IDENTIFICATION AND COUNTERACTION OF BENIGN, DDOS, DOS, AND MITM UTILSING AI CALCULATIONS

A PROJECT REPORT

Submitted by

VELLANKI VENKAT ADITYA [RA2111003011799]

Under the Guidance of

Dr. S. Nikkath Bushra

Assistant Professor, Department of Computing Technologies

in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE ENGINEERING

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

DEPARTMENT OF COMPUTING TECHNOLOGIES
COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR- 603 203

NOVEMBER 2024

Department of Computing Technologies
SRM Institute of Science & Technology
Own Work* Declaration Form

This sheet must be filled in (each box ticked to show that the condition has been met). It must be signed and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done.

To be completed by the student for all assessments

Degree/ Course	**: B-Tech Computer Science and Engineering**
Student Name	**: Vellanki Venkat Aditya**
Registration Number	**: RA2111003011799**
Title of Work	**: Identification and Counteraction of Benign, DDOS, DOS AND MITM utilising AI calculations**

We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism as listed in the University Website, Regulations, and the Education Committee guidelines.

We confirm that all the work contained in this assessment is our own except where indicated, and that we have met the following conditions:

- Clearly referenced / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:

I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.

V Venkat Aditya
RA2111003011799

S Karthikeya Reddy
RA2111003011812

Date:

If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

SRM

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that 18CSP107L - Minor Project [18CSP108L- Research] report titled "**Identification and Counteraction of Benign, DDOS, DOS, and MITM utilising AI calculations**" is the bonafide work of "**VELLANKI VENKAT ADITYA [RA2111003011799], SUDARAKUNTLA KARTHIKEYA REDDY [RA2111003011812]**" who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

[illegible]

Dr. S. Nikkath Bushra
SUPERVISOR
Assistant Professor
Department of Computing Technologies

[illegible]

Dr. K. Sreekumar
PANEL HEAD
Associate Professor
Department of Computing Technologies

of Computing Technologies [illegible]

[illegible]

Dr. G. Niranjana
HEAD OF THE DEPARTMENT
PROFESSOR
Department of Computing Technologies

ACKNOWLEDGEMENTS

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to **Dr. T. V. Gopal,** Dean-CET, SRM Institute of Science and Technology, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor and Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We encompass our sincere thanks to, **Dr. M. Pushpalatha**, Professor and Associate Chairperson, School of Computing and **Dr. C. Lakshmi,** Professor and Associate Chairperson, School of Computing, SRM Institute of Science and Technology, for their invaluable support.

We are incredibly grateful to our Head of the Department, **Dr. G. Niranjana**, Professor and Head, Department of Computing Technologies, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We are incredibly grateful to our Academic Advisor, **Dr. B. Muruganantham**, Associate Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for his suggestions and encouragement at all stages of the project work.

We want to convey our thanks to our Project Coordinators, **Dr. K Sreekumar**, Panel Head and **Dr. A. Sibi** and **Dr. S. Nikkath Bushra**, Panel Members, Department of Computing Technologies, SRM Institute of Science and Technology, for their inputs during the project reviews and support.

We register our immeasurable thanks to our Faculty Advisor, **Dr. P. Saravanan**, Assistant Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide, **Dr. S. Nikkath Bushra**, Assistant Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under her mentorship. She provided us with the freedom and support to explore the research topics of our interest. Her passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank all the staff and students of Computing Technolgies, School of Computing, S.R.M Institute of Science and Technology, for their help during our project. Finally, we would like to thank our parents, family members, and friends for their unconditional love, constant support and encouragement.

V. Venkat Aditya

ABSTRACT

An increasingly well-connected digital world brings with it more and more cyber-threats that may comprise data integrity and security of any network. Such types of common cyber-attacks include DDOS (Distributed Denial of Service), DOS (Denial of Service), MITM (Man-in-the-Middle). They are constantly being used to disrupt offered services, gain access to confidential data, and, thereby, jeopardise the entire network system reliability. Traditional security systems do not manage to keep abreast of the dynamic nature of these threats and how attackers keep on updating their tactics to evade detection. Therefore, there is a great need for much more advanced or automated systems that can detect cyber-attacks and remediate them immediately in real-time. The target is real-time protection of the infrastructure in the network.

This project brings in a much more robust, machine learning-based framework for detecting and mitigating such threats with high accuracy. This system suggests a layering architecture on the application of two algorithms that work pretty well. They include XGBoost and KNN. This XGBoost is an algorithm categorised in the class of gradient-boosting machines. Being fast and strong for predictions makes this algorithm great for detecting these specific attacks like DDOS, DOS, or even MITM attacks and, as their features describe the characteristics of these respective attack types and kind of behaviour demonstrated during their activities. This is the central part of the model, working mainly as the classifier, which identifies various malicious activities.

The system has an additional classifier known as K-Nearest Neighbours (KNN) in it that distinguishes benign and malignant network behaviours further. KNN can classify the data based on its similarity to previous instances, hence making it suitable for identification of benign traffic, reduces false positives, and minimises the interruptions of regular network operations. These algorithms, together, ensure a holistic multi-layered cyber threat detection approach that helps in precise classification of threats and efficient handling of legitimate traffic.

The system was stringently tested based on real-world and achieved a respectable 94.88% in the identification and classification of different types of cyber threats. This level of accuracy alone validates the effectiveness of the algorithms and highlights the aptness of the framework to be deployed in a live network environment where fast and reliable threat detection is important. This project contributes a scalable and adaptable solution to the evolving field of cybersecurity by combining the strengths of XGBoost and KNN, thus providing a solid foundation for future advancements in automated threat mitigation systems.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

DDOS - Distributed Denial of Service

DOS - Denial of Service

MITM - Man-in-the-Middle

ML - Machine Learning

KNN - K-Nearest Neighbour

XGBOOST - Extreme Gradient Boosting

IDS/IPS - Intrusion Detection System/Intrusion Prevention System

BP - Back Propagation

LMI - Linear Matrix Inequality

AETM - Adaptive Event-Triggered Mechanism

TTD - Time-To-Defend

LDOS - Low-Rate Denial of Service

SDN - Software Defined Networking

MSR - Modular Square Root

ERF-KMC - Ensemble Random Forest-Kmeans Clustering

SON - Software-Defined Networking

NB - Naïve Bayes

CHAPTER 1

INTRODUCTION

1.1 Project Overview

Regarding this, most current research concentrates on identifying and categorizing DDOS attacks using various mechanisms and a single controller. They also concentrate on accuracy or efficiency rather than both. In data centers, several controllers must be shielded from Benign, DDOS, DOS, and MITM attacks. The tolerance level for network traffic varies among various controllers. One way to conceal the identity of the attacker in an attack of this nature is to spoof the source, often known as a phony source address. For the malicious packets to be served, the attackers also try to overwhelm the victim with false packets. The following are the reasons behind these attacks: When there is a conflict between two groups or two individuals,[1] DDOS and DOS are powerful weapons that can be used to obstruct the opponent's applications and infrastructure. An individual may also deliberately become an attack and carry out unwanted activities in response to a perceived injustice through this attack. Finally, a terrorist cell may attempt to attack some sensitive zones through cyber warfare (which is motivated by politics or geopolitics) to destroy the economic system. Several methods for identifying, categorizing, and reducing DDOS, DOS and MITM attacks using machine learning algorithms are known in the literature.

Thus, DDOS, DOS, and MITM attacks become an important threat to the availability and reliability of services delivered online in the fast pace of cyber threats. The intent of the offenses is to flood target systems or networks with traffic to such a point that the intended systems or networks cannot be reached by authorized users.

Traditional methodologies of DDOS, DOS, as well as MITM attack identification and counteraction are mostly static and not that sophisticated compared to these threats. The simplified research works simplifies the finding of Benign, DDOS, DOS, as well as MITM attacks through easier as well as more effective ways. It also shows that the machine learning approach is quite efficient. The results have solely relied on a single data set those forms one of the model's limitations. Hence, one can analyse a biased dataset to provide guidance on future improvements.

