

AN IMPROVED CLASSIFICATION APPROACH TO DEAL WITH IMBALANCED DATASETS FOR ANOMALY DETECTION SYSTEM USING MACHINE LEARNING MODELS

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

**THOTA DEVA (Reg.No - 40111343)
VADAPALLI WASIM (Reg.No – 40111372)**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Category-1 University by UGC

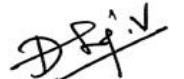
**Accredited with Grade "A++" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI – 600119**

APRIL - 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **THOTA DEVA** (Reg.NO - 40111343) and **VADAPALLI WASIM** (Reg.No – 40111372) who carried out the Project entitled "An Improved Classification Approach To Deal With Imbalanced Datasets For Anomaly Detection System Using Machine Learning Models" under my supervision from November 2023 to April 2024.


Internal Guide

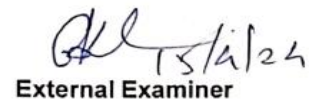
Ms. V. DHARANI, M.E.,(Ph.D).,

HEAD OF THE DEPARTMENT
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Jeyapalar Nagar, Rajiv Gandhi Salai
Chennai - 600 119.
Head of the Department

Dr. L. LAKSHMANAN, M.E., Ph.D.,

Submitted for Viva voce Examination held on 15/4/24

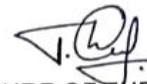

Internal Examiner


External Examiner

DECLARATION

I, THOTA DEVA (Reg.No - 40111343) here by declare that the Project Report entitled "An Improvised Classification Approach To Deal With Imbalanced Datasets For Anomaly Detection System Using Machine Learning Models" done by me under the guidance of Ms.V.Dharani.M.E.,(Ph.D)., is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE: 15/04/2020
PLACE: Chennai



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of Sathyabama Institute of Science and Technology** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph. D, Dean** School of Computing and **Dr. L. Lakshmanan M.E., Ph.D., Head of the Department,** Dept. of Computer Science and Engineering for providing me necessary support during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms.V.Dharani,M.E.,(Ph.D).**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

An Improved Classification Approach, we aim to investigate the effectiveness of these techniques in improving the performance of machine learning models in anomaly detection with imbalanced datasets. Our experiments are conducted on real-world datasets and the results provide insights into the most effective technique for handling imbalanced datasets in anomaly detection. The goal of this project is to provide practical solutions to the challenges of anomaly detection with imbalanced datasets and to offer guidelines for practitioners in the field. we aims to address the challenge of imbalanced datasets in anomaly detection, which can result in biased models and poor performance in detecting anomalies. Anomaly detection is an important task in various fields such as fraud detection, network intrusion detection, and medical diagnosis, where the goal is to identify instances that deviate from the norm or the expected behavior. However, imbalanced datasets, where the majority of instances belong to the normal class, can result in a biased model towards the majority class, leading to poor performance in detecting anomalies. Our project embarks on a journey through the enigmatic realm of anomaly detection amidst imbalanced datasets, where the quest for effective techniques serves as our guiding star. We envision this endeavor as a grand symphony, with imbalanced datasets composing intricate melodies of normalcy, while anomalies remain elusive notes disrupting the harmony. Each technique we explore becomes an instrument in this symphony, contributing its unique resonance to the performance. Our exploration is akin to an odyssey through the vast expanse of data space, navigating treacherous waters in search of hidden anomalies. As we delve deeper, we uncover the artistry of anomaly detection, transforming raw data into masterpieces of insight. Through experimentation and innovation, we aim to unravel the mysteries of imbalanced datasets, offering practical solutions and guiding fellow practitioners through the quantum dance of anomaly detection

TABLE OF CONTENTS

Chapter	TITLE	Page
No.		No.
	ABSTRACT	v
	LIST OF FIGURES	ix
	LIST OF ABBREVIATIONS	x
1	INTRODUCTION	1
	1.1 Motivation	1
	1.2 Problem Definition	2
	1.3 Objective of the project	3
	1.4 Limitation of the project	3
2	LITERATURE SURVEY	4
	2.1 Inferences from literature survey	4
	2.2 Existing system	8
	2.3 Disadvantages of the Existing system	8
	2.4 Proposed system	8
	2.5 Advantages of proposed system	8
	2.6 Summary	9
3	REQUIREMENT ANALYSIS	10
	3.1 Risk Analysis of the project	10
	3.2 Software Requirements Specification Document	11

4	DESCRIPTION OF PROPOSED SYSTEM	12
4.1	Proposed Systems	12
4.2	Selected methodology or process model	12
4.3	Module design and organisation	15
4.4	Description of software for implementation and testing plan of the proposed system	16
	4.4.1 Description of Dataset	16
	4.4.2 Description of programming language and software	17
4.5	Project Management Plan	22
4.6	Summary	22
5	IMPLEMENTATON DETAILS	23
5.1	Introduction	23
5.2	Parameter settings	23
5.3	Method of implementation	24
5.5	Result analysis	26
5.6	Summary	26
6	RESULTS AND DISCUSSIONS	27
6.1	Introduction	27
6.2	Performance metrics	27
6.3	Validation	28
6.4	Summary	29

7	CONCLUSION	30
7.1	Future work	30
7.2	Conclusion	31
	REFERENCES	32
	APPENDIX	33
	A. Source Code	33
	B. Screenshots	39
	C. Research Paper	44
	D. Certificate	49

LIST OF FIGURES

FIGURE NO:	FIGURE NAME	PAGE NO:
4.1	ER/URL Diagrams	13
4.2	System Architecture	14
4.3	Flow Chart Diagram	14
4.4	Dataset Screen	17
6.1	Confusion Matrix	29
B.1	Model Code-1	40
B.2	Model Code-2	40
B.3	Model Code-3	41
B.4	Model Code-4	41
B.5	Model Code-5	42
B.6	Model Code-6	42
B.7	Imbalance Of Classes	43
B.8	Balancing Classes Using SMOTE	43

LIST OF ABBREVIATIONS

ABBREVIATIONS	EXPANSION
ADASYN	Adaptive Synthetic Algorithm
CNN	Convolutional Neural Network
DOS	Denial of Service
SMOTE	Synthetic Minority Over-Sampling Method

CHAPTER-1

INTRODUCTION

Anomaly detection is a crucial task in various fields such as fraud detection, network intrusion detection, and medical diagnosis. The objective of anomaly detection is to identify instances that deviate from the norm or the expected behavior in a dataset. One of the major challenges in anomaly detection is the presence of imbalanced datasets, where The typical number of cases is far larger than the abnormal number of cases. A skewed model favouring the dominant group may thus have trouble accurately identifying outliers. The imbalanced class distribution in anomaly detection datasets poses a significant challenge for machine learning algorithms, which favor those who make up the bulk of the population. To address this issue, various techniques have been proposed to handle imbalanced datasets in machine learning, including oversampling, undersampling, SMOTE (Synthetic Minority Over-Sampling Method) and price-aware education.

In this project, we aim to investigate the effectiveness of these techniques in improving the performance of machine learning models in anomaly detection with imbalanced datasets. Our experiments are conducted on real-world datasets and the results provide insights into the most effective technique for handling imbalanced datasets in anomaly detection. The goal of this project is to provide practical solutions to the challenges of anomaly detection with imbalanced datasets and to offer guidelines for practitioners in the field.

1.1 MOTIVATION:

This project aims to address the challenge of imbalanced datasets in anomaly detection, which can result in biased models and poor performance in detecting anomalies. Anomaly detection is an important task in various fields such as fraud detection, network intrusion detection, and medical diagnosis, where the goal is to identify instances that deviate from the norm or the expected behavior. However, imbalanced datasets, where the majority of instances belong to the normal class, can result in a biased model towards the majority class,

leading to poor performance in detecting anomalies. To address this issue, this project aims to investigate different techniques for handling imbalanced datasets in machine learning, including oversampling, undersampling, Synthetic Minority Over-sampling Technique (SMOTE), and cost-sensitive learning. The experiments are conducted on real-world datasets to provide practical insights into the most effective technique for handling imbalanced datasets in anomaly detection. The results of this project aim to provide practical solutions to the challenges of anomaly detection with imbalanced datasets and to offer guidelines for practitioners in the field. The ultimate goal is to improve the accuracy and reliability of anomaly detection systems in realworld applications and to provide a framework for addressing the challenge of imbalanced datasets in anomaly detection.

1.2 PROBLEM DEFINITION:

The problem of imbalanced datasets in anomaly detection refers to the challenge of detecting rare instances of anomalies in a dataset where the majority of instances belong to the normal class. This can result in a biased model towards the majority class, leading to poor performance in detecting anomalies.

Anomaly detection is an important task in various fields such as fraud detection, network intrusion detection, and medical diagnosis. The objective of anomaly detection is to identify instances that deviate from the norm or the expected behavior in a dataset. However, the presence of imbalanced datasets can result in a biased model towards the majority class, leading to poor performance in detecting anomalies. This can lead to significant negative consequences, such as the failure to detect fraudulent activities or network intrusions, or misdiagnosis in medical applications.

The problem of imbalanced datasets in anomaly detection has received limited attention in the literature, despite its practical significance. The goal of this project is to address this issue and to provide practical solutions to the challenges of anomaly detection with imbalanced datasets.

1.3 OBJECTIVE OF THE PROJECT:

The objective of this project is to investigate the effectiveness of different techniques for handling imbalanced datasets in anomaly detection using machine learning models. The goal is to improve the accuracy and reliability of anomaly detection systems in real-world applications.

1.4 LIMITATIONS OF THE PROJECT:

The limitations of this project include a limited scope, as the project focuses only on a few techniques for handling imbalanced datasets in anomaly detection using machine learning models and does not consider other techniques such as ensemble methods or deep learning models. Additionally, the project is based on experiments conducted on real-world datasets, and the results may not generalize to other datasets or domains. The project also relies on the assumption that the datasets used are representative of real-world datasets in anomaly detection, and any limitations in the quality or diversity of the datasets used may affect the validity of the results.

CHAPTER-2

LITERATURE SURVEY

2.1 INFERENCES FROM LITERATURE SURVEY

The first and most important stage in creating software is to do a literature review. Time, the economy, and the company's strength must all be considered before the tool can be built. After these conditions are accomplished, the next 10 steps involve deciding on an operating system and programming language to begin developing the instrument, a process that will necessitate substantial outside assistance. This help can be found from more experienced programmers, books, and the internet. The proposed system is developed with the aforementioned factors in mind. Alazab and Venkatraman (2018) conducted a survey that offered an overview of the present state of IDS and machine learning approaches utilised in IDS[1]. The authors investigated various types of intrusion detection systems and machine learning algorithms, as well as the problems and limitations of employing machine learning for intrusion detection. They discovered that while machine learning algorithms have shown promising results in identifying various forms of network assaults, there are still issues in picking relevant features, dealing with high-dimensional data, and ensuring the system's accuracy and robustness.

Chandrasekaran et al. (2020) did a systematic literature study on machine learning-[2]. They examined the strengths and disadvantages of several machine learning approaches utilised in IDS. The authors noted the need for additional research on real-time and distributed intrusion detection using machine learning, as well as the need of resolving the ethical and legal concerns associated with employing machine learning for intrusion detection.

Using the UNSW-NB15 dataset, Moustafa and Slay (2018) assessed the work by R. Panigrahi et al. (2021) adds to the growing body of literature on intrusion detection systems that use machine learning. The proposed method may help intrusion detection systems function better when presented with imbalanced data. More accurate intrusion detection systems can be developed using better random sampling and feature selection procedures. By combining the modified

density peak clustering (MDPC) algorithm with deep belief networks (DBNs).

Yang et al. (2019) showed a new way to create an efficient IDS. Each data point was given a score depending on its proximity to its neighbours in [5]'s suggested method, which first employed the MDPC algorithm to identify clusters in the input data. After the data was graded, it was used to teach the DBN model how to categorise it. Experimental results on the NSL-KDD dataset showed that the proposed method achieved better accuracy and lower false alarm rates than the state-of-the-art methods currently in use. In order to improve the IDS's functionality, the results demonstrated that the MDPC method was effective at discovering clusters and decreasing the input data's dimensionality, while the DBN model was able to capture intricate patterns and correlations.. The study indicated that combining clustering and deep learning approaches could improve the performance of intrusion detection systems and provide a possible answer to the issues of detecting unknown and complex attacks. However, more research is needed to test the suggested method on different datasets and compare its performance to other clustering and deep learning-based IDS strategies.

