NETWORK ANOMALY DETECTION USING BORDERLINE SMOTE ALGORITHM AND SUPPORT VECTOR MACHINES

BY

DINESH M	**71382006012**
SABARISH C S	**71382006040**
YOGESHWARAN S	**71382006048**

Report submitted in partial fulfilment of the requirements for the degree of Bachelor of Technology in Information Technology

SRI RAMAKRISHNA INSTITUE OF TECHNOLOGY

Coimbatore – 641010

May 2024

SRI RAMAKRISHNA INSTITUTE OF TECHNOLOGY

PACHAPALAYAM - PERUR CHETTIPALAYAM

COIMBATORE -10

Department of Information Technology

BONAFIDE CERTIFICATE

Certified that the project titled **NETWORK ANOMALY DETECTION USING BORDERLINE SMOTE ALGORITHM AND SUPPORT VECTOR MACHINES** is the Bonafide work done by **DINESH M, SABARISH C S, YOGESHWARAN S** in the **FINAL YEAR (PHASE - II) PROJECT** of this institution, as prescribed by Sri Ramakrishna Institute of Technology for the EIGHTH Semester / B. Tech, Programme during the year 2023- 2024.

Supervisor Head of the Department

Submitted for the Project Viva - voce held on …….

Internal Examiner External Examiner

APPROVAL AND DECLARATION

This project report titled **NETWORK ANOMALY DETECTION USING BORDERLINE SMOTE ALGORITHM AND SUPPORT VECTOR MACHINES** was prepared and submitted by **DINESH M (71382006012), SABARISH C S (71382006040), YOGESHWARAN S (71382006048)** and has been found satisfactory in terms of scope, quality, and presentation as partial fulfilment of the requirement for the Bachelor of Technology (Information Technology) in Sri Ramakrishna Institute of Technology, Coimbatore.

Checked and approved by

_____

Dr. J. J. Adri Jovin
Associate Professor / IT
Project Supervisor

Department of Information Technology
Sri Ramakrishna Institute of Technology, Coimbatore-10
May 2024

ACKNOWLEDGEMENT

First, we record our grateful thanks to the almighty for his blessings to make this project a grand success.

We express our sincere thanks to our respected and honourable Principal **Dr. M. Paulraj,** for granting us permission to undergo the project.

We record our heartfelt gratitude to **Dr. M. Suresh Kumar**, Professor and Head (in-charge), Department of Computer Science and Engineering and **Dr. J. J. Adri Jovin**, Associate Professor and Head (in-charge), Department of Information Technology for rendering timely help for the successful completion of this project.

We thank our Project Coordinator **Dr. T. C. Ezhil Selvan,** Associate Professor, Department of Information Technology, Sri Ramakrishna Institute of Technology for his guidance throughout the entire period for the completion of the project.

We are greatly indebted to our Project Supervisor, **Dr. J. J. Adri Jovin, Ph.D.,** Associate Professor, Department of Information Technology, Sri Ramakrishna Institute of Technology for his inspirational guidance, valuable suggestions and providing us a chance to express our technical capabilities in every respect for the completion of the project work.

We thank our parents and friends for the strong support and inspiration they have provided us in bringing out this project successfully.

ABSTRACT

Network Security is a major challenge in the digital world. Intrusion is common in many applications and intruders are sophisticated enough to change their attack pattern very often. To address this issue, the development of a model for the detection of network anomalies and intrusions. The approach utilizes the Borderline Synthetic Minority Over-Sampling Technique (SMOTE) along with Support Vector Machines (SVM) to enhance anomaly detection capabilities. By intelligently oversampling the minority class using SMOTE and training SVM, the proposed model exhibits a robust defence mechanism against network intruders. The utilization of these advanced techniques aims to augment the accuracy and efficiency of anomaly detection, minimizing false positives and ensuring prompt response to genuine threats. The results obtained from this study add to the ongoing efforts to secure data in the digital age, by combining SMOTE and SVM for network intrusion detection.

TABLE OF CONTENT

LIST OF FIGURES

ABBREVIATIONS & ACRONYMS

ABBREVIATION	EXPANSION
ML	Machine Learning
SMOTE	Synthetic Minority Over-Sampling Technique
SVM	Support Vector Machines
CIC	Canadian Institute for Cybersecurity
IDS	Intrusion detection system
NIDS	Network intrusion detection systems
CIA	Confidentiality, Integrity, and Availability
SID	Signature-Based Intrusion Detection
AID	Anomaly-Based Intrusion Detection

CHAPTER – 1

INTRODUCTION

1.1 Overview

Digital transformation is taking place in most sectors around the world. A drastic effort of networking systems globally has given the advantage of unlimited access to data and knowledge. With the growth in digital systems the growth of users with a malicious intent has also started to grow. Early intrusions have been made in the late 1970s and early 1980s. This was a period when most systems were not networked with one another, and internet was in a rudimentary state. Hence, the impact of any such intrusions cost only a few individuals and to the worst case, an organisation. As the years passed, more organisations entered the digital domain and started doing business online. The malicious users also started experimenting with sophisticated tools for intruding into systems over which business is done. Over the years, the pattern of intrusion has also changed. Some users with a malicious intent still try the classical intrusion techniques, which are usually detected by intrusion detection systems. These classical methods follow a specific signature and hence can be detected by a signature-based intrusion detection system. However, advanced malicious users use different patterns of intrusion which are least detected by signature-based intrusion detection systems. Such patterns can only be detected using an anomaly detection system. These systems harness the power of artificial intelligence to perform anomaly detection. In the recent days, machine learning is applied in a higher level to detect such intrusions[1][2].

1.2 Network Security.

In the modern digital era, with the increasing number of networked devices, including critical systems like power networks, increases the opportunity for malicious intrusions and increases the possible consequences of a successful breach. The increasing strain on current security monitoring systems surpasses their inherent capacities, presenting substantial hurdles for their immediate threat identification abilities. As a result, the current cybersecurity competition leans toward favouring attackers, highlighting the ongoing need for persistent contributions from the research sphere to

fortify existing defences and introduce pioneering methods aimed at mitigating these risks.

It heavily relies on the internet and technology introduces a critical need to safeguard sensitive data and personal information held by service providers. Network intrusions send out a global "red alert" to both service providers and consumers, as they compromise user data and services, incurring substantial financial costs. To combat these network intrusions and their adverse impacts on consumers, Intrusion Detection Systems (IDS) play a crucial role. Figure 1.1 shows the architecture of an IDS.

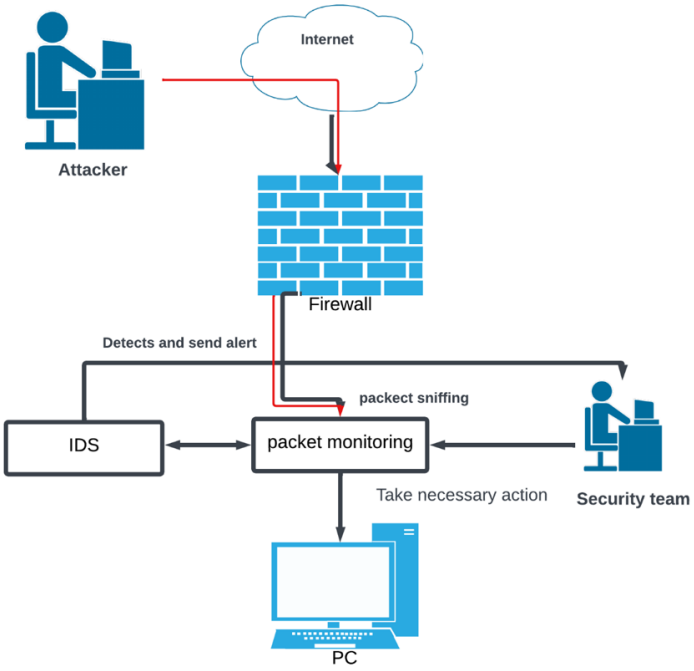

Figure 1.1 Intrusion Detection systems

The automation and precision of intrusion/attack detection are pivotal for an effective IDS. Therefore, leveraging machine learning techniques becomes highly desirable in achieving automated and accurate attack detection in this digital age.

1.3 Network Intrusion

Intrusion is breaking and entering in an unauthorized manner to access something sensitive that poses a threat to a personal property. In terms of network security, intrusion means unauthorized entry into the network area. It involves accessing network computers and stealing important information and resources, by compromising the network security. The intrusion can be both passive and active. Passive means illegal and secret access without being noticed. Active intrusion impacts network resources and bandwidth. As discussed in the overview, the technology used in intrusion detection is divided into two types: signature-based intrusion detection and anomaly-based intrusion detection.

1.4 Machine Learning

Machine learning is revolutionizing network security by improving network intrusion detection systems (NIDS). The network intrusion detection system is used to monitor and analyse network traffic to detect and respond to unauthorized activity. Traditionally, rule-based, or signature-based methods struggled to keep up with the evolving cyber threat landscape. Machine learning enables NIDS to adapt and learn from historical network data, improving their ability to detect anomalies in real-time, identifying zero-day attacks, and analysing user and network device behaviour over time and new threats. It aids in anomaly detection, behaviour analysis, reduced false positives, rapid response, scalability, and improved threat intelligence. Various algorithms and techniques can be employed for NIDS, including supervised learning for classification, unsupervised learning for anomaly detection, and deep learning for complex pattern recognition. Open-source machine learning frameworks and tools make it more accessible for organizations to deploy machine learning-based NIDS solutions.

CHAPTER-2

LITERATURE SURVEY

Sun et al. meticulously designed a Borderline SMOTE Algorithm and Feature Selection-Based Network Anomalies Detection Strategy marking a significant advancement in the domain of network security[3]. The research introduces a groundbreaking framework for network anomaly detection that masterfully addresses the intricate task of classifying diverse network intrusions. The approach gracefully incorporates a resampling strategy, blending data from Borderline SMOTE[4] and random sampling to create a harmonious balance for the dataset. The essence of the technique is the use of feature selection, which is largely based on information acquisition rate. To ensure the utmost effectiveness, the researchers conducted a series of rigorous experiments employing three popular machine learning algorithms, namely the K-Nearest Neighbour, Decision Tree, and Random Forest. The results of these experiments unequivocally point towards an optimal feature selection scheme, thereby overcoming the perennial challenge of data imbalance in network intrusion detection datasets.

Shadman Latif et al. conducted a thorough exploration using the NSL-KDD dataset [5]. The research mainly concentrated on using different machine learning approaches such as AdaBoost, Decision Tree, Support Vector Machine, Naïve Bayes, Random Forest, and Neural Networks. The researchers employed a comprehensive approach to data preprocessing, encompassing the conversion of categorical features into numeric representations using one-hot encoding. Additionally, they experimented with diverse feature scaling techniques and excluded Naïve Bayes, due to its subpar performance. The novel part of the study was the deployment of several sampling strategies to produce machine learning pairings that were scaled, feature-reduced, and over-sampled.

Sharafaldin et al. created a fresh dataset for Intrusion detection and traffic characterization [6]. The initiative aims to address the problems that the existing datasets have, including inadequate feature sets, limited attack kinds, anonymised packet information, and a lack of diversity in traffic. The dataset includes a mix of benign and prevalent attack network flows, presenting an exhaustive set of scenarios for assessment. In addition, the researchers have thoroughly assessed the machine learning methods and network traffic aspects. This comprehensive analysis aims to empower the community with a robust tool for detecting and categorizing a spectrum of attacks, thereby augmenting network security against growing threats.

Miel Verkerken reviewed and presented a novel multi-stage approach for hierarchical intrusion detection [2][7]. In the era of digital transformation, there is a demand for robust intrusion detection systems which identifies previously unknown, zero-day attacks. The study offers a multi-phase hierarchical intrusion detection system. The method is validated using two publicly available, well-known datasets CIC-IDS-2017 and CSE-CIC-IDS-2018. The methodology achieves an outstanding performance in classification with 96% balanced accuracy, outperforming both baseline methods and existing approaches. The flexibility of this method is that it does not require retraining and it uses n-tier installations to reduce bandwidth and compute costs while adhering to strict privacy limitations, is especially remarkable. The top-performing model in the study reduced bandwidth demands by as much as 69% while accurately categorizing 41 out of 47 zero-day assaults with an astonishing 87% accuracy. This research presents a significant leap forward in the field of intrusion detection, offering an efficient and powerful tool to enhance network security.

Garcia Teodoro et al. conducted a comprehensive review on anomaly-based Network Intrusion Detection [8]. The increasing proliferation of security threats across the internet and computer networks emphasizes the necessity for the evolution of adaptable and flexible security approaches. Anomaly-based network intrusion detection solutions are useful in this situation to protect target systems from malicious activity. Moreover, they emphasized the existence of accessible platforms, continuing efforts to construct systems, and research initiatives in this field. The major contribution is to outline the key obstacles to the widespread utilization of anomaly-based systems for intrusion detection, with a focus on the evaluation of these technologies. It offers insights into the landscape of anomaly-based intrusion detection, which is crucial for enhancing network security.

