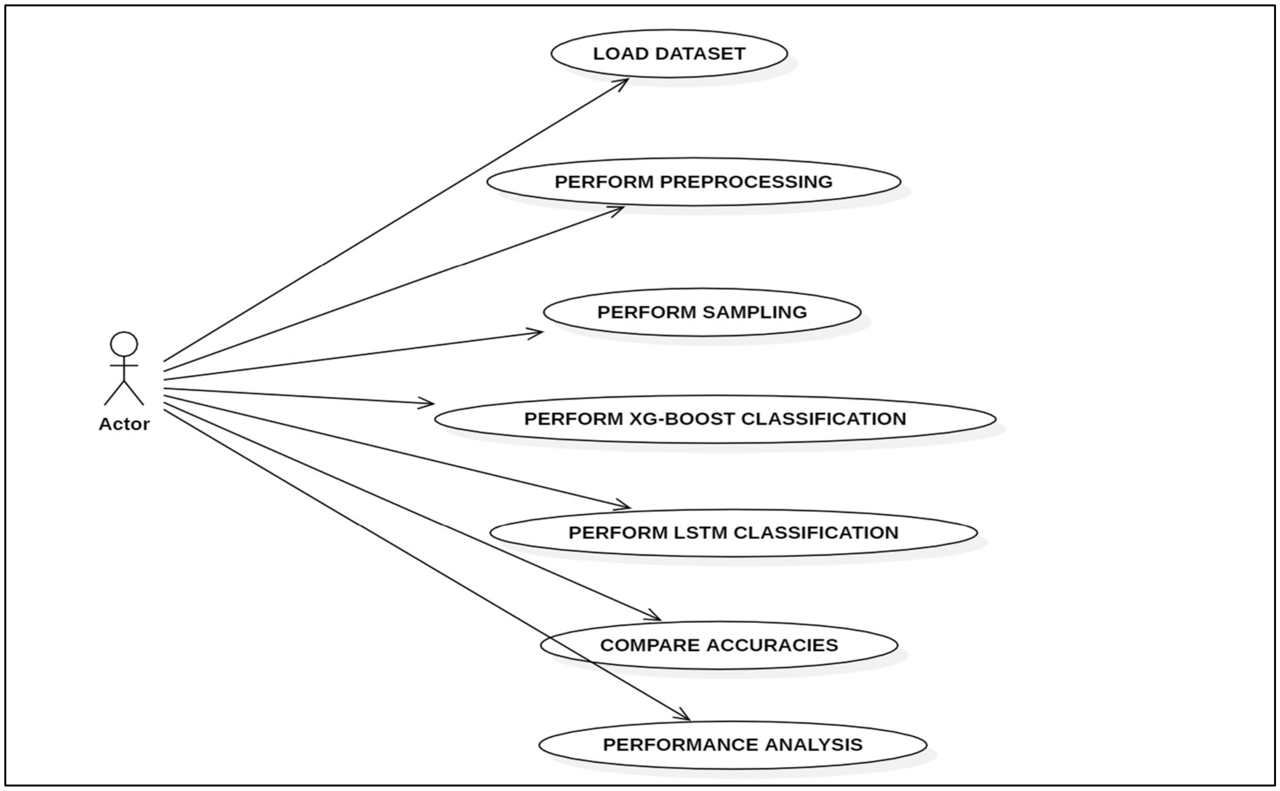


Fig 5.3: Use Case Diagram

The **figure 5.4** shows the class diagram of the system. It represents static view of application. The below figure describes attributes and operations of classes and the constraints imposed on the system.

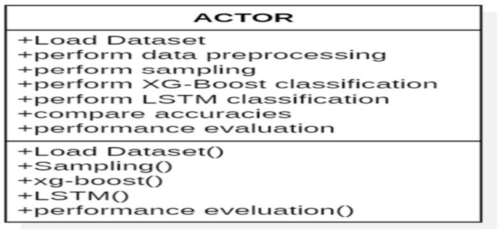


Fig 5.4: Class Diagram

The **figure 5.5** shows the sequence diagram for admin module of the system. The below figure depicts the processes involved and the sequences of messages exchanged between the processes needed to carry out the functionality in admin module.

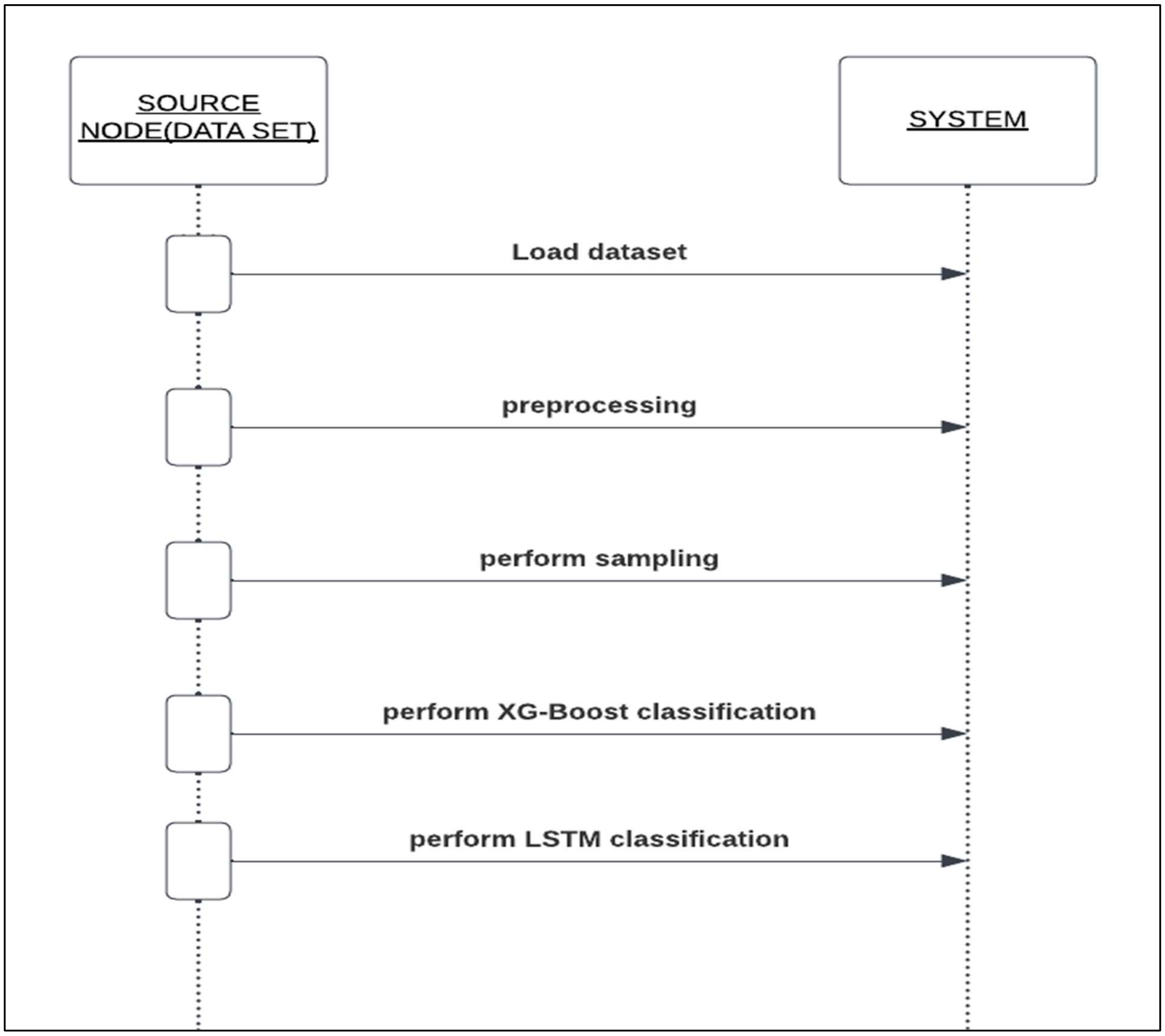


Fig 5.5: Sequence Diagram

The **figure 5.6** shows the activity diagram for admin module of the system. The below figure basically represents the flow from one activity to another activity in admin module. In the below figure each activity can be described as an operation of admin module of the system.

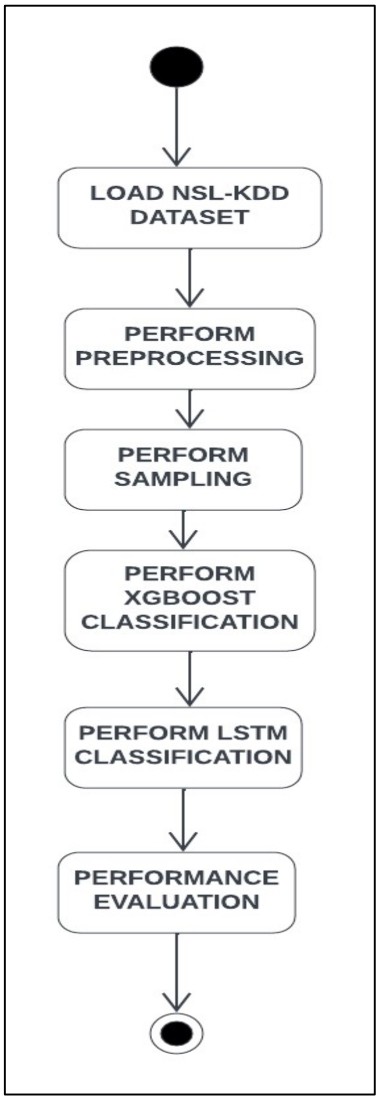


Fig 5.6: Activity Diagram

6. SYSTEM IMPLEMENTATION

6.1 Technologies

6.1.1 Python:

Python is a high-level programming language that is designed to be simple to read and use. It's free to use, even for commercial purposes, because it's open-source. Python is available for Mac, Windows, and Unix systems, as well as Java and .NET virtual machines.

Python, like Ruby or Perl, is a scripting language that is frequently used to create Web applications and dynamic Web content. Python is also supported by a variety of 2D and 3D imaging programs, allowing users to write custom plug-ins and extensions. GIMP, Inkscape, Blender, and Autodesk Maya are examples of applications that support a Python API. Python scripts (.PY files) can be parsed and executed right away. They can also be saved as compiled programs (.PYC files), which are commonly used as programming modules that other Python programs can reference.

Python Features

Python has a variety of useful features that distinguish it from other programming languages. It supports object-oriented programming, procedural programming, and memory allocation that is dynamic. A few essential features are listed below:

Easy to Learn and Use:

Python is a simple programming language to learn when compared to other programming languages. Its syntax is simple and like that of the English language. The semicolon and curly brackets are not used; instead, the indentation defines the code block. For beginners, it is the recommended programming language.

Expressive Language:

Python can perform complex tasks with just a few lines of code. For example, to run the hello world program, simply type print ("Hello World"). It will only require one line of code to run, whereas Java or C will require multiple lines.

Interpreted Language:

Python is an interpreted language, which means that each line of a Python program is executed separately. The benefit of being an interpreted language is that debugging is simple and portable.

Cross-platform Language:

Python can run on a variety of platforms, including Windows, Linux, UNIX, and Macintosh. As a result, we can say that Python is a portable programming language. It allows programmers to create software for multiple competing platforms by writing only one program.