1.2 Motivation

The motives behind this project implementation arise from an ever-increasing cyber threat phenomenon in the digital world. DDOS, DOS, and MITM attacks are just some of the impressive risks in security in terms of the possibility of these threats invading data: insecurity, availability, and confidentiality. Online services and networks become critical factors in securing their integrity, interwoven with the fast pace of growing digital transformations across all industries.

Traditional security mechanisms have only proved to be effective in fighting cyber-attacks partly, as the cleverness of cyberattacks is increasing rapidly. Therefore, this project envisions implementing AI-based more robust, adaptive, and scalable defense through a myriad of AI algorithms that can speed up the identification of abnormal patterns of behavior, detect potential threats in real time, and execute the right countermeasures before any significant damage has occurred.

Since it employs AI, the system is in constant learning, and thus, it evolves based on new and emerging threat vectors. This comes to correct the disadvantages found in traditional security methods wherein it relies on pre-defined rules, and which can be slow to react to new threats. This project aims at contributing to a safer digital ecosystem by creating an intelligent solution which can fight complex attacks such as DDOS, DOS, and MITM.

In addition, the knowledge and innovations gained here could possibly form the kernel on which future breakthroughs in cyber security will evolve to assist further research work for academia and to be translated into new practices for the business world.

CHAPTER 2

LITERATURE SURVEY

Research on Detection and Defense Mechanisms of DoS Attacks Based on BP Neural Network and Game Theory. The BP neural network DoS attacks detection model uses KDDCUP99 as the dataset and selects multiple feature vectors from the dataset that can efficiently identify DoS attacks by large-scale training. Algorithms: The DoS attacks detection model based on the BP neural networks
The simulation results show that the proposed detection method and defence strategy are effective for DoS attacks which improves the accuracy of detecting DoS attacks to 99.977%. Denial of Firewalling Attacks (DoF): The Case Study of the Emerging BlackNurse Attack (2019). This research also concentrates on BlackNurse attack principles, practical attack generation, and its general effect on impacted firewalls and the networks. The performance evaluations are conducted on commercial grades, namely, Juniper NetScreen SSG 20 and Cisco ASA 5540 firewalls. Algorithms: Time-To-Defend (TTD) Modelling. The evaluation is conducted to simulate the proposed mechanism of defence against novice and expert BlackNurse attackers. Low-Rate DoS Attacks, Detection, Defence, and Challenges: A Survey (2020). Classifies the LDOS attacks and existing defence methods according to time domain and frequency domain in which detection and defence are performed. Algorithms: Naïve Bayes, Support Vector Machines. The proposed algorithms have accuracy values of 75.33% and 63.75% respectively. Ddos attack detection method based on improved KNN with the degree of ddos attack in software-defined networks (2020). Method uses the improved k-nearest neighbours (KNN) algorithm based on machine learning (ML) to discover the ddos attack. Algorithms: method uses the improved K-Nearest Neighbours (KNN). Test results of test-bed experiments indicate that the accurate detection probability of proposed approach is 88.69%
Detecting dos attacks based on multi-features in SDN (2020). A new approach by extracting six features of the flow table and using the back propagation (BP) neural network to construct the classifier. Algorithms: Software Defined Network (SDN) architecture. Test results of test-bed experiments indicate that the accurate detection probability of the proposed approach is 98.9%. A Survey of Denial-of-Service Attacks and Solutions in the Smart Grid (2020). Provides a holistic and methodical presentation of the DoS attack taxonomies as well as a survey of potential solution techniques to help draw a more concentrated and coordinated research into this area, lack of which may have performed consequences. Algorithms: Naive bayes, Decision Tree. The proposed algorithms have accuracy values of 67.11% and 83.54% respectively. An Efficient IDS Framework for DDoS Attacks in SDN Environment (2021). This mechanism, transmissions occur while a specified event is triggered to prevent unessential utilization of communication resources.

The asymptotic stability of the ET-based controller is shown formally by using Lyapunov stability via linear matrix inequality (LMI) conditions. Algorithms: Software Defined Network (SDN) architecture. The proposed algorithms have accuracy values of 89.36% and 73.02% respectively. An Improved Protocol to Consensus of Delayed MASs with UNMS and Aperiodic DoS Cyber-Attacks (2021). Secondly, the changes of communication topologies due to frequently aperiodic DoS cyber-attacks are taken into consideration, which may destroy the chains of communication and lead to network paralysis in MASs. Algorithms: Random Forest Algorithm, SVM. The proposed algorithms have accuracy values of 98.36% and 75.02% respectively.

Detection and Mitigation of Low-Rate Denial-of-Service Attacks: A Survey (2022). Evolving type of attack, known as LDoS (Low-rate Denial of Service) attacks, has the potential to produce more damage than its predecessor due to its stealth nature and the lack of suitable detection and defence methods. Algorithms: Support Vector Machine (SVM), Random Forest (RF). The proposed algorithms have accuracy values of 93.80% and 88.55% respectively. Anomaly Detection IDS for Detecting DoS Attacks in IoT Networks Based on Machine Learning Algorithms (2022). According to the adaptive threshold technique, a resilient adaptive event-triggered mechanism (AETM) is developed to reduce the transmission frequency and combat the aperiodic DoS attacks. Algorithms: Random Forest, Support Vector Machine. The proposed algorithms have accuracy values of 99.80% and 98.55% respectively. MSR-DoS: Modular Square Root-Based Scheme to Resist Denial of Service (DoS) Attacks in 5G-Enabled Vehicular Networks (2022). This research proposes modular square root-based to resist denial of service (DoS) attacks (MSR-DoS) scheme in 5G-enabled vehicular networks. Algorithms: Modular Square Root (MSR) Algorithms. The proposed MSR-DoS scheme reduces the computation overhead of signing the message and verifying the message by 99.80% and 98.55% respectively. LSTM-Based Collaborative Source-Side DDoS Attack Detection (2022). Proposed framework applies LSTM-based adaptive thresholds to each source-side network to mitigate performance degradation caused by irregular network traffic behaviour. Algorithms: KNN, Decision Tree. The proposed algorithms have accuracy values of 78.08% and 81.69% respectively. Observer-Based Secure Control for Vehicular Platooning Under DoS Attacks (2023)
To deal with DoS attacks, it considers an observer-based mechanism to estimate the state of vehicles based on available sensor measurements which significantly improves the resilience and tolerance of the platooning system during the attack interval. Algorithms: K-Means Clustering, ERF-KMC Algorithm. The proposed algorithm has accuracy values of 85.33% and 83.75% respectively.

CHAPTER 3

PROBLEM STATEMENT

Advanced threats are constantly changing the landscape of modern cybersecurity. This includes Denial of Service and Distributed Denial of Service attacks, that are disruptive in nature because they affect the availability and reliability of targeted systems and networks. In other words, these are designed to overflow the targeted system or network, hence blocking legitimate users from gaining access to its services. This is because traditional defense mechanisms such as signature-based or rule-based detection approaches are useless in these attacks because they are dynamic and complex. Therefore, there is a greater need for more adaptive and more intelligent solutions that would look at differentiating benign from malicious network traffic.

It is expected that the system applies machine learning techniques for the mitigation and the detection of DDOS, DOS, MITM, and any other type of attacks, as well as benign traffic. This system would improve the accuracy and response time in the attack detection. Machine learning algorithms such as KNN and XGBoost can be used in real-time network traffic data analysis to identity complex attack patterns that otherwise evade detection through conventional methods. The system, therefore, can dynamically identify and neutralize malicious activity, thereby strengthening the resilience of the network against emerging cybersecurity threats.