Many literatures that present a comprehensive review on various security threats and classification of malware attacks, vulnerabilities, and detection techniques are available [9][10][11][12][13]. The works delve into the growing concerns surrounding security threats in the context of both wired and wireless networks. They emphasize the disruptive impact of malicious behaviour exhibited by nodes under attack on network operations. To counter such malevolent behaviours, the works outlines multitude of security resolutions that have been developed. It specifies the role of malware in security threats concerning both computers and the internet. The studies provide an elaborate vision of many types of malwares, vulnerabilities, and the current defences against these security threats.

2.1 PROJECT CHARTER

1. General Project Information	
Project Name:	**NETWORK ANOMALY DETECTION USING BORDERLINE SMOTE ALGORITHM AND SUPPORT VECTOR MACHINES**
Executive Sponsors:	-
Department Sponsor:	-
Impact of project:	This project aims to significantly improve network security practices, contributing to ongoing efforts to protect corporate data in today's digital and interconnected world.

2. Project Team

	Name	Department	Telephone	E-mail
Project Manager:	**ADRI JOVIN J J**	IT	9994797284	adrijovin.it@srit.org
Team Members:	**DINESH M**	IT	9789453361	dineshm.2006@srit.org
	SABARISH C S	IT	9095957717	sabarish.2006@srit.org
	YOGESHWARAN S	IT	9791773088	yogeshwaran.2006@srit.org

3. Stakeholders

Corporate Management

IT and Security Teams

End-Users and Data Owners

4. Project Scope Statement

Project Purpose / Business Justification

To improve network security by implementing advanced anomaly detection.

To protect sensitive data from intrusions, thereby justifying its importance in safeguarding invaluable corporate assets and data in the digital age.

Objectives (in business terms)

Strengthening Data Security.

Minimizing False Alarms.

Cost-Efficiency.

Competitive Advantage.

Deliverables

Anomaly Detection Model
Documentation
Testing and Validation Protocols

Scope: This project addresses the need for safeguarding of sensitive data from network intruders. The expected outcome is to improved the accuracy of detecting anomaly intrusion and to reduce False Positives(alert

Project Milestones

Phases	Start date (dd-mm-yyyy)	End date (dd-mm-yyyy)
Project analysis and planning	07-08-2023	17-08-2023
Gathering Required data	21-08-2023	04-09-2023
Intrusion ML Model development	07-09-2023	25-10-2023
Implementation of Backend	05-11-2023	01-12-2023
Implementation of frontend	20-01-2024	19-02-2024
Connecting Backend with frontend	20-02-2024	27-02-2024
Testing and Documentation	01-03-2024	23-03-2024

Major Known Risks (including significant Assumptions)

Risk	Risk Rating (High, Medium, Low)
The team is not well versed with networking, network security and machine learning. This could lead to schedule and budget overruns	Medium

Constraints

Time	
Budget	
Quality	The project should be of high quality with assured data integrity and security.
Scope	The scope of the project shall be as defined previously in this document under section titled “scope” and no more.

External Dependencies

The project will require to access to network data and logs for anomaly detection, third-party software, libraries, or tools for implementing SMOTE, SVM.

These dependencies must be effectively managed to ensure the effective implementation of the anomaly detection system and enhance network security.

5. Communication Strategy

The Project Manager will communicate with team members every week about the project. The project manager and/or the team members will communicate with the other stakeholders once every two weeks or after the completion of a milestone whichever is later. The project manager will provide a status report during every stakeholder meeting. Project team meetings will happen every two weeks. The agenda for every team meeting should include any milestones reached in the previous weeks, work progress report from every team member and a mini plan for the next two weeks.

6. Sign-off

	Name	Signature	Date (DD/MM/YYYY)
Executive Sponsor			
Department Sponsor			
Project Manager	**ADRI JOVIN J J**		

7. Notes

Any changes to be made to this document and subsequent project documents need to be approved by the change control board.

Any new features not listed in the SRS need to be approved by the change control board before being implemented.

Employees are encouraged to keep all project related communication restricted to the official channels established for the purpose and to keep all related communication transparent.

CHAPTER – 3

PROPOSED SYSTEM

3.1 Introduction

A main objective is to improve network security by implementing advanced anomaly detection and to protect sensitive data from intrusions, thereby justifying its importance in safeguarding invaluable corporate assets and data in the digital age the use borderline smote algorithm and support vector machines.

3.2 System Requirement

3.2.1 Hardware requirement.

- Processor (CPU): Intel core i5 or AMD Ryzen 5.
- Memory: 8 Gigabyte RAM
- Storage space: 128 Gigabyte SSD
- Internet Connectivity: Stable

3.2.2 Software requirement.

- An operating system for the hardware, such as Linux or Windows.
- Programming languages such as Python.
- Libraries and packages for machine learning, such as Pandas, NumPy, spark, matplotlib, seaborn, Borderline SMOTE.
- Integrated Development Environment (IDE) such as PyCharm or Visual Studio or Jupyter Notebook (anaconda3) for coding and debugging.
- Wireshark tool to capture the real time Data packets.
- Other requirements may include power supplies, cooling systems, and any necessary peripherals like displays or input devices

3.3 Datasets used

As an improvement on their previous NSL-KDD dataset which was initially drawn from the well-known KDD99 dataset Sharafaldin et al. (2018) created the new dataset called CIC-IDS-2017. The primary objective behind the creation of CIC-IDS-2017 was to adhere to a set of 11 specific criteria essential for a robust intrusion detection dataset, rendering it a valuable resource for benchmarking purposes. This dataset was meticulously constructed over a span of 5 days, employing 14 machines, and encompasses both legitimate and illegitimate network traffic.

B-profiles, which are created from the typical behavioural patterns of 25 people using statistical methods and machine learning, are used to artificially create the benign traffic present in the dataset. On the other hand, within predetermined timeframes, the execution of proven attack tools generates harmful traffic. Both kinds of traffic are combined into a single dataset, which is then made available in formats such as CSV files and packet captures (PCAP).

```
: print('First read version of the dataset: \n',data_value)

First read version of the dataset:
  Label
BENIGN              2273097
DoS Hulk             231073
PortScan             158930
DDoS                 128027
DoS GoldenEye         10293
FTP-Patator            7938
SSH-Patator            5897
DoS slowloris          5796
DoS Slowhttptest       5499
Bot                    1966
Brute Force            1507
XSS                     651
Infiltration             36
Sql Injection            21
Heartbleed               11
 XSS                      1
Name: count, dtype: int64

: print('First (row,Column) number of the dataset: {} '.format(data.shape))

First (row,Column) number of the dataset: (2830743, 79)
```

Figure 3.1 Attack Occurrences count

CHAPTER - 4

METHODOLOGY

The steps for the development of the method for intrusion detection is as follows:

- The necessary libraries are imported together with the CIC-IDS-2017 dataset.
- The dataset is pre-processed by eliminate any missing values.
- We choose to employ Borderline SMOTE to balance the dataset because it contains imbalanced classification.
- A training and testing dataset are divided from the dataset.
- A choice of machine learning algorithms is Support Vector Classifier (SVC), Based on the training data, a classifier model is created for the algorithms.
- The machine learning algorithm classifier model is tested using the test set, and then the performance of each classifier is compared using a variety of metrics.

4.1 Modules Description

- Dataset Collection
- Data Preprocessing
- Random Sampling
- Training the data
- Classification methods
- Packages used
- Data Evaluation

4.2 Dataset Collection

Using the Pandas package, a complete collection of network traffic data has been incorporated into the CIC-IDS-2017 dataset. This dataset comprises 78 distinct features that serve as the input variables, while the labels are used as the output variable. These labels map to several different types of network attacks. Then it has been classified as normal or anormal and created a new dataset and named it as NormalAnormal_dataset.csv which is further in use.

4.3 Data Preprocessing

Preprocessing data is a crucial stage, especially when dealing with intrusion data, which frequently contains contaminates and missing values that could compromise the accuracy and efficiency of the data mining procedure. As a result, data preprocessing is done to improve the efficacy and quality of the data that is collected following the mining process.

To achieve accurate results and good predictions utilizing machine learning techniques applied to the dataset, data preparation is essential.

```
: print('dataset label after reducing noise:\n',df2_value)

dataset label after reducing noise:
  Label
BENIGN              2096484
DoS Hulk             172849
DDoS                 128016
PortScan              90819
DoS GoldenEye         10286
FTP-Patator            5933
DoS slowloris          5385
DoS Slowhttptest       5228
SSH-Patator            3219
Bot                    1953
Brute Force            1470
XSS                     651
Infiltration             36
Sql Injection            21
Heartbleed               11
 XSS                      1
Name: count, dtype: int64

: print('First (row,Column) number of the dataset: {} '.format(df2.shape))

First (row,Column) number of the dataset: (2522362, 79)
```

Figure 4.1 Dataset After reducing Noise

4.4 Random Sampling

Random sampling is a key technique in data analysis, especially when dealing with large data sets. For label values, a common approach is to check the label values to ensure balanced representation in the dataset. For example, in a scenario where label values exceed 1,00,000, you can down-sample these values to a more manageable 1,00,000, ensuring a more even distribution across categories. This strategic use of random sampling techniques results in a more even distribution of label values, increasing the accuracy and reliability of data analysis.

```python
def perform_random_sampling(df, label_column):
    unique_labels = df[label_column].unique()

    under_sampled_df = pd.DataFrame()

    for label in unique_labels:
        label_data = df[df[label_column] == label]
        num_samples_label = len(label_data)

        if num_samples_label > 400000:
            under_sampled_label = label_data.sample(n=100000, random_state=42)
            under_sampled_df = pd.concat([under_sampled_df, under_sampled_label])
        else:
            under_sampled_df = pd.concat([under_sampled_df, label_data])
    return under_sampled_df

under_sampled_data = perform_random_sampling(data, ' Label')
```

```python
Sampled = under_sampled_data[' Label'].value_counts()
print(Sampled)
```

```
 Label
Normal     100000
Anormal    100000
Name: count, dtype: int64
```

Figure 4.2 Random Sampling

4.5 Training the model

The training dataset, which makes up 70% of the data to create the classification model, whereas the testing dataset, which makes up 30% of the data, is used to predict outcomes. To find the algorithm accuracy scores, the model is trained on training and validation sets.

4.6 Borderline Synthetic Minority Over-Sampling Technique (SMOTE).

It is required to address unbalanced classification while developing prediction models for classification of datasets with a significant class imbalance. The main problem with these kinds of datasets is that a lot of machine learning methods tend to ignore the minority class, which leads to poor performance.

Figure 4.3 Oversampling the minority class

To use the Synthetic Minority Over-Sampling Technique (SMOTE), a random instance 'A' from the minority class is first chosen, and its 'K' closest neighbours within the same minority class are then identified. Following that, 'k' nearest neighbours, designated as 'b,' is chosen at random to produce a synthetic instance. 'A' and 'b' are connected to form a line segment in the feature space. 'A' and 'b', the two selected instances, are combined to create convex combinations that provide the synthetic instances.

4.7 Support Vector Machines

The support vector machine is an algorithm for supervised machine learning. SVM is used to create a hyperplane that divides two classes. It can generate a hyperplane or a group of hyperplanes in high-dimensional space. It is possible to use this hyperplane for classification or regression. SVM distinguishes examples within certain classes and can categorise things that lack data support. The nearest training point for every class is used for separation, which is carried out using a hyperplane. For calculations to arrange and categorise data into extreme levels, going beyond the X/Y expectation, a Support vector machine (SVM) is utilised. To provide a simple visual clarity, two labels—red and blue are used along with two informational highlights—X and Y. The classifier will be trained to determine if an X/Y coordinate is blue or red. The SVM then determines the ideal hyperplane for dividing the tags. This line is limited to two dimensions. On one side of the line, everything is blue, and on the other, red. SVM makes it possible for machine learning to be more precise since it is multidimensional.

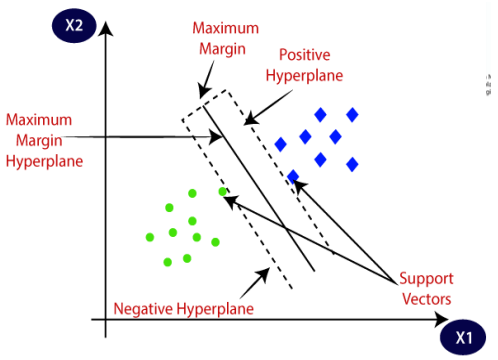

Fig: 4.4 Support Vector Machine (Source: https://ars.els-cdn.com/content/image/1-s2.0-S2772662222000261-gr11.jpg)

4.8 Performance Metrics

The evaluation of the performance of the suggested method is done using confusion matrix, as presented in the tables. The confusion matrix can yield four different outcomes: True positive (TP), False positive (FP), True negative (TN), and False negative (FN).