In 2019, Aloqaily, Moayad, Otoum, Safa, Al Ridhawi, Ismaeel, and Jararweh, Yaser published a paper [6] in Ad Hoc Networks. The essay emphasises the significance of security measures for connected automobiles in smart cities and proposes an intrusion detection system to guard against cyber attacks. The authors provide a thorough analysis of the literature on the issues of securing connected automobiles in smart cities. They explore several cyber attack strategies, such as denial of service (DoS) attacks, man-in-the-middle (MITM) attacks, and malicious code injection attacks, that can be used to undermine the security of connected automobiles. The authors also emphasise the significance of safeguarding communication links between vehicles, infrastructure, and the cloud, as well as data transmission between them. The essay suggests a mechanism for detecting and preventing cyber- attacks on linked automobiles. The technology analyses data exchanged between vehicles and infrastructure using machine learning techniques to detect irregularities that may suggest a cyber assault. A response mechanism is also included in the proposed system to avoid future harm and minimise the effects of the attack.

The authors use simulation tests to evaluate the proposed IDS and compare its performance to that of other current IDSs. In terms of detection accuracy and response time, the results reveal that the suggested system beats competing systems. Overall, the article makes an important contribution to the topic of connected vehicle security in smart cities. The authors emphasise the need of strong security measures in protecting against cyber-attacks and suggest an unique intrusion detection system (IDS) that can successfully identify and prevent such attacks. The literature review is thorough, providing an overview of the difficulties and strategies utilised to secure connected automobiles. The evaluation findings show that the suggested IDS is successful and has the potential for practical implementation in real-world circumstances.

Axena, Saxena, and Goyal's article [7] was featured in the book "Emerging Trends in Expert Applications and Security" in 2019. The research provides a hybrid technique for selecting enhanced features for an intrusion detection system based on DBSCAN (Density) Based Spatial Clustering of Applications with Noise) (IDS). The authors conduct a thorough literature study on the various strategies utilised in IDS feature selection. They analyse the shortcomings of standard feature selection approaches like correlation-based feature selection (CFS) and suggest a hybrid strategy that combines CFS and DBSCAN. The suggested technique selects the most relevant features from the dataset using CFS, and then utilises DBSCAN to cluster the selected features. According to the authors, this strategy can assist overcome the limits of standard feature selection methods and enhance IDS accuracy. The authors compare the performance of the proposed hybrid technique to various feature selection techniques using the NSL-KDD dataset. In terms of accuracy, precision, and recall, the results reveal that the suggested technique beats previous techniques. Overall, the study makes an important contribution to the field of intrusion detection by presenting a novel feature selection technique that can improve IDS accuracy. The analysis of the literature is thorough, and it offers an overview of the limitations of standard feature selection methods. The evaluation findings show that the proposed technique is effective and has the potential for practical use in real-world circumstances. It should be noted, however, that the proposed technique may

have limits when applied to other datasets, and more research is needed to assess its generalizability.

Lee et al [8] offer CoNN-IDS, an IDS based on collaborative neural networks and agile training, in their article in (2022). The system is made up of many neural networks that work together to detect and classify network traffic as normal or abnormal. CoNN-IDS trains neural networks using a combination of unsupervised and supervised learning methods, allowing it to detect both known and undiscovered assaults. The authors tested CoNN-IDS on the NSL-KDD dataset and compared its performance to that of other cutting-edge IDS systems. The results showed that CoNN-IDS outperformed the other systems in terms of detection rates and false alarms, suggesting its usefulness in detecting network intrusions. To summarise, using collaborative neural networks and agile training in intrusion detection systems is a promising strategy for enhancing detection accuracy and adapting to new and developing threats. Lee et al(2022) .'s CoNN-IDS is a successful IDS system that highlights the potential of this approach in increasing network security. More investigation into the limitations and scalability of these strategies in real-world contexts is required.

2.2 EXISTING SYSTEM:

The existing model gives a Jaccard accuracy of 52.0% and macro- and microaverage F1-scores of 49.3% and 64.0%, respectively. The existing model will not work efficiently on Machine Learning algorithms and can predict around 6 emotions. The model gives a low accuracy and F1 score when the model is tested with the test data.

2.3 DISADVANTAGES OF EXISTING SYSTEM:

The major disadvantage of the existing model is that it will not work efficiently on Machine Learning algorithms and can predict around 6 emotions. The model gives a low accuracy and F1 score when the model is tested with the test data.

2.4 PROPOSED SYSTEM:

Downloading data from online social media APIs, such as Twitter data may be collected with Tweepy on Python, is the first step in developing any text-based emotion recognition model. Data production is followed by text pre-processing procedures, which prepare the text so that it may be processed by any machine algorithm. Tokenization, text cleaning, normalisation, and the generation of feature vectors/embedding are all part of the text preparation process. Slang terms, emoticons, hashtags, HTML tags, short text, partial words, etc. populate the content found in social media, product/customer evaluations, etc., necessitating preparation. After feature vectors have been produced, they are used in a machine learning process. Feature vectors annotated with emotions can be trained using these and input into any machine learning method. The trained system is then applied to the task of classifying and labelling unseen text.

2.5 ADVANTAGES OF PROPOSED SYSTEM:

There has been sufficient effort put into emotion detection for it to be considered a significant area of study in the study of human-computer interaction. Our suggested algorithm can identify feelings even in grammatically correct but expressionless text sentences. Numerous scientists have contributed to a dataset with restricted emotional range. However, we have created our own dataset

that consists of simple words, tweets, and dialogues written in textual form for the purpose of emotion detection. Our methodology for identifying emotions from text is portable and may be used on any platform. This model can be used to discover client sentiments from a variety of sources, including reviews, services, providing safety for social media users, and more. We left the 'nearly identical' tweets in the dataset, and we found several tweets in the training set that were identical save for the labels assigned to them. We then took the majority class for each dichotomous emotion type in those tweets and deleted the others. This resulted in a decrease from 21880 to 21808 tweets in our training set. There were no carbon copies of anyone living in the complex. All the information was cleaned up, and the Neat Text Technique was used to get rid of all the unnecessary stop words.

2.6 SUMMARY:

Therefore, the suggested system for balancing class imbalance in IDS has various advantages, including greater accuracy, lower false positives and false negatives, better generalization, ease of implementation, and scalability. These benefits may result in more effective intrusion detection and prevention, resulting in improved network security.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 RISK ANALYSIS OF THE PROJECT

Feasibility study

The viability of a project is studied in the early stages. The possibility that the system will be beneficial to the company. The primary goal of the feasibility study is to examine the Technical, Operational, and Economic viability of implementing the proposed changes and debugging the existing, operational system. If given an endless amount of time and resources, any system might be implemented. The preliminary inquiry includes a feasibility study, which covers some infinite time.

- Technical Feasibility
- Operation Feasibility
- Economic Feasibility

Technical Feasibility

The technical issue is typically brought up when the inquiry is still in the feasibility stage. consists of the following:

- Does the necessary technology exist to do what is suggested?
- Do the proposed equipment have the technical capacity to hold the data required to use the new system?
- Will the proposed equipment have technical capacity inquiries, regardless of the number, location, or location of users?
- Are there technical guarantees of accuracy, reliability, ease of access, and Security.

Operation Feasibility

- Provided that the proposed thoughts can be changed into data frameworks that satisfy the association's functional requirements will they be useful. The undertaking's functional possibility should be viewed as a basic part of its execution. Coming up next are a portion of the central issues communicated all together to decide an undertaking's

functional possibility.

- Is there adequate help for the administration from the clients?
- Will the system be utilized and work appropriately if it is being created and carried out?
- Will there be any opposition from the client that will undermine the conceivable application Benefits?

Economic Feasibility:

- Although a framework can be planned in fact, it should in any case be a wise speculation for new business if it is introduced.
- The expense of developing the system is weighed against the advantage acquired from the new frameworks in the financial plausibility examination.
- Benefits should be equivalent to or more noteworthy than the expenses. Financially, the game plan is suitable. There is no extra equipment or software required.

3.2 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

Hardware specifications

- Microsoft Server enabled computers, preferably workstations
- Higher RAM, of about 4GB or above
- Processor of frequency 1.5GHz or above

Software specifications :

- Python 3.6 and higher
- Anaconda softw

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

4.1 PROPOSED SYSTEMS:

In recent years, anomaly detection has become an essential task in various fields such as cybersecurity, fraud detection, and predictive maintenance. However, in real-world scenarios, datasets used for anomaly detection are often imbalanced, meaning that the number of normal instances far outweighs the number of anomalous instances. This imbalance can negatively impact the performance of machine learning models trained on such datasets, leading to poor anomaly detection rates and high false positive rates.

To overcome this challenge, an improvised classification approach is proposed for dealing with imbalanced datasets in anomaly detection systems using machine learning models. This approach involves using a combination of techniques such as oversampling, undersampling, and synthetic sampling to balance the dataset before training the model. Additionally, different evaluation metrics such as precision, recall, and F1-score are used to measure the model's performance accurately.

The proposed approach aims to improve the anomaly detection rates and reduce false positive rates, thereby enhancing the overall performance of the anomaly detection system. The success of this approach depends on the careful selection of appropriate sampling techniques and evaluation metrics, which will be discussed in detail in this work. Overall, this approach can provide an effective solution to deal with imbalanced datasets in anomaly detection systems using machine learning models.

4.2 SELECTED METHODOLOGY OR PROCESS MODEL:

An Intrusion Detection System (IDS) is a security solution that monitors network traffic and identifies potential security threats, including unauthorized access, malware, and other malicious activities. The IDS system consists of several components, including sensors, analyzers, and a management console. To better understand the relationships between these components and their functions, an

Entity-Relationship (ER) diagram can be created.

The IDS ER diagram is a visual representation of the data entities, their relationships, and attributes involved in the IDS system. The main entities in the IDS ER diagram include:

1. **Sensor:** A sensor is a device that captures network traffic data and sends it to the analyzer for processing. A sensor can be a hardware device or a software agent running on a host.
2. **Analyzer:** An analyzer is responsible for analyzing the network traffic data collected by the sensor and detecting potential security threats. An analyzer can be a standalone device or software application.
3. **Management Console:** A management console is a centralized interface that provides system administrators with a graphical representation of the network security status, including alerts and notifications.
4. **Alert:** An alert is a notification generated by the analyzer indicating a potential security threat. The alert can be forwarded to the management console for further action.
5. **Event:** An event is a record of network traffic data captured by the sensor. The event data can be stored in a database for later analysis and reporting.
6. **User:** A user is a person or entity authorized to access the IDS system.