Cyber security uses methods of machine learning and can adapt and do highly intelligent analysis on the fly based on network traffic, contrary to methods that rely solely upon the pattern recognition process; those do learn real time based on emerging novel threats. For example, machine learning algorithms like KNN and XGBoost might find patterns of very multi-dimensional attacks, hence very sophisticated attacks that even through some forms of rule-based models are not recognised in its class. The training of these models on such a diverse dataset causes the system to learn and recognise fine subtleties that distinguish ordinary traffic from anomalous activity in an attack. This makes proactive detection not only more accurate in terms of identifying attacks but also less prone to false positives in the process, thereby protecting legitimate users from being cut off from the network service. Furthermore, it can respond to threats quicker by automating the process of detection and response so that the damage may be reduced as much as possible, making it a more resilient cybersecurity framework.

CHAPTER 4

PROPOSED MODEL

The solution under consideration employs a multi-tiered approach, combining advanced machine learning algorithms with real-time data analysis to detect and counteract threats such as DDoS, Man-in-the-Middle (MITM), and benign ones. The methodology comprises multiple crucial elements, one of which is alerting others of an attack.

1. Real-Time Data Collection and Processing:

Data Sources - Several network traffic sources, including routers, firewalls, intrusion detection/prevention systems (IDS/IPS), and network logs, provide real-time data to the system.

Preprocessing - Raw data undergoes preprocessing steps, such as data cleaning, normalization, and feature extraction, to ensure high-quality inputs for machine learning models. Packet capture can be performed using tools such as Wireshark or Tshark whereas preprocessing is done using Python bandwidth libraries like Pandas or Scikit-learn.

2. Creating new features:

Some useful features like size of packets, source and destination Ips, number of requests nd types of said requests etc are extracted and altered to optimise the given model’s ability to detect and classify attacks.

Using, for instance, NumPy and Scipy, statistical and temporal features are also calculated to detect abnormal behaviour of the network.

3. Choosing and Educating of Machine Learning Model:

Supervised Learning – As a first step, and a few standard supervised learning techniques are applied on labeled datasets for many times wherein Support Vector Machines (SVM), K-Nearest Neighbours (KNN) and the like techniques are employed.

Unsupervised Learning – In addition to the above, several kinds of unsupervised learning are also used by the system for the purpose of mimicking or generating ‘new’ attacks where in the case if K-Means clustering is used for the Autoencoders application to detect traffic patterns anomalies.

The training takes place with the use of Scikit-learn and TensorFlow/Keras especially for the models which are deep learning based, while Jupyter Notebooks are used as the working environment.

4. Model Evaluation and Tuning

Model metrics for example accuracy, precision, recall, and F-1 score are used to evaluate the models developed to ensure that there is good separation between the normal and the attack data.

Hyperparameters are tuned using Cross-Validation and Grid Search techniques as applicable, those techniques increasing the performance of models with respect to detection accuracy and response time.

5. Real-Time Monitoring and Counteractions

The system can perform real time classification of the incoming and outgoing network traffic using the trained models.

In case of an attack, the system quickly adapts network parameters such as rerouting existing path, changing firewall settings or altering access control policies, when required to reduce the level of the threat. Open Daylight and ONOS network management systems, for instance Software Defined Networking (SDN) controllers, allow for reconfiguration of the network on the fly.

6. Clinical Practice Guidelines for the Management of Anomalies and Diabetic Foot: A Combination of Anomaly Detection Based and Signature Detection Based Attacks

The system applies hybrid detection shields comprising also signature detection techniques, to increase hallmark detection capabilities and decrease any false alarm rates.

In this case, Snort or Suricata tools are designed for the signature-based approach detection, while the systems are additionally supported by the anomaly detection models to present new forms of undesired weapons previously unknown.

7. The Progression of Endless Learning and Tuning

The new attack approaches identified with the help of unsupervised techniques in this approach are added to the supervised learning phase which aids in reloading and retraining the incumbent models from time to time.

This helps in the faster recognition of the threats and helps in updating the system based on the recognition of different threats.

8. Deployment and Governance Monitoring

The entire framework is deployed on a scalable architecture and microservices using Docker and Kubernetes enabling deployment and scaling across different networks with ease.

Prometheus and Grafana help in monitoring the performance of the system and the metrics of attack detection by providing the user with an up-to-date view of the security status.

This multi-tiered approach from the proposed solution will afford an effective defense strategy that adjusts to different types of cyber threats. The architecture for this system is proactive in addition to reactive, giving full protection at the important stages of detection, mitigation, and response. Since it combines real-time processing with machine learning models, the system can identify emergent attack patterns in real time and hence reduce the window open to attackers to exploit a given vulnerability. In addition, the system evolves with the threat landscape through its adaptability and ensures that its defenses remain effective as new attack strategies are developed.

The system has the capability to alert stakeholders and automatically take countermeasures when an attack is detected. The integration of tools such as Open Daylight and ONOS gives the ability to do quick reconfiguration of a network in response to attacks. For instance, it will reroute traffic and maybe change firewall rules and modify access control policies whenever the system detects a DDOS attack. What really makes the system very relevant in large-scale networks is its capability to respond autonomously. In such a network, responding manually will be too slow, and clearly, this system shows the strength of rapid autonomous mitigation of threats.

However, this is more than just responding to threats. Because the system learns and models get updated, it can counter the changing threats in real time. For greater accuracy over time, attack signatures found by unsupervised learning are fed into supervised learning models. As such, because of the loop of learning, the model is always training for finding new threat signatures so that it does not become dependent on history. Such kind of proactive framework of adaptive learning will help in huge efficiency by thwarting unknown types of attacks, due to which the system will not hold zero-day vulnerabilities and very complex adaptive attack vectors mainly.

CHAPTER 5

SYSTEM ARCHITECTURE AND METHODOLOGY

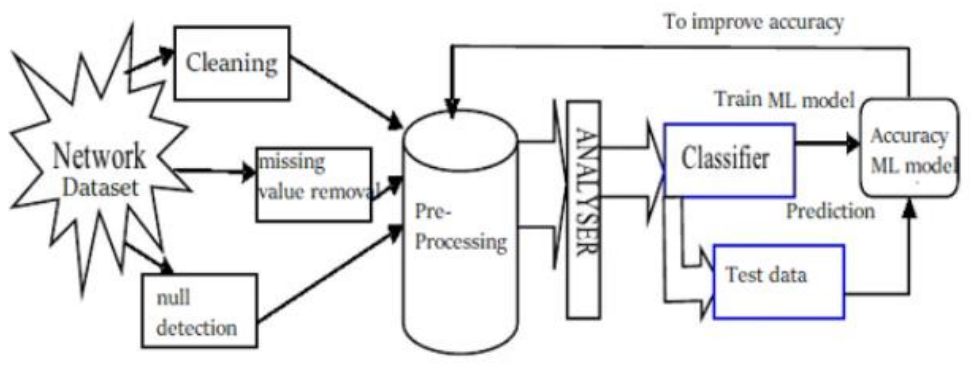


Fig. 1 – System Architecture for Real-Time Detection and Prevention of Benign, DdoS, DoS, and MiTM Attacks Using Machine Learning

This architecture aims to real-time identify and counteract upon benign, DOS, DOS, and Man-in-the-Middle (MITM) attacks by creating a solid framework that applies the potential power of machine learning for effective identification and mitigation. Therefore, in the whole architecture, the main entity that will comprise the Cybersecurity Attack Identifier is operating in the environment of the system, where rapid response with a necessity for very high accuracy becomes extremely necessary while detecting all sorts of attacks, ranging from DDOS that arises in a cloud environment. This architecture involves multiple important components that work in sequence as follows-from acquiring the data, then performing all the necessary preprocessing operations on it and analysing this such that its proper classification followed by validation may occur.

- Network Dataset Collection – This design starts with a comprehensive collection of a Network Dataset. Such a dataset can be collected from sources such as routers, firewalls, and intrusion detection/prevention systems (IDS/IPS). The raw dataset consisting of network traffic with features like packet size, source and destination IPs, types of requests is fed to the architecture. These are the features that would go into training the machine learning model. At this very first level, data cleansing passes through the sieve in a try to leave behind the lesser value or just "noise" data and save crucial information. Handling missing values also falls at this very step. Depending upon varied criteria, they can fill up the gaps or blank spaces to discover the consistency of importance and relevance by the formation of relevant training set.