- **Accuracy:** This gauge how many accurate predictions the model has made overall. For any model, it can be stated as the total number of test instances by the number of accurate forecasts.

$$\text{Accuracy} = TP + / TP + FP + T\text{N}.$$

- **Precision:** Precision is the ratio of accurate positive predictions to all positive predictions.

$$\text{Pre}cison = TP / TP + T\text{N}$$

- **Recall:** Recall is the difference between total positive forecasts and actual positive values.

$$\text{Re}call = \text{I}/TP + F\text{N}$$

- **F1-score:** This score, which considers both recall and precision, is defined as follows:

$$F1\text{-}score = 2 * (precision * recall / precision + recall)$$

4.9 Packages Used

PANDAS:

A set of tools for data analysis and manipulation designed for the Python programming language is called Pandas. It offers techniques and data structures for working with numerical tables and time series. The BSD three-clause license governs the use of this free software. Pandas is a Python module that offers meaningful, flexible, and quick data structures. It helps to perform real-world, routine data analysis.

NUMPY:

NumPy library supports large, multi-dimensional arrays and matrices, and may be used to perform a wide range of complex mathematical operations on them. The core Python package is used for scientific computing is called NumPy. It contains a multidimensional array object, several derived objects, and numerous routines for fast array operations.

MATPLOT:

Matplotlib is a Python framework for charting, and NumPy is an expansion on numerical mathematics. It offers an API for integrating plots into applications using GUI toolkits. The Python application Matplotlib allows you to create static, animated, and interactive graphics. Matplotlib is a tool that simplifies complex and tough tasks.

SKLEARN:

For the Python programming language, Scikit-learn is a free ML package which is scikits.learn It provides several techniques for classification, regression, and clustering, including k-means, DBSCAN, random forests, and support-vector machines. It is also designed to be compatible with the scientific and numerical libraries for Python, NumPy and SciPy. Additionally, it offers a range of tools for different utilities, such as data pretreatment, model evaluation, and model selection.

PICKLE:

The pickle library in Python is a powerful tool used for serializing and deserializing Python objects. In machine learning (ML), it's commonly employed for saving trained models to disk and later loading them back into memory. This is useful for reusing models, sharing them with others, or deploying them in production environments without having to retrain the model every time.

STREAMLIT:

Streamlit is an open-source Python library that allows you to create web applications for machine learning and data science projects with minimal effort. It's designed to enable rapid prototyping and sharing of ML models and data visualizations through simple and intuitive Python scripts.

REQUESTS:

The requests are a Python library used to make HTTP requests. It simplifies the process of sending HTTP requests and handling responses, making it a widely used tool for interacting with web services and APIs. The requests are a versatile library used in various applications for interacting with APIs, fetching web content, and more, thanks to its simplicity and effectiveness in handling HTTP requests and responses in Python.

PIL (PYTHON IMAGING LIBRARY):

The Python Imaging Library (PIL) was a widely used library for working with images in Python. However, its development ceased around 2011, and since then, a fork called Pillow has taken over as the more actively maintained and updated library for image processing in Python. Pillow is backward-compatible with PIL and includes several improvements and additional features. Its ease of use and extensive functionality make it a go-to choose for working with images within Python applications, whether it's for basic operations or complex image manipulations.

MYSQL.CONNECTOR:

MySQL. Connector is a Python driver used to connect to MySQL databases from Python programs. It provides an interface for Python scripts to interact with MySQL databases, allowing you to execute SQL queries, perform database operations, and manage data. It provides a straightforward way to interact with MySQL databases in Python, making it a valuable tool for managing and working with database operations within Python applications.

Smtplib:

This module provides an SMTP (Simple Mail Transfer Protocol) client session object that can be used to send mail to any internet machine with an SMTP or ESMTP listener daemon. It encapsulates the functionality for connecting to an SMTP server and sending emails.

email. mime.text.MIMEText:

This module is a part of the email package and specifically deals with creating email messages. It's used to create MIME (Multipurpose Internet Mail Extensions) objects for representing textual information in emails.

CHAPTER - 5

TECHNOLOGIES USED

5.1 Jupyter Notebook (Anaconda 3 Framework-based)

A web-based programming environment called Jupyter Notebook enables the creation and sharing of documents containing narrative prose, equations, live code, and visualizations. It provides a cell-based interface for task organization and supports Python. It supports Markdown and data visualization using libraries like Matplotlib, Seaborn, and Plotly. It is commonly integrated with the Anaconda distribution for data science and machine learning.

5.2 Python

Python is well-known for being readable high-level programming languages and is easy to use. Its extensive ecosystem of libraries and frameworks serves a wide range of applications, including scientific computing, web development, data analysis, and artificial intelligence. In the context of our machine learning project, Python serves as a cornerstone, powering algorithm implementation, data analysis, and model development. The prominence of python in machine learning, coupled with its potent libraries and user-friendly nature, positions it as an essential tool for our project. This proficiency empowers us to construct intricate models, extract valuable insights from data, and successfully attain the project objectives.

5.3 Wireshark

In Wireshark, WinPcap 4.1.3 tool is used to capture the packets within predetermined timeframes. The captured packets are made available in formats such as CSV files and packet captures (PCAP) for further processing. Fig.5.1 shows an instance of packet capture in the system.

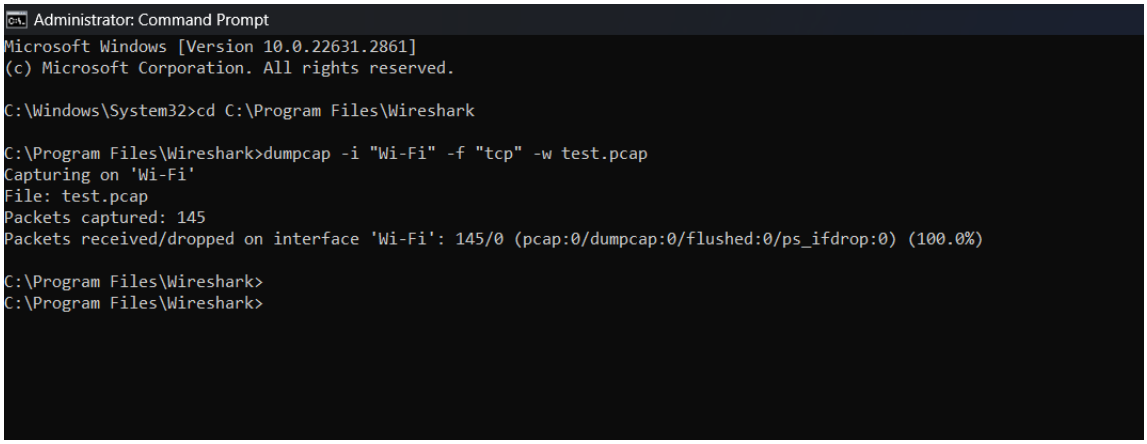

Figure 5.1 Wireshark Packet capture using command line interface

5.4 CICFlowMeter

To transform the raw PCAP files into bidirectional flows, the CICFlowMeter tool is employed. A "biflow" is a network connection that combines and calculates Eighty network features from all the packets that are sent across that connection. The source IP, destination IP addresses, ports, and the timestamp are used to uniquely identify it. The primary purpose of this dataset is to serve as a comprehensive and dependable resource for the assessment of intrusion detection algorithms and systems. Fig. 5.2 shows the capturing of packets by the CICFlowMeter.

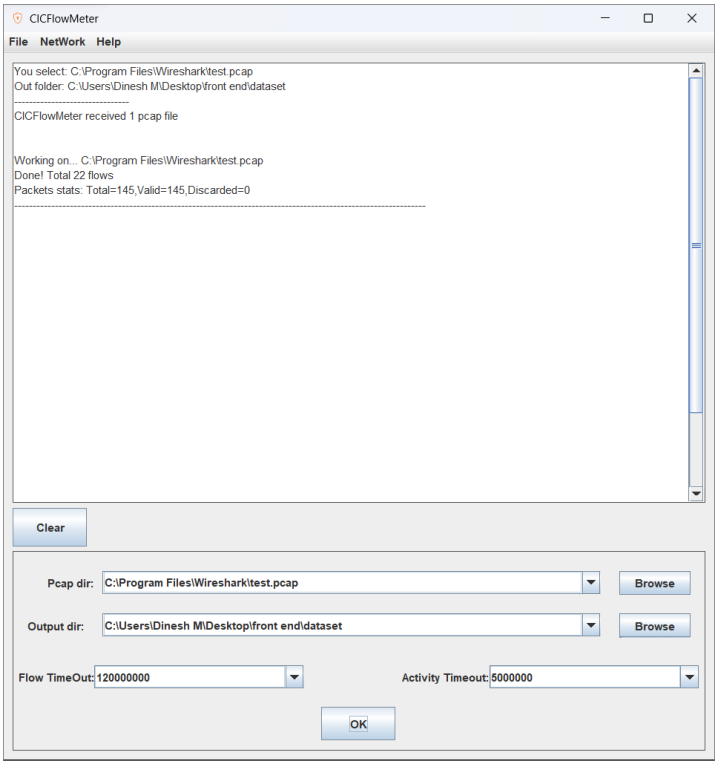

Figure 5.2 CICFlowMeter Packet capture

CHAPTER - 6

EXPERIMENTAL RESULTS AND DISCUSSION

6.1 Dataset Before and After Removing Noise

Noise removal is one of the essential processes before deploying any machine learning algorithm. Removing the noise helps in maintaining the consistency of the data and results.

a. Before noise removal b. After noise removal

Figure 6.1 Dataset Value count

The impact of noise removal can be seen Figure 6.1. The data count in the data before noise removal (Figure 6.1 a) will be high compared to the data count after noise removal (Figure 6.1 b).

6.2 Exploratory Data Analysis (EDA)

In data science and statistics in particular, Exploratory Data Analysis is a crucial first stage in the data analysis process. In order to better comprehend a dataset, find trends, spot abnormalities, and produce hypotheses about the data, it entails analyzing and summarizing the data. To help with later analysis, exploratory data analysis (EDA) is frequently carried out prior to formal statistical modeling or hypothesis testing.

EDA is a flexible and iterative process, and the specific steps and techniques used which can vary depending on the dataset and the goals of the analysis. The goal of EDA is to provide a foundation for more advanced statistical and machine learning techniques while also ensuring a better understanding of the data and its underlying structure.

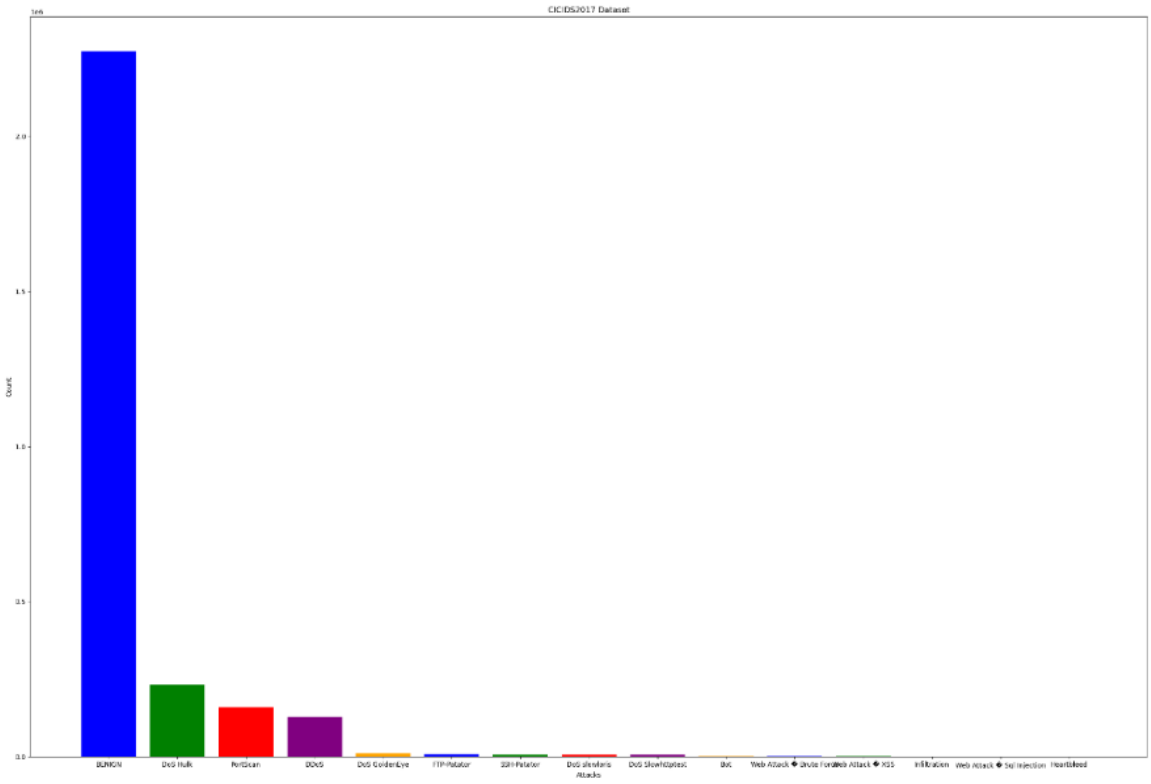

Figure 6.2 Data visualization

Figure 6.2 shows the density of data available in the overall dataset. This is classified based on the labels associated with the data. It could be found that the concentration of benign data is very high amidst the total data available in the dataset.