Free and Open Source:

Python is a free programming language that anyone can use. On its official website, www.python.org, it is freely available. It has a large community all over the world working hard to create new Python modules and functions. The Python community welcomes contributions from anyone. "Anyone can download its source code without paying a penny," says open-source.

Object-Oriented Language:

Python supports object-oriented programming, which introduces the concepts of classes and objects. It allows for an inheritance, polymorphism, and encapsulation, among other things. The object-oriented method aids programmers in writing reusable code and developing applications with less code.

Extensible:

It means that other languages, such as C/C++, can be used to compile the code, allowing us to use it in our Python code. It converts the program to byte code, which can be run on any platform.

Large Standard Library:

It offers a diverse set of libraries for a variety of fields, including machine learning, web development, and scripting. Tensor flow, Pandas, NumPy, Keras, and Pytorch are just a few examples of machine learning libraries. Python web development frameworks include Django, Flask, and Pyramids.

GUI Programming Support:

For the development of a desktop application, a graphical user interface is used. The libraries used to develop the web application are PyQT5, Tkinter, and Kivy.

Integrated:

It's simple to integrate with languages like C, C++, and JAVA, among others. Python, like C, C++, and Java, executes code line by line. It makes it easy to debug the code.

Embeddable:

Other programming languages' code can be used in the Python source code. Python source code can also be used in other programming languages. It can embed other languages into our code.

Dynamic Memory Allocation:

We don't need to specify the variable's data type in Python. When we assign a value to a variable, the variable's memory is automatically allocated at run time. If the integer value 15 is assigned to x, we don't need to write int x = 15. Simply write x = 15 on a piece of paper.

6.1.2 Google Colab:

Google is quite aggressive in AI research. Over many years, Google developed AI framework called **TensorFlow** and a development tool called **Colaboratory**. Today TensorFlow is open-sourced and since 2017, Google made Colaboratory free for public use. Colaboratory is now known as Google Colab or simply **Colab**.

Another attractive feature that Google offers to the developers is the use of GPU. Colab supports GPU and it is totally free. The reasons for making it free for public could be to make its software a standard in the academics for teaching machine learning and data science. It may also have a long term perspective of building a customer base for Google Cloud APIs which are sold per-use basis.

Irrespective of the reasons, the introduction of Colab has eased the learning and development of machine learning applications. If you have used **Jupyter** notebook previously, you would quickly learn to use Google Colab. To be precise, Colab is a free Jupyter notebook environment that runs entirely in the cloud. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited by your team members - just the way you edit documents in Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your notebook.

Advantages of Google Colab:

There are the following advantages of Google Colab

1. **Sharing**:

   You can share your Google Colab notebooks very easily. Thanks to Google Colab everyone with a Google account can just copy the notebook on his own Google Drive account. No need to install any modules to run any code, modules come preinstalled within Google Colab.

2. **Versioning:**

   You can save your notebook to Github with just one simple click on a button. No need to write "git add git commit git push git pull" codes in your command client (this is if you did use versioning already)!

3. **Code Snippets:**

   Google Colab has a great collection of snippets you can just plug in on your code. For example, if you want to write data to a Google Sheet automatically, there's a snippet for it in the Google Library

4. **Forms for non-technical users**:

   Not only programmers have to analyze data and Python can be useful for almost everyone in an office job. The problem is non-technical people are scared to death of making even the tiniest change to the code.

   But Google Colab has the solution for that. Just insert the comment #@param {type: "string"} and you turn any variable field in a easy-to-use form input field. Your non-technical user needs to change form fields and Google Colab will automatically update the code.

5. **Performance and Price:**

Use the computing power of the Google servers instead of your own machine. Running python scripts often requires a lot of computing power and can take time. By running scripts in the cloud, you don't need to worry. Your local machine performance won't drop while executing your Python scripts. Best of all its freeware and cost-free to use

6.1.3 Model Information:

Machine Learning Classification

Classification is a process of categorizing a given set of data into classes; it can be performed on both structured and unstructured data. The process starts with predicting the class of given data points. The classes are often referred to as target, label, or categories.

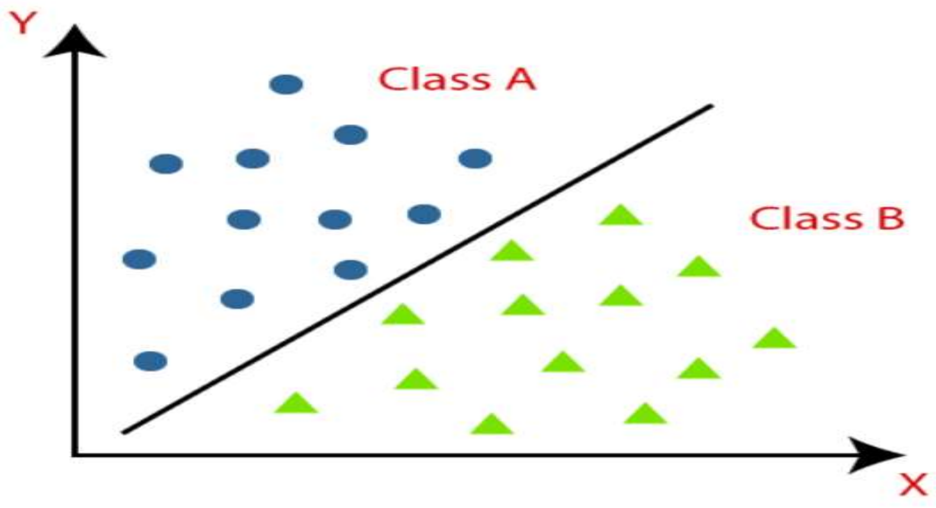


Fig 6.1 Machine Learning Classifier

The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations based on training data. In Classification, a program learns from the given dataset or observations and then classifies new observation into several classes or groups. Such as**, Yes or No, 0(Goodware) or 1(Malware),** etc. Classes can be called as targets/labels or categories.

SMOTE Techniques

The SMOTE algorithm

- SMOTE is an algorithm that performs data augmentation by creating synthetic data points based on the original data points.
- SMOTE can be seen as an advanced version of oversampling, or as a specific algorithm for data augmentation.
- The advantage of SMOTE is that you are not generating duplicates, but rather creating synthetic data points that are slightly different from the original data points.
- SMOTE is an improved alternative for oversampling.

The SMOTE algorithm works as follows:

- You draw a random sample from the minority class.
- For the observations in this sample, you will identify the k nearest neighbor.
- You will then take one of those neighbours and identify the vector between the current data point and the selected neighbor.
- You multiply the vector by a random number between 0 and 1.
- To obtain the synthetic data point, you add this to the current data point.
- This operation is very much like slightly moving the data point in the direction of its neighbor.
- This way, you make sure that your synthetic data point is not an exact copy of an existing data point while making sure that it is also not too different from the known observations in your minority class.

XGBoost Algorithm

XGBoost is one of the most popular machine learning algorithms these days. Regardless of the type of prediction task at hand; regression or classification.

XGBoost is well known to provide better solutions than other machine learning algorithms. In fact, since its inception, it has become the "state-of-the-art” machine learning algorithm to deal with structured data.

But what makes XGBoost so popular?

- **Speed and performance** :
  Originally written in C++, it is comparatively faster than other ensemble classifiers.

- **Core algorithm is parallelizable :**
  Because the core XGBoost algorithm is parallelizable it can harness the power of multi-core computers.
- It is also parallelizable onto GPU's and across networks of computers making it feasible to train on very large datasets as well.
- **Consistently outperforms other algorithm methods** :
  It has shown better performance on a variety of machine learning benchmark datasets.
- **Wide variety of tuning parameters** :
  XGBoost internally has parameters for cross-validation, regularization, user-defined objective functions, missing values, tree parameters, scikit-learn compatible API etc.

XGBoost (Extreme Gradient Boosting) belongs to a family of boosting algorithms and uses the gradient boosting (GBM) framework at its core.

It is an optimized distributed gradient boosting library.

LSTM Classifier

LSTM stands for Long-Short Term Memory. LSTM is a type of recurrent neural network but is better than traditional recurrent neural networks in terms of memory. Having a good hold over memorizing certain patterns LSTMs perform better.

As with every other NN, LSTM can have multiple hidden layers and as it passes through every layer, the relevant information is kept, and all the irrelevant information gets discarded in every single cell.

With an emerging field of deep learning, performing complex operations has become faster and easier. As you start exploring the field of deep learning, you are going to come across words like Neural networks, recurrent neural networks, LSTM, GRU, etc. This article explains LSTM and its use in Text Classification. So, what is LSTM? And how can it be used?

What is LSTM?

LSTM stands for Long-Short Term Memory. LSTM is a type of recurrent neural network but is better than traditional recurrent neural networks in terms of memory. Having a good hold over memorizing certain patterns

LSTMs perform better. As with every other NN, LSTM can have multiple hidden layers and as it passes through every layer, the relevant information is kept, and all the irrelevant information gets discarded in every single cell. How does it do the keeping and discarding you ask?

Why LSTM?

Tradition neural networks suffer from short-term memory. Also, a big drawback is the vanishing gradient problem. (While backpropagation the gradient becomes so small that it tends to 0 and such a neuron is of no use in further processing.) LSTMs efficiently improves performance by memorizing the relevant information that is important and finds the pattern.

LSTM for Text Classification

There are many classic classification algorithms like Decision trees, RFR, SVM, that can fairly do a good job, then why to use LSTM for classification?

One good reason to use LSTM is that it is effective in memorizing important information. If we look and other non-neural network classification techniques they are trained on multiple word as separate inputs that are just word having no actual meaning as a sentence, and while predicting the class it will give the output according to statistics and not according to meaning. That means, every single word is classified into one of the categories.

This is not the same in LSTM. In LSTM we can use a multiple word string to find out the class to which it belongs. This is very helpful while working with Natural language processing. If we use appropriate layers of embedding and encoding in LSTM, the model will be able to find out the actual meaning in input string and will give the most accurate output class.

6.1.4 Libraries:

NumPy

NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open-source project.

Operations using NumPy

Using NumPy, a developer can perform the following operations:

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

NumPy – A Replacement for MATLAB

NumPy is often used along with packages like SciPy (Scientific Python) and Mat−plotlib (plotting library). This combination is widely used as a replacement for MATLAB, a popular platform for technical computing. However, Python alternative to MATLAB is now seen as a more modern and complete programming language. It is open source, which is an added advantage of NumPy.

Pandas

Pandas is an open-source library that is made mainly for working with relational or labelled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas is fast and it has high performance & productivity for users.

Pandas were initially developed by Wes McKinney in 2008 while he was working at AQR Capital Management. He convinced the AQR to allow him to open source the Pandas. Another AQR employee, Chang She, joined as the second major contributor to the library in 2012. Over time many versions of pandas have been released. The latest version of the pandas is 1.4.1.

Advantages

- Fast and efficient for manipulating and analysing data.

- Data from different file objects can be loaded.
- Easy handling of missing data (represented as NaN) in floating point as well as non-floating-point data
- Size mutability: columns can be inserted and deleted from Data Frame and higher dimensional objects
- Data set merging and joining.
- Flexible reshaping and pivoting of data sets
- Provides time-series functionality.
- Powerful group by functionality for performing split-apply-combine operations on data sets.