- Null Detection and Handling - After data cleaning, Null Detection is performed to identify gaps in the data that may affect the accuracy of the model. In this step, null values or blanks within the dataset are identified because such may compromise the quality of the training process. The proper handling of such null values is essential to have an accurate model built; thus, any detected null values are either filled or removed. The process results in a fully processed dataset such that all entries are neither missing nor present partially; otherwise, it would skew the learning process.

- Data Preprocessing and Transformation - Following null detection, Data Preprocessing transforms and molds the dataset to suit the optimization of machine learning. The two main techniques used in preprocessing include normalization, which is the scaling of numeric values within a standard range and encoding of categorical variables like an IP address or device type so that data is prepared for input into the model. Preprocessing is an important part of ensuring consistency and compatibility within the dataset for the purposes of the needs of the machine learning algorithms.

- Feature Extraction and Engineering - This feature precedes Feature Extraction and Engineering. It extracts features from the dataset, identifying major features from different network entities as sources. Determining factors affecting attack detection accuracy include traffic frequency, time-based statistics, and pattern of behaviour. This, especially the feature engineering technique enables the model to improve its ability to differentiate between good and bad traffic by allowing it to include derived features that would capture the hidden patterns of an attack. For instance, average packet size over a period will capture the patterns and represent an attack. It is where meaningful selection of features plays a prominent role in building a model effectively to identify very complex attack scenarios.

- Selection and Training of Machine Learning Model - During the Model Training stages; after pre-processing the dataset and feature engineering, this is used for the actual training. This architecture combines Supervised Learning to utilize labeled attack data with Unsupervised Learning for the need to identify anomalies. One of the supervised methods comprises SVM, KNN, and XGBoost on supervised techniques that classify it as benign or malicious upon recognised patterns of attacks. Unscrewed methods: K-methods clustering and autoencoders. These are expected to identify anomalies as well as new kind of attacks, unrelated to the previously discovered patterns. Training models depend upon whether it is a more classical machine learning model, so Scikit-learn libraries will be used. TensroFlow/Keras should be applied in case of deep learning model.

- Model Testing and Validation – This testing involves the test dataset to be used in training the classifier. This kind of dataset contains unseen data that can simulate real situations; therefore, it checks how the model generalizes well. The metrics considered include accuracy, precision, recall, and F1 score, all of which determine how well a model can identify malicious activity. Improvements based on test results will determine the model’s parameters. Techniques such as Grid Search and Cross-Validation optimize the model for the best possible accuracy in threat identification and reduce false positives.

- Real-Time Classification and Threat Mitigation – The validated model is then deployed for Real-Time Classification and threat mitigation. This model is deployed to monitor incoming and outgoing network traffic and classify data in real time as either benign or malicious. Once a attack has been detected, the system compels the countermeasures in the form of modification to firewall configuration, updating of access policies, as well as network traffic rerouting. The reason for this is that is uses the speed with which an SDN controller may alter the configuration of a network with the purpose of least amount of disruption towards legit users.

- Hybrid Detection System with Signature-Based Integration – It includes Hybrid Detection Approach, which has machine learning-based anomaly detection and signature-based detection with known attack signatures recognised with tools like Snort or Suricata for strong detection accuracy. In this combined model, the additional security layer is provided to the signature-based tool with the identification of known attacks in addition to emerging threats while reducing the potential occurrence of false positives with enhanced detection efficiency.

- Continuous Learning and Model Update – Maintaining the pace of continuously changing attack landscapes, Continuous Learning with Model Update has variations. For unknown types of attacks or attack patterns, the system begins identifying patterns through unsupervised learning, which feeds the pattern back for supervised learning. These updates happen in real-time so that it keeps addressing the latest techniques with reasonable accuracy levels in detection capabilities.

- Deployment and System Monitoring – The architecture is finally Deployed on a Scalable Infrastructure that uses Docker and Kubernetes to make it easy to scale across different network environments. As far as monitoring is concerned, Prometheus and Grafana ensure real-time performance insight about the model's performance as well as network security status. All the above metrics are tracked; hence this provides the administrator with an idea about network traffic as well as model's accuracy about its detection. This total deployment ensures the architecture stays flexible, scalable, and responsive in fending off a wide array of cyber threats.

A multi-layered architecture includes real-time data analysis, machine learning, and automated responses in a highly adaptive framework for cloud security, making it strong in dealing with shifting security challenges in the cloud. The system incorporates the various critical stages-or steps-of data collection and preprocessing, through feature engineering to model training and validation-to create a streamlined flow that increases both accuracy and speed in threat detection. Real-time classification and hybrid detection enter the scene here, which inherently bridges both anomaly as well as signature-based methods. It thus enables both known as well as unknown threats being identified and responded to in real time while ensuring the overall downtime is seen to be kept at a minimum in network traffic.
This architecture has an important characteristic of learning constantly. The system will be able to proactively realize emerging threats that have not yet been in the known attack signatures by unleashing unsupervised learning techniques for finding new and unusual traffic patterns. With every threat data, it is immediately fed into the supervised learning algorithms, hence the model evolves with the change in continuous updating of emerging threats against a set of attackers. This incessant evolution is of extreme importance in today's cybersecurity world, which sees attackers constantly finding new ways to attack. It ensures that the architecture remains resolute and responsive, thus continuously managing to adapt dynamically to face novel attack vectors as they emerge.
The architecture in this system also provides extreme benefits, especially its scalability in that it accommodates every organization, regardless of their industry or size. With the use of technologies like Docker and Kubernetes for containerization, it can be efficiently deployed in the diverse network environments, from small setup to expansive and enterprise levels. Scalability can also be flexible through this technology; it will grow with an organization's adaptive security requirements. It can also be supported by the real-time insights that monitoring tools like Prometheus and Grafana give on network traffic and model performance, allowing data-driven decision-making by security teams. With such a strong, adaptive, and always- learning system, it secures an organisation's cloud environment against a very vast array of cyber threats, offering much peace of mind.

CHAPTER 6

TECHNICAL REQUIREMENTS

Technical requirements are critical for the development of a DNA Sequence classifier using machine learning that provides accurate and reliable air quality information to the public and relevant authorities. These requirements encompass various aspects, including hardware, software, data, and infrastructure. Collaborating with domain experts and stakeholders is essential to ensure that the system meets the specific needs of the region and adheres to environmental standards and regulations.