6.3Borderline Synthetic Minority Over-Sampling Technique (SMOTE)

6.3.1 Before SMOTE

```
print('Original dataset shape %s' % Counter(y_train))
```

```
Original dataset shape Counter({'Normal': 70116, 'Anormal': 69807})
```

Figure 6.3 Before Borderline SMOTE Value count

Figure 6.3 shows the source code relevant to the classification of Normal and Anormal data before the application of the Borderline SMOTE technique. It could be seen that the data set contains 69807 abnormal records and 70116 Normal data records in the data set. The same could be visualised as a graph in Figure 6.4. The source code relevant to the visualization of the graph is also made available in the figure.

```
original_counts = Counter(y_train)
class_labels = list(original_counts.keys())
class_counts = list(original_counts.values())
plt.figure(figsize=(10, 6))
sns.set_style("whitegrid")  # Optional, set the style
sns.barplot(x=class_labels, y=class_counts)
plt.xlabel('Class Labels')
plt.ylabel('Count')
plt.title('Class Distribution in y_train')
plt.show()
```

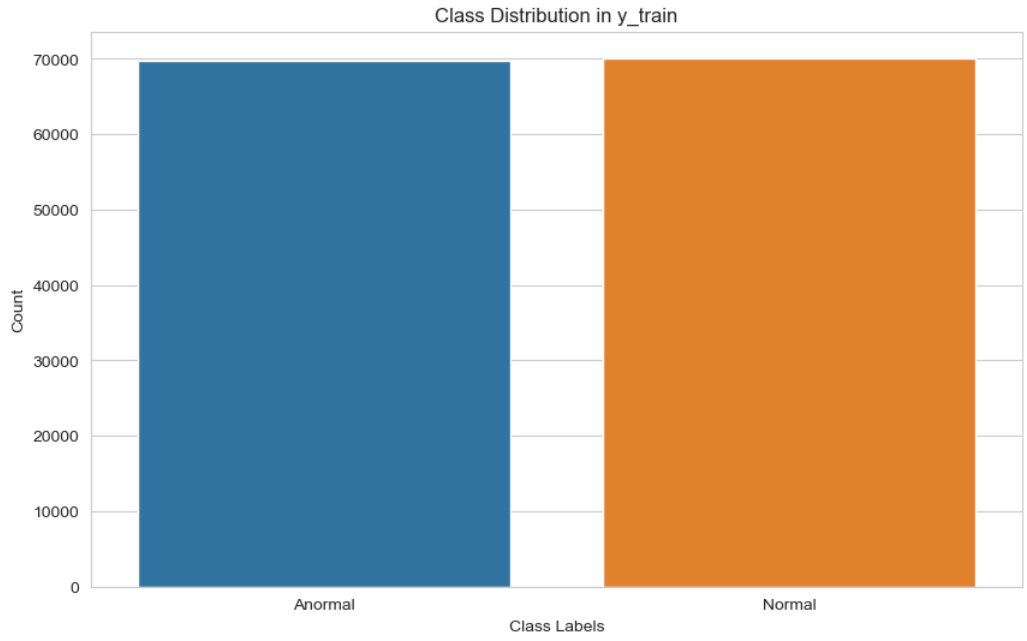

Figure 6.4 Before Borderline SMOTE

6.3.2 After SMOTE

Unbalanced data is one of the major challenges encountered while processing data in large scale. This will create a skewness in the decision-making process. To encounter this issue, the Borderline SMOTE is applied to the dataset. It could be observed in Figure 6.5 and Figure 6.6 that the balancing achieved in terms of the data values. In Figure 6.5, it could be observed that the Abnormal and Normal values are equal after the application of SMOTE, that is 70116.

```
# Perform borderline SMOTE over-sampling
smote = BorderlineSMOTE()
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)
print('Resampled dataset shape %s' % Counter(y_resampled))
```

```
Resampled dataset shape Counter({'Anormal': 70116, 'Normal': 70116})
```

Figure 6.5 After Borderline SMOTE Value count

The same is visualised in the form of a graph in Figure 6.6. The source code corresponding to the visualisation is also made available in the figure.

```
resampled_counts=Counter(y_resampled)
class_labels = list(original_counts.keys())
class_counts = list(resampled_counts.values())
# Create a bar chart using Seaborn
plt.figure(figsize=(10, 6))
sns.set_style("whitegrid")  # Optional, set the style
sns.barplot(x=class_labels, y=class_counts)
plt.xlabel('Class Labels')
plt.ylabel('Count')
plt.title('Class Distribution in y_resampled')
plt.show()
```

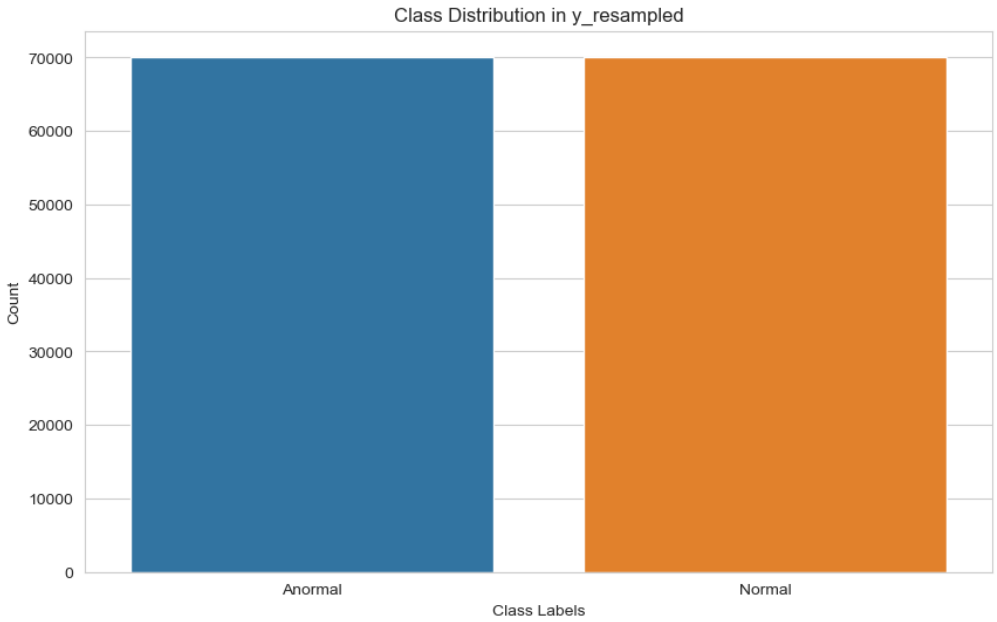

Figure 6.6 After Borderline SMOTE

6.4 Support Vector Machine Algorithm

Support Vector Machines is a significant algorithm in machine learning. However, in case of the implementation, it is found that the deployment of SVM is less effective in case of the dataset used. There are many factors that influence the results such as a skewed normalization or overfitting of data to optimization parameters. The effectiveness of the SVM depends on the kernel and kernel parameters. It also depends on the soft margin parameter also. The selection of an appropriate optimizer also plays a major role in the results obtained by using a multiclass classifier like Support Vector Machines. Any factor that impacts the choice of the above-mentioned parameters steeply affects the performance of SVMs. These have been evident from the works of multiple researchers (Watanachaturaporn, 2004). In this case also the choice of certain parameters related to SVM might have influenced the downtrend of the results.

The performance of the proposed system is measured in terms of precision, recall and f1- score. The accuracy is also measured. The True positive (TP), False positive (FP), True negative (TN), and False negative (FN) measures are represented as a confusion matrix. It could be observed in Fig.9, that there are 18500 True Positive outputs, 20352 True Negative outputs, 663 False Positive outputs and 2555 False Negative outputs. It could be observed that the False Positive and False Negative outputs are comparatively less than the True Positive and True Negative outputs.

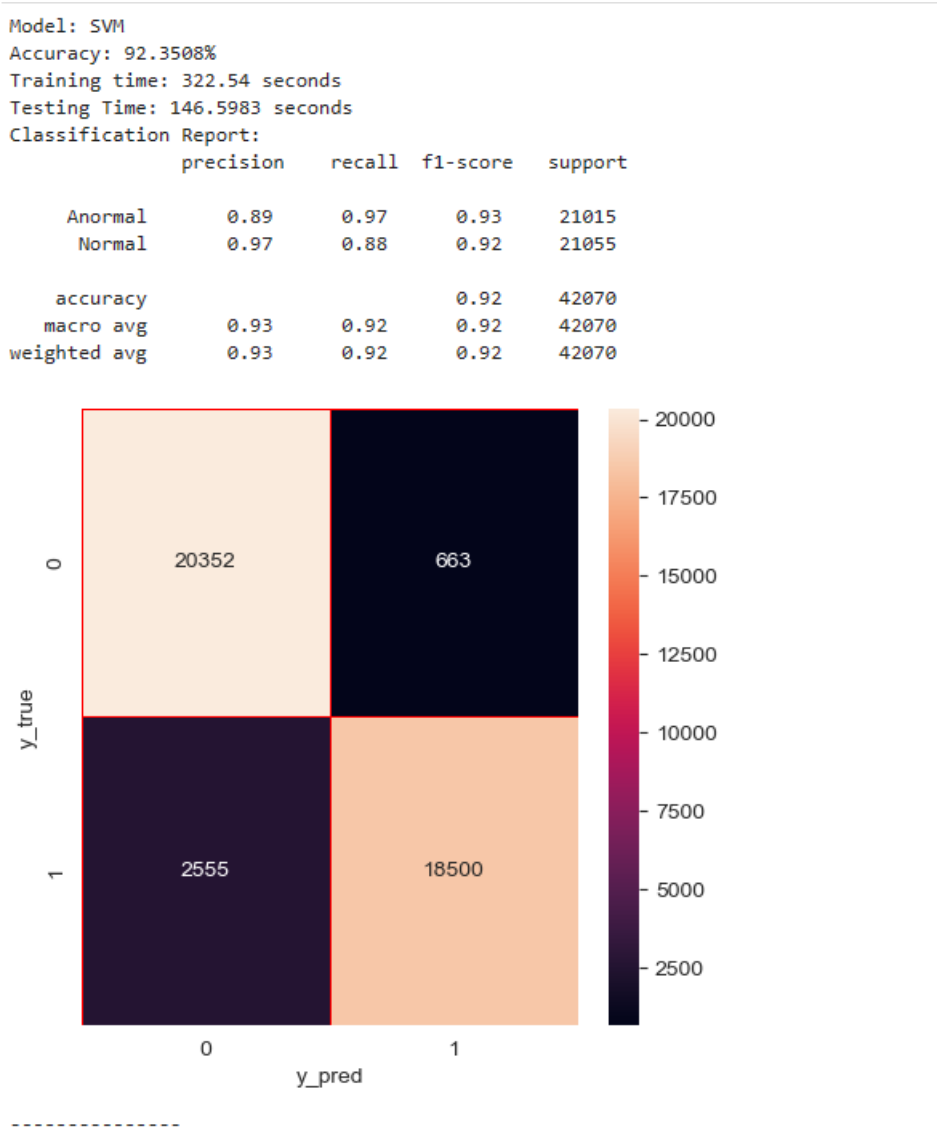

Figure 6.7 Performance Analysis

Further, it is observed that the proposed solution produces results with an accuracy of 92.35%. This may not be as good as the system proposed by Miel [7] but is better compared to most other existing systems. It could be inferred that some of the parameters selected might not have effectively contributed to the decision making in the system.

6.5 Web Interface

Home page

Figure 6.8 Home Page

Model Testing page.

A simple web application to process and to display the results to the user is designed. The web application is interfaced with the CICFlowMeter and the csv files produced by the software are sought as input by this application and is processed. Fig.6.9 shows the design of the application. This page is accessible only to registered users.

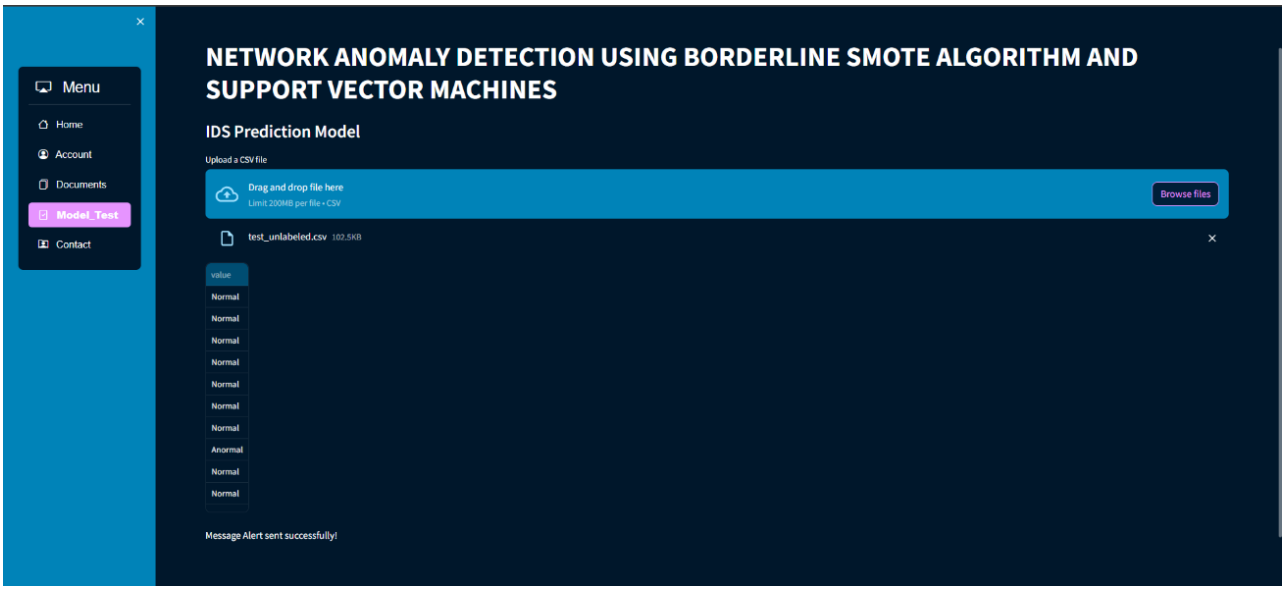

Figure 6.9 Model Testing Page

Mail Alert.