Scikit learn

In Python, Scikit-learn (Sklearn) is the most usable and robust machine learning package. It uses a Python consistency interface to give a set of fast tools for machine learning and statistical modelling, such as classification, regression, clustering, and dimensionality reduction. NumPy, SciPy, and Matplotlib are the foundations of this package, which is mostly written in Python. SciKit Learn includes everything from dataset manipulation to processing metrics One of the best things about SciKit Learn are the built-in algorithms for Machine Learning which you can just try out with minimal adjustments Functions such as classification, regression, clustering, mode, model selection and others are generally built-in.

- **Train_Test_Split:**
  train_test_split is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data. With this function, you don't need to divide the dataset manually. By default, Sklearn train_test_split will make random partitions for the two subsets.

Matplotlib

- Matplotlib is an amazing visualization library in Python for 2D plots of arrays.
- Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.
- It was introduced by John Hunter in the year 2002.

- One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals.
- Matplotlib consists of several plots like line, bar, scatter, histogram etc.

Installation:

Windows, Linux and macOS distributions have matplotlib and most of its dependencies as wheel packages. Run the following command to install matplotlib package:

python -mpip install -U matplotlib

Basic plots in Matplotlib:

Matplotlib comes with a wide variety of plots. Plots helps to understand trends, patterns, and to make correlations.

They're typically instruments for reasoning about quantitative information.

Seaborn

- Seaborn is a Python visualization library for statistical plotting.
- It comes equipped with preset styles and color palettes so you can create complex, aesthetically pleasing charts with a few lines of code.
- It's designed to work with NumPy and pandas.) data structures and to support statistical tasks completed in SciPy and stats models.
- Seaborn is built on top of Python's core visualization library matplotlib, but it's meant to serve as a complement, not a replacement.
- In most cases, you'll still use matplotlib for simple plotting, and you'll need a knowledge of matplotlib to tweak Seaborn's default plots.

Seaborn tutorials

- **Data visualization with Seaborn** - This tutorial gives a quick overview of the code needed to create statistical data visualizations such as histograms, pairplots, and factor plots.
- **Data Visualization in Python**- Advanced Functionality in Seaborn - This tutorial covers some of Seaborn's most useful functions, such as conditional plotting and showing interactions between variables.
- **Visualizing Google Forms Data with Seaborn** - A great example of using Seaborn for business problems, this tutorial explores bar plots, time series graphs, and heatmaps, and cluster maps.
- **Learning Python**- This NBA dataset makes for a fun time learning how to make line, bar, box, and violin plots.

Seaborn features

- Visualizing the distribution of a dataset
- Plotting univariate and bivariate distributions
- Visualizing pairwise relationships
- Creating a plot with a regression line
- Fitting linear models
- Exploring interactions between multiple variables
- Plotting with categorical data
- Plotting wide-form data
- Making point plots, box plots, violin plots. , and categorical scatter plots

Evaluation Metrics

The performance of the proposed architecture is evaluated based on several statistical measures in addition to our new metric, defined as the corona score.

Accuracy

Accuracy is a metric that quantifies the competency of the method in defining the correct predicted cases

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

- True Positive: True Positive is equal to the number of correct predicted positive cases.
- False Positive: False Positive is equal to the number of incorrect predicted positive cases.
- True Negative: True Negative is equal to the number of correct predicted negative cases.
- False Negative: False Negative is equal to the number of incorrect predicted negative cases

Recall

Recall is the sensitivity of the method

$$\text{Recall} = \frac{TP}{TP + PN}$$

Precision

Precision is the ratio of the unnecessary positive case to the total number of positives.

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

Specificity

Specificity is the ratio of correct predicted negatives over negative observations.

$$\text{Specificity} = \frac{TN}{TN + FP}$$

F1-Score

F1-Score is the measure of the quality of detection.

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

6.2 Dataset Used

This system uses NSL-KDD as the benchmark dataset. It is the most classic dataset in the field of intrusion detection, each sample in the NSL-KDD includes 41 features. NSL-KDD is highly imbalanced dataset with normal traffic accounting for the vast majority, which conforms to traffic data distribution in the whole network world. The dataset was set up, then the training and testing datasets were checked for missing values and then concatenated to form a combined dataset.

Table 6.1 Dataset attribute description

Attribute	Description
1-9	Basic features of network connection
10-22	Content-related traffic features
23-31	Time-related traffic features
32-41	Host-based traffic features

Features of NSL-KDD

The NSL-KDD dataset was used, an improved version of the KDD dataset, as the initial KDD dataset had redundancy and classifiers tended to be biased. The 42nd column in both train and test had the binary labelling of each row as “anomaly” or “normal”. This feature was used as a target value to classify the data and plot the confusion matrix. The combined dataset was converted into the all-numeric data type, for the calculations to be done easily without any errors.

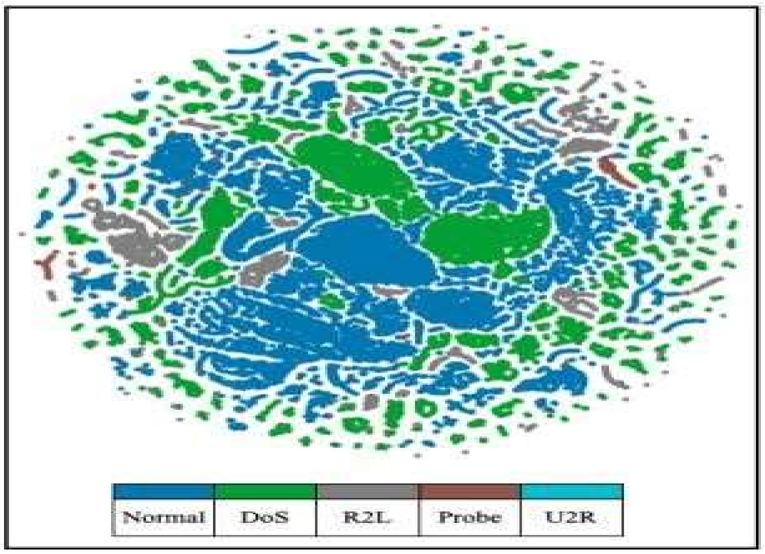


Fig 6.2 NSL-KDD dataset

6.2.1 Data Preparation

Data Pre-Processing:

Data pre-processing is the one of the important concepts to perform on each data before training the model.

Checking missing values:

The concept of missing values is important to understand to successfully manage data. If the missing values are not handled properly, then it may lead to drawing an inaccurate inference about the data. Due to improper handling, the result obtained will differ.

Finding the outliers:

It starts with the collection of data and that's when outliers first introduced to the population. The outliers will not be known at the collection phase, they can be the result of a mistake during data collection, or it can be just an indication of variance in the data.

Normalize the data:

Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values or losing information.

Splitting the Data:

Splitting the data into train and test is based on all random images. Train data is sent to the model and the test data is used for the model performance. Always train data must be high because it will be trained.

Model Applying:

Classification models are a subset of supervised machine learning. A classification model reads some input and generates an output that classifies the input into some category. For example, a model might read an email and classify it as either spam or not - binary classification.

Model Performance

To calculate the model performance, the type 1 and type 2 errors are checked with TP, FP, FN, TN to create confusion matrix and calculating the accuracy, precision, recall and sensitivity.

Predictions:

When the data is passed into the saved model, model must extract the features of the data and classify the predictions.

6.2.2 SMOTE Technique for Oversampling

Oversampling is a method that involves randomly duplicating examples from the minority class in the training dataset. This means that the data contains an attack column with different classes such as Normal, Neptune and others. When compared to the normal class, the remaining classes are less, so when the data is sent into the model, it will fall into the normal class and this problem can be avoided using an oversampling technique like SMOTE(Synthetic Minority Oversampling Technique).

SMOTE first selects a minority class instance at random and finds its k-nearest minority class neighbours. The synthetic instance is then created by choosing one of the k-nearest neighbours at random and connecting a and b to form a line segment in the feature space. The synthetic instances are generated as a convex combination of the two chosen instances a and b .

One of the most used oversampling methods to solve the imbalance problem is SMOTE (synthetic minority oversampling technique). It tries to balance class distribution by replicating minority class examples at random. SMOTE creates new minority instances by combining existing minority instances.

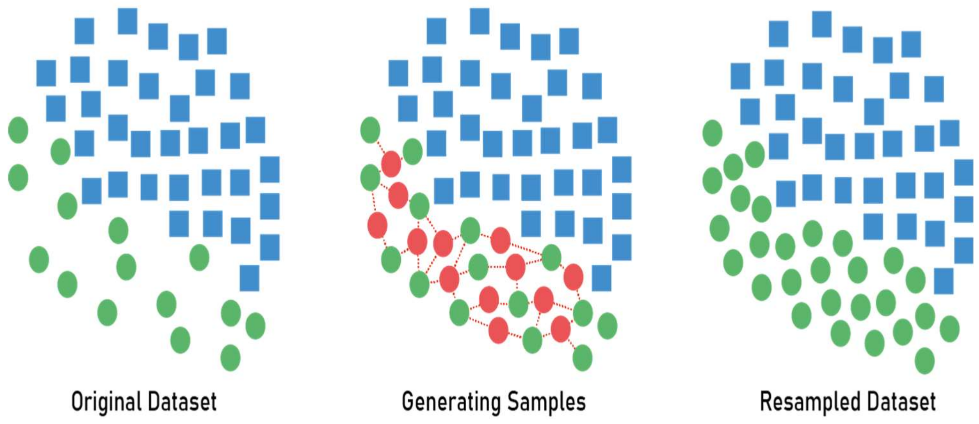


Fig 6.3 Synthetic Minority Oversampling Technique

SMOTE Technique:

- Choose a random sample from the minority class.
- For the observations in this sample, identify the k nearest neighbours.
- Take one of those neighbours and identify the vector between the current data point and the selected neighbour.
- Multiply the vector by a random number between 0 and 1.
- To obtain the synthetic data point, add this to the current data point.

- This operation is very much like, slightly moving the data point in the direction of its neighbour.
- This way, synthetic data point is not an exact copy of an existing data point while making sure that it is also not too different from the known observations in minority class.

6.2.3 XGBoost Algorithm

XGBoost is one of the most popular machine learning algorithms these days. Regardless of the type of prediction task at hand; regression or classification. XGBoost is well known to provide better solutions than other machine learning algorithms. XGBoost has become the "state-of-theart" machine learning algorithm to deal with structured data.

The objective function comprises training loss and regularization.

$$obj(\theta) = TL(\theta) + R(\theta)$$

where obj is objective function

TL is training loss

R is regularization function

θ denotes an epoch

The regularization term is represented by R and TL is for Training Loss. The TL is just a measure of how predictive the model is. Regularization helps to keep the model's complexity within desired limits, eliminating problems such as over-stacking or overfitting of data which can lead to a less accurate model. XGBoost simply adds the prediction of all trees formed from the dataset and then optimizes the result.

XGBoost has many parameters and one can use these parameters to perform specific tasks. The following are some of the parameters that are used to get the results. The "learning rate" (also referred to as "eta") parameter is basically set to get rid of overfitting problems. It performs the step size shrinkage and weights relating to the new features can be easily extracted (set as 0.1).