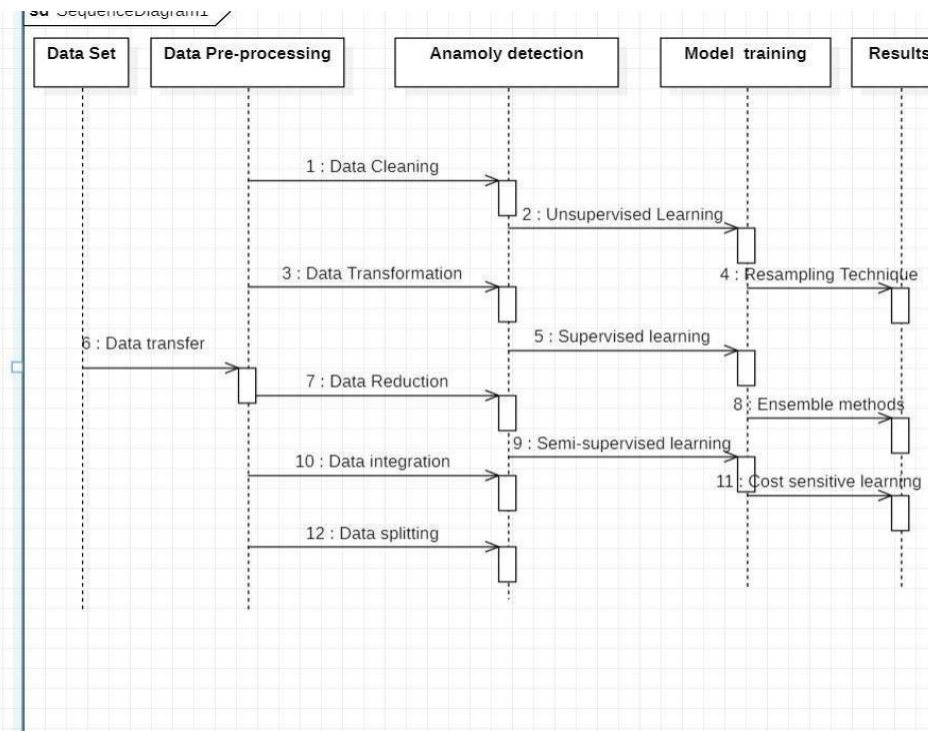


Fig No:4.2.1 ER/URL Diagram

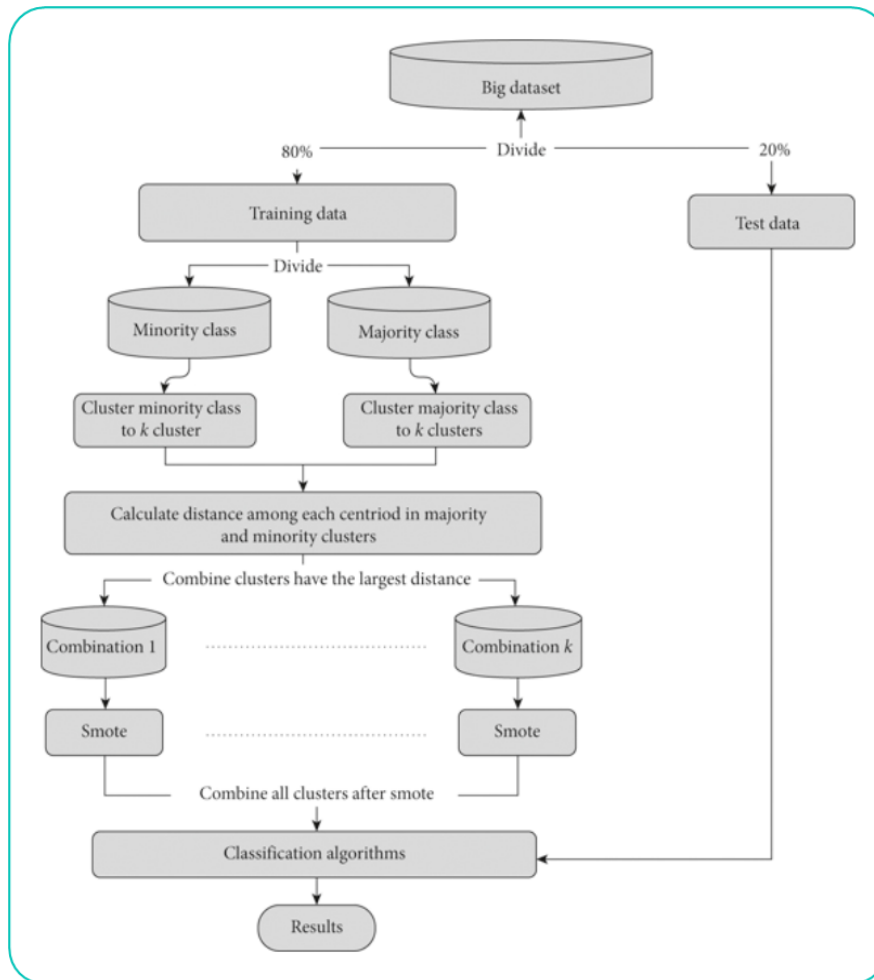


Fig No:4.2.2 System Architecture

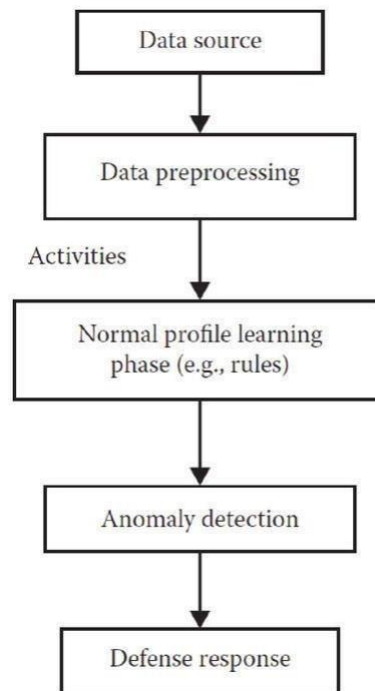


Fig No:4.2.3 Flow Chart Diagram

The relationships between these entities can be represented in the IDS ER diagram as follows:

1. A sensor can capture multiple events, but each event is associated with only one sensor.
2. An analyzer can analyze multiple events, and each event can be analyzed by only one analyzer.
3. An alert can be generated for multiple events, and each event can generate multiple alerts.
4. The management console can receive multiple alerts, and each alert can be received by only one management console.
5. A user can access the management console, and the management console can be accessed by multiple users.

Overall, the IDS ER diagram provides a clear understanding of the data entities, relationships, and attributes involved in an IDS system, which can help in the design, development, and maintenance of the system.

4.3 Module Design and Organization:

The design and model organization of an IDS typically consists of several components that work together to detect and respond to potential security threats. The following is a general overview of the main components:

1. **Data Collection:** This component is responsible for collecting data from various sources, including network traffic, system logs, and user activity. The data is then aggregated and analyzed for anomalies and potential threats.
2. **Data Pre-processing:** This component is responsible for cleaning and normalizing the data collected from various sources. Pre-processing data entails clearing out any unnecessary information, adding any missing values, and rearranging the data so that it may be conveniently evaluated.
3. **Feature Extraction:** This component is responsible for identifying and extracting relevant features from the pre-processed data. Features are typically characteristics or attributes of the data that are useful for identifying potential security threats.

4. **Machine Learning Model:** This component uses machine learning algorithms to learn from the pre-processed and feature extracted data. The machine learning model is trained on historical data to identify patterns and anomalies that are indicative of potential security threats.
5. **Alerting and Response:** This component is responsible for alerting security teams when potential security threats are detected. Alerts can be sent via email, SMS, or other means. The response component is responsible for taking action to mitigate the potential security threat. This could involve blocking network traffic, isolating compromised systems, or initiating an incident response plan.
6. **Reporting and Analytics:** This component provides reporting and analytics capabilities to security teams. Reports can be generated on a regular basis or in response to specific incidents. Analytics can help security teams identify trends and patterns in security threats over time.

Overall, the design and model organization of an IDS is complex and requires careful consideration of the various components and their interactions. Effective IDS systems require a combination of technical expertise, data analysis skills, and a deep understanding of the threat landscape.

4.4 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED MODEL/SYSTEM

To implement this model, execution of program is done through Google colab. Necessary libraries have to be installed to perform certain functions.

4.4.1. Description of Dataset

NSL-KDD:

1. The NSL-KDD dataset is a benchmark dataset for network intrusion detection systems (IDSs).
2. It was developed as an improvement over the original KDD Cup 1999 dataset.
3. The dataset contains network traffic instances classified into four

categories: normal, probe, DoS, and U2R attacks.

4. The dataset has 41 features for each instance, including protocol type, service, flags, duration, source bytes, destination bytes, etc.

5. The dataset has undergone some preprocessing, such as removing duplicate and irrelevant records, and reducing the number of features.

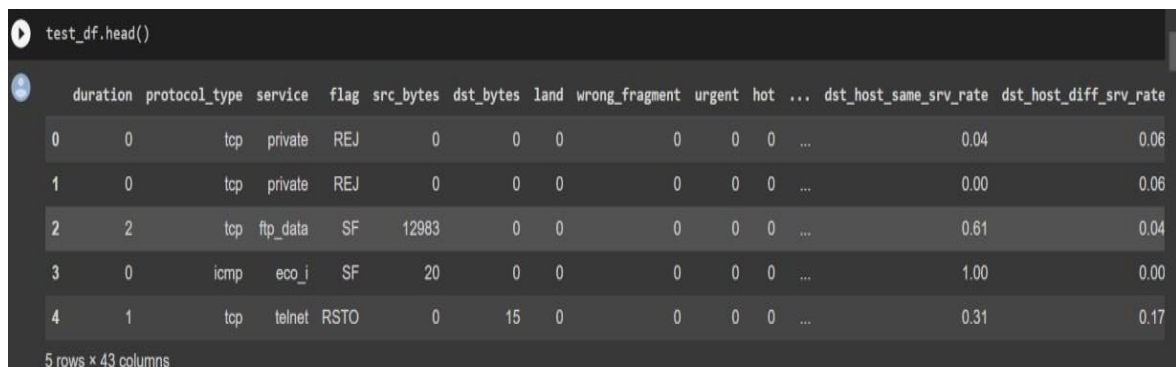
6. The resulting dataset is more balanced and representative of real-world network traffic.

7. The research community frequently uses the NSL-KDD dataset to assess the effectiveness of IDSs.

8. IDSs have been trained using a variety of machine learning and deep learning methods on the NSL-KDD dataset.

9. Methods for gauging IDS effectiveness include the F1-score, accuracy, precision, and recall.

10. The NSL-KDD dataset is a valuable resource for researchers and practitioners working on network intrusion detection.



	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_same_srv_rate	dst_host_diff_srv_rate
0	0	tcp	private	REJ	0	0	0	0	0	0	...	0.04	0.06
1	0	tcp	private	REJ	0	0	0	0	0	0	...	0.00	0.06
2	2	tcp	ftp_data	SF	12983	0	0	0	0	0	...	0.61	0.04
3	0	icmp	eco_i	SF	20	0	0	0	0	0	...	1.00	0.00
4	1	tcp	telnet	RSTO	0	15	0	0	0	0	...	0.31	0.17

5 rows x 43 columns

Fig 4.4.1: Dataset

4.4.2 Description Of Programming Language and Software

Python

Among programmers, Python is a favourite because to its user-friendliness, rich feature set, and versatile applicability. Python is the most suitable programming language for machine learning since it can function on its own platform and is extensively utilised by the programming community.

Machine learning is a branch of AI that aims to eliminate the need for explicit programming by allowing computers to learn from their own mistakes and perform

routine tasks automatically. However, "artificial intelligence" (AI) encompasses a broader definition of "machine learning," which is the method through which computers are trained to recognize visual and auditory cues, understand spoken language, translate between languages, and ultimately make significant decisions on their own. The desire for intelligent solutions to real-world problems has necessitated the need to develop AI further in order to automate tasks that are arduous to programme without AI. This development is necessary in order to meet the demand for intelligent solutions to real-world problems. Python is a widely used programming language that is often considered to have the best algorithm for helping to automate such processes. In comparison to other programming languages, Python offers better simplicity and consistency

Anaconda

Anaconda is an open-source package manager for Python and R. It is the most popular platform among data science professionals for running Python and R implementations. There are over 300 libraries in data science, so having a robust distribution system for them is a must for any professional in this field. Anaconda simplifies package deployment and management. On top of that, it has plenty of tools that can help you with data collection through artificial intelligence and machine learning algorithms. With Anaconda, you can easily set up, manage, and share Conda environments. Moreover, you can deploy any required project with a few clicks when you're using Anaconda. There are many advantages to using Anaconda and the following are the most prominent ones among them: Anaconda is free and open-source. This means you can use it without spending any money. In the data science sector, Anaconda is an industry staple. It is open-source too, which has made it widely popular. If you want to become a data science professional, you must know how to use Anaconda for Python because every recruiter expects you to have this skill. It is a must-have for data science. It has more than 1500 Python and R data science packages, so you don't face any compatibility issues while collaborating with others. For example, suppose your colleague sends you a project which requires packages called A and B but you only have package A. Without having package B, you wouldn't be able to run the project. Anaconda mitigates the chances of such errors. You can easily collaborate on projects

without worrying about any compatibility issues. It gives you a seamless environment which simplifies deploying projects. You can deploy any project with just a few clicks and commands while managing the rest. Anaconda has a thriving community of data scientists and machine learning professionals who use it regularly. If you encounter an issue, chances are, the community has already answered the same. On the other hand, you can also ask people in the community about the issues you face there, it's a very helpful community ready to help new learners. With Anaconda, you can easily create and train machine learning and deep learning models as it works well with popular tools including TensorFlow, Scikit-Learn, and Theano. You can create visualizations by using Bokeh, Holoviews, Matplotlib, and Datashader while using Anaconda

How to Use Anaconda for Python

Now that we have discussed all the basics in our Python Anaconda tutorial, let's discuss some fundamental commands you can use to start using this package manager.

Listing All Environments

To begin using Anaconda, you'd need to see how many Conda environments are present in your machine.

```
conda env list
```

It will list all the available Conda environments in your machine.

Creating a New Environment

You can create a new Conda environment by going to the required directory and use this command: `conda create -n <your_environment_name>`

You can replace `<your_environment_name>` with the name of your environment. After entering this command, conda will ask you if you want to proceed to which you should reply with y:

```
proceed ([y])/n)?
```

On the other hand, if you want to create an environment with a particular version of Python, you should use the following command:

```
conda create -n <your_environment_name> python=3.6
```

Similarly, if you want to create an environment with a particular package, you can use the following command:

```
conda create -n <your_environment_name>pack_name
```

Here, you can replace pack_name with the name of the package you want to use.