The system was experimented with various inputs that were sought through the CICFlowMeter. A minor setback with the system is that the data obtained through the CICFlowMeter cannot be fed directly to the developed system. The csv or pcap file obtained is fed at regular intervals to the system enabling it to alert the user at regular intervals if an incident occurs.

The users registered with this system also get an email alert from the system, which is shown in Fig. 6.10.

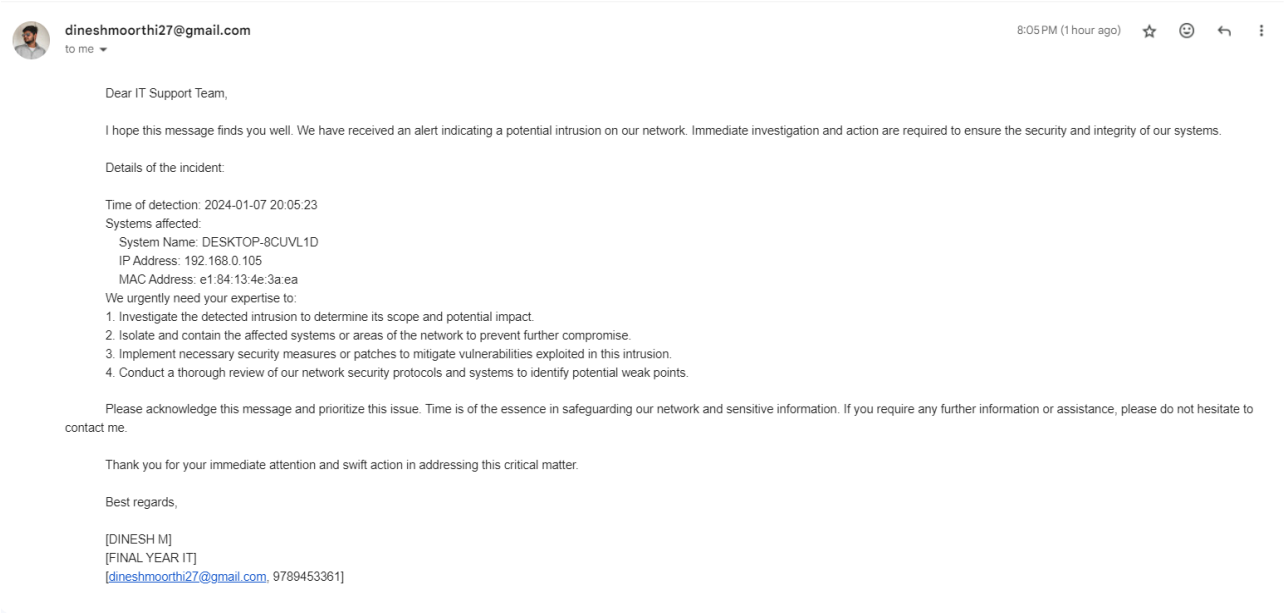

Figure 6.10 Mail Alert

The email alert is sent to the user if and only if an intrusion is attempted or an abnormal traffic is encountered.

Database

The database is set up using MySQL database. The database can be accessed with ease through the phpMyAdmin interface. The purpose of the database is to store user information registered with the intrusion detection system and to store the intrusion incidents recorded by the system. The schema is presented in Fig.6.11

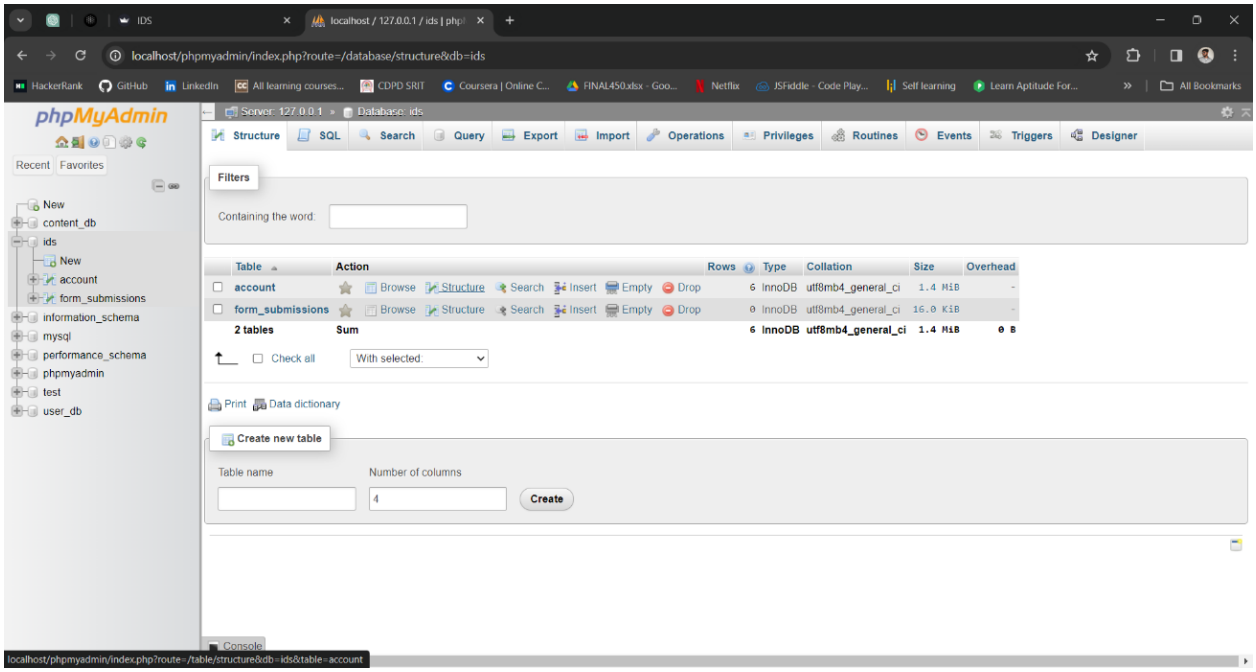

Figure 6.11 Database Schema

CHAPTER - 7

CONCLUSION

In conclusion, this study underscores the criticality of robust network security measures in safeguarding sensitive business data in the digital age. The proposed Synthetic Minority Over-Sampling Technique and Support Vector Machines (SMOTE-SVM) model demonstrates a commendable anomaly detection accuracy of 92.35%, reaffirming the efficacy of machine learning techniques in fortifying network security. The successful application of the SMOTE-SVM model not only enhances anomaly detection but also presents potential implications for wider deployment in practical network security setups. This research contributes substantively to advancing the understanding about the field and practical implementation of innovative approaches to combat evolving network threats, marking a significant stride toward more resilient network infrastructures.

7.1 Future Work

The future enhancements revolve around the seamless integration of live data into the model for real-time prediction. The aim is to elevate our monitoring capabilities by adopting more sophisticated analysis techniques, such as deploying deep learning models for anomaly detection and harnessing big data frameworks to expedite data processing. Furthermore, the focus lies on the development of automation and response mechanisms. This includes the implementation of automated incident response protocols that enable swift mitigation or containment of threats upon their detection.

REFERENCES

[1] N. Meemongkolkiat and V. Suttichaya, “Analysis on network traffic features for designing machine learning based IDS,” in *Journal of Physics: Conference Series*, 2021, vol. 1993, no. 1, p. 12029.

[2] M. Verkerken, L. D’hooge, T. Wauters, B. Volckaert, and F. De Turck, “Towards model generalization for intrusion detection: Unsupervised machine learning techniques,” *J. Netw. Syst. Manag.*, vol. 30, pp. 1–25, 2022.

[3] Y. Sun *et al.*, “Borderline smote algorithm and feature selection-based network anomalies detection strategy,” *Energies*, vol. 15, no. 13, p. 4751, 2022.

[4] H. Han, W.-Y. Wang, and B.-H. Mao, “Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning,” in *International conference on intelligent computing*, 2005, pp. 878–887.

[5] S. Latif, F. F. Dola, M. D. Afsar, I. J. Esha, and D. Nandi, “Investigation of Machine Learning Algorithms for Network Intrusion Detection.,” *Int. J. Inf. Eng. \& Electron. Bus.*, vol. 14, no. 2, 2022.

[6] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization.,” *ICISSp*, vol. 1, pp. 108–116, 2018.

[7] M. Verkerken *et al.*, “A Novel Multi-Stage Approach for Hierarchical Intrusion Detection,” *IEEE Trans. Netw. Serv. Manag.*, 2023.

[8] P.Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, “Anomaly-based network intrusion detection: Techniques, systems and challenges,” *Comput. & Secur.*, vol. 28, no. 1–2, pp. 18–28, 2009.

[9] S. Divya, “A survey on various security threats and classification of malware attacks, vulnerabilities and detection techniques,” *Int. J. Comput. Sci. \& Appl.*, vol. 2, no. 04, 2013.

[10] N. Das and T. Sarkar, “Survey on host and network based intrusion detection system,” *Int. J. Adv. Netw. Appl.*, vol. 6, no. 2, p. 2266, 2014.

[11] M. Ahsan, K. E. Nygard, R. Gomes, M. M. Chowdhury, N. Rifat, and J. F. Connolly, “Cybersecurity threats and their mitigation approaches using Machine Learning—A Review,” *J. Cybersecurity Priv.*, vol. 2, no. 3, pp. 527–555, 2022.

[12] C.-C. Sun, D. J. S. Cardenas, A. Hahn, and C.-C. Liu, “Intrusion detection for cybersecurity of smart meters,” *IEEE Trans. Smart Grid*, vol. 12, no. 1, pp. 612–622, 2020.

[13] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, “Survey of intrusion detection systems: techniques, datasets and challenges,” Cybersecurity, vol. 2, no. 1, pp. 1–22, 2019.

[14] N. V Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: synthetic minority over-sampling technique,” *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, 2002.

[15] P. Watanachaturaporn, P. K. Varshney, and M. K. Arora, “Evaluation of factors affecting support vector machines for hyperspectral classification,” in the American Society for Photogrammetry \& Remote Sensing (ASPRS) 2004 Annual Conference, Denver, CO, 2004.

APPENDICES

Appendix A Source Code

SVM MODEL CODE:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from imblearn.under_sampling import RandomUnderSampler
from collections import Counter
from imblearn.over_sampling import BorderlineSMOTE
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectKBest, mutual_info_classif
from sklearn.preprocessing import MinMaxScaler
from sklearn.impute import SimpleImputer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score,confusion_matrix, classification_report
import time
# Load the CIC-IDS2017 dataset (replace 'dataset.csv' with the actual file path)
data = pd.read_csv('dataset/Normal_Anormal_dataset.csv')
print (data.shape)
count=data[' Label'].value_counts()
print(count)
def perform_random_sampling(df, label_column):
    unique_labels = df[label_column].unique()

    under_sampled_df = pd.DataFrame()

    for label in unique_labels:
        label_data = df[df[label_column] == label]
        num_samples_label = len(label_data)

        if num_samples_label > 400000:
            under_sampled_label = label_data.sample(n=100000, random_state=42)
            under_sampled_df = pd.concat([under_sampled_df, under_sampled_label])
        else:
            under_sampled_df = pd.concat([under_sampled_df, label_data])
    return under_sampled_df
```

```
    under_sampled_data = perform_random_sampling(data, ' Label')
    Sampled = under_sampled_data[' Label'].value_counts()
    print(Sampled)
    # Assuming 'df' is your DataFrame
    # Remove NaN and Inf values
    df = under_sampled_data.replace([np.inf, -np.inf], np.nan).dropna()

    # Remove null values
    df = df.dropna()
    count_res=df[' Label'].value_counts()
    print(count_res)
    X = df.drop(' Label', axis=1)
    y = df[' Label']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
    print(X_train.shape)
    print(X_test.shape)

    print('Original dataset shape %s' % Counter(y_train))
    # Perform borderline SMOTE over-sampling
    smote = BorderlineSMOTE()
    X_resampled, y_resampled = smote.fit_resample(X_train, y_train)
    print('Resampled dataset shape %s' % Counter(y_resampled))

    original_counts = Counter(y_train)
    class_labels = list(original_counts.keys())
    class_counts = list(original_counts.values())
    plt.figure(figsize=(10, 6))
    sns.set_style("whitegrid")  # Optional, set the style
    sns.barplot(x=class_labels, y=class_counts)
    plt.xlabel('Class Labels')
    plt.ylabel('Count')
    plt.title('Class Distribution in y_train')
    plt.show()
    resampled_counts=Counter(y_resampled)
    class_labels = list(original_counts.keys())
    class_counts = list(resampled_counts.values())