Hardware Requirements-

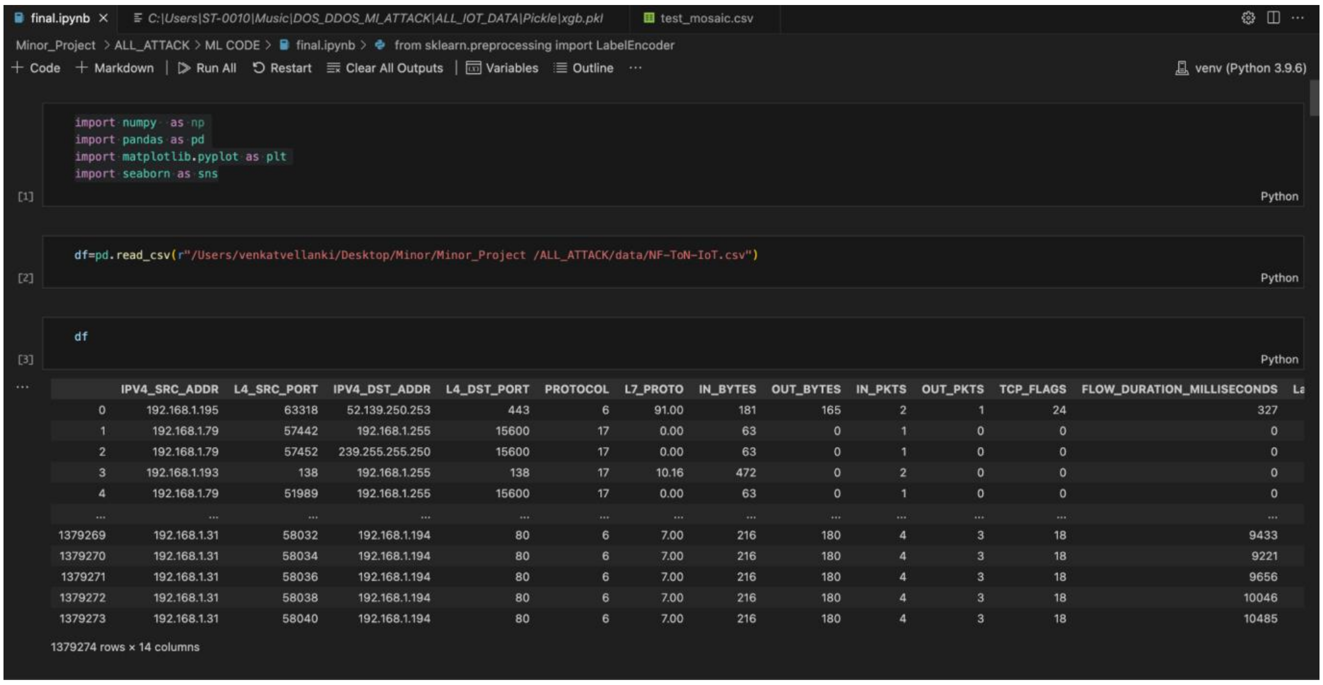
- Laptop / PC with any OS (Window 7/8/10 or, Mac OS (any version), Linux (any version)
- RAM storage of minimum 4GB and Hard Disk storage of minimum 256GB is required.
- Constant internet connection
- Laptops or PCs with an i3 processor or better.
- Virus-free environment

Software Requirements-

- Operating System: Windows 7/8/10 or Mac OS or Linux
- Programming Language: Python
- Libraries: numpy, pandas, matplot, seaborn, sklearn

CHAPTER 7

CODING AND TESTING



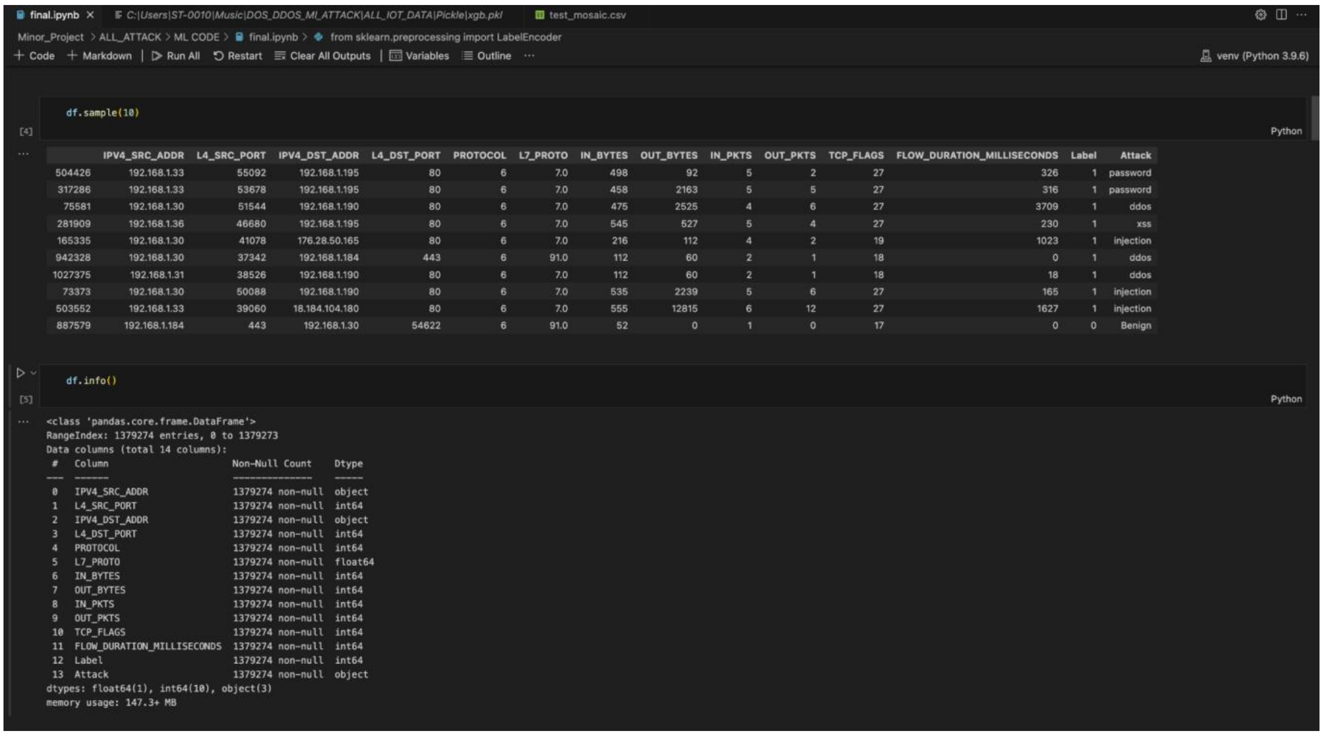
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```python
df=pd.read_csv(r"/Users/venkatvellanki/Desktop/Minor/Minor_Project /ALL_ATTACK/data/NF-ToN-IoT.csv")
```

```python
df
```

	IPV4_SRC_ADDR	L4_SRC_PORT	IPV4_DST_ADDR	L4_DST_PORT	PROTOCOL	L7_PROTO	IN_BYTES	OUT_BYTES	IN_PKTS	OUT_PKTS	TCP_FLAGS	FLOW_DURATION_MILLISECONDS	La
0	192.168.1.195	63318	52.139.250.253	443	6	91.00	181	165	2	1	24	327	
1	192.168.1.79	57442	192.168.1.255	15600	17	0.00	63	0	1	0	0	0	
2	192.168.1.79	57452	239.255.255.250	15600	17	0.00	63	0	1	0	0	0	
3	192.168.1.193	138	192.168.1.255	138	17	10.16	472	0	2	0	0	0	
4	192.168.1.79	51989	192.168.1.255	15600	17	0.00	63	0	1	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
1379269	192.168.1.31	58032	192.168.1.194	80	6	7.00	216	180	4	3	18	9433	
1379270	192.168.1.31	58034	192.168.1.194	80	6	7.00	216	180	4	3	18	9221	
1379271	192.168.1.31	58036	192.168.1.194	80	6	7.00	216	180	4	3	18	9656	
1379272	192.168.1.31	58038	192.168.1.194	80	6	7.00	216	180	4	3	18	10046	
1379273	192.168.1.31	58040	192.168.1.194	80	6	7.00	216	180	4	3	18	10485	

1379274 rows × 14 columns

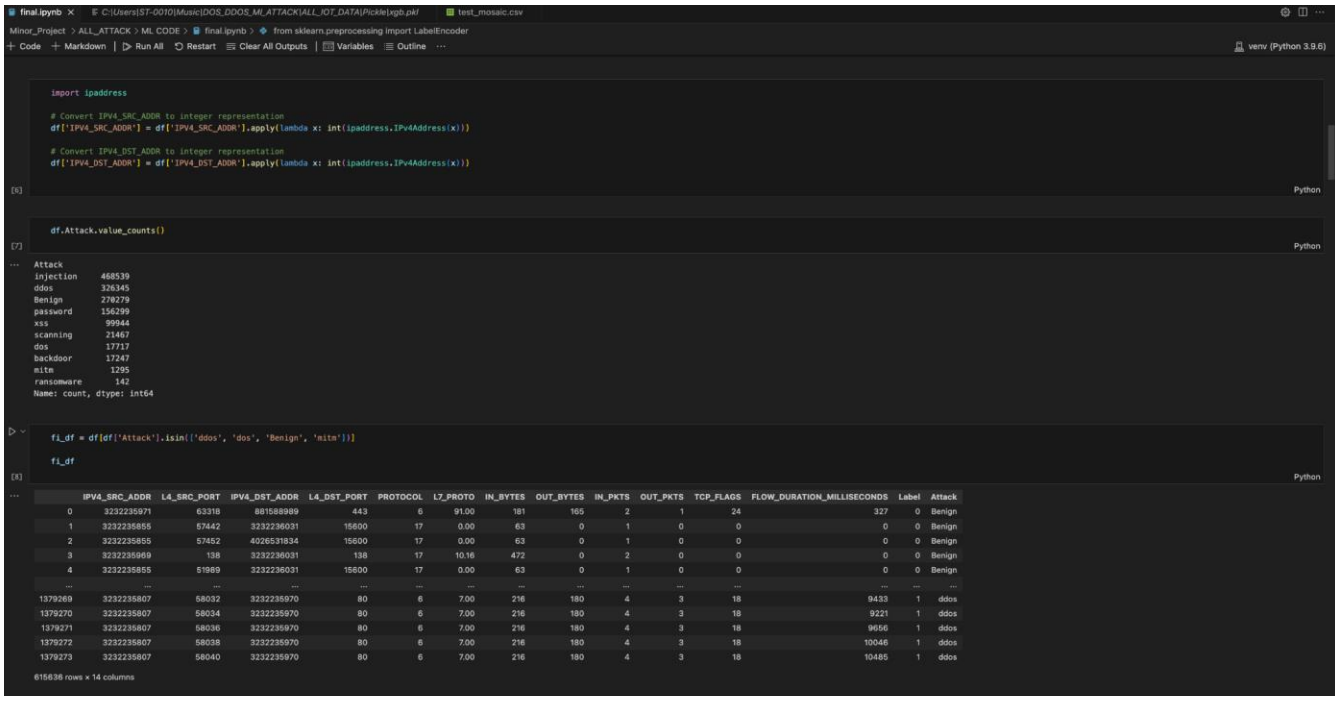