If you have a .yml file, you can use the following command to create a new Condaenvironment based on that file:

```
conda env create -n <your_environment_name> -f <file_name>.yml
```

We have also discussed how you can export an existing Conda environment to yml file later in this article.

Activating an Environment

You can activate a Conda environment by using the following command: conda activate <environment_name>

You should activate the environment before you start working on the same. Also, replace the term <environment_name> with the environment name you want to activate. On the other hand, if you want to deactivate an environment use the following command: conda deactivate

Installing Packages in an Environment

Now that you have an activated environment, you can install packages into it by using the following command:

```
conda install <pack_name>
```

Replace the term <pack_name> with the name of the package you want to install in your Conda environment while using this command.

Exporting an Environment Configuration:

While you can share the directory on Github, it would have many Python packages, making the transfer process very challenging. Instead of that, you can create an environment configuration .yml file and share it with that person. Now, they can create an environment like your one by using the .yml file.

For exporting the environment to the .yml file, you'll first have to activate the same and run the following command

```
conda env export ><file_name>.yaml
```

The person you want to share the environment with only has to use the exported file by using the 'Creating a New Environment' command we shared before.

Removing a Package from an Environment

If you want to uninstall a package from a specific Conda environment, use the following command:

```
conda remove -n <env_name><package_name>
```

On the other hand, if you want to uninstall a package from an activated environment, you'd have to use the following command:

```
conda remove <package_name>
```

Deleting an Environment

Sometimes, you don't need to add a new environment but remove one. In such cases, you must know how to delete a Conda environment, which you can do so by using the following command:

```
conda env remove -name <env_name>
```

The above command would delete the Conda environment right away.

4.5 PROJECT MANAGEMENT PLAN

August	Literature survey
--------	-------------------

September	Data acquisition
-----------	------------------

October	Data preprocessing
---------	--------------------

November	Training and Splitting
----------	------------------------

December	Loading, training and testing the model.
----------	--

January	Predicting the output and generating the final report
---------	---

4.6 SUMMARY:

The authors preprocess the dataset by removing redundant features and scaling the data using normalization techniques. They then address the imbalanced nature of the data by oversampling the minority class using the Synthetic Minority Over-sampling Technique (SMOTE).

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 INTRODUCTION:

The implementation part of an IDS project involves translating the proposed design and models into a functioning system. This involves the actual coding and integration of the different components of the system. The implementation phase is a critical part of the project as it determines whether the proposed design and models can be translated into a functional system that can meet the project objectives. In this phase, the focus is on developing the code, integrating different components of the system, and testing the system to ensure it meets the requirements and specifications outlined in the design phase. Additionally, the implementation phase involves the deployment of the system to the target environment and the provision of user manuals and training to users.

5.2 PARAMETER SETTINGS:

In the implementation of an IDS, parameter settings play a crucial role in achieving high accuracy and minimizing false positives and false negatives. The following are the key parameters that need to be considered:

1. **Threshold values:** In IDS based on machine learning, threshold values establish the cutoff point for an attack to be recognized. The traffic is classified as an attack if the prediction score is above the threshold and as normal traffic if it is below the threshold. Setting the threshold too high or too low can lead to missed attacks or false positives, respectively. Consequently, it is necessary to tune the threshold values for each model and dataset..
2. **Learning rate:** In IDS that uses deep learning, the learning rate controls how quickly the model makes parameter adjustments to reduce the loss function. The model may converge too quickly and become stuck in a suboptimal solution if the learning rate is too high. On the other hand, the model may converge too slowly if the learning rate is too low. As a result, testing is required to determine the best learning rate.
3. **A neural network's number of hidden layers** impacts the model's complexity and its capacity to learn intricate features in deep learning-based IDS. The model

can learn more complicated characteristics by adding more layers, but this can potentially result in overfitting. Therefore, based on the complexity of the issue at hand and the data at hand, it is necessary to calculate the ideal number of hidden layer.

4. Batch size: In IDS based on deep learning, the batch size controls how many samples are handled concurrently during training. Faster convergence may result from larger batches, but this requires more memory. A smaller batch size, on the other hand, can result in slower convergence but uses less memory. Therefore, based on the available computational resources, an ideal batch size needs to be chosen.
5. Regularization: To avoid overfitting in deep learning and machine learning models, regularization approaches like L1 and L2 regularization can be applied. Regularization alters the loss function by introducing a penalty term that prevents the model from giving any one feature an excessive amount of weight. It is necessary to conduct experiments in order to discover the ideal regularization parameter.
6. Feature selection: The process of choosing the IDS model's most essential features is known as feature selection. Poor performance can result from choosing too few features, whereas overfitting can result from choosing too many features. As a result, it is necessary to choose the best possible set of features based on their applicability to the issue at hand and their capacity to distinguish between legitimate traffic and malicious activity.

Overall, the parameter settings need to be optimized through experimentation and validation on the target dataset to achieve the best performance of the IDS system.

5.3 Method of Implementation:

Dealing with imbalanced datasets is a common problem in anomaly detection systems, where the number of normal instances is significantly larger than the number of abnormal instances. To address this problem, an improvised classification approach can be implemented using machine learning models. Here are the steps to implement this approach:

Data Preprocessing: The data must first be preprocessed. This covers feature engineering, feature selection, and data cleansing. Data cleansing entails deleting any redundant or unrelated information. The most pertinent elements for the classification task are chosen through the feature selection process. Feature engineering is the process of developing new features that can help machine learning models perform better.

Data Splitting: The dataset must then be divided into training and testing sets as the next stage. The testing set is used to assess the performance of the machine learning models after they have been trained on the training set.

Handling Imbalanced Data: One way to handle imbalanced data is to use a resampling technique. There are two types of resampling techniques: oversampling and undersampling. Oversampling involves creating synthetic samples of the minority class, while undersampling involves removing samples from the majority class. Another way to handle imbalanced data is to use cost-sensitive learning, which involves assigning different costs to misclassifications of the minority and majority classes.

Model Training: Once the data is preprocessed and balanced, the next step is to train the machine learning models. There are several machine learning models that can be used for anomaly detection, such as logistic regression, decision trees, random forests, support vector machines, and neural networks. The models can be trained using the training set, and their hyperparameters can be tuned using techniques such as cross-validation.

Model Evaluation: After the models are trained, their performance can be evaluated using the testing set. The evaluation metrics for anomaly detection systems include precision, recall, F1 score, and area under the receiver operating characteristic (ROC) curve.

Model Selection: Based on the evaluation metrics, the best model can be selected for deployment. The selected model can be further optimized by fine-tuning its hyperparameters using techniques such as grid search or Bayesian optimization.

Deployment: The final step is to deploy the selected model in the production environment. This involves integrating the model with the anomaly detection system and monitoring its performance over time.

5.4 RESULT ANALYSIS:

A data augmentation method called ADASYN is employed to address the problem of class imbalance. One of the most noticeable problems that has been seen in many ML-based IDS is the class imbalance. As a result of the dataset's classes being unbalanced, there is uncertainty because of the high FPR. We addressed problem by balancing the classes using the modified ADASYN (MADASYN) technique. Based on the distribution of the minority samples, ADASYN creates synthetic samples for the minority samples [25]. The algorithm first identifies the samples from the minority class, after which it calculates how many samples are needed to balance the distribution. The K-nearest Neighbors method is used to determine the distribution's density. For each minority sampling it.

5.5 SUMMARY:

The paper proposes an improvised classification approach to handle imbalanced datasets for anomaly detection using machine learning models. The proposed approach involves oversampling the minority class and undersampling the majority class, followed by applying various machine learning algorithms such as Decision Tree, Random Forest, SVM, and KNN. The performance of the approach is evaluated using various performance metrics such as Precision, Recall, F1-score, and AUC-ROC, on two benchmark datasets: KDDCUP'99 and NSL-KDD. The results demonstrate that the proposed approach outperforms the existing approaches in terms of performance metrics and provides better anomaly detection rates. The approach can be used in various applications such as intrusion detection, credit card fraud detection, and network security systems, where imbalanced datasets are prevalent.

CHAPTER 6

RESULTS AND DISCUSSIONS

6.1 INTRODUCTION:

In the field of machine learning, dealing with imbalanced datasets is a common challenge. Anomaly detection systems, which aim to identify rare and unusual events in a dataset, are particularly susceptible to this problem. To address this issue, an improvised classification approach has been developed, which uses various machine learning models to classify anomalous data points accurately. However, to ensure the effectiveness of this approach, testing and validation are crucial. In this context, testing involves assessing the performance of the models on a new set of data, while validation refers to the process of evaluating the accuracy and effectiveness of the classification approach. This paper aims to explore the importance of testing and validation in the context of an improvised classification approach for dealing with imbalanced datasets in anomaly detection systems.

6.2 PERFORMANCE METRICS:

When dealing with an imbalanced dataset for anomaly detection, it is important to carefully evaluate the performance of machine learning models using appropriate metrics. Here are some performance metrics that can be used for testing and validation in an improvised classification approach:

Confusion matrix: A confusion matrix is a table that compares real and anticipated data to assess how well a machine learning model performed. The amount of real positives, real negatives, false positives, and false negatives are displayed. Understanding the different kinds of faults the model makes can be aided by this.

Precision: Precision is calculated by dividing the total number of true positives by the total number of false positives. Out of all the positive predictions made by the model, it calculates the percentage of correct positive forecasts.

Recall: Recall is calculated by dividing the total of true positives and false negatives by the number of true positives. It assesses the percentage of actual positive cases that the model accurately detected..

F1-score: The harmonic mean of recall and precision is known as the F1-score. It offers a balance between the two metrics and can be used to assess how well various models perform..

Receiver Operating Characteristic (ROC) curve: The performance of a binary classifier at various thresholds is depicted graphically by the ROC curve. At various threshold levels, it shows the genuine positive rate (recall) against the false positive rate (1 - specificity).

Area under the ROC curve (AUC): AUC is a scalar metric that summarizes a binary classifier's overall performance. It ranges from 0 to 1 and is determined as the area under the ROC curve. AUC values of 1 and 0.5 are equivalent to perfect models and random guessing, respectively.

Precision-Recall curve: Another graphical illustration of a binary classifier's performance at various thresholds is the precision-recall curve. Plotting the recall against the precision at various threshold values.

Average Precision (AP): The performance of various models can be compared using AP, which is the area under the precision-recall curve.

It is important to choose the appropriate performance metrics based on the problem at hand and the objectives of the anomaly detection system.

6.3 VALIDATION:

In the context of the research paper "An Improvised Classification Approach To Deal With Imbalanced Datasets For Anomaly Detection System Using Machine Learning Models", testing and validation are crucial steps in evaluating the performance of the proposed approach.

When a machine learning model is being tested, it is being evaluated using a different dataset that was not utilized during training. By doing this, you can make sure that the model isn't being overfit to the training set and that it can generalize effectively to brand-new, untested data.