The "max_depth" parameter is used to define how deep a tree runs: the bigger the value, the more complex the model becomes (set as 3). The "n_estimators" parameter refers to the number of rounds or trees used in the model (set as 100). The "random_state" parameter is also a learning parameter. It is also sometimes referred to as "seed" (set as 7). The "n_splits" parameter is used to split up the dataset into k parts (set as 10).

If parameters are not set, XGBoost picks up the default values, though one can define the parameters as per the desired model.

The XGBoost method is used for both classification and regression problems. There is a categorization issue with the dataset and must categorise attack levels like Normal, Neptune and others. By combining the predictions of several weaker, simpler models, the XGBoost algorithm seeks to accurately forecast a target variable by comparing to other classification methods. This is how XGBoost comes up with the results and out of the three results, the best results were for recall, followed by accuracy and precision. As the number of splits increases, there is a decrease in the values because more splits allow the model to learn relations very specific to a dataset.

6.2.4 LSTM Algorithm

LSTM stands for Long-Short Term Memory. LSTM is a type of recurrent neural network but is better than traditional recurrent neural networks in terms of memory. Having a good hold over memorizing certain patterns LSTMs perform better. As with every other neural network, LSTM can have multiple hidden layers and as it passes through every layer, the relevant information is kept, and all the irrelevant information gets discarded in every single cell. Traditional neural networks suffer from short-term memory. Also, a big drawback is the vanishing gradient problem. LSTM efficiently improves performance by memorizing the relevant information that is important.

LSTM for Text Classification

There are many classic classification algorithms like Decision trees, RFR, SVM, that can fairly do a good job, then one good reason to use LSTM is that it is effective in memorizing important information. Other non-neural network classification techniques are trained on multiple word as separate inputs that are just word having no actual meaning as a sentence and while predicting the class it will give the output according to statistics and not according to meaning.

That means, every single word is classified into categories.

In LSTM ,multiple word strings can be used to find out the class to which it belongs. This is very helpful while working with Natural language processing. If appropriate layers of embedding and encoding are used in

LSTM, the model will be able to find out the actual meaning in input string and will give the most accurate output class.

In comparison to traditional algorithms, LSTM is also used for classification and regression problems. Neural networks are more accurate. It is essentially just a standard neural network that takes a hidden state from the previous step as input in addition to the input from the present step.

As a result, a Neural Network can be used for classification.

The principal component of LSTM is the cell state. To add or remove information from the cell state, the gates are used to protect it, using sigmoid function (one means allows the modification, while a value of zero means denies the modification.).

Three different gates are identified :

• **Forget gate layer** :

Looks at the input data, and the data received from the previously hidden layer, then decides which information LSTM is going to delete from the cell state, using a sigmoid function (One means keeps it, 0 means delete it). It is calculated as:

$f_t = \sigma (W_f .[h_{t-1}, x_t] + b_f)$ where ft = output of forget gate

Wf = weight matrix

ht-1 = hidden state of step t-1

xt = current input

bf = bias value

• **Input/Update gate layer** :

Decides which information LSTM is going to store in the cell state. At first, input gate layer decides which information will be updated using a sigmoid function, then a Tanh layer proposes a new vector to add to the cell state. Then the LSTM updates the cell state, by forgetting the information that is decided to forget and updating it with the new vector values.

It is calculated as:

$$it = \sigma\,(Wi.[ht-1, xt] + bi) \text{ and } \tilde{C}t = \tanh(Wc.[ht-1, xt] + bC)$$

where it decides which values we will update and $\tilde{C}$ t are the new candidate values that could be added to the cell state. bi and bC are bias values and Wi is weight matrix.

• **Output Layer** :

It decides what the output will be by executing a sigmoid function ,that decides which part of the cell LSTM is going to output, the result is passed through a Tanh layer (value between -1 and 1) to output, only the information that was decided to pass to the next neuron. It is calculated as:

$$Ot = \sigma\,(Wo[ht-1, xt] + bo) \text{ and } ht = ot * \tanh(Ct)$$

where Ot and ht are output and hidden states, Wo is weight matrix and Ct denotes cell state.

Gathered binary classification results and multi-classification (3- class) results. In both groups of classification, PCA and Mutual information are used for dimensionality reduction and applied LSTM as a classification algorithm.

Implementation of this model is done using Google Colab, using Keras library (Which is an opensource neural network library written in Python). In Google Colab, pre-processed Dataset is split into 60% for training Data, 20% for validation and 20% for testing purposes. On the other hand, LSTM model is configured with the parameters. To determine the performance of the proposed algorithm, a confusion matrix is plotted. The matrix in Figure 6.3 represents what a confusion matrix is made up of.

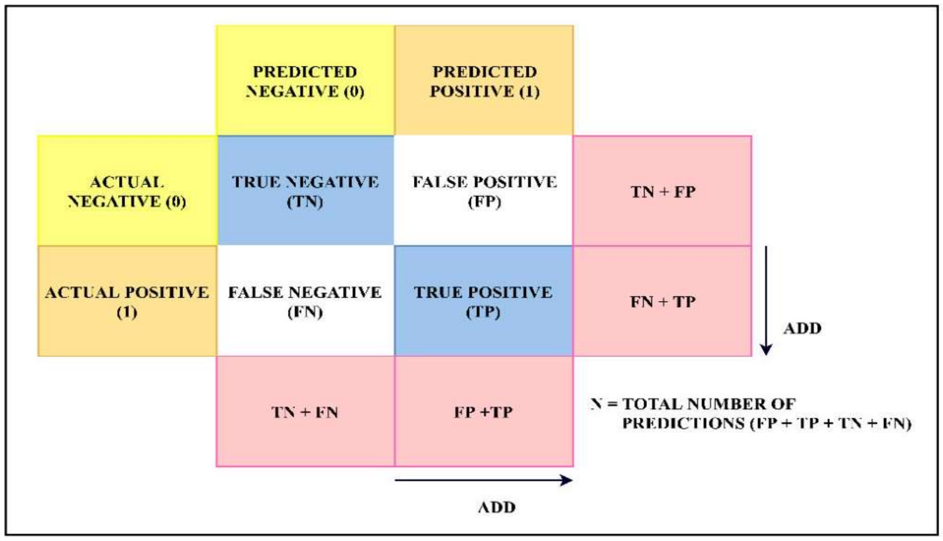


Fig 6.4 Confusion Matrix

The Confusion Matrix represents the quality of the output of a classifier on any dataset. The diagonal elements (blue box in fig 6.4) represent correct or true labels, whereas the off diagonal (white box in fig 6.4) elements represent the elements misclassified by the classification model. Therefore, the higher the values of the diagonal elements of the confusion matrix, the better and more accurate the classification model becomes. Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class. Therefore, confusion matrix allows to calculate the true positives, true negatives, false positives and false negatives such as: True positive (TP) is when the model classifies an attack as an attack. True negative (TN) is when the model classifies a normal entry as normal. False positive (FP) is when the model classifies a normal entry as an attack. False negative (FN) is when the model classifies an attack as a normal entry. Usually, intrusion detection systems try to reduce the false positive and false negative rate, knowing that the latter (FN) will have severe consequences on information systems. Multiple metrics are considered , namely: Accuracy, Sensitivity (or Recall), precision and F1 Score. These metrics are calculated as:

Accuracy = (TP + TN)/All Predictions

Sensitivity(Recall) = TP/(FN + TP)

Precision = TP/(TP + FP)

F1Score = 2 $*$ (Precision $*$ Sensitivity)/(Precision + Sensitivity)

It is observed that LSTM-PCA provided the best results, especially using 02 components. This may be due to PCA had low noise sensitivity, remove correlated features and smaller dimensions help the classifier to learn easily. It can also notify that Mutual information is better with 4 features than 10 features and LSTM performs well in the binary classification, but for multi-classification is much less efficient, it may be due to the data set is quite noisy.

The LSTM model is capable to learn successfully the features extracted from the dataset in the training period. This capability allows the models to effectively distinguish the normal traffic from the network attacks. To implement the proposed architecture, Keras library and TensorFlow are used on Google Colab platform. Principal Component Analysis and Mutual Information are applied as dimensionality reduction algorithms. The intrusion detection is based on binary and multiclass classification. For binary and multiclass classification, the accuracy is 83.68%. The accuracy and sensitivity are compared with previous approach, which also demonstrates the efficiency of the proposed method.

After inserting the features into the model, it will learn data patterns to classify the attack. Then evaluate model performance using evaluation metrics such as confusion matrix, which visualises and summarises the performance of a classification algorithm and will give class wise performance, as well as classification report, which gives accuracy, precession, recall, F1Score.

Flow of Implementation:

Step 1: Importing necessary libraries and installing the modules like XGBoost algorithm.

Step 2: Mention the columns in train and test data by checking data information, display numerical and categorical columns and check unique values in categorical columns by visualizing the classes using count plot.

Step 3: The Target column is attack. Check the unique values and plot the columns using count plot.

Step 4: Plot a pie plot for attack column, create a function for the attack column's classes that will return 0 if normal is present, 1 if Neptune is present, and 2 if other classes are present.

Step 5: Again, plot a pie plot for attack column and check its percentage of distribution. Convert the categorical columns to numerical columns to make predictive models and models require all input and output variables to be numeric.

Step 6: Plot a bar plot of attack column for value counts, the data features are taken into the X variable, and the target is taken into the Y variable.

Step 7: Use an oversampling technique SMOTE. If this model gets trained, it will overfit. In order to avoid this, employ the SMOTE technique.

Step 8: After using SMOTE, send the data to the XGBoost classifier to train. Once the model is trained, plot a confusion matrix for model performance and then print a classification report.

Step 9: Use the LSTM algorithm for training, checking model loss, plotting the confusion matrix for model performance, and printing the classification report.

Step 10: In the final step, Compare the accuracies of XGBoost and LSTM models.