```python
df.sample(10)
```

	IPV4_SRC_ADDR	L4_SRC_PORT	IPV4_DST_ADDR	L4_DST_PORT	PROTOCOL	L7_PROTO	IN_BYTES	OUT_BYTES	IN_PKTS	OUT_PKTS	TCP_FLAGS	FLOW_DURATION_MILLISECONDS	Label	Attack
504426	192.168.1.33	55092	192.168.1.195	80	6	7.0	498	92	5	2	27	326	1	password
317286	192.168.1.33	53678	192.168.1.195	80	6	7.0	458	2163	5	5	27	316	1	password
75581	192.168.1.30	51544	192.168.1.190	80	6	7.0	475	2525	4	6	27	3709	1	ddos
281909	192.168.1.36	46680	192.168.1.195	80	6	7.0	545	527	5	4	27	230	1	xss
165335	192.168.1.30	41078	176.28.50.165	80	6	7.0	216	112	4	2	19	1023	1	injection
942328	192.168.1.30	37342	192.168.1.184	443	6	91.0	112	60	2	1	18	0	1	ddos
1027375	192.168.1.31	38526	192.168.1.190	80	6	7.0	112	60	2	1	18	18	1	ddos
73373	192.168.1.30	50088	192.168.1.190	80	6	7.0	535	2239	5	6	27	165	1	injection
503552	192.168.1.33	39060	18.184.104.180	80	6	7.0	555	12815	6	12	27	1627	1	injection
887579	192.168.1.184	443	192.168.1.30	54622	6	91.0	52	0	1	0	17	0	0	Benign

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1379274 entries, 0 to 1379273
Data columns (total 14 columns):
 #   Column                      Non-Null Count    Dtype
---  ------                      --------------    -----
 0   IPV4_SRC_ADDR               1379274 non-null  object
 1   L4_SRC_PORT                 1379274 non-null  int64
 2   IPV4_DST_ADDR               1379274 non-null  object
 3   L4_DST_PORT                 1379274 non-null  int64
 4   PROTOCOL                    1379274 non-null  int64
 5   L7_PROTO                    1379274 non-null  float64
 6   IN_BYTES                    1379274 non-null  int64
 7   OUT_BYTES                   1379274 non-null  int64
 8   IN_PKTS                     1379274 non-null  int64
 9   OUT_PKTS                    1379274 non-null  int64
 10  TCP_FLAGS                   1379274 non-null  int64
 11  FLOW_DURATION_MILLISECONDS  1379274 non-null  int64
 12  Label                       1379274 non-null  int64
 13  Attack                      1379274 non-null  object
dtypes: float64(1), int64(10), object(3)
memory usage: 147.3+ MB
```


final.ipynb
C:|Users|ST-0010|Music|DOS_DDOS_MI_ATTACK|ALL_IOT_DATA|Pickle|xgb.pkl
test_mosaic.csv
Minor_Project > ALL_ATTACK > ML CODE > final.ipynb > from sklearn.preprocessing import LabelEncoder
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
venv (Python 3.9.6)

```python
import ipaddress

# Convert IPV4_SRC_ADDR to integer representation
df['IPV4_SRC_ADDR'] = df['IPV4_SRC_ADDR'].apply(lambda x: int(ipaddress.IPv4Address(x)))

# Convert IPV4_DST_ADDR to integer representation
df['IPV4_DST_ADDR'] = df['IPV4_DST_ADDR'].apply(lambda x: int(ipaddress.IPv4Address(x)))
```

[6]

```python
df.Attack.value_counts()
```

[7]

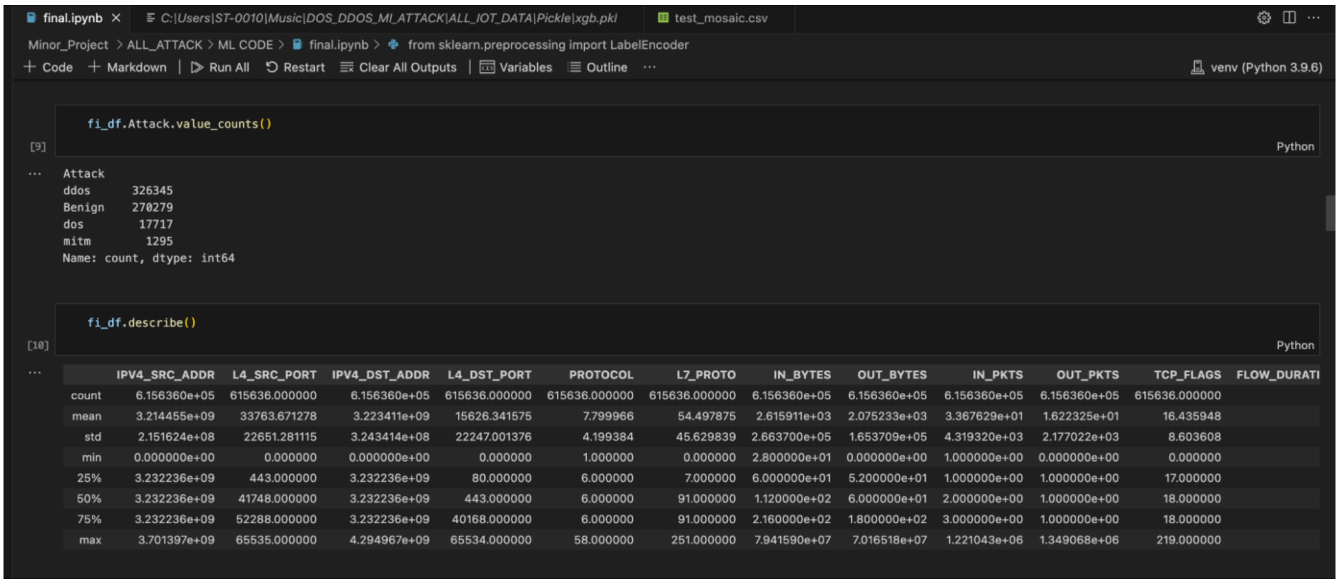
```
Attack
injection    468539
ddos         326345
Benign       270279
password     156299
xss           99944
scanning      21467
dos           17717
backdoor      17247
mitm           1295
ransomware      142
Name: count, dtype: int64
```

```python
fi_df = df[df['Attack'].isin(['ddos', 'dos', 'Benign', 'mitm'])]

fi_df
```