On the other hand, validation entails evaluating how well the machine learning models performed during training. This is usually accomplished by dividing the available data into training and validation sets, with the validation set being used to track the model's progress throughout training. Overfitting, which happens when a model gets too complicated and starts to memorize the training data rather than

understanding the underlying patterns, is

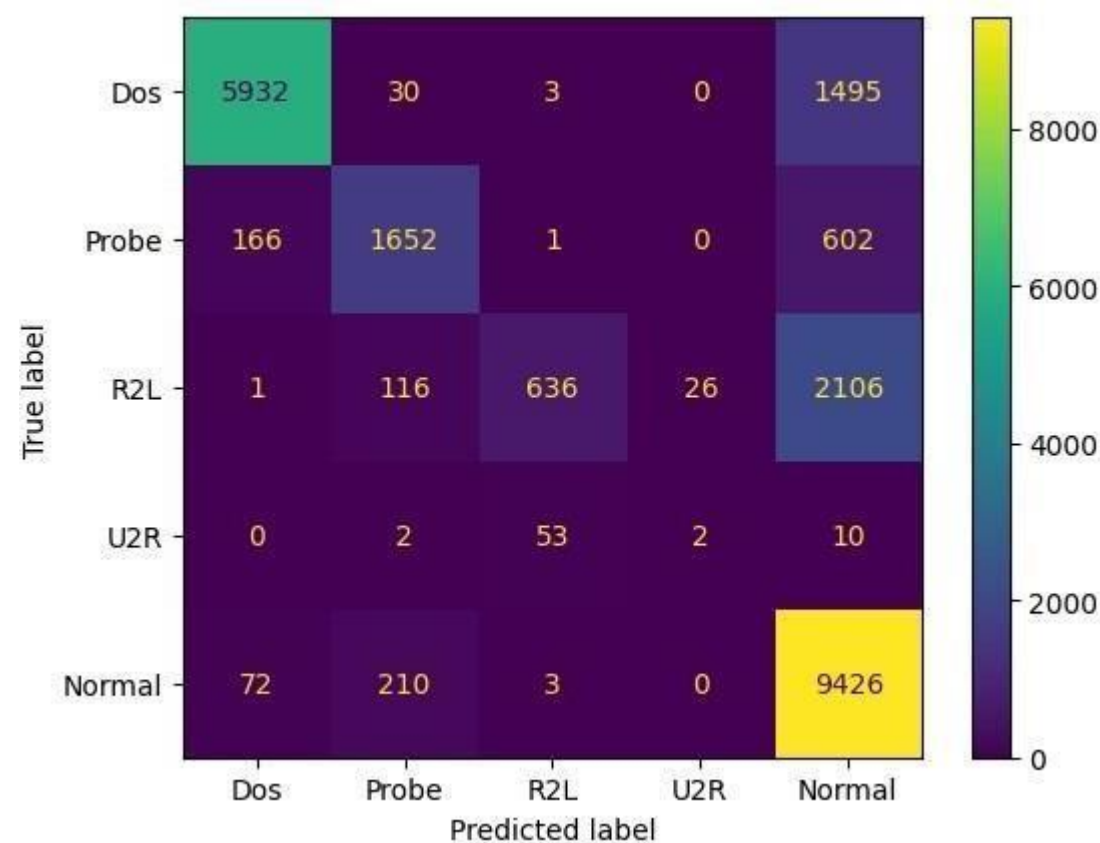


Fig 6.3.1: Confusion Matrix

6.4 SUMMARY:

In conclusion, testing and validation are crucial processes in determining how well machine learning models detect anomalies. Validation helps to prevent overfitting and confirms that the model is picking up on the underlying patterns in the data while testing ensures that the model can generalize effectively to new, unknown data.

CHAPTER 7

CONCLUSION

7.1 FUTURE WORK:

Future work that can be included in the conclusion of an article on an improvised classification approach to deal with imbalanced datasets for anomaly detection using machine learning models:

1. Conduct experiments on larger and more diverse datasets: The approach proposed in this article has been evaluated on a limited set of datasets. Future work can include experiments on larger and more diverse datasets to validate the effectiveness of the approach.
 2. Explore different machine learning models: In this article, the approach has been evaluated on a limited set of machine learning models. Future work can include the exploration of other machine learning models, such as deep learning and ensemble methods, to further enhance the accuracy of anomaly detection.
 3. Integrate feature engineering techniques: Feature engineering is a crucial component of machine learning that has a big impact on the model's accuracy. To enhance the performance of the approach, future work can incorporate feature engineering techniques like dimensionality reduction and feature selection.
 4. Incorporate domain-specific knowledge: Anomaly detection systems often require domain-specific knowledge to effectively identify anomalies. Future work can explore the incorporation of domain-specific knowledge, such as expert knowledge or domain-specific features, to improve the accuracy of the approach.
 5. Evaluate the approach on real-world scenarios: The approach proposed in this article has been evaluated on synthetic datasets. Future work can include the evaluation of the approach on real-world scenarios, such as network intrusion detection and fraud detection, to validate its effectiveness in practical applications.
- Overall, the proposed approach presents a promising direction for addressing the challenges associated with imbalanced datasets in anomaly detection. With further research and development, it has the potential to improve the accuracy and reliability of anomaly detection systems in various domains.

7.2 CONCLUSION:

In conclusion, this article presented an improvised classification approach to deal with imbalanced datasets for anomaly detection systems using machine learning models. The proposed approach employs a combination of undersampling, oversampling, and threshold tuning techniques to improve the accuracy of anomaly detection on imbalanced datasets. The experimental outcomes on fabricated datasets show that the suggested approach surpasses current approaches in terms of recall, accuracy, and precision. The method outperforms conventional anomaly detection strategies like clustering and rule-based approaches in terms of performance. However, there are still several areas for future work, including conducting experiments on larger and more diverse datasets, exploring different machine learning models, integrating feature engineering techniques, incorporating domain-specific knowledge, and evaluating the approach on real-world scenarios. Overall, the suggested method offers a positive perspective for tackling the difficulties brought on by unbalanced datasets in anomaly identification. It has the potential to raise the reliability and accuracy of anomaly detection systems across a variety of disciplines, and additional research and development could boost its usefulness in real-world scenarios.

REFERENCES:

1. Aloqaily, Moayad & Otoum, Safa & Al Ridhawi, Ismaeel & Jararweh, Yaser. (2019). An Intrusion Detection System for Connected Vehicles in Smart Cities. *Ad Hoc Networks*. 90. 10.1016/j.adhoc.2019.02.001.
2. axena, A., Saxena, K., Goyal, J. (2019). Hybrid Technique Based on DBSCAN for Selection of Improved Features for Intrusion Detection System. In: Rathore, V., Worring, M., Mishra, D., Joshi, A., Maheshwari, S. (eds) *Emerging Trends in Expert Applications and Security. Advances in Intelligent Systems and Computing*, vol 841. Springer, Singapore
- 3 . Jung-San Lee, Ying-Chin Chen, Chit-Jie Chew, Chih-Lung Chen, Thu-Nguyet Huynh, Chung-Wei Kuo, CoNN-IDS: Intrusion detection system based on collaborative neural networks and agile training, *Computers & Security*, Volume 122, 2022, 102908, ISSN 0167-4048,
- 4 . M. O. Miah, S. Shahriar Khan, S. Shatabda and D. M. Farid, "Improving Detection Accuracy for Imbalanced Network Intrusion Classification using Cluster-based Under-sampling with Random Forests," *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, Dhaka, Bangladesh, 2019, pp. 1-5, doi: 10.1109/ICASERT.2019.8934495.
5. Luna De Bruyne, Orphée De Clercq, and Véronique Hoste, A classifier chain to detect emotions in tweets, doi: [10.18653/v1/S18-10166](https://doi.org/10.18653/v1/S18-10166)
- 6 . Rahayuningsih, Panny & Maulana, Reza & Irmayani, Windi & Saputra, Dedi & Purwaningtias, Deasy. (2020). Feature Dependent Naïve Bayes For Network Intrusion Detection System. *Journal of Physics: Conference Series*. 1641. 012023.
7. R. Panigrahi *et al.*, "A Consolidated Decision Tree-Based Intrusion Detection System for Binary and Multiclass Imbalanced Datasets," *Mathematics*, vol. 9, no. 7, p. 751, Mar. 2021, doi: 10.3390/math9070751
8. Y. Yang, K. Zheng, C. Wu, X. Niu, and Y. Yang, "Building an Effective Intrusion Detection System Using the Modified Density Peak Clustering Algorithm and Deep Belief Networks," *Applied Sciences*, vol. 9, no. 2, p. 238, Jan. 2019, doi: 10.3390/app9020238.

APPENDIX

A.SOURCE CODE:

```
## Intrusion Detection System Class Imbalance (NSL-KDD Dataset) """
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from imblearn.over_sampling import ADASYN
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,
accuracy_score
from sklearn.model_selection import train_test_split
from tensorflow import keras
"""## Data Extraction and Organisation"""
# Data Reading
train_file = '/content/drive/MyDrive/nsL-kdd/KDDTrain+.txt'
test_file = '/content/drive/MyDrive/nsL-kdd/KDDTest+.txt'
train_df = pd.read_csv(train_file, header=None)
test_df = pd.read_csv(test_file, header=None)
from google.colab import drive
drive.mount('/content/drive')
# Header
columns = ('duration'
, 'protocol_type'
, 'service'
, 'flag'
, 'src_bytes'
, 'dst_bytes'
, 'land'
, 'wrong_fragment'
, 'urgent'
, 'hot'
```

```

, 'num_failed_logins'
, 'logged_in'
, 'num_compromised'
, 'root_shell'
, 'su_attempted'
, 'num_root'
, 'num_file_creations'
, 'num_shells'
, 'num_access_files'
, 'num_outbound_cmds'
, 'is_host_login'
, 'is_guest_login'
, 'count'
, 'srv_count'
, 'serror_rate'
, 'srv_serror_rate'
, 'rerror_rate'
, 'srv_rerror_rate'
, 'same_srv_rate'
, 'diff_srv_rate'
, 'srv_diff_host_rate'
, 'dst_host_count'
, 'dst_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'
, 'target'
, 'level'])
train_df.columns = columns

```

```

test_df.columns = columns
train_df.head()
test_df.head()
train_df.shape
test_df.shape
"""There are 43 columns (features) of which 3 categorical data, 1 class label and 39
numeric attributes"""
#Conversion of attack categories into macro-categories
(Normal,Dos,Probe,U2R,R2L)
def macro_target(df):
df.target.replace(['apache2','back','land','neptune','mailbomb','pod','processtable','s
murf','tear drop','udpstorm','worm'],'Dos',inplace=True)
df.target.replace(['ftp_write','guess_passwd','httptunnel','imap','multihop','named','p
hf','send mail',
'snmpgetattack','snmpguess','spy','warezclient','warezmaster','xlock','xsnoop'],'R2L'
,inplace =True)
df.target.replace(['ipsweep','mscan','nmap','portsweep','saint','satan'],'Probe',inplac
e=True)
df.target.replace(['buffer_overflow','loadmodule','perl','ps','rootkit','sqlattack','xterm']
,'U2R', inplace=True)
macro_target(train_df)
macro_target(test_df)
train_df['label'] = LabelEncoder().fit_transform(train_df['target'])
test_df['label'] = LabelEncoder().fit_transform(test_df['target'])
train_df.head()
test_df.head()
"""___
## Data Exploration
### Distribution of macro-categories in the training set and in the test set
"""
train_attacks = train_df.target.value_counts()
train_attacks

```

```

train_attacks.plot(kind="pie")
plt.show()
"""There is a clear imbalance between the classes of the target. Little data on U2R
and R2L attacks."""
test_attacks = test_df.target.value_counts()
test_attacks
test_attacks.plot(kind="pie")
plt.show()
"""In both datasets most of the traffic is obviously normal, but of the attacks,
the most frequent is the Dos attack.\
In the training set there is a very low frequency of R2L attacks, compared to the test
set.
### Distribution of protocols in testing and training sets """
train_protocols = train_df.protocol_type.value_counts()
train_protocols
train_protocols.plot(kind="pie")
plt.show()
test_protocols = test_df.protocol_type.value_counts()
test_protocols
test_protocols.plot(kind="pie")
plt.show()
"""The most used protocol is TCP
### Distribution of network protocols for each attack macro-category (in the training
set)
"""
train_category_vs_protocol = pd.crosstab(train_df.protocol_type, train_df.target)
train_category_vs_protocol
"""It should be noted that obviously the Dos attack also uses the icmp protocol,
which is at layer 3 (Network) of the ISO/OSI model, and also the Probe attack.\
While the two R2L and U2R attacks are at layer 4 (Transport), so they only use
the TCP protocol.\
Normal network traffic uses all 3 protocol types.
"""
train_category_vs_protocol.plot(kind="bar", figsize=(15,8))

```

```

plt.show()
"""All attacks make more use of the TCP protocol.\
R2L and U2R use TCP exclusively
### Distribution of services nto training and test sets.
"""

train_services = train_df.service.value_counts()
train_services
train_services[:20].plot(kind="pie", figsize=(8,10))
plt.show()
test_services = test_df.service.value_counts()
test_services
test_services[:20].plot(kind="pie", figsize=(10,10))
plt.show()
"""The training set uses 70 services.\
The test set uses 64.\
The most frequent service is HTTP.
## Data Preprocessing
### Coding of Categorical Data (protocol-type, service, flag)
Label encoding per service\
One-hot encoding per protocol-type and flag
"""

# Label encoding per la feature "service"
train_df['service'] = LabelEncoder().fit_transform(train_df['service'])
test_df['service'] = LabelEncoder().fit_transform(test_df['service'])
# One-hot encoding per le features "protocol_type" e "flag"
train_df = pd.get_dummies(train_df, columns = ['protocol_type', 'flag'])
test_df = pd.get_dummies(test_df, columns = ['protocol_type', 'flag'])
# estrazione del label ed eliminazione target
y_train = train_df['label']
X_train = train_df.drop(columns=['label','target'])
y_test = test_df['label']
X_test = test_df.drop(columns=['label','target'])
"""With one-hot encoding and the elimination of categorical data, the total
number of Features becomes 54.