    # Create a bar chart using Seaborn
    plt.figure(figsize=(10, 6))
    sns.set_style("whitegrid")  # Optional, set the style
    sns.barplot(x=class_labels, y=class_counts)
    plt.xlabel('Class Labels')
    plt.ylabel('Count')
    plt.title('Class Distribution in y_resampled')

    plt.show()
```

```
    X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled,
test_size=0.3, random_state=42)
    print(X_train.shape)
    print(X_test.shape)

    # Feature selection using mutual information gain ratio
    selector = SelectKBest(mutual_info_classif, k=14)
    X_train_selected = selector.fit_transform(X_train, y_train)
    # Get indices of the selected features
    selected_indices = selector.get_support(indices=True)

    # Get the names of the selected features from the original dataframe (assuming X is
a DataFrame)
    selected_features = X.columns[selected_indices]

    # Print the names of the selected features
    print("Selected Features:")
    print(selected_features)
    # Scaling features using MinMaxScaler
    scaler = MinMaxScaler()
    X_train_scaled = scaler.fit_transform(X_train_selected)
    X_test_scaled = scaler.transform(selector.transform(X_test))

    # Deleting NaN/inf values
    imputer = SimpleImputer(strategy='mean')
    X_train_scaled = imputer.fit_transform(X_train_scaled)
    X_test_scaled = imputer.transform(X_test_scaled)

    # Training and evaluating models
    models = {
        'SVM': SVC()
    }

    for name, model in models.items():
        start_time = time.time()
        model.fit(X_train_scaled,y_train )
        training_time = time.time() - start_time
        # Testing time calculation
        start_test_time = time.time()
        y_pred = model.predict(X_test_scaled)
        end_test_time = time.time()
        test_time = end_test_time - start_test_time
        accuracy = accuracy_score(y_test, y_pred)
        accuracy_percentage = accuracy * 100
        report = classification_report(y_test, y_pred)
```

```
        print(f"Model: {name}")
        print(f"Accuracy: {accuracy_percentage:.4f}%")
        print(f"Training time: {training_time:.2f} seconds")
        print(f"Testing Time: {test_time:.4f} seconds")
        print("Classification Report:\n", report)
        y_true=y_test
        cm=confusion_matrix(y_true,y_pred)
        f,ax=plt.subplots(figsize=(5,5))
        sns.heatmap(cm,annot=True,linewidth=0.5,linecolor="red",fmt=".0f",ax=ax)
        plt.xlabel("y_pred")
        plt.ylabel("y_true")
        plt.show()
        print("---------------")


    # Code to save code the model
    import pickle
    filename = '_14_model.sav'
    pickle.dump(model, open(filename, 'wb'))
```

MAIN PAGE:

```
import streamlit as st
import login
from PIL import Image
import requests
from streamlit_option_menu import option_menu
from streamlit_lottie import st_lottie
import predict
import contact
st.set_page_config(page_title="IDS",layout="wide")
class MultiApp:
    def __init__(self):
        self.apps = []

    def run(self,):
        # Function to load Lottie URL
        def load_lottieurl(url):
            r = requests.get(url)
            if r.status_code != 200:
                return None
            return r.json()

        def local_css(filename):
            with open(filename) as f:
```

```
            st.markdown(f"<style>{f.read()}</style>",unsafe_allow_html=True)
    local_css("style/style.css")

    # ---- LOAD ASSETS ----
    # Consider resizing images for responsiveness
    lottie_coding=
load_lottieurl("https://assets5.lottiefiles.com/packages/lf20_fcfjwiyb.json")
    img_contact_form = Image.open("images/Learn-about-benefits-of-document-
management-system-banner-1024x332.webp").resize((500, 430))

    st.title("NETWORK ANOMALY DETECTION USING BORDERLINE SMOTE
ALGORITHM AND SUPPORT VECTOR MACHINES")
    with st.sidebar:
        selected = option_menu(
            menu_title="Menu",
            options=["Home","Account", "Documents", "Model_Test", "Contact"],
            icons=["house","person-circle","files","file-check","person-rolodex"],#bootstrp
icon names are included here
            menu_icon="cast",
            default_index=0,
        )

    if selected == "Home":
        with st.container():
            st.subheader("Hi, I am Dinesh :wave:")
            st.write("#")
            st.write("This is the project to detect the  Network intrusion ")
        with st.container():
            # Abstract section
            st.write("---")
            left_column, right_column = st.columns(2)
            with left_column:
                st.header("Abstract of our project")
                st.write("##")
                st.write("""Network Security is a major challenge in the digital world.
Intrusion is common in many applications and intruders are sophisticated enough to
change their attack pattern very often.
                To address this issue, the development of a model for the detection
of network anomalies and intrusions. The approach utilizes the Borderline Synthetic
Minority Over-Sampling Technique (SMOTE) along with Support Vector Machines
(SVM) to enhance anomaly detection capabilities. By intelligently oversampling the
minority class using SMOTE and training SVM, the proposed model exhibits a robust
defence mechanism against network intruders. The utilization of these advanced
techniques aims to augment the accuracy and efficiency of anomaly detection,
minimizing false positives and ensuring prompt response to genuine threats.
                The results obtained from this study add to the ongoing efforts to
secure data in the digital age, by combining SMOTE and SVM for network intrusion
detection.
```

```
                """)
            with right_column:
                st_lottie(lottie_coding, height=300, key="coding")

        # Projects section
        with st.container():
            st.write("---")
            st.header("project overview")
            st.write("##")
            image_column, text_column = st.columns((1, 2))
            with image_column:
                st.image(img_contact_form)
            with text_column:
                st.write(
                    """Digital transformation is taking place in most sectors around the world. A drastic effort of networking systems globally
                    has given the advantage of unlimited access to data and knowledge. With the growth in digital systems the growth of users with a malicious intent has also started to grow.
                    Early intrusions have been made in the late 1970s and early 1980s. This was a period when most systems were not networked with one another, and internet was in a rudimentary state.
                    Hence, the impact of any such intrusions cost only a few individuals and to the worst case, an organisation. As the years passed, more organisations entered the digital domain and started doing business online.
                    The malicious users also started experimenting with sophisticated tools for intruding into systems over which business is done. Over the years, the pattern of intrusion has also changed. Some users with a malicious intent still try the classical intrusion techniques, which are usually detected by intrusion detection systems.
                    These classical methods follow a specific signature and hence can be detected by a signature-based intrusion detection system. However, advanced malicious users use different patterns of intrusion which are least detected by signature-based intrusion detection systems.
                    Such patterns can only be detected using an anomaly detection system.
                    These systems harness the power of artificial intelligence to perform anomaly detection. In the recent days, machine learning is applied in a higher level to detect such intrusions."""
                )
    if selected == "Account":
        login.ma()

    if selected == "Documents":
        try:
            if login.get_session_id():
                if st.session_state.logged_in:
                    st.write("Here is the document for our project")
                else:
```

```
                st.warning("Please login to access this page.")
            except:
                st.warning("Please login to access this page.")


        if selected == "Model_Test":
            try:
                if login.get_session_id():
                    if st.session_state.logged_in:
                        #testing.testing_csv()
                        predict.testing_csv()
                    else:
                        st.warning("Please login to access this page.")
            except Exception as e:
                st.warning(f"Error {e}")
                #st.warning("Please login to access this page.")

        if selected == "Contact":
            try:
                if login.get_session_id():
                    if st.session_state.logged_in:
                        contact.cont()
                    else:
                        st.warning("Please login to access this page.")
            except:
                st.warning("Please login to access this page.")
my_app = MultiApp()
my_app.run()
```

LOGIN PAGE:

```
import streamlit as st
import mysql.connector
from PIL import Image
from io import BytesIO
import uuid

# Establish connection to your local MySQL database
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="ids"
)
mycursor = mydb.cursor()

def create_session_id():
```

```
    return str(uuid.uuid4())

# Check if session exists and return session ID
def get_session_id():
    if 'session_id' not in st.session_state:
        st.session_state.session_id = create_session_id()
    return st.session_state.session_id

def creation():
    def create_user_table():
        mycursor.execute("""
            CREATE TABLE IF NOT EXISTS ACCOUNT (
                email VARCHAR(255) PRIMARY KEY,
                password VARCHAR(255) NOT NULL,
                username VARCHAR(255) NOT NULL UNIQUE,
                name VARCHAR(255) NOT NULL,
                phone VARCHAR(25) NOT NULL,
                profile_picture LONGBLOB
            )
        """)
        mydb.commit()

    create_user_table()
    img_login = Image.open("images/bg.jpg").resize((300, 300))

    def display_profile_picture(profile_pic):
        try:
            if profile_pic:
                image = Image.open(BytesIO(profile_pic))
                image = image.resize((300, 300))
                st.image(image)
            else:
                st.image(img_login)
        except Exception as e:
            st.error(f"Error loading profile picture: {e}")

    def user_authentication():
        st.title("User Authentication")
        if 'logged_in' not in st.session_state:
            st.session_state.logged_in = False

        if not st.session_state.logged_in:
            choice = st.selectbox("Login/Signup", ["Login", "Register"])
            if choice == "Login":
                # Login Section
                st.subheader("Login Section")
                email = st.text_input("Email")
                password = st.text_input("Password", type='password')
```

```
            if st.button("Login"):
                try:
                    query = "SELECT * FROM ACCOUNT WHERE email = %s"
                    values = (email,)
                    mycursor.execute(query, values)
                    user = mycursor.fetchone()

                    if user:
                        if user[1] == password:  # Check if fetched password matches input
password
                            st.success("Login successful")
                            st.session_state.username = user[2]  # Assuming username is at
index 3
                            st.session_state.useremail = user[0]  # Assuming email is at index
1
                            st.session_state.logged_in = True
                            profile_pic = user[5]
                            display_profile_picture(profile_pic)
                        else:
                            st.warning("Login failed. Incorrect password.")
                    else:
                        st.warning("Login failed. Incorrect email.")
                except Exception as e:
                    st.error(f"Error: {e}")

        else:
            # Create New Account Section
            st.subheader("Create New Account")
            new_email = st.text_input("Email")
            new_password = st.text_input("Password", type='password')
            new_username = st.text_input("Unique Username")
            new_name = st.text_input("Full Name")
            new_phone = st.text_input("Phone Number")
            new_profile_pic = st.file_uploader("Upload Profile Picture", type=['jpg', 'png',
'jpeg'])

            if st.button("Register"):
                mycursor.execute(f"SELECT * FROM ACCOUNT WHERE
username='{new_username}'")
                existing_user = mycursor.fetchone()
                if existing_user:
                    st.warning("Username already exists! Try a different one.")
                else:
                    try:
                        query = "INSERT INTO ACCOUNT (email, password, username,
name, phone, profile_picture) VALUES (%s, %s, %s, %s, %s, %s)"
                        values = (new_email, new_password, new_username, new_name,
```

```
new_phone,
                        new_profile_pic.read() if new_profile_pic else None)
                    mycursor.execute(query, values)
                    mydb.commit()
                    st.success('Account created successfully!')
                    st.balloons()
                    st.markdown("Login using your email and password")
                except Exception as e:
                    st.warning("Account creation failed.")
                    st.error(f"Error: {e}")

    # Run the authentication function
    user_authentication()

def ma():
    creation()  # Run creation function

    if st.session_state.logged_in:
        # Display logged-in user's information
        st.write("User is logged in.")
        # Add the content you want to display for logged-in users here
        st.subheader(f"Welcome, {st.session_state.username}")
        st.write(f"Email: {st.session_state.useremail}")

        # Add a logout button
        if st.button("Logout"):
            st.session_state.logged_in = False
            # Clear session on logout
            if 'username' in st.session_state:
                del st.session_state.username
            if 'useremail' in st.session_state:
                del st.session_state.useremail
            st.write("Logged out successfully.")
```

MODEL TESTING CODE:

```
import numpy as np
import pandas as pd
import pickle
import streamlit as st
from sklearn.preprocessing import MinMaxScaler
from sklearn.impute import SimpleImputer
import smtplib
from email.mime.text import MIMEText
import datetime
import socket
import uuid
```

```
def preprocess_and_predict(model, new_data):
    new_data_clean = new_data.replace([np.inf, -np.inf], np.nan).dropna()
    new_data_clean = new_data_clean.dropna()

    scaler = MinMaxScaler()
    imputer = SimpleImputer(strategy='mean')