[8]

	IPV4_SRC_ADDR	L4_SRC_PORT	IPV4_DST_ADDR	L4_DST_PORT	PROTOCOL	L7_PROTO	IN_BYTES	OUT_BYTES	IN_PKTS	OUT_PKTS	TCP_FLAGS	FLOW_DURATION_MILLISECONDS	Label	Attack
0	3232235971	63318	881588989	443	6	91.00	181	165	2	1	24	327	0	Benign
1	3232235855	57442	3232236031	15600	17	0.00	63	0	1	0	0	0	0	Benign
2	3232235855	57452	4026531834	15600	17	0.00	63	0	1	0	0	0	0	Benign
3	3232235969	138	3232236031	138	17	10.16	472	0	2	0	0	0	0	Benign
4	3232235855	51989	3232236031	15600	17	0.00	63	0	1	0	0	0	0	Benign
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1379269	3232235807	58032	3232235970	80	6	7.00	216	180	4	3	18	9433	1	ddos
1379270	3232235807	58034	3232235970	80	6	7.00	216	180	4	3	18	9221	1	ddos
1379271	3232235807	58036	3232235970	80	6	7.00	216	180	4	3	18	9656	1	ddos
1379272	3232235807	58038	3232235970	80	6	7.00	216	180	4	3	18	10046	1	ddos
1379273	3232235807	58040	3232235970	80	6	7.00	216	180	4	3	18	10485	1	ddos

615636 rows × 14 columns


final.ipynb
C:|Users|ST-0010|Music|DOS_DDOS_MI_ATTACK|ALL_IOT_DATA|Pickle|xgb.pkl
test_mosaic.csv
Minor_Project > ALL_ATTACK > ML CODE > final.ipynb > from sklearn.preprocessing import LabelEncoder
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
venv (Python 3.9.6)

```python
fi_df.Attack.value_counts()
```

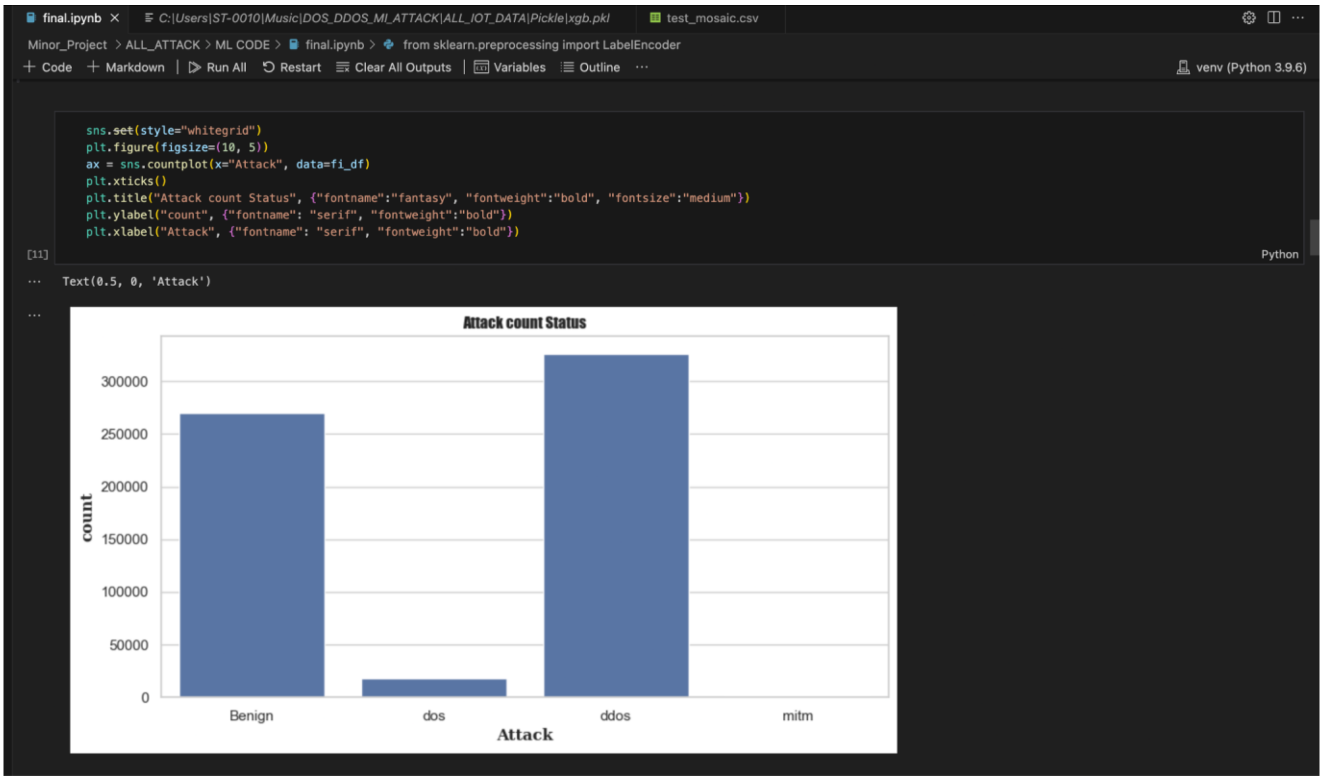
[9]

```
Attack
ddos      326345
Benign    270279
dos        17717
mitm        1295
Name: count, dtype: int64
```

```python
fi_df.describe()
```

[10]