6.3 Sample code:

Data Pre-processing:

```
import os
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from imblearn.over_sampling import SMOTE
train = pd.read_csv('/content/drive/MyDrive/IDS/KDDTrain+.txt',delimiter=',')
test = pd.read_csv('/content/drive/MyDrive/IDS/KDDTest+.txt',delimiter=',')
```

```
display(train.head(4))

display(test.head(4))

columns = (['duration','protocol_type','service','flag','src_bytes','dst_bytes','la
nd','wrong_fragment','urgent','hot','num_failed_logins','logged_in','num_compromise
d','root_shell','su_attempted','num_root','num_file_creations','num_shells','num_ac
cess_files','num_outbound_cmds','is_host_login','is_guest_login','count','srv_count
','serror_rate','srv_serror_rate','rerror_rate','srv_rerror_rate','same_srv_rate','
diff_srv_rate','srv_diff_host_rate','dst_host_count','dst_host_srv_count','dst_host
_same_srv_rate','dst_host_diff_srv_rate','dst_host_same_src_port_rate,'dst_host_srv
_diff_host_rate','dst_host_serror_rate','dst_host_srv_serror_rate','dst_host_rerror
_rate','dst_host_srv_rerror_rate','attack','level'])

train.columns = columns

train.head()

test.columns = columns

test.head()

train.info()

#Numerical Columns

train.drop(['protocol_type', 'service', 'flag', 'attack'],axis=1)

# Categorical columns

train[['protocol_type', 'service', 'flag', 'attack']]

train['protocol_type'].unique()

train['service'].unique()

plt.figure(figsize=(7,5))

sns.countplot(data = train , x=train['protocol_type'])

plt.show()

protocol_type = {}
```

```
li = list(train['protocol_type'].unique())

for i in range(len(train['protocol_type'].unique())):

    protocol_type[li[i]]=i

protocol_type

plt.figure(figsize=(7,15))

sns.countplot(data = train , y=train['service'])

plt.show()

service_type = {}

li = list(train['service'].unique())

for i in range(len(train['service'].unique())):

    service_type[li[i]]=i

service_type

plt.figure(figsize=(15,7))

sns.countplot(data = train , x=train['flag'])

plt.show()

flag_type = {}

li = list(train['flag'].unique())

for i in range(len(train['flag'].unique())):

    flag_type[li[i]]=i

flag_type

train['attack'].unique()

train['attack'].value_counts().plot.bar(figsize=(15,5))

train['attack'].value_counts().plot(kind = 'pie',figsize=(10,10),autopct= "%0.2f")

#Replacing all attacks with 'other' except normal and neptune

def attack_type(k):
    if k=='normal':
        return 0
    elif k=='neptune':
        return 1
    else:
        return 2
train1=train
test1=test
train1['attack']=train1['attack'].apply(attack_type)
train1['attack'].value_counts().plot(kind = 'pie',figsize=(10,10),autopct= "%0.2f")
train1=train
test1=test
#LABEL ENCODING
test1['protocol_type'] = test1['protocol_type'].map(protocol_type)
test1['service']=test1['service'].map(service_type)
test1['flag']=test1['flag'].map(flag_type)
test1['attack']=test1['attack'].apply(attack_type)
train1['attack'].value_counts().plot.bar(figsize=(7,5))
X_train = train1.drop(['attack'],axis=1)
y_train = train1['attack']
X_test = test1.drop(['attack'],axis=1)
y_test = test1['attack']
y_test
```

SMOTE Sampling Technique:

```
from sklearn.preprocessing import LabelEncoder, MinMaxScaler

scaler = MinMaxScaler()

le = LabelEncoder()

labels_dict = {'normal':0,'neptune':1,'others':2}

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)

print("Instances per label in training set\n", y_train.value_counts())

print(X_train.shape)

print(X_test.shape)

labels_dict = {'normal':0,'neptune':1,'others':2}

print(labels_dict)

# training data sampling

from imblearn.over_sampling import SMOTE

from imblearn.pipeline import Pipeline

from collections import Counter

over = SMOTE(sampling_strategy='not majority', n_jobs=3)

steps = [('o', over)]

pipeline = Pipeline(steps=steps)

X_train, y_train = pipeline.fit_resample(X_train, y_train)

counter = Counter(y_train)

print(counter)
```

XG Boost Classifier:

```
from xgboost import XGBClassifier

xg=XGBClassifier()
```

```
#fitting the model

xg.fit(X_train,y_train)

y_pred=xg.predict(X_test)

xg_acc=accuracy_score(y_test,y_pred)

print(f"Accuracy of xgboost classifier: \n {xg_acc}")

#CLASSIFICATION REPORT

from sklearn.metrics import classification_report

print(classification_report(y_test,y_pred))

#CONFUSION MATRIX

from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

cm=confusion_matrix(y_test,y_pred)

print(cm)

fig,ax=plt.subplots(figsize=(7,7))

sns.heatmap(cm,annot=True,cmap='GnBu',fmt="0.1f").plot(ax=ax)

plt.show()
```

LSTM Classifier:

```
#reshape input data to LSTM format i.e., [samples,time_steps,features]

X_train_lstm=X_train.reshape(X_train.shape[0],1,X_train.shape[1])

X_test_lstm=X_test.reshape(X_test.shape[0],1,X_test.shape[1])

print(f"Shape of X_train:",X_train_lstm.shape)
print(f"Shape of X_test:",X_test_lstm.shape)
from tensorflow.keras import Model,Sequential,Input
from tensorflow.keras.layers import LSTM,Dense,Dropout
print(f"num of classes:{3}")
n_classes=3
n_features=X_train_lstm.shape[2]
def multiClassModel(n_features, n_classes):
    model = Sequential()
    model.add(Input(shape=(None, n_features)))
    model.add(LSTM(units=30))
    model.add(Dropout(0.2))
model.add(Dense(n_classes, activation="softmax", name="softmax"))
    model.compile(loss="sparse_categorical_crossentropy", optimizer='Adam')
    model.summary()
    return model
# predicting on training set
y_train_pred_prob = model.predict(X_train_lstm)
y_test_pred_prob = model.predict(X_test_lstm)
y_train_pred = np.argmax(y_train_pred_prob, axis=1)
y_test_pred = np.argmax(y_test_pred_prob, axis=1)
#CLASSIFICATION REPORT
print(classification_report(y_test, y_test_pred))
lstm_acc= accuracy_score(y_test, y_test_pred)
print(f"Accuracy of LSTM: \n {lstm_acc}")
#CONFUSION MATRIX
```

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
cm = confusion_matrix(y_test, y_test_pred)
print(cm)
fig, ax = plt.subplots(figsize=(7, 7))
sns.heatmap(cm, annot=True,cmap='GnBu',fmt='0.1f').plot(ax=ax)
plt.show()
```

Accuracy Comparison of Models:

```
x =models = ['XGBoost classifier','LSTM']
y = accuracy = [xg_acc*100 , lstm_acc*100]
width = 0.4
plt.figure(figsize=(15,10))
fig, ax = plt.subplots(figsize=(15,7))
pps = ax.bar(x, y, width, align='center')
for p in pps:
   height = p.get_height()
   ax.text(x=p.get_x() + p.get_width() / 2, y=height+1,
      s="{}%".format(height),
      ha='center')
plt.title('Accuracies of both the models')
plt.show()
```

6.4 Results:

All experiments were conducted on Google Colab, data is a cleaned data, so no pre-processing steps are required. splitting the data in 80-20 ratio and apply SMOTE techniques, XGBoost and LSTM Algorithms.

The figure 6.5 depicts the count of protocol_type in NSL-KDD dataset. The dataset consists of 3 distinct types of protocols udp, tcp and icmp.

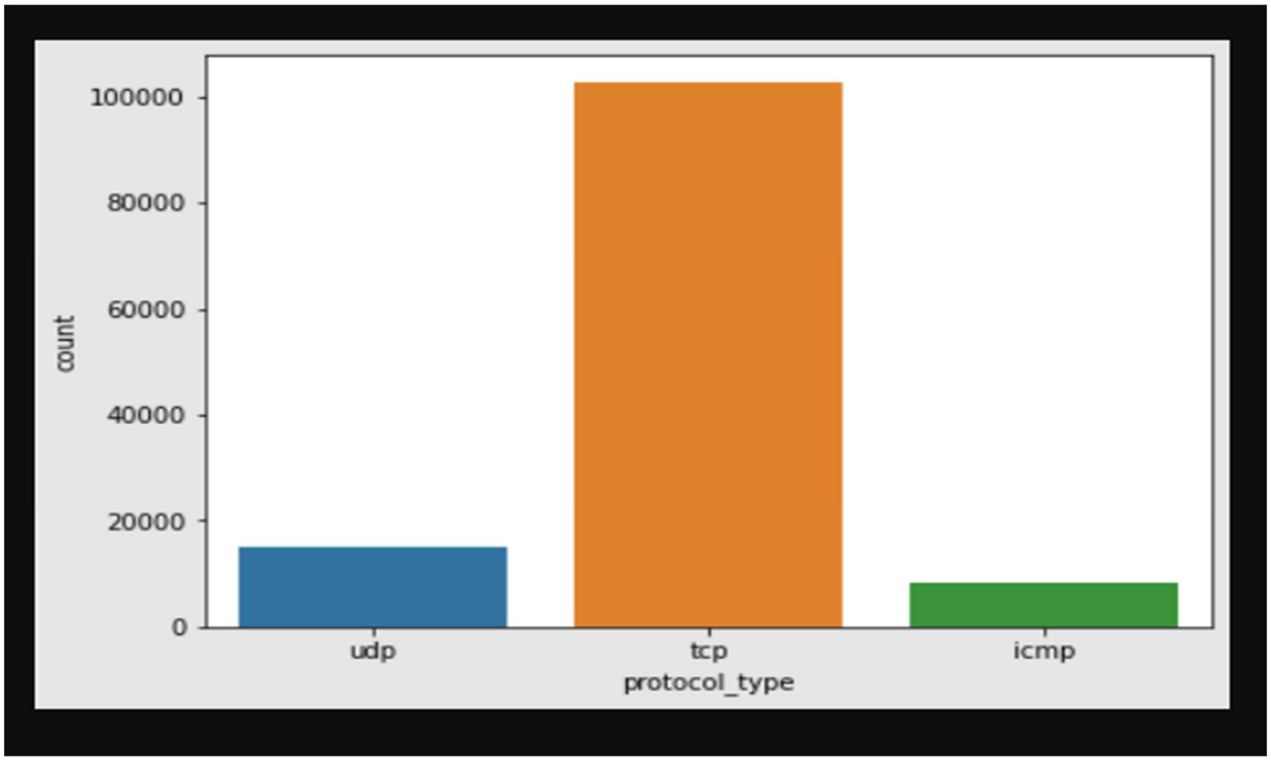


Fig.6.5 protocol_type plot

The figure 6.6 depicts about the count of service types and their names in NSL-KDD dataset.

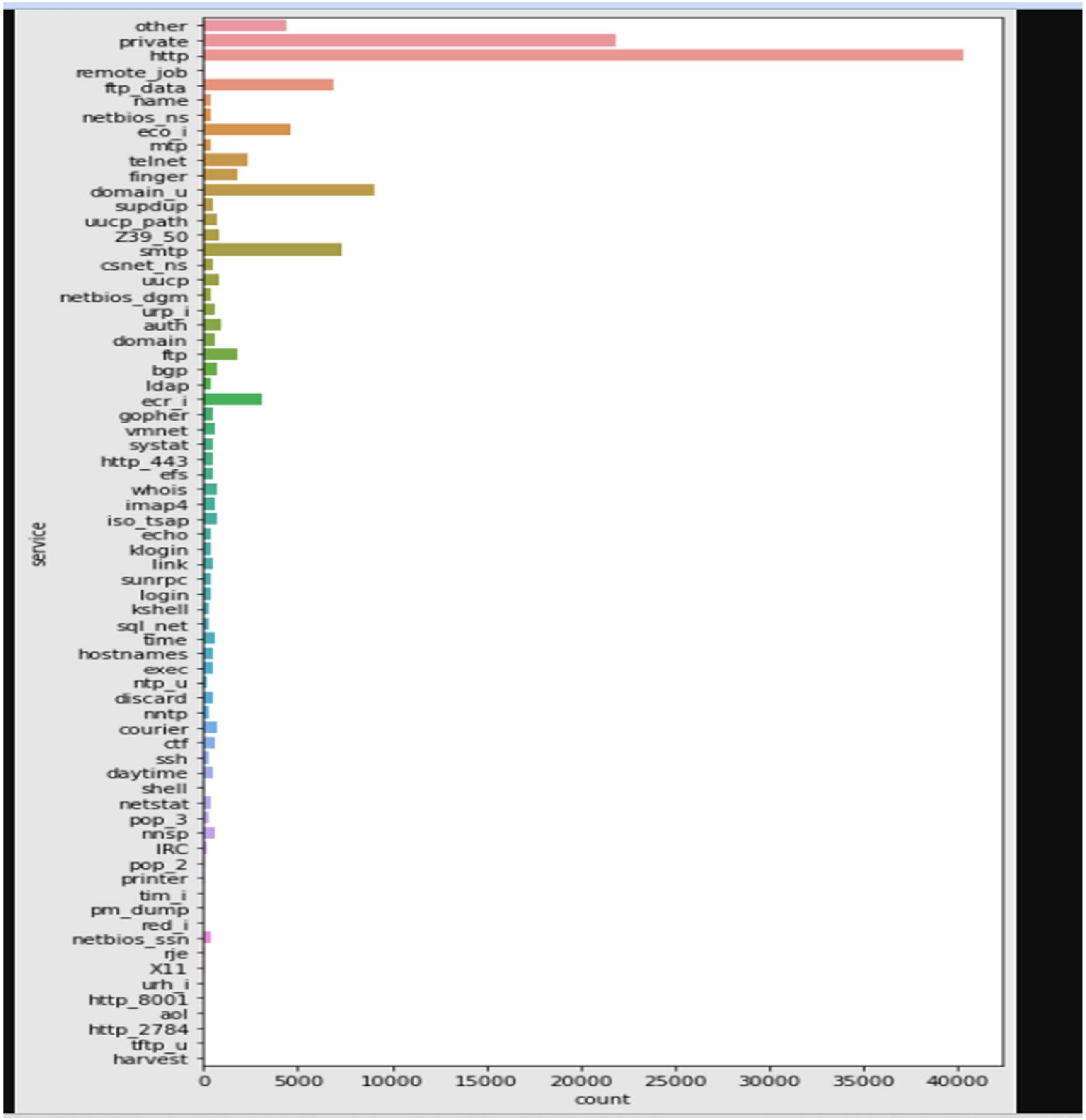


Fig.6.6 service_types of plots

The figure 6.7 depicts about the data of flag attribute in NSL-KDD dataset.

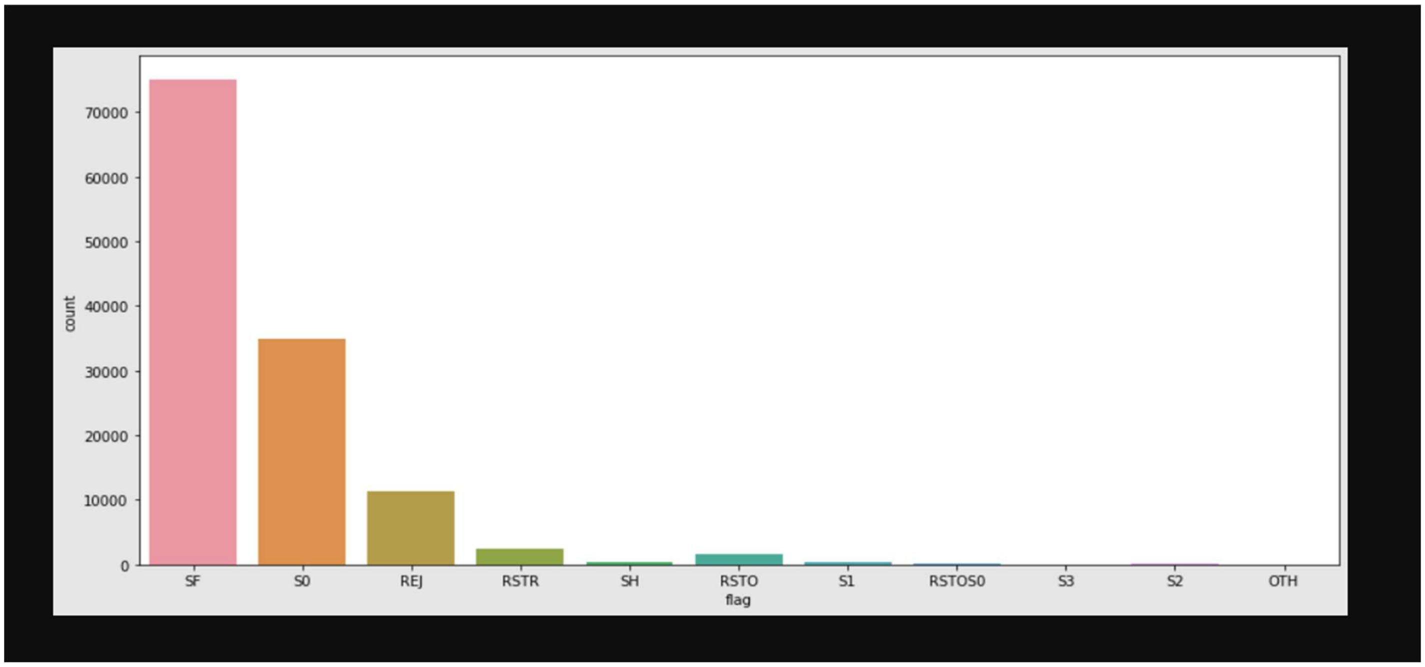


Fig.6.7 flag data plot

The figure 6.8 depicts about the data of attack attribute in NSL-KDD dataset.

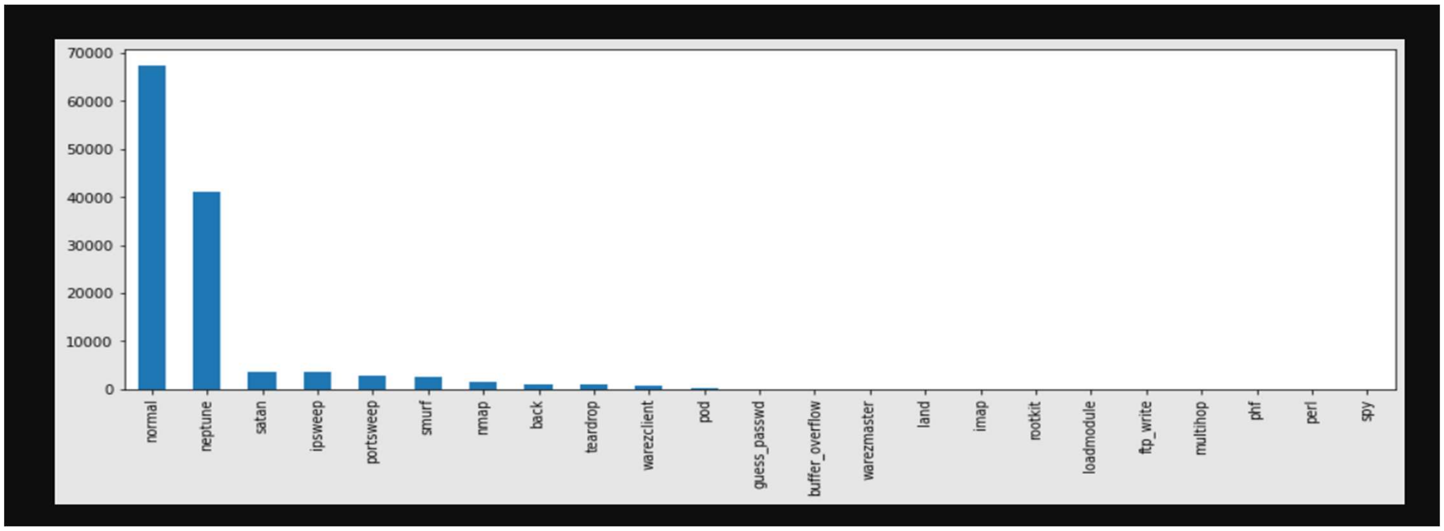


Fig.6.8 attack plot

The figure 6.9 depicts about the percentage of type of attacks before replacing all attacks with label 'other' except normal and neptune.

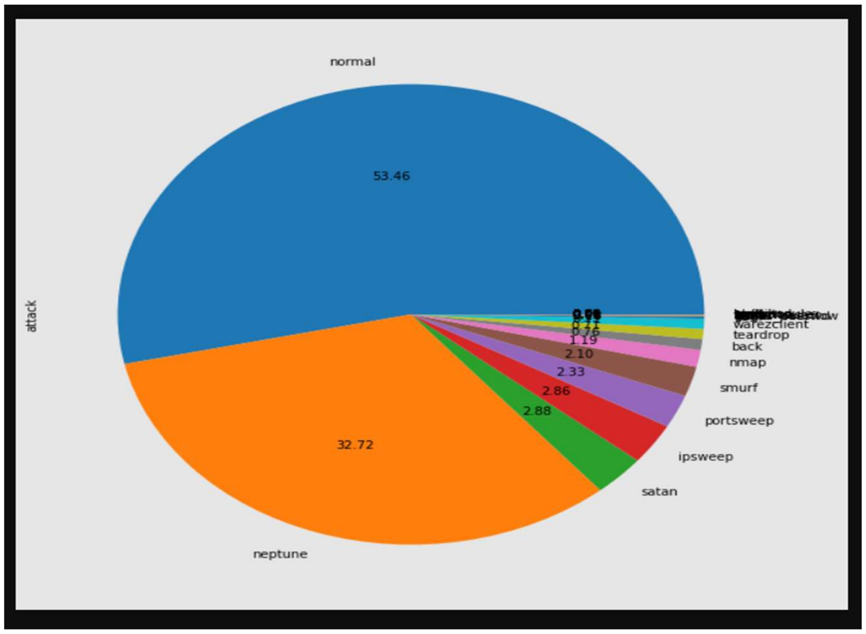


Fig.6.9 pie chart of attack_type before replacement

The figure 6.10 depicts about the percentage of type of attacks after replacing all attacks with label 'other' except normal and neptune.

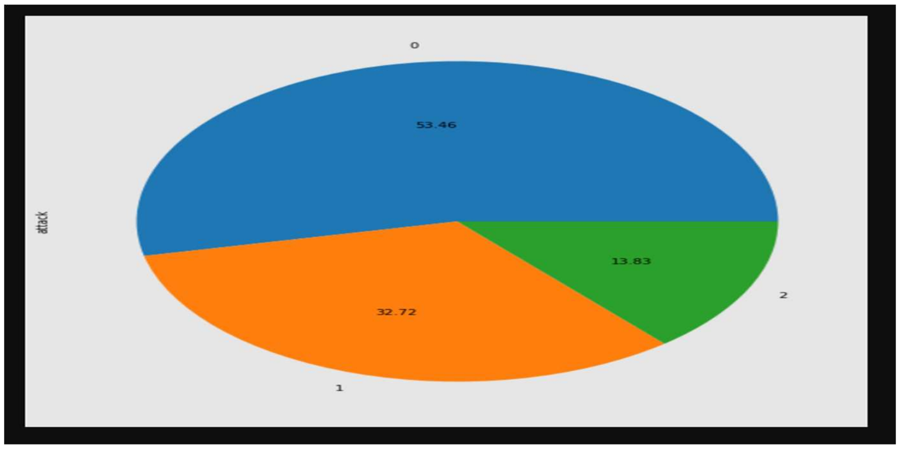


Fig.6.10 pie chart of attack_type after replacement

Confusion Matrix and Classification Report for XGBoost classifier:

The figure 6.11 shows the confusion matrix of the XGBoost model.

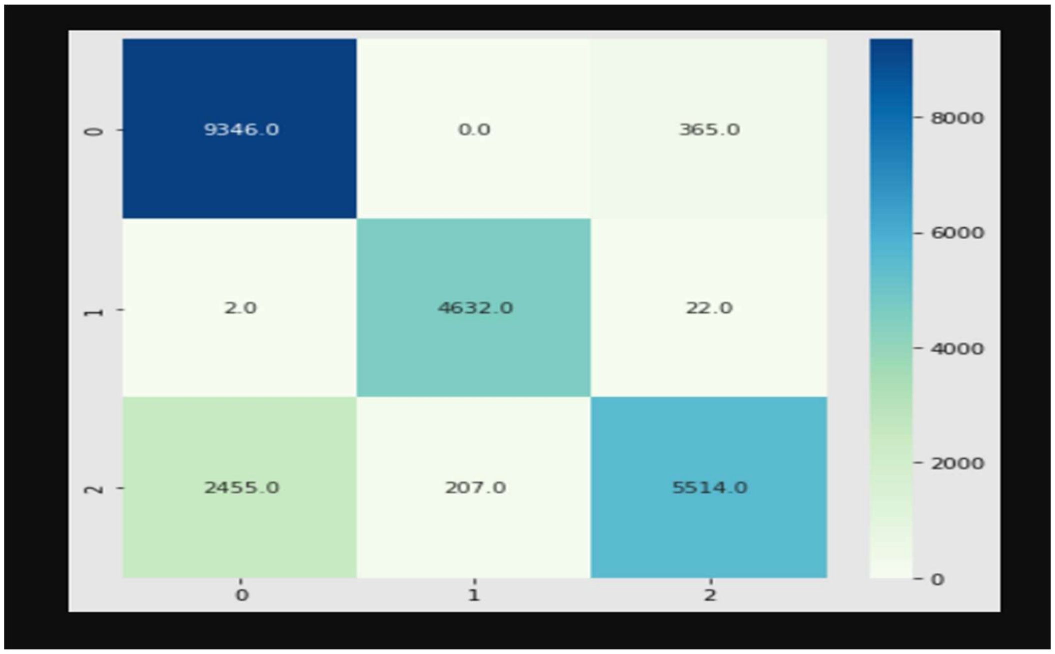
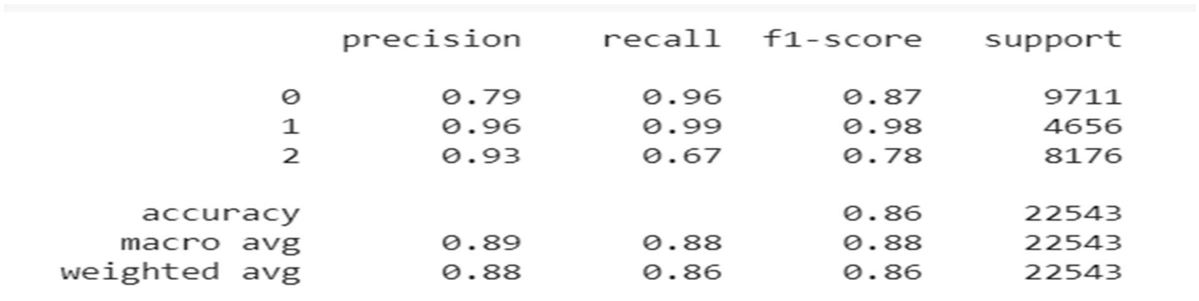


Fig.6.11 Confusion matrix of XG-Boost model

The figure 6.12 shows the classification report obtained after model execution of XG-Boost classifier.