    # Fit the imputer and scaler to the data
    imputed_data = imputer.fit_transform(new_data_clean)
    scaler.fit(imputed_data)  # Fit scaler to the imputed data
    X_new_selected_scaled = scaler.transform(imputed_data)

    y_pred_new = model.predict(X_new_selected_scaled)
    return y_pred_new


def testing_csv():
    st.subheader(' IDS Prediction Model')
    # loading the saved model
    loaded_model = pickle.load(open('_14_model.sav', 'rb'))
    uploaded_file = st.file_uploader("Upload a CSV file", type=["csv"])
    if uploaded_file is not None:
        new_data = pd.read_csv(uploaded_file)
        predictions = preprocess_and_predict(loaded_model, new_data)
        st.write(predictions)
        if predictions[0] == 1:
            pass
        else:
            current_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")

            # Get the hostname
            system_name = socket.gethostname()
            # Get the IP address
            ip_address = socket.gethostbyname(system_name)

            # Get the MAC address
            mac_address = ':'.join(['{:02x}'.format((uuid.getnode() >> elements) & 0xff) for
elements in range(0, 2 * 6, 2)][::-1])
            subject = "Urgent: Network Intrusion Detected - Immediate Investigation
Required"
            body = f"""
            Dear IT Support Team,

            I hope this message finds you well. We have received an alert indicating a
potential intrusion on our network. Immediate investigation and action are required to
ensure the security and integrity of our systems.
```

Details of the incident:

Time of detection: {current_time}
Systems affected:
System Name: {system_name}
IP Address: {ip_address}
MAC Address: {mac_address}
We urgently need your expertise to:
1. Investigate the detected intrusion to determine its scope and potential impact.
2. Isolate and contain the affected systems or areas of the network to prevent further compromise.
3. Implement necessary security measures or patches to mitigate vulnerabilities exploited in this intrusion.
4. Conduct a thorough review of our network security protocols and systems to identify potential weak points.

Please acknowledge this message and prioritize this issue. Time is of the essence in safeguarding our network and sensitive information. If you require any further information or assistance, please do not hesitate to contact me.

Thank you for your immediate attention and swift action in addressing this critical matter.

Best regards,

[DINESH M]
[FINAL YEAR IT]
[dineshmoorthi27@gmail.com, 9789453361]
"""
sender = "dineshmoorthi27@gmail.com"
recipients = ["projecttestingfyp@gmail.com"]
password = "wtai ayeq zfwa tiwr"

```
def send_email(subject, body, sender, recipients, password):
    msg = MIMEText(body)
    msg['Subject'] = subject
    msg['From'] = sender
    msg['To'] = ', '.join(recipients)
    with smtplib.SMTP_SSL('smtp.gmail.com', 465) as smtp_server:
        smtp_server.login(sender, password)
        smtp_server.sendmail(sender, recipients, msg.as_string())
    st.write("Message Alert sent successfully!")

send_email(subject, body, sender, recipients, password)
```

CONTACT PAGE:

```
import streamlit as st
import mysql.connector
# Establish connection to your local MySQL database
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",  # Enter your MySQL password here
    database="ids"
)
cursor = conn.cursor()
# Function to create the table
def create_form_submissions_table():
    cursor.execute("""
    CREATE TABLE IF NOT EXISTS form_submissions (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    message TEXT NOT NULL,
    submission_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP)
    """)
    conn.commit()
create_form_submissions_table()
def cont():
    st.header("Get in touch with me!")
    # Define form inputs
    form_name = st.text_input("Your name")
    form_email = st.text_input("Your email")
    form_message = st.text_area("Your message here")
    submit_button = st.button("Send")
    if submit_button:
        # Insert form data into the database
        query = "INSERT INTO form_submissions (name, email, message) VALUES (%s,
%s, %s)"
        cursor.execute(query, (form_name, form_email, form_message))
        conn.commit()
        # Display a link to navigate back to the home page
        st.success("Form submitted successfully!")
        st.balloons()
```

Appendix B Plagiarism Report

RE-2022-220914 - Turnitin Plagiarism Report

by Adri Jovin John Joseph

Submission date: 22-Mar-2024 06:17PM (UTC+0300)
Submission ID: 271711140641
File name: RE-2022-220914.docx (1.03M)
Word count: 5506
Character count: 31775

RE-2022-220914-plag-report

ORIGINALITY REPORT

10%	7%	5%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.coursehero.com Internet Source	1%
2	backoffice.biblio.ugent.be Internet Source	1%
3	Submitted to University of Liverpool Student Paper	1%
4	Faeiz Alserhani, Alaa Aljared. "Evaluating Ensemble Learning Mechanisms for Predicting Advanced Cyber Attacks", Applied Sciences, 2023 Publication	<1%
5	Submitted to University of East London Student Paper	<1%
6	"Machine Learning Applications", Wiley, 2023 Publication	<1%
7	patents.google.com Internet Source	<1%
8	ebin.pub Internet Source	<1%

Appendix C Published Conference Paper

Network Anomaly Detection Using Borderline Smote Algorithm and Support Vector Machines

M. Dinesh[1], **C. S. Sabarish**[2], **S. Yogeshwaran**[3], **Dr. J. J. Adri Jovin**[4]

[1,2,3] Under Graduate Student(s), Department of Information Technology,

Sri Ramakrishna Institute of Technology, Coimbatore, Tamil Nadu, India

[4] Associate Professor, Department of Information Technology,

Sri Ramakrishna Institute of Technology, Coimbatore, Tamil Nadu, India

Corresponding Author: dineshm.2006@srit.org, adrijovin.it@srit.org.

***Abstract*— Network Security is a major challenge in the digital world. Intrusion is common in many applications and intruders are sophisticated enough to change their attack pattern very often. To address this issue, the development of a model for the detection of network anomalies and intrusions. The proposed approach utilizes the Borderline Synthetic Minority Over-Sampling Technique (SMOTE) along with Support Vector Machines (SVM) to enhance anomaly detection capabilities. By intelligently oversampling the minority class using SMOTE and training SVM, the proposed model exhibits a robust defence mechanism against network intruders. The utilization of these advanced techniques aims to augment the accuracy and efficiency of anomaly detection, minimizing false positives and ensuring prompt response to genuine threats. Theresults obtained from this study add to the ongoing efforts to secure data in the digital age, by combining SMOTE and SVM for network intrusion detection.**

Keywords*— *Network Security, Anomaly detection, Robust Defence Mechanism, Borderline SMOTE and SVM.

I. INTRODUCTION

Digital transformation is taking place in most sectors around the world. A drastic effort of networking systems globally has given the advantage of unlimited access to data and knowledge. With the growth in digital systems the growthof users with a malicious intent has also started to grow. Earlyintrusions have been made in the late 1970s and early 1980s.This was a period when most systems were not networked with one another, and internet was in a rudimentary state. Hence, the impact of any such intrusions cost only a few individuals and to the worst case, an organisation. As the yearspassed, more organisations entered the digital domain and started doing business online. The malicious users also startedexperimenting with sophisticated tools for intruding into systems over which business is done. Over the years, the pattern of intrusion has also changed. Some users with a malicious intent still try the classical intrusion techniques, which are usually detected by intrusion detection systems. These classical methods follow a specific signature and hencecan be detected by a signature-based intrusion detection system. However, advanced malicious users use different patterns of intrusion which are least detected by signature- based intrusion detection systems. Such patterns can only be detected using an anomaly detection system. These systems harness the power of artificial intelligence to perform anomaly detection. In the recent days, machine learning is applied in ahigher level to detect such intrusions[1][2].

II. LITERATURE REVIEW

Sun et al. meticulously designed a Borderline SMOTE Algorithm and Feature Selection-Based Network Anomalies Detection Strategy marking a significant advancement in the domain of network security[3]. The research introduces a groundbreaking framework for network anomaly detection that masterfully addresses the intricate task of classifying diverse network intrusions. The approach gracefully incorporates a resampling strategy, blending data from Borderline SMOTE[4] and random sampling to create aharmonious balance for the dataset. The essence of the technique is the use of feature selection, which is largely basedon information acquisition rate. To ensure the utmost effectiveness, the researchers conducted a series of rigorous experiments employing three popular machine learning algorithms, namely the K-Nearest Neighbour, Decision Tree,and Random Forest. The results of these experiments unequivocally point towards an optimal feature selectionscheme, thereby overcoming the perennial challenge of data imbalance in network intrusion detection datasets.

Shadman Latif et al. conducted a thorough exploration using the NSL-KDD dataset [5]. The research mainly concentrated on using different machine learning approaches such as AdaBoost, Decision Tree, Support VectorMachine, Naïve Bayes, Random Forest, and Neural Networks. The researchers employed a comprehensive approach to data preprocessing, encompassing the conversion of categorical features into numeric representations using one-hot encoding. Additionally, they experimented with diverse feature scaling techniques and excluded Naïve Bayes,due to its subpar performance. The novel part of the study was the deployment of several sampling strategies to produce machine learning pairings that were scaled, feature-reduced, and over-sampled.

Sharafaldin et al. created a fresh dataset for Intrusion detection and traffic characterization [6]. The initiative aims to address the problems that the existing datasets have, including inadequate feature sets, limited attack kinds, anonymised packet information, and a lack of diversity in traffic. The dataset includes a mix of benign and prevalent attack network flows, presenting an exhaustive

set of scenarios for assessment. In addition, the researchers have thoroughly assessed the machine learning methods and network traffic aspects. This comprehensive analysis aims to empower the community with a robust tool for detecting andcategorizing a spectrum of attacks, thereby augmenting network security against growing threats.

Miel Verkerken reviewed and presented a novelmulti-stage approach for hierarchical intrusion detection [2][7]. In the era of digital transformation, there is a demandfor robust intrusion detection systems which identifies previously unknown, zero-day attacks. The study offers a multi-phase hierarchical intrusion detection system. The method is validated using two publicly available, well-knowndatasets CIC-IDS-2017 and CSE-CIC-IDS-2018. The methodology achieves an outstanding performance in classification with 96% balanced accuracy, outperforming both baseline methods and existing approaches. The flexibility of this method is that it does not require retrainingand it uses n-tier installations to reduce bandwidth and compute costs while adhering to strict privacy limitations, is especially remarkable. The top-performing model in the study reduced bandwidth demands by as much as 69% whileaccurately categorizing 41 out of 47 zero-day assaults with an astonishing 87% accuracy. This research presents a significant leap forward in the field of intrusion detection, offering an efficient and powerful tool to enhance network security.

Garcia Teodoro et al. conducted a comprehensive review on anomaly-based Network Intrusion Detection [8]. The increasing proliferation of security threats across the internet and computer networks emphasizes the necessity for the evolution of adaptable and flexible security approaches. Anomaly-based network intrusion detection solutions areuseful in this situation to protect target systems from malicious activity. Moreover, they emphasized the existence of accessible platforms, continuing efforts to construct systems, and research initiatives in this field. The major contribution is to outline the key obstacles to the widespread utilization of anomaly-based systems for intrusion detection,with a focus on the evaluation of these technologies. It offers insights into the landscape of anomaly-based intrusion detection, which is crucial for enhancing network security.

Many literatures that present a comprehensive review on various security threats and classification of malware attacks, vulnerabilities, and detection techniques areavailable [9][10][11][12][13]. The works delve into the growing concerns surrounding security threats in the context of both wired and wireless networks. They emphasize the disruptive impact of malicious behaviour exhibited by nodes under attack on network operations. To counter such malevolent behaviours, the works outlines multitude of security resolutions that have been developed. It specifies the role of malware in security threats conceming both computersand the internet. The studies provide an elaborate vision of many types of malwares, vulnerabilities, and the currentdefences against these security threats.

III. METHODOLOGY

A. *Datasets*

As an improvement on their previous NSL-KDD dataset which was initially drawn from the well-known KDD99 dataset Sharafaldin et al. [3] created the new dataset called CIC-IDS-2017. The primary objective behind the creation of CIC-IDS-2017 was to adhere to a set of 11 specific criteria essential for a robust intrusion detection dataset, rendering it a valuable resource for benchmarking purposes. This dataset was meticulously constructed over a span of 5 days, employing 14 machines, and encompasses both legitimate and illegitimate network traffic. The various aspects of the dataset are summarized in Fig.1.

B-profiles, which are created from the typical behavioural patterns of 25 people using statistical methods and machine learning, are used to artificially create the benign traffic present in the dataset. On the other hand, withinpredetermined timeframes, the execution of proven attack tools generates harmful traffic. Both kinds of traffic are combined into a single dataset, which is then made availablein formats such as CSV files and packet captures (PCAP).