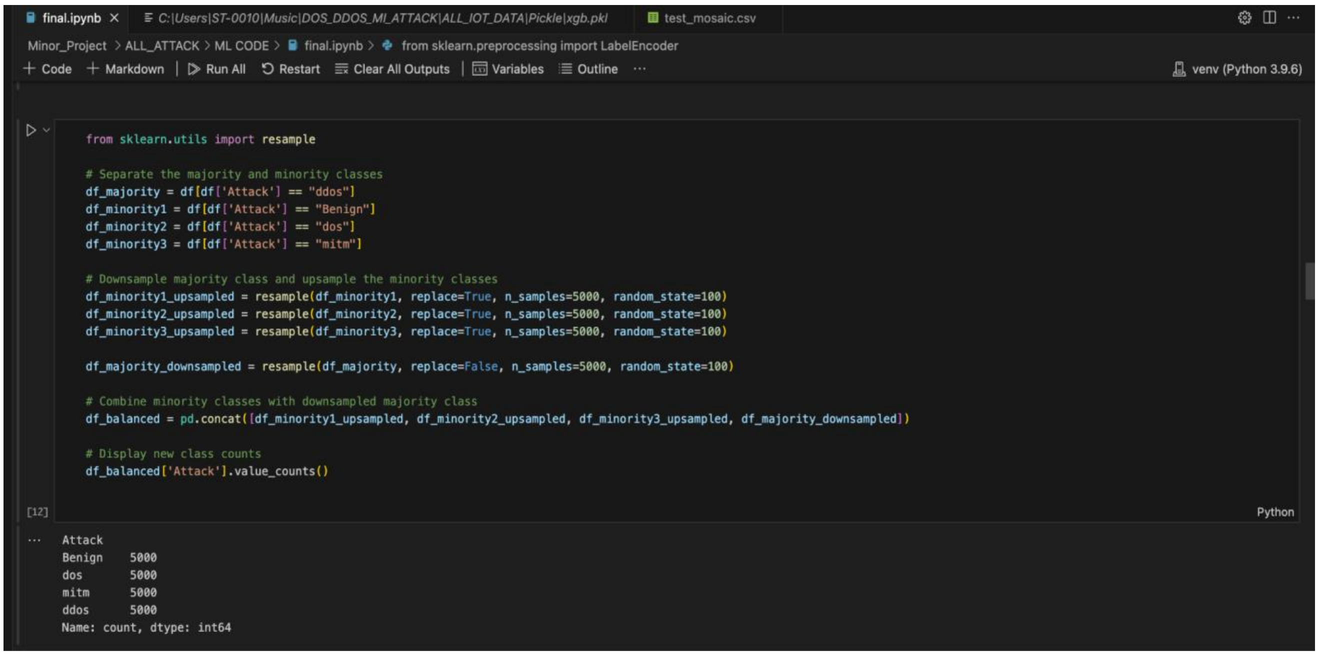
	IPV4_SRC_ADDR	L4_SRC_PORT	IPV4_DST_ADDR	L4_DST_PORT	PROTOCOL	L7_PROTO	IN_BYTES	OUT_BYTES	IN_PKTS	OUT_PKTS	TCP_FLAGS	FLOW_DURATI
count	6.156360e+05	615636.000000	6.156360e+05	615636.000000	615636.000000	615636.000000	6.156360e+05	6.156360e+05	6.156360e+05	6.156360e+05	615636.000000	
mean	3.214455e+09	33763.671278	3.223411e+09	15626.341575	7.799966	54.497875	2.615911e+03	2.075233e+03	3.367629e+01	1.622325e+01	16.435948	
std	2.151624e+08	22651.281115	3.243414e+08	22247.001376	4.199384	45.629839	2.663700e+05	1.653709e+05	4.319320e+03	2.177022e+03	8.603608	
min	0.000000e+00	0.000000	0.000000e+00	0.000000	1.000000	0.000000	2.800000e+01	0.000000e+00	1.000000e+00	0.000000e+00	0.000000	
25%	3.232236e+09	443.000000	3.232236e+09	80.000000	6.000000	7.000000	6.000000e+01	5.200000e+01	1.000000e+00	1.000000e+00	17.000000	
50%	3.232236e+09	41748.000000	3.232236e+09	443.000000	6.000000	91.000000	1.120000e+02	6.000000e+01	2.000000e+00	1.000000e+00	18.000000	
75%	3.232236e+09	52288.000000	3.232236e+09	40168.000000	6.000000	91.000000	2.160000e+02	1.800000e+02	3.000000e+00	1.000000e+00	18.000000	
max	3.701397e+09	65535.000000	4.294967e+09	65534.000000	58.000000	251.000000	7.941590e+07	7.016518e+07	1.221043e+06	1.349068e+06	219.000000	


final.ipynb
C:|Users|ST-0010|Music|DOS_DDOS_MI_ATTACK|ALL_IOT_DATA|Pickle|xgb.pkl
test_mosaic.csv
Minor_Project > ALL_ATTACK > ML CODE > final.ipynb > from sklearn.preprocessing import LabelEncoder
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
venv (Python 3.9.6)

```
sns.~~set~~(style="whitegrid")
plt.figure(figsize=(10, 5))
ax = sns.countplot(x="Attack", data=fi_df)
plt.xticks()
plt.title("Attack count Status", {"fontname":"fantasy", "fontweight":"bold", "fontsize":"medium"})
plt.ylabel("count", {"fontname": "serif", "fontweight":"bold"})
plt.xlabel("Attack", {"fontname": "serif", "fontweight":"bold"})
```

[11] Python

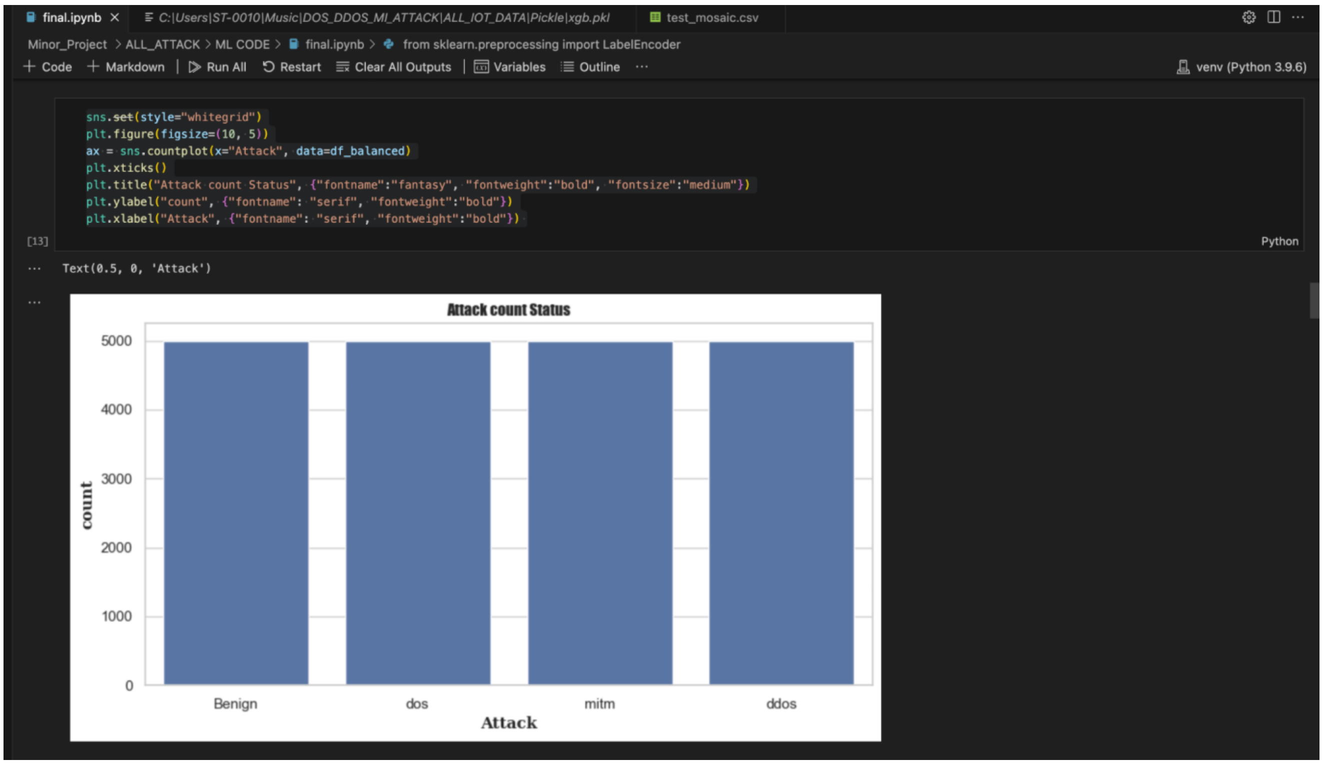
```
Text(0.5, 0, 'Attack')
```


final.ipynb
C:|Users|ST-0010|Music|DOS_DDOS_MI_ATTACK|ALL_IOT_DATA|Pickle|xgb.pkl
test_mosaic.csv
Minor_Project > ALL_ATTACK > ML CODE > final.ipynb > from sklearn.preprocessing import LabelEncoder
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
venv (Python 3.9.6)

```
from sklearn.utils import resample

# Separate the majority and minority classes
df_majority = df[df['Attack'] == "ddos"]
df_minority1 = df[df['Attack'] == "Benign"]
df_minority2 = df[df['Attack'] == "dos"]
df_minority3 = df[df['Attack'] == "mitm"]

# Downsample majority class and upsample the minority classes
df_minority1_upsampled = resample(df_minority1, replace=True, n_samples=5000, random_state=100)
df_minority2_upsampled = resample(df_minority2, replace=True, n_samples=5000, random_state=100)
df_minority3_upsampled = resample(df_minority3, replace=True, n_samples=5000, random_state=100)

df_majority_downsampled = resample(df_majority, replace=False, n_samples=5000, random_state=100)

# Combine minority classes with downsampled majority class
df_balanced = pd.concat([df_minority1_upsampled, df_minority2_upsampled, df_minority3_upsampled, df_majority_downsampled])

# Display new class counts
df_balanced['Attack'].value_counts()
```

[12] Python