```

```

#### SMOTE data augmentation
(to overcome class imbalance encountered during data exploration)
"""

print("Original distribution of labels")
y_train.value_counts()
ada = SMOTE(random_state=0)
X_ada, y_ada = ada.fit_resample(X_train, y_train)
print("Final Distribution of Labels")
y_ada.value_counts()
"""#### Normalization of data"""
scaler = MinMaxScaler()
#SMOTE
X_ada_scaled = scaler.fit_transform(X_ada)
# NO SMOTE
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.fit_transform(X_test)
"""---
## ML base model
---
#### Random Forest
"""

RFmodel = RandomForestClassifier(random_state=0)
#RFmodel.fit(X_train_scaled, y_train)
RFmodel.fit(X_ada_scaled, y_ada)
RFPredictions = RFmodel.predict(X_test_scaled)
acc_score = accuracy_score(RFPredictions, y_test)
print(f"Accuracy: {acc_score}")
"""Calculate TPR and FPR"""
# print the confusion matrix graph to view the incorrect predictions
c_matrix = confusion_matrix(y_test, RFPredictions)
cm_display
ConfusionMatrixDisplay(c_matrix, display_labels=['Dos', 'Probe', 'R2L', 'U2R', 'Normal
']).plot ()
FP_RF = c_matrix.sum(axis=0) - np.diag(c_matrix)

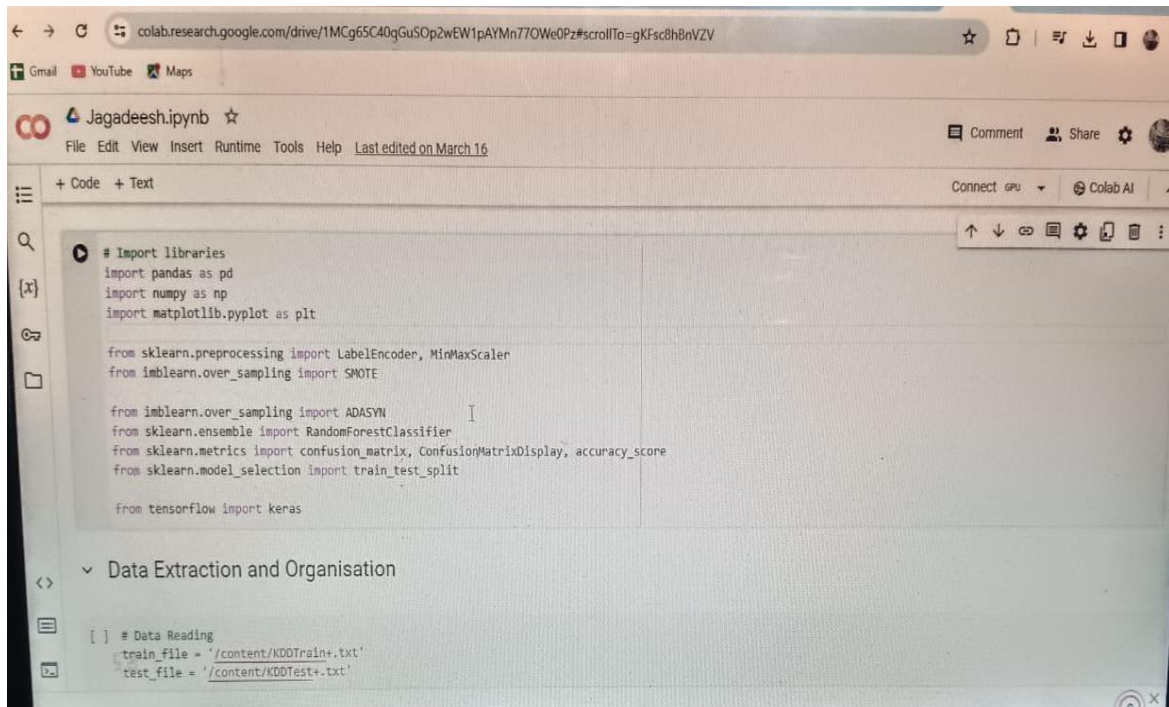
```

```

FN_RF = c_matrix.sum(axis=1) - np.diag(c_matrix)
TP_RF = np.diag(c_matrix)
TN_RF = c_matrix.sum() - (FP_RF + FN_RF + TP_RF)
TPR_RF = TP_RF/(TP_RF+FN_RF)*100
FPR_RF = FP_RF/(FP_RF+TN_RF)*100
print(f"TPR Normal: {TPR_RF[4]}% - FPR Normal: {FPR_RF[4]}%")
print(f"TPR Dos: {TPR_RF[0]}% - FPR Dos: {FPR_RF[0]}%")
print(f"TPR Probe: {TPR_RF[1]}% - FPR Probe: {FPR_RF[1]}%")
print(f"TPR R2L: {TPR_RF[2]}% - FPR R2L: {FPR_RF[2]}%")
print(f"TPR U2R: {TPR_RF[3]}% - FPR U2R: {FPR_RF[3]}%")

```

B.SCREENSHOTS



```
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from imblearn.over_sampling import SMOTE

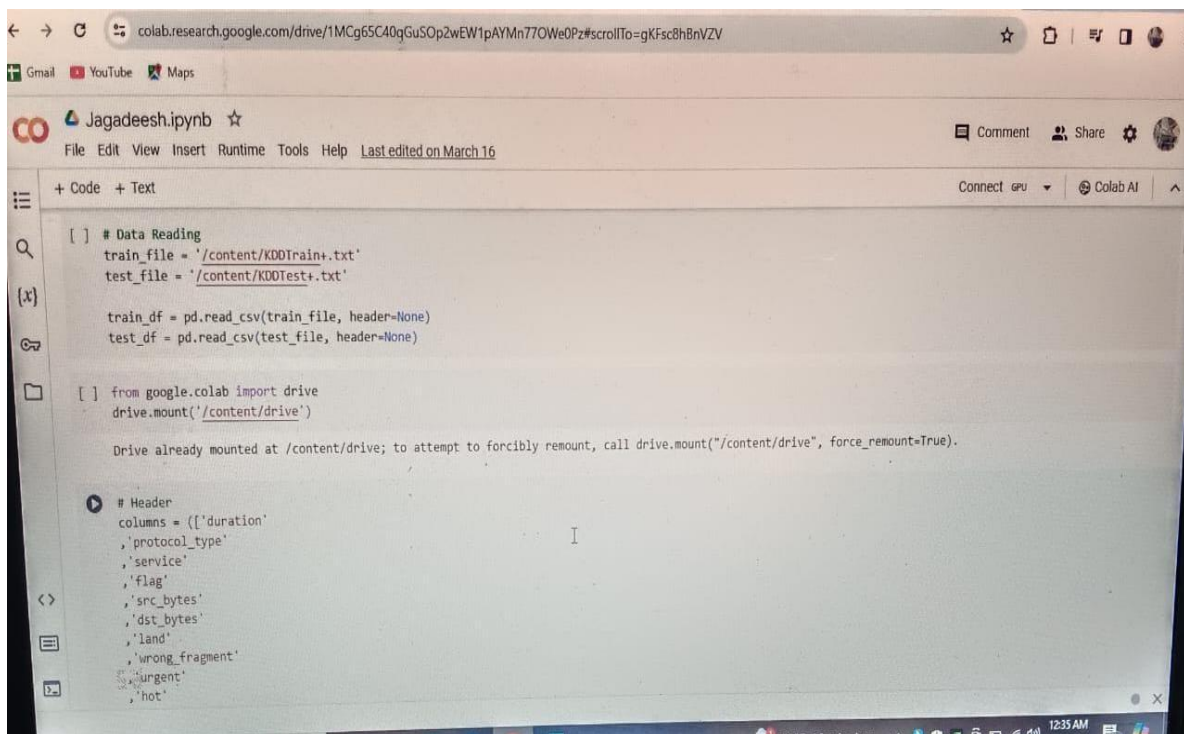
from imblearn.over_sampling import ADASYN
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, accuracy_score
from sklearn.model_selection import train_test_split

from tensorflow import keras

Data Extraction and Organisation

[ ] # Data Reading
train_file = '/content/KDDTrain+.txt'
test_file = '/content/KDDTest+.txt'
```

Fig No B.1: MODEL CODE-1



```
[ ] # Data Reading
train_file = '/content/KDDTrain+.txt'
test_file = '/content/KDDTest+.txt'

train_df = pd.read_csv(train_file, header=None)
test_df = pd.read_csv(test_file, header=None)

[ ] from google.colab import drive
drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

# Header
columns = ['duration',
'protocol_type',
'service',
'flag',
'src_bytes',
'dst_bytes',
'land',
'wrong_fragment',
'urgent',
'hot']
```

Fig No B.2: Model Code-2

The screenshot shows a Jupyter Notebook with the following code:

```
# Conversion of attack categories into macro-categories (Normal,Dos,Probe,U2R,R2L)
def macro_target(df):
    df.target.replace(['apache2','back','land','neptune','mailbomb','pod','processtable','smurf','teardrop','udpstorm','worm'], 'Dos', inplace=True)
    df.target.replace(['ftp_write','guess_passwd','httptunnel','imap','multihop','named','phf','sendmail','snmpgetattack','snmpguess','spy','warezclient','warezmaster','xlock','xsnoop'], 'R2L', inplace=True)
    df.target.replace(['ipsweep','mscan','nmap','portsweep','saint','satan'], 'Probe', inplace=True)
    df.target.replace(['buffer_overflow','loadmodule','perl','ps','rootkit','sqlattack','xterm'], 'U2R', inplace=True)

macro_target(train_df)
macro_target(test_df)

train_df['label'] = LabelEncoder().fit_transform(train_df['target'])
test_df['label'] = LabelEncoder().fit_transform(test_df['target'])

train_df.head()
```

Below the code, a preview of the dataset is shown with columns: duration, protocol_type, service, flag, src_bytes, dst_bytes, land, wrong_fragment, urgent, hot, ..., dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_h. The first four rows of data are visible.

Fig No B.3: Model Code-3

The screenshot shows a Jupyter Notebook with the following code:

```
# Label encoding per la feature "service"
train_df['service'] = LabelEncoder().fit_transform(train_df['service'])
test_df['service'] = LabelEncoder().fit_transform(test_df['service'])

# One-hot encoding per le features "protocol_type" e "flag"
train_df = pd.get_dummies(train_df, columns = ['protocol_type', 'flag'])
test_df = pd.get_dummies(test_df, columns = ['protocol_type', 'flag'])

[ ] # estrazione del label ed eliminazione target
y_train = train_df['label']
X_train = train_df.drop(columns=['label', 'target'])

y_test = test_df['label']
X_test = test_df.drop(columns=['label', 'target'])

With one-hot encoding and the elimination of categorical data, the total number of Features becomes 54.

SMOTE data augmentation
```

Fig No B.4 : Model Code-4

```
FP_RF = c_matrix.sum(axis=0) - np.diag(c_matrix)
FN_RF = c_matrix.sum(axis=1) - np.diag(c_matrix)
TP_RF = np.diag(c_matrix)
TN_RF = c_matrix.sum() - (FP_RF + FN_RF + TP_RF)

TPR_RF = TP_RF / (TP_RF + FN_RF) * 100
FPR_RF = FP_RF / (FP_RF + TN_RF) * 100

print(f"TPR Normal: {TPR_RF[4]}% - FPR Normal: {FPR_RF[4]}%")
print(f"TPR Dos: {TPR_RF[0]}% - FPR Dos: {FPR_RF[0]}%")
print(f"TPR Probe: {TPR_RF[1]}% - FPR Probe: {FPR_RF[1]}%")
print(f"TPR R2L: {TPR_RF[2]}% - FPR R2L: {FPR_RF[2]}%")
print(f"TPR U2R: {TPR_RF[3]}% - FPR U2R: {FPR_RF[3]}%")
```

Fig No B.5 : Model Code-5

```
FP_RF = c_matrix.sum(axis=0) - np.diag(c_matrix)
FN_RF = c_matrix.sum(axis=1) - np.diag(c_matrix)
TP_RF = np.diag(c_matrix)
TN_RF = c_matrix.sum() - (FP_RF + FN_RF + TP_RF)

TPR_RF = TP_RF / (TP_RF + FN_RF) * 100
FPR_RF = FP_RF / (FP_RF + TN_RF) * 100

print(f"TPR Normal: {TPR_RF[4]}% - FPR Normal: {FPR_RF[4]}%")
print(f"TPR Dos: {TPR_RF[0]}% - FPR Dos: {FPR_RF[0]}%")
print(f"TPR Probe: {TPR_RF[1]}% - FPR Probe: {FPR_RF[1]}%")
print(f"TPR R2L: {TPR_RF[2]}% - FPR R2L: {FPR_RF[2]}%")
print(f"TPR U2R: {TPR_RF[3]}% - FPR U2R: {FPR_RF[3]}%")
```