```
print('First read version of the dataset: \n',data_value)
```

```
First read version of the dataset:
                Label
BENIGN         2273097
DoS Hulk        231073
PortScan        158930
DDoS            128027
DoS GoldenEye    10293
FTP-Patator       7938
SSH-Patator       5897
DoS slowloris     5796
DoS Slowhttptest  5499
Bot               1966
Brute Force       1507
XSS                652
Infiltration        36
Sql Injection       21
Heartbleed          11
 XSS                 1
Name: count, dtype: int64
```

```
print('First (row,Column) number of the dataset: {} '.format(data.shape))
```

```
First (row,Column) number of the dataset: (2830743, 79)
```

Fig. 1. CIC-IDS2017 Dataset value count

B. *Dataset Collection*

Using the Pandas package, a complete collection of network traffic data has been incorporated into the CIC-IDS- 2017 dataset. This dataset comprises 78 distinct features thatserve as the input variables, while the labels are used as the output variable. These labels map to several different types of network attacks. Then it has been classified as normal or anormal. The newly created dataset is named NormalAnormal_dataset.csv.

C. *Data Preprocessing*.

Preprocessing data is a crucial stage, especially when dealing with intrusion data, which frequently contains contaminates and missing values that could compromise the accuracy and efficiency of the data mining procedure. As a result, data preprocessing is done to improve the efficacy and quality of the data that is collected following the mining process. To achieve accurate results and good predictions utilizing machine learning techniques applied to the dataset, data preparation is essential. The unnecessary noises areremoved and the resulting dataset containing the clean data isshown in Fig. 2

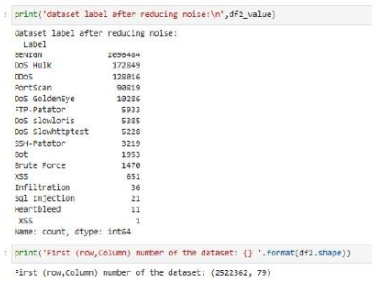

Fig. 2. Data after Preprocessing

D. *Random Sampling*

Random sampling is a key technique in data analysis, especially when dealing with large data sets. For label values, a common approach is to check the label values to ensure balanced representation in the dataset. For example, in a scenario where label values exceed 1,00,000, you can down-sample these values to a more manageable 1,00,000, ensuringa more even distribution across categories. This strategic useof random sampling techniques results in a more even distribution of label values, increasing the accuracy and reliability of data analysis.

E. *Borderline Synthetic Minority Over-Sampling Technique (SMOTE).*

The Borderline Synthetic Minority Over-Sampling Technique (SMOTE) [4][14][3] is employed to augment the minority class instances intelligently. This algorithm selectively synthesizes instances along the decisionboundary, focusing on borderline instances, thereby balancing class distributions. The implementation utilizes the imbalanced-learn library in Python, customizing the algorithm's parameters to ensure optimal oversampling. The outcome of the sampling process is shown in Fig.3.

Fig. 3. Outcome of Sampling

F. *Support Vector Machines*

Support Vector Machines [15] is a significant algorithm in machine learning. However, in case of the implementation, it is found that the deployment of SVM is less effective in case of the dataset used. There are many factors that influence the results such as a skewed normalization or overfitting of data to optimization parameters. The effectiveness of the SVM depends on the kernel and kernel parameters. It also depends on the soft margin parameter. The selection of an appropriate optimizer also plays a major role in the results obtained by using a multiclass classifier like Support Vector Machines. Any factor that impacts the choice of the above-mentioned parameters steeply affects the performance of SVMs. In this case also, the choice of certain parameters related to SVM might influence the downtrend of the results.

G. *Performance Metrics*

The evaluation of the performance of the suggested method is done using confusion matrix, as presented in the tables. The confusion matrix can yield four different outcomes: True positive (TP), False positive (FP), True negative (TN), and False negative (FN).

Accuracy: This measures the number of accurate predictionsmade by the model. For any model, it can be stated as the total number of test instances by the number of accurate forecasts.

$$\text{Accuracy} = TP + / TP + FP + TN.$$

Precision: Precision is the ratio of accurate positive predictions to all positive predictions.

$$\text{Pre}cison = TP / TP + TN$$

Recall: Recall is the difference between total positive forecasts and actual positive values.

$$\text{Re}call = I/TP + FN$$

F1-score: This score, which considers both recall and precision, is defined as follows:

$$F1\text{-}score = 2 * (precision * recall / precision + recall)$$

IV. IMPLEMENTATION OF PROPOSED SYSTEM

A. *Live Packet Capture using Wireshark*

In Wireshark, WinPcap 4.1.3 tool is used to capture thepackets within predetermined timeframes. The captured packets are made available in formats such as CSV files and packet captures (PCAP) for further processing. Fig.4 shows an instance of packet capture in the system.

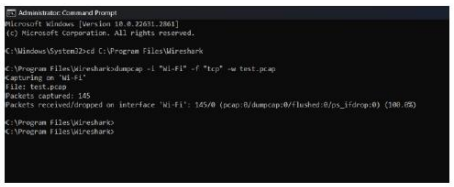

Fig. 4. Wireshark Packet capture using command line interface

B. CICFlowMeter

To transform the raw PCAP files into bidirectional flows, the CICFlowMeter tool is employed. A "biflow" is a network connection that combines and calculates Eighty network features from all the packets that are sent across that connection. The source IP, destination IP addresses, ports, and the timestamp are used to uniquely identify it. The primary purpose of this dataset is to serve as a comprehensive and dependable resource for the assessment of intrusion detection algorithms and systems. Fig. 5 shows the capturing of packets by the CICFlowMeter.

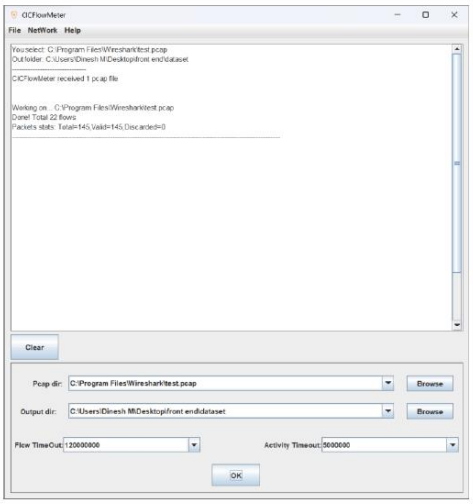

Fig. 5. CICFlowMeter Packet capture

C. Database setup

The database is set up using MySQL database. The database can be accessed with ease through the phpMyAdmininterface. The purpose of the database is to store user information registered with the intrusion detection system

and to store the intrusion incidents recorded by the system. The schema is presented in Fig.6.

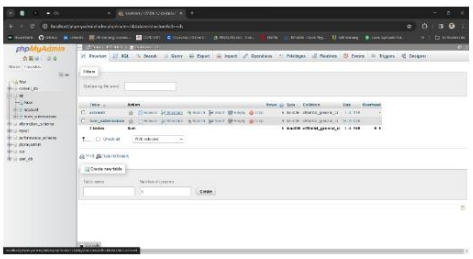

Fig. 6. Database Schema

D. Web Application

A simple web application to process and to display the results to the user is designed. The web application is interfaced with the CICFlowMeter and the csv files produced by the software are sought as input by this application and is processed. Fig.7 shows the design of the application. This page is accessible only to registered users.

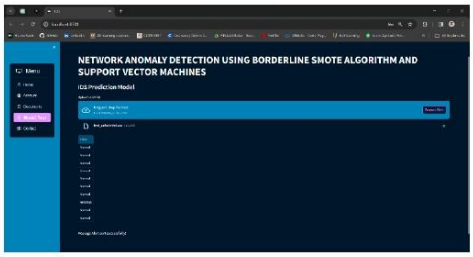

Fig. 7. Web Application

V. EXPERIMENTAL RESULT

The system was experimented with various inputs that were sought through the CICFlowMeter. A minor setback with the system is that the data obtained through the CICFlowMeter cannot be fed directly to the developed system. The csv or pcap file obtained is fed at regular intervals to the system enabling it to alert the user at regular intervals if an incident occurs.

The users registered with this system also get an email alert from the system, which is shown in Fig. 8.

Mail Alert.

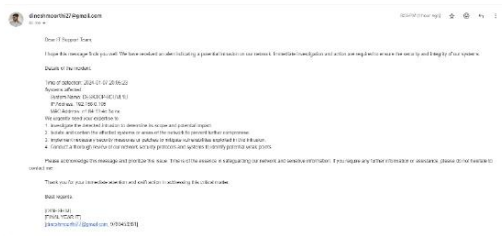

Fig. 8. Email alert for registered user

The email alert is sent to the user if and only if an intrusion is attempted or an abnormal traffic is encountered.

The performance of the proposed system is measured in terms of precision, recall and f1- score. The accuracy is also measured. The True positive (TP), False positive (FP), True negative (TN), and False negative (FN) measures are represented as a confusion matrix. It could be observed in Fig.9, that there are 18500 True Positive outputs, 20352 TrueNegative outputs, 663 False Positive outputs and 2555 FalseNegative outputs. It could be observed that the False Positiveand False Negative outputs are comparatively less than the True Positive and True Negative outputs.

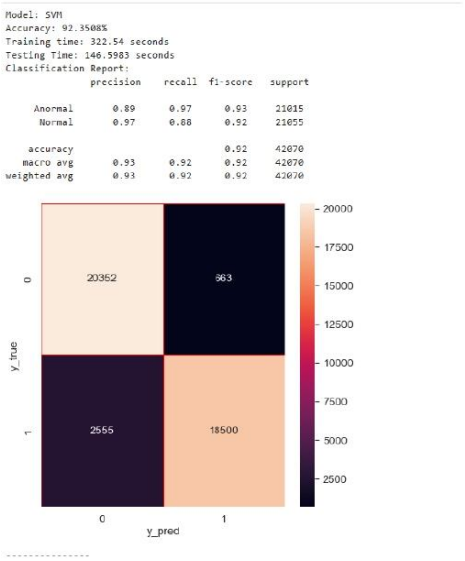

Fig. 9. Performance Analysis

Further, it is observed that the proposed solution producesresults with an accuracy of 92.35%. This may not be as goodas the system proposed by Miel [7] but is better compared to most other existing systems. It could be inferred that some ofthe parameters selected might not have effectively contributed to the decision making in the system.

VI. Conclusion

In conclusion, this study underscores the criticality of robust network security measures in safeguarding sensitive business data in the digital age. The proposed Synthetic Minority Over-Sampling Technique and Support Vector Machines (SMOTE-SVM) model demonstrates a commendable anomaly detection accuracy of 92.35%, reaffirming the efficacy of machine learning techniques in fortifying network security. The successful application of the SMOTE-SVM model not only enhances anomaly detection but also presents potential implications for wider deployment in practical network security setups. This research contributes substantively to advancing the understanding about the field and practical implementation of innovative approaches to combat evolving network threats, marking a significant stridetoward more resilient network infrastructures.

VII. Future Work

The future enhancements revolve around the seamless integration of live data into the model for real-time prediction. The aim is to elevate our monitoring capabilities by adopting more sophisticated analysis techniques, such as deploying deep learning models for anomaly detection and harnessing big data frameworks to expedite data processing. Furthermore, the focus lies on the development of automation and response mechanisms. This includes the implementation of automated incident response protocols that enable swift mitigation or containment of threats upon their detection.

References

[1] N. Meemongkolkiat and V. Suttichaya, "Analysis on network traffic features for designing machine learning based IDS," in *Journal of Physics: Conference Series*, 2021, vol. 1993, no. 1, p. 12029.

[2] M. Verkerken, L. D'hooge, T. Wauters, B. Volckaert, and F. De Turck, "Towards model generalization for intrusion detection: Unsupervised machine learning techniques," *J. Netw. Syst. Manag.*, vol. 30, pp. 1–25, 2022.

[3] Y. Sun *et al.*, "Borderline smote algorithm and feature selection-based network anomalies detection strategy," *Energies*, vol. 15, no. 13, p. 4751, 2022.

[4] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-SMOTE: a newover-sampling method in imbalanced data sets learning," in *International conference on intelligent computing*, 2005, pp. 878–887.

[5] S. Latif, F. F. Dola, M. D. Afsar, I. J. Esha, and D. Nandi, "Investigation of Machine Learning Algorithms for Network Intrusion Detection.," *Int. J. Inf. Eng. \& Electron. Bus.*, vol. 14, no. 2, 2022.

[6] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization.," *ICISSp*, vol. 1, pp. 108–116, 2018.

[7] M. Verkerken *et al.*, "A Novel Multi-Stage Approach for Hierarchical Intrusion Detection," *IEEE Trans. Netw. Serv. Manag.*, 2023.

[8] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Comput. \& Secur.*, vol. 28,no. 1–2, pp. 18–28, 2009.

[9] S. Divya, "A survey on various security threats and classification of malware attacks, vulnerabilities and detection techniques," *Int. J. Comput. Sci. \& Appl.*, vol. 2, no. 04, 2013.

[10] N. Das and T. Sarkar, "Survey on host and network based intrusiondetection system," *Int. J. Adv. Netw. Appl.*, vol. 6, no. 2, p. 2266, 2014.

[11] M. Ahsan, K. E. Nygard, R. Gomes, M. M. Chowdhury, N. Rifat, and J. F. Connolly, "Cybersecurity threats and their mitigation approaches using Machine Learning—A Review," *J. Cybersecurity Priv.*, vol. 2, no. 3, pp. 527–555, 2022.

[12] C.-C. Sun, D. J. S. Cardenas, A. Hahn, and C.-C. Liu, "Intrusion detection for cybersecurity of smart meters," *IEEE Trans. Smart Grid*, vol. 12, no. 1, pp. 612–622, 2020.

[13] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.

[14] N. V Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, 2002.

[15] P. Watanachaturaporn, P. K. Varshney, and M. K. Arora, "Evaluation of factors affecting support vector machines for hyperspectral classification," in *the American Society for Photogrammetry \& Remote Sensing (ASPRS) 2004 Annual Conference, Denver, CO*, 2004.