```
              precision    recall  f1-score   support

           0       0.79      0.96      0.87      9711
           1       0.96      0.99      0.98      4656
           2       0.93      0.67      0.78      8176

    accuracy                           0.86     22543
   macro avg       0.89      0.88      0.88     22543
weighted avg       0.88      0.86      0.86     22543
```

Fig.6.12 Classification report of XG-Boost model

Confusion Matrix and Classification Report for LSTM Classifier:

The figure 6.13 shows the confusion matrix of the LSTM model.

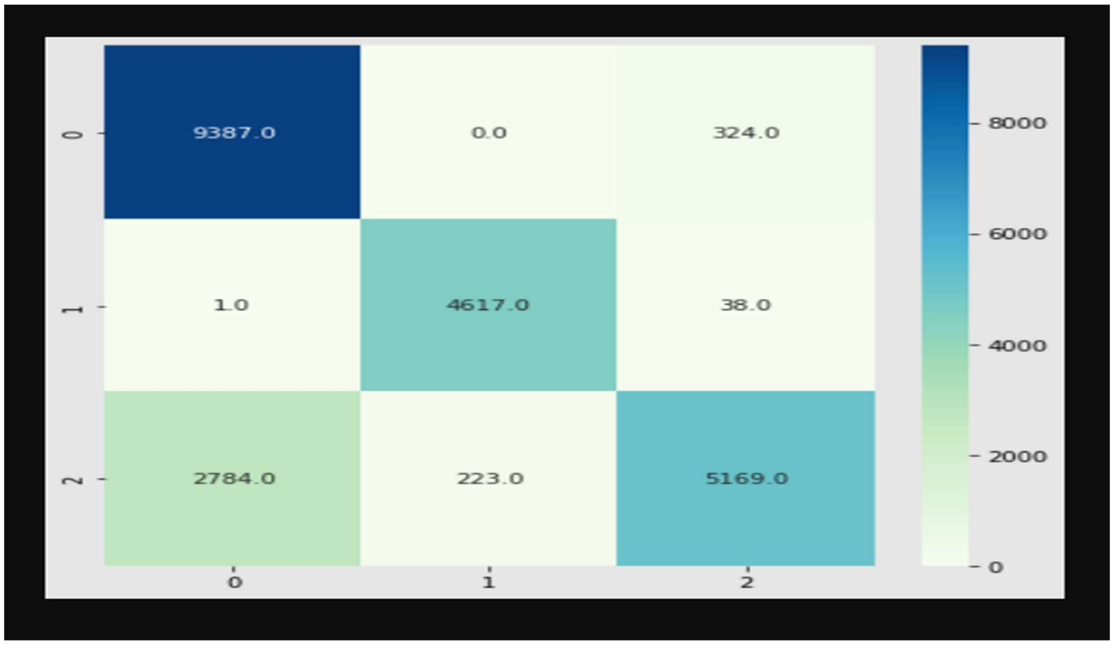
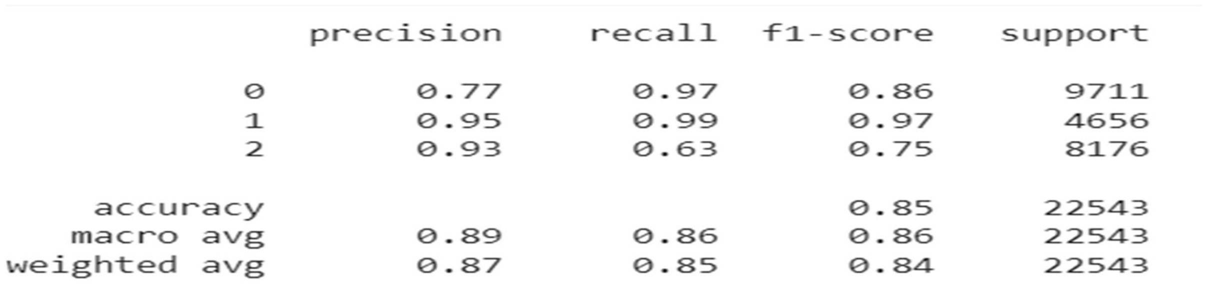


Fig.6.13 Confusion matrix of LSTM model

The figure 6.14 shows the classification report obtained after model execution of LSTM classifier.



```
              precision    recall  f1-score   support

           0       0.77      0.97      0.86      9711
           1       0.95      0.99      0.97      4656
           2       0.93      0.63      0.75      8176