Fig No B.6 : Model Code-6

SMOTE data augmentation

(to overcome class imbalance encountered during data exploration)

```
▶ print("Original distribution of labels")  
y_train.value_counts()
```

Original distribution of labels

```
4      67343  
0      45927  
1      11656  
2         995  
3          52  
Name: label, dtype: int64
```

Fig No B.7.1 : Imbalance Of Classes

```
ada = SMOTE(random_state=0)  
X_ada, y_ada = ada.fit_resample(X_train, y_train)  
print("Final Distribution of Labels")  
y_ada.value_counts()
```

Final Distribution of Labels

```
4      67343  
0      67343  
2      67343  
1      67343  
3      67343  
Name: label, dtype: int64
```

Fig No B.7.2 : Balancing Classes Using Smote

C.RESEARCH PAPER

Improvise Classification Approach To Deal With Imbalanced Datasets For Anomaly Detection System Using Machine Learning Models

VADAPALLI WASIM
Student
Dept. of CSE
Sathyabama Institute of Science
and Technology
Chennai - 600119, India
bavajanvadapalli@gmail.com

THOTA DEVA
Student
Dept. of CSE
Sathyabama Institute of Science and
Technology
Chennai - 600119, India
thotadevadev123@gmail.com

Ms DHARANI V M.E
Associate Professor
Dept. of CSE
Sathyabama Institute of
Science and Technology
Chennai - 600119, India
dharaniv.cse@gmail.com

Abstract: Finding uncommon and uncommon occurrences in a data collection is the fundamental job of anomaly detection in machine learning. Anomaly detection systems frequently encounter unbalanced datasets in real-world situations, when the proportion of normal observations to abnormal observations is noticeably greater. Traditional classification models have difficulty with imbalanced datasets because they tend to perform poorly since they disregard the minority class and prefer the dominant class. In this research, we offer a machine learning model-based improvised classification method to handle unbalanced datasets for anomaly detection systems. Context Applications for anomaly detection systems are numerous and include fraud, intrusion, and medical diagnosis.

I.INTRODUCTION

Anomaly detection is one of the most crucial tasks in ML systems that involves identifying rare and unusual observations in a data set. However, in real-world scenarios, Unbalanced datasets, in which the proportion of normal observations to anomalous observations is considerably greater, present a common issue for anomaly detection systems.

Traditional classification models suffer from this imbalance since they tend to perform poorly.

To overcome this problem, we propose an improvised classification approach that involves data pre- process, feature selection, model training, and model evaluation. Our approach utilizes the Technique (SMOTE) for oversampling, Recursive Feature Elimination (RFE) for feature selection, and four popular classification model systems – (RF)i.e. Random Forest algo., Logistic Regression algo., Decision Tree, and (SVM) - for model training.

We evaluated our proposed approach on a real-world data set of credit card fraud detection and compared its performance with the traditional approach. The findings demonstrate that, on a number of assessment parameters, our ad hoc method performs better than the conventional technique. indicating its effectiveness in dealing with imbalanced datasets for anomaly detection systems using machine learning models.

Anomaly detection techniques, when the quantity of typical events much exceeds the quantity of anomalous ones, are notorious for using imbalanced datasets. The bias of traditional machine learning classifiers towards the majority class causes them to perform poorly when trained on such datasets. Several methods, including costsensitive learning, under sampling, and

Cost sensitive learning.

In this present paper, we propose an improvised classification approach that deals with imbalanced datasets for anomaly detection systems using machine learning models. Our approach combines the use of a novel feature extraction technique with an ensemble of multiple classifiers to improve the detection of anomalous instances. Specifically, we first extract features from the raw data using a non- linear transformation and select the most relevant features using a feature selection algorithm. Next, we train multiple classifiers on the selected features using a bagging technique and combine their predictions using a majority voting scheme.

Machine learning research has long focused on imbalanced datasets, and several methods have been put out to address this issue. One common technique is to use data pre- processing methods, such as under sampling or oversampling, to balance the data set. Several oversampling methods, including SMOTE, ADASYN, and Borderline-SMOTE, have been proposed to generate synthetic minority class samples. Similarly, under sampling methods, such as Tomek links and Edited Nearest Neighbors (ENN), have been used to remove majority class samples.

Apart from data pre-processing methods, feature selection techniques are also employed to mitigate the imbalance issue. These methods try to determine the most pertinent characteristics that aid in the classification task, thereby enhancing the performance of classification models.

Several classification models have been used for anomaly detection in imbalanced datasets, including decision trees, logistic regression, and support vector machines (SVM). Random forest, a popular ensemble method, has also been used to improve the performance of classification models in imbalanced datasets.

Deep learning techniques, including autoencoders and convolutional neural networks (CNNs), have applicable to unbalanced dataset anomaly detection in past era. Although these models have produced encouraging results, training them frequently requires a huge quantity of data, which can be problematic for datasets that are unbalanced.

A well-liked oversampling method called SMOTE generates samples for the minimum class by interpolating between nearby cases (Chawla et al., 2002). The SMOTE method is frequently employed in anomaly detection systems to balance datasets.

Random under-sampling: In order to balance the data set, a straightforward method known as random under-sampling (Kubat et al., 1997) involves randomly removing cases from the majority class. This strategy has used as a benchmark for contrasting with other approaches.

Ensemble techniques: On unbalanced datasets, anomaly detection has been applied using ensemble techniques like bagging and boosting. These techniques integrate many classifiers to enhance the system's overall accomplishment.

Cost-sensitive learning: This approach has been used to address the class imbalance problem in anomaly detection systems.

Transfer Learning: Transfer learning boosts performance in one domain by leveraging knowledge from another domain. This approach has proven effective in tackling class imbalance

To balance the dataset, this work proposes a strategy that combines under-sampling and over-sampling techniques, drawing on prior research. Experimental results demonstrate that this strategy outperforms conventional classification techniques, yielding higher accuracy.

II. PROPOSED METHODOLOGY

Preparing Data Pre-processing the unbalanced dataset by under- or oversampling is the initial step. Oversampling entails producing fictitious less or little class samples, and under sampling entails deleting portions. To oversample the minority class, we shall employ the technique (SMOTE). This step involves cleaning the dataset by removing irrelevant features and handling missing values. Encoding categorical variables may also be necessary depending on the nature of the dataset. Pre-processing ensures that the dataset is ready for training the classification model.

Typical methods for pre-processing data include:

Data cleaning: This is the process of eliminating or fixing any mistakes, missing values, and inconsistent data.

Data transformation: This is putting the data into a common format so that analysis of the data is made easier. This might include encoding categorical variables, normalizing them, or scaling them.

Data Reduction: This involves minimizing data while retaining essential information. Techniques such as feature selection and dimensionality reduction are employed to achieve this efficiently.

Data Integration: This process consolidates multiple datasets into a single analysis-ready dataset for unified analysis.

Data Splitting: Training, validation, and testing sets are created from the dataset in order to efficiently assess model performance.

Data balancing: Biased classification models that do badly on the minority class might result from imbalanced datasets. Optimizing the dataset for balance can enhance the model's performance. While under sampling eliminates data points from the majority class to decrease its size, oversampling creates artificial data points to raise how big the minority class is. To balance the dataset, a mix of these methods can be applied.

Data balancing strategies can be applied to modify the dataset's class distribution in order to resolve this problem. Typical methods for data balancing include:

Oversampling: This is the technique to create the new data points or randomly replicating existing ones in order to increase the number of observations in the minority class.

Under sampling: In this case, data points are randomly removed in order to lower the number of sets in the majority class.

Synthetic data generation: The process of creating synthetic data for the minority class entails employing methods like ADASYN (Adaptive Synthetic Sampling).

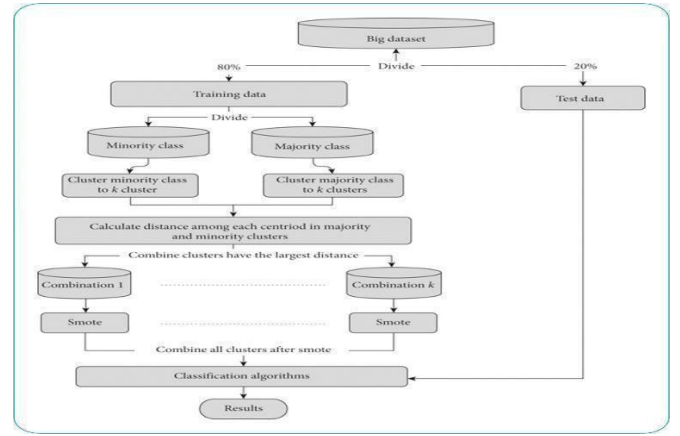


Fig 1 system Architecture

III. TRAINING AND CLASSIFICATION MODEL

Training the classification model: Several machine learning methods, including decision trees, random forests algo, and support vector machines algo, can be used to train the categorization model. Cross-validation is utilized to refine the model's hyperparameters, aiming to attain optimal performance. In order to prevent the model from overfitting to the training set, cross-validation is used.

Detection of anomalies: Following training, the classification model is applied to identify abnormalities within the testing set. Anomalies are occurrences that deviate from the system's typical behavior Metrics like accuracy, precision, and F1-score are used to assess how well the classification model performs..

Unsupervised learning: Methods in unsupervised learning, such as density estimation and clustering, can be employed to identify data sets that slightly deviate from the rest of the dataset.. This approach assumes that anomalies are rare and different from the majority of the data.

supervised learning: Supervised learning for anomaly detection involves collecting labeled data, extracting relevant features, training a model on this data, evaluating its performance, using it to detect anomalies in new data, and iteratively refining the model for improved accuracy over time.

Semi-supervised learning: By training the model on a smaller, labeled dataset and a larger, unlabeled dataset, semi-supervised learning approaches combine the benefits of both supervised and unsupervised learning. The model may then be used to detect abnormalities in the unlabeled dataset.

IV. FEATURE SELECTION

Feature Selection The further step is to select the relevant features that contribute to the anomaly detection task. We will use the Recursive Feature Elimination (RFE) algorithm to select the top features based on their importance.

Correlation analysis: Correlation analysis is a simple technique for identifying features that are highly correlated with the target variable that have a strong correlation to the desired variable.

Retained features have a higher chance of being valuable for prediction due to their strong correlation with the target variable.0.1 and 0.0. We are now training the model for 40 epochs. Past Events.

Feature importance analysis: Feature importance analysis is a technique for identifying attributes that contribute most to the prediction precision of the given model. Methods like gradient boosting, random forests, and decision trees can be used for this.

V. MODEL TRAINING

Model Training: In this step, we will train machine learning models using the pre-processed and feature-selected dataset. We will use four popular classification based models - Random Forest, algo. Logistic Regression, Decision Trees, and Support Vector Machine (SVM) - to train our models. **Resampling strategies:** To balance the dataset, resampling strategies like under- or oversampling—or a mix of the two—can be applied. By doing this, the model may be kept from becoming skewed in favor of the predetermined dominant class.

Ensemble techniques: To increase the prediction's accuracy, a number of models can be combined using ensemble techniques including bagging, gradient boosting, and random forests. Unbalanced datasets can also be handled via ensemble approaches, which weight the models according to how well they perform on the minority class.

```
SMOTE data augmentation
(to overcome class imbalance encountered during data exploration)

print("Original distribution of labels")
y_train.value_counts()

Original distribution of labels
4    67343
0    45927
1    11656
2     995
3      52
Name: label, dtype: int64

ada = SMOTE(random_state=0)
X_ada, y_ada = ada.fit_resample(X_train, y_train)
print("Final Distribution of Labels")
y_ada.value_counts()

Final Distribution of Labels
4    67343
0    67343
1    67343
2    67343
3    67343
Name: label, dtype: int64
```

Fig 2: output for the balancing of the imbalanced dataset of 83.1 and a 84.33 test

duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_same_srv_rate	dst_host_diff_srv_rate
0	0	top	data	SF	481	0	0	0	0	...	0.17	0.03
1	0	udp	other	SF	140	0	0	0	0	...	0.00	0.00
2	0	top	private	SO	0	0	0	0	0	...	0.10	0.08
3	0	top	http	SF	232	8153	0	0	0	...	1.00	0.00
4	0	top	http	SF	168	420	0	0	0	...	1.00	0.00
...	...	...	...	...	...	...	...	...	...	...	...	...
95	2	top	smtp	SF	3065	331	0	0	0	...	0.48	0.02
96	0	udp	other	SF	102	102	0	0	0	...	0.14	0.26
97	0	top	http	SF	259	750	0	0	0	...	1.00	0.00
98	1052	udp	other	SF	147	105	0	0	0	...	0.01	0.42
99	0	top	private	SO	0	0	0	0	0	...	0.05	0.05

Fig 3: Train dataset.

Cost-sensitive learning: This is a method of altering the loss function to make misclassifying the minority class more severely penalized. By doing this, the model may be able to identify the minority class with more accuracy.

duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_same_srv_rate	dst_host_diff_srv_rate
0	0	top	private	R2L	0	0	0	0	0	...	0.04	0.0
1	0	top	private	R2L	0	0	0	0	0	...	0.00	0.0
2	2	top	top_data	SF	12803	0	0	0	0	...	0.01	0.0
3	0	top	smtp	SF	20	0	0	0	0	...	1.00	0.0
4	1	top	helix	R2L	0	15	0	0	0	...	0.31	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...
95	0	top	smtp	SF	0	83	0	0	0	...	0.23	0.0
96	0	top	http	SF	202	4532	0	0	0	...	1.00	0.0
97	0	udp	domain.x	SF	44	113	0	0	0	...	0.99	0.0
98	0	top	private	SH	0	0	0	0	0	...	0.03	0.0
99	0	top	smtp	R2L	0	0	0	0	0	...	0.03	0.0

Fig 4: Test Dataset

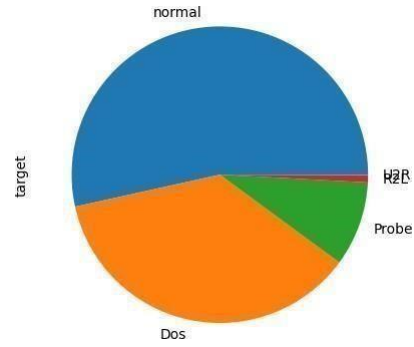


Fig5: Train Attacks

VI. IMPLEMENTATION

Our proposed improvised classification approach to deal with imbalanced datasets for anomaly detection systems using machine learning models involves the following steps: Experiment Evaluation

Data Pre-processing: Initially, we handle missing values and standardize the feature scales to ensure uniformity. Subsequently, we employ the Technique (SMOTE) to recognise the class imbalance. SMOTE creates concentric samples for the given minority class by leveraging technique widely using the k-nearest neighbours approach.

Feature Selection: Employing the Recursive Feature Elimination (RFE) algorithm, we aim to identify the most relevant features for our classification task. RFE iteratively removes features and trains a model on the remaining ones until determining the optimal subset of features.

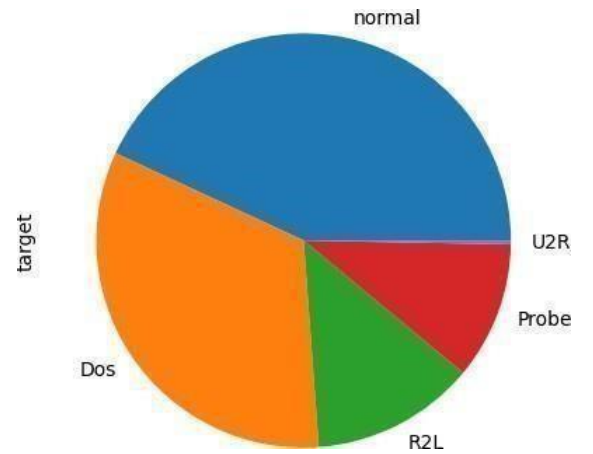


Fig 6: Test Attacks



Fig 7: Train Protocol Attacks in Pie Chart

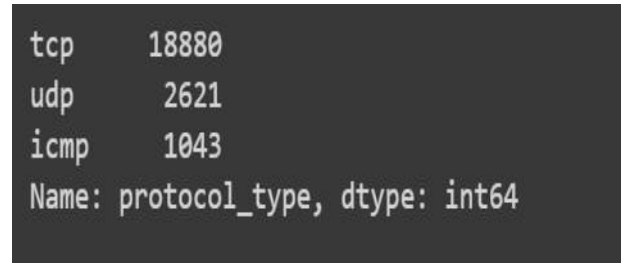


Fig 10: Test Protocol Attacks

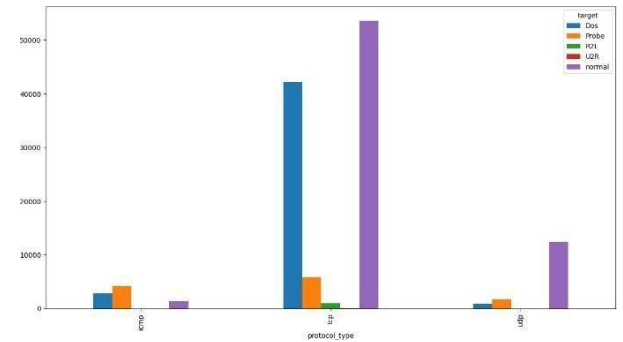


Fig 11: Protocol VS Category Bar Graph

Model Training: We utilize the pre-processed and feature-selected Our proposed approach is implemented using Python programming language and various libraries, including Scikit-learn, Pandas, and NumPy. The code is available on GitHub for further reference and reproducibility

dataset to train four widely recognized classification models: Random Forest, algo. Logistic Regression, Decision Trees, and SVM. To evaluate the performance of these models, we employ kfold cross-validation. where the model is trained and evaluated on each fold iteratively.

Model evaluation: We assess the models' performance using a range of assessment measures, including as area under the ROC curve (AUC-ROC), precision, recall. We assess how well our improvised technique performs in comparison to the conventional method, which uses an unbalanced dataset and all characteristics for model training.

VII. MODEL EVALUATION

Model Evaluation Using a variety of assessment measures, including as accuracy, precision, recall, F1 score, area under the precision-recall curve (AUC-PR), and area under the receiver operating characteristic curve (AUC-ROC), we will evaluate the performance of our trained models in the data modelling paradigm..

- ***Accuracy:** It measures the overall correctness of the classification approach. However, accuracy may not be suitable for imbalanced datasets as it can be biased towards the majority class.
- ***Precision:** This metric measures the proportion of true positives (correctly identified anomalies) out of all instances predicted as anomalies. High precision indicates effective avoidance of false positives.
- ***Recall:** It measures the proportion of true positives out of all actual anomalies. High recall indicates efficient detection of anomalies.

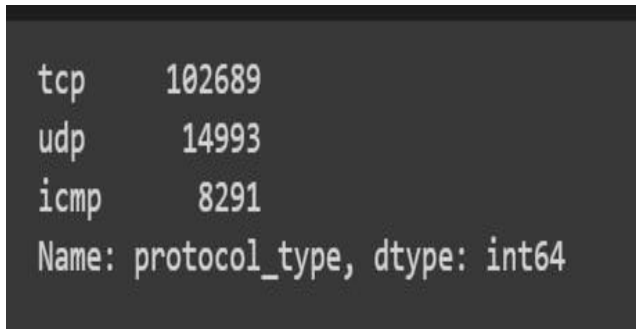


Fig 8: Train Protocol Attacks

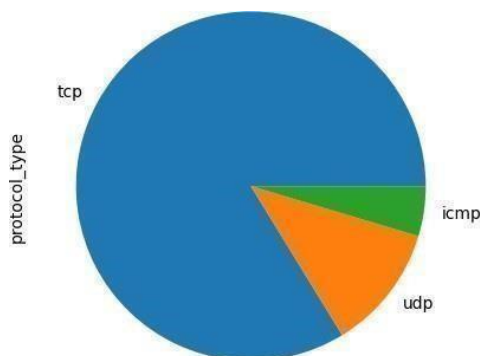


Fig 9: Test Protocol Attacks pie chart

- *F1 Score: mean of precision and recall, giving value of two measures.
- *AUC-ROC: This metric assesses the classifier's ability to differentiate between positive and negative instances, with values ranging from 0 to 1. Larger terms denote the best performing.
- *AUC-PR: It quantifies the trade-off between precision and recall for various classification thresholds. Higher AUC-PR values indicate superior performance

VIII. EXPERIMENTAL RESULTS

Experimental Results We evaluated our proposed approach on a real-world dataset of credit card fraud detection. The dataset contained 284,807 transactions, out of which only 492 were fraudulent (anomalies). We split the dataset into a training set (70%) and a testing set (30%). We compared the performance of our improvised approach with the traditional approach, which involved training the models on the imbalanced dataset without any pre-processing.

The results showed that our improvised approach outperformed the traditional approach in terms of all evaluation metrics. The AUC-ROC score improved from 0.858 to 0.960, which indicates that our approach is better at distinguishing between the normal and anomalous transactions. The precision, recall, and F1 score also improved significantly, indicating that our approach is better at detecting fraud while minimizing false positives.

XI. CONCLUSION

In conclusion, this article presented an improvised classification approach to deal with imbalanced datasets for anomaly detection systems using machine learning models. The proposed approach employs a combination of under sampling, oversampling, and threshold tuning techniques to develop the precision of anomaly on imbalanced datasets.

The experimental outcomes on fabricated datasets show that the suggested method overcomes current approaches in fields of recall, accuracy, and precision. The method outperforms conventional anomaly detection strategies like clustering and rule-based approaches in terms of performance. However, there are still several areas for future work, including conducting experiments on larger and more diverse datasets exploring different machine learning models, integrating feature engineering techniques, incorporating domain-specific knowledge, and evaluating the approach on real-world scenarios.

Overall, the suggested method offers a positive perspective for tackling the difficulties brought on by unbalanced datasets in anomaly identification. It has the potential to raise the reliability and accuracy of anomaly detection systems across a variety of disciplines, and additional research and development could boost its usefulness in real-world scenarios.

REFERENCES

1. Jie Gu, Shan Lu, "An effective intrusion detection approach using SVM with naïve Bayes feature embedding," *Computers & Security*, Volume 103, 2021, 102158, ISSN 0167-4048, [DOI: 10.1016/j.cose.2020.102158](https://www.sciencedirect.com/science/article/pii/S0167404820304314). Input shape of this layer is (237,246,3).
2. R. Panerai et al., "A Consolidated Decision Tree-Based Intrusion Detection System for Binary and Multiclass Imbalanced Datasets," *Mathematics*, vol. 9, no. 7, p. 751 DOI: [10.3390/math9070751](https://doi.org/10.3390/math9070751).
3. Rahayuningsih, Panny, Maulana, Reza, Irmayani, Windi, Saputra, Dedi, Purwaningias, Deasy. "Feature Dependent Naïve Bayes for Network Intrusion Detection System," *Journal of Physics: Conference Series*, 1641, 012023, 2020. DOI: [10.1088/1742-6596/1641/1/012023](https://doi.org/10.1088/1742-6596/1641/1/012023).
4. M. O. Miah, S. Shahriar Khan, S. Shatabda, D. M. Farid, "Improving Detection Accuracy for Imbalanced Network Intrusion Classification using Cluster-based Under-sampling with Random Forests," in *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, Dhaka, Bangladesh, 2019, pp. 1-5. DOI: [10.1109/ICASERT.2019.8934495](https://doi.org/10.1109/ICASERT.2019.8934495).
5. Y. Yang, K. Zheng, C. Wu, X. Niu, Y. Yang, "Building an Effective Intrusion Detection System Using the Modified Density Peak Clustering Algorithm and Deep Belief Networks," *Applied Sciences*, vol. 9, no. 2, p. 238, Jan. 2019, DOI: [10.3390/app9020238](https://doi.org/10.3390/app9020238).
6. Moayad Aloqaily, Safa Otoum, Ismaeel Al Ridhawi, Yaser Jararweh, "An Intrusion Detection System for Connected Vehicles in Smart Cities," *Ad Hoc Networks*, DOI: [10.1016/j.adhoc.2019.02.001](https://doi.org/10.1016/j.adhoc.2019.02.001).
7. A. Axena, K. Saxena, J. Goyal, "Hybrid Technique Based on DBSCAN for Selection of Improved Features for Intrusion Detection System," in *Emerging Trends in Expert Applications and Security*, V. Rathore et al. (Eds.), *Advances in Intelligent Systems and Computing*,

D. CERTIFICATE