    accuracy                           0.85     22543
   macro avg       0.89      0.86      0.86     22543
weighted avg       0.87      0.85      0.84     22543
```

Fig.6.14 Classification report of LSTM model

Accuracy comparison of models:

The figure 6.15 shows the comparison of model accuracies of XG-Boost and LSTM using entire test and training set of NSL-KDD dataset.

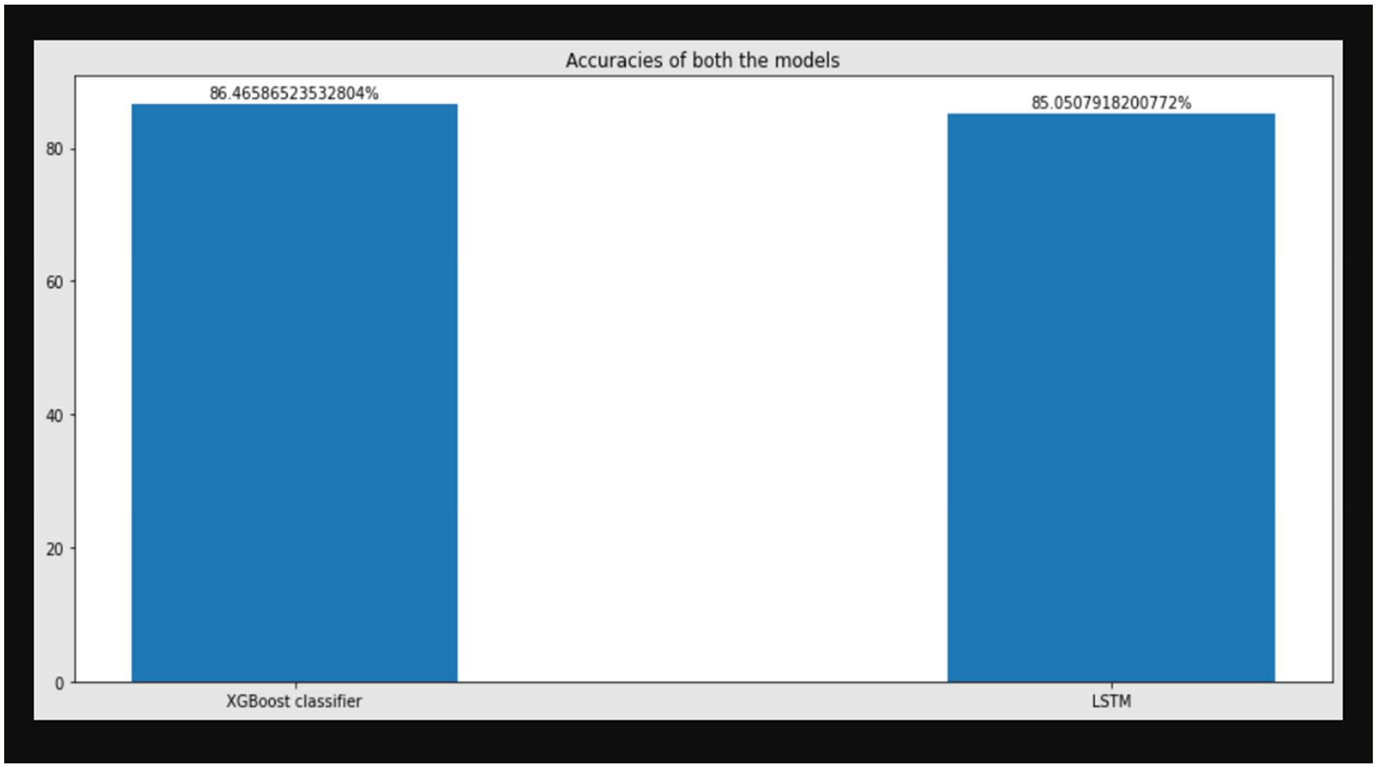


Fig.6.15 Accuracy comparison of models

The figure 6.16 shows the comparison of model accuracies of XG-Boost and LSTM using sub datasets provided in NSL-KDD dataset.

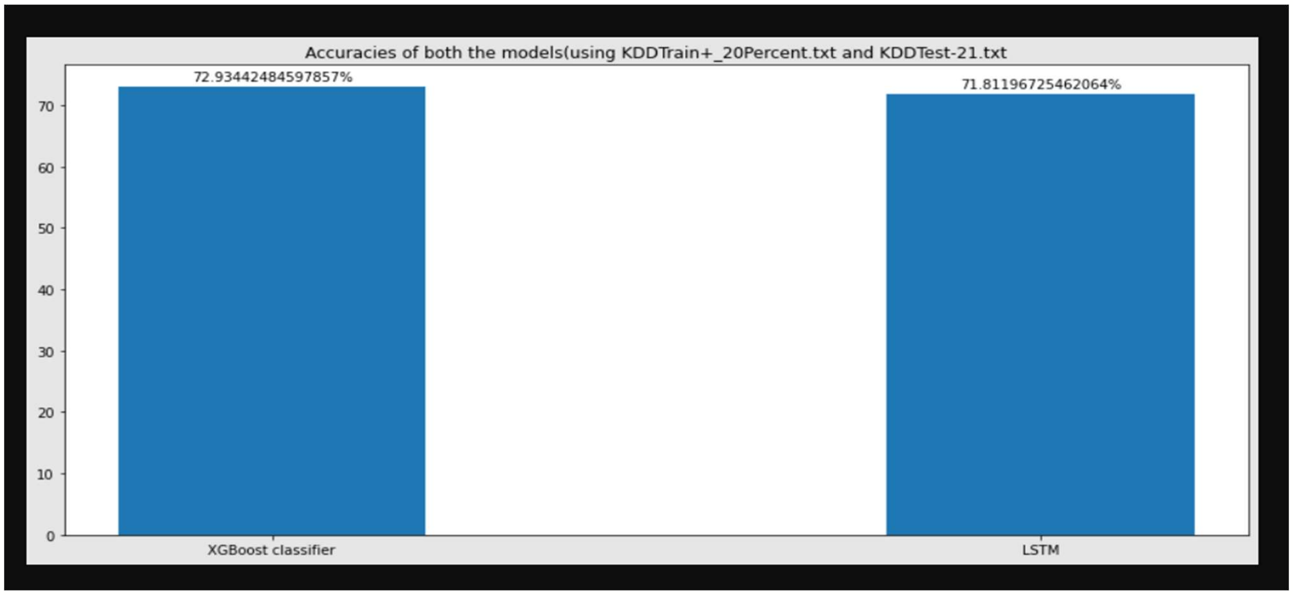


Fig.6.16 Accuracy comparison of models using sub-datasets

7. SYSTEM TESTING

7.1 Introduction:

Testing forms an integral part of any software development project. Testing helps in ensuring that the final product is by and large, free of defects and it meets the desired requirements. Proper testing in the development phase helps in identifying the critical errors in the design and implementation of various functionalities thereby ensuring product reliability. Even though it is a bit time-consuming and a costly process at first, it helps in the long run of software development.

Although machine learning systems are not traditional software systems, not testing them properly for their intended purposes can lead to a huge impact in the real world. This is because machine learning systems reflect the biases of the real world. Not accounting or testing for them will inevitably have lasting and sometimes irreversible impacts.

Testing Traditional Software Systems v/s Machine Learning Systems

In traditional software systems, code is written for having a desired behaviour.as the outcome. Testing them involves testing the logic behind the actual behaviour. and how it compares with the expected behaviour.

In machine learning systems, however, data and desired behavior are the inputs and the models learn the logic as the outcome of the training and optimization processes. In this case, testing involves validating the consistency of the model’s logic and our desired behavior.

Due to the process of models learning the logic, there are some notable obstacles in the way of testing Machine Learning systems. They are:

- **Indeterminate outcomes**-On retraining, it’s highly possible that the model parameters vary significantly
- **Generalization**-It’s a huge task for Machine Learning models to predict sensible outcomes for data not encountered in their training.
- **Coverage**-There is no set method of determining test coverage for a Machine Learning model**.**
- **Interpretability**-Most ML models are black boxes and don’t have a comprehensible logic for a certain decision made during prediction

These issues lead to a lower understanding of the scenarios in which models fail and the reason for that behavior; not to mention, making it more difficult for developers to improve their behaviours.

Difference between Model Testing and Model Evaluation

From the discussion above, it may feel as if model testing is the same as model evaluation but that's not true. Model evaluations focus on the performance metrics of the models like accuracy, precision, the area under the curve, f1 score, log loss, etc. These metrics are calculated on the validation dataset and remain confined to that. Though the evaluation metrics are necessary for assessing a model, they are not sufficient because they don't shed light on the specific behaviours of the model.

It is fully possible that a model's evaluation metrics have improved but its behavior on a core functionality has regressed. Or retraining a model on new data might introduce a bias for marginalized sections of society all the while showing no difference in the metrics values. This is extra harmful in the case of ML systems since such problems might not come to light easily but can have devastating impacts.

In summary, model evaluation helps in covering the performance on validation datasets while model testing helps in explicitly validating the nuanced behaviours of our models. During the development of ML models, it is better to have both model testing and evaluation to be executed in parallel.

Writing Test Cases

We usually write two different classes of tests for Machine Learning systems:

- Pre-train tests
- Post-train tests

Pre-train tests:

The intention is to write such tests which can be run without trained parameters so that we can catch implementation errors early on. This helps in avoiding the extra time and effort spent in a wasted training job.

We can test the following in the pre-train test:

- The model predicted output shape is proper or not.

- Test dataset leakage i.e., checking whether the data in training and testing datasets have no duplication.
- Temporal data leakage which involves checking whether the dependencies between training and test data do not lead to unrealistic situations in the time domain like training on a future data point and testing on a past data point.
- Check for the output ranges. In the cases where we are predicting outputs in a certain range (for example when predicting probabilities), we need to ensure the final prediction is not outside the expected range of values.
- Ensuring a gradient step training on a batch of data leads to a decrease in the loss
- Data profiling assertions

Post-train tests:

Post-train tests are aimed at testing the model's behavior. We want to test the learned logic and it could be tested on the following points and more:

- Invariance tests which involve testing the model by tweaking only one feature in a data point and checking for consistency in model predictions. For example, if we are working with a loan prediction dataset then change in sex should not affect an individual's eligibility for the loan given all other features are the same or in the case of titanic survivor probability prediction data, change in the passenger's name should not affect their chances of survival.
- Directional expectations wherein we test for a direct relation between feature values and predictions. For example, in the case of a loan prediction problem, having a higher credit score should increase a person's eligibility for a loan.
- Apart from this, you can also write tests for any other failure modes identified for your model.

7.2 Test Cases

Table 7.1 Test Cases

Test No	**Test Cases**	**Expected Output**	**Actual Output**	**Pass/ Fail**
1	Importing the Libraries	Imported	Imported without errors	Pass
2	Collecting the data and by using the Authentications and mounting	Collected	Got Access to the data using the authentications	Pass
3	Pre-process the data	Data is cleaned	Data is cleaned without errors	Pass
4	Normalizing the data	Normalize the data	Normalize the data	Pass
5	Splitting data into Train and Test	Train and Test 80-20%	Train and Test 80-20%	Pass
6	Splitting data into Test and Validation	Test and Validation 50-50%	Test and Validation 50-50%	Pass
7	Running model	Training the SMOTE	Trained the SMOTE	Pass

		techniques like XGBoost Classifier and LSTM models.	techniques like XGBoost Classifier and LSTM models.	
8	Model Evaluation	Classification Report	Classification Report Generated	Pass
9	Predictions	User sending the input from the validation for the prediction to classify	Model predicting the output	Pass

The above table consists of description of test cases tested against the system for verification of its functionality. Various test cases are used to perform testing on different modules of the system.

8. CONCLUSION

The pressure on network intrusion detection increases as network intrusion continues to evolve. An Intrusion Detection System using machine learning algorithms is proposed to detect the intrusions in the network. The proposed model is predictive and is capable of distinguishing between anomaly connections(intrusions) and normal connections. The increased number of minority samples can reduce the imbalance of network traffic and improve the classification accuracy. Two classification methods, XGBoost and LSTM are combined with SMOTE sampling technique. Experiments are performed on NSL-KDD dataset, the results, and accuracies of XGBoost algorithm(Extreme Gradient Boosting) and LSTM algorithm(Long Short Term Memory) are compared. XGBoost algorithm classified better than LSTM algorithm with the accuracy of 87.15% .

9. FUTURE ENHANCEMENTS

- In future, we can apply the pre-defined models to check the better accuracy.
- Researchers are working to combine probabilistic approaches with deep learning algorithms to define some uncertainties and their results.

BIBLIOGRAPHY

[1] 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.

[2] M. Zamani and M. Movahedi, ‘‘Machine learning techniques for intrusion detection,’’ 2013, arXiv:1312.2177. [Online]. Available: http://arxiv. org/abs/1312.2177.

[3] M. S. Pervez and D. M. Farid, ‘‘Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs,’’ in Proc. 8th Int. Conf. Softw., Knowl., Inf. Manage. Appl. (SKIMA), Dec. 2014, pp. 1–6.

[4] H. Shapoorifard and P. Shamsinejad, ‘‘Intrusion detection using a novel hybrid method incorporating an improved KNN,’’ Int. J. Comput. Appl., vol. 173, no. 1, pp. 5–9, Sep. 2017. [5] S. Bhattacharya, P. K. R. Maddikunta, R. Kaluri, S. Singh, T. R. Gadekallu, M. Alazab, and U. Tariq, ‘‘A novel PCA-firefly based XGBoost classification model for intrusion detection in networks using GPU,’’ Electronics, vol. 9, no. 2, p. 219, Jan. 2020.

[6] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, ‘‘A deep learning approach for network intrusion detection system,’’ in Proc. 9th EAI Int. Conf. Bioinspired Inf. Commun. Technol.

(Formerly BIONETICS), 2016, pp. 21–26.

[7] P. Torres, C. Catania, S. Garcia, and C. G. Garino, ‘‘An analysis of recurrent neural networks for botnet detection behavior,’’ in Proc. IEEE Biennial Congr. Argentina (ARGENCON), Jun. 2016, pp. 1–6.

[8] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, ‘‘Malware traffic classification using convolutional neural network for representation learning,’’ in Proc. Int. Conf. Inf. Netw. (ICOIN), 2017, pp. 712–717.

[9] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, ‘‘A survey of deep learningbased network anomaly detection,’’ Cluster Comput., vol. 22, pp. 949–961, 2019.

[10] B. A. Tama, M. Comuzzi, and K.-H. Rhee, ‘‘TSE-IDS: A two-stage classifier ensemble for intelligent anomaly-based intrusion detection system,’’ IEEE Access, vol. 7, pp. 94497– 94507, 2019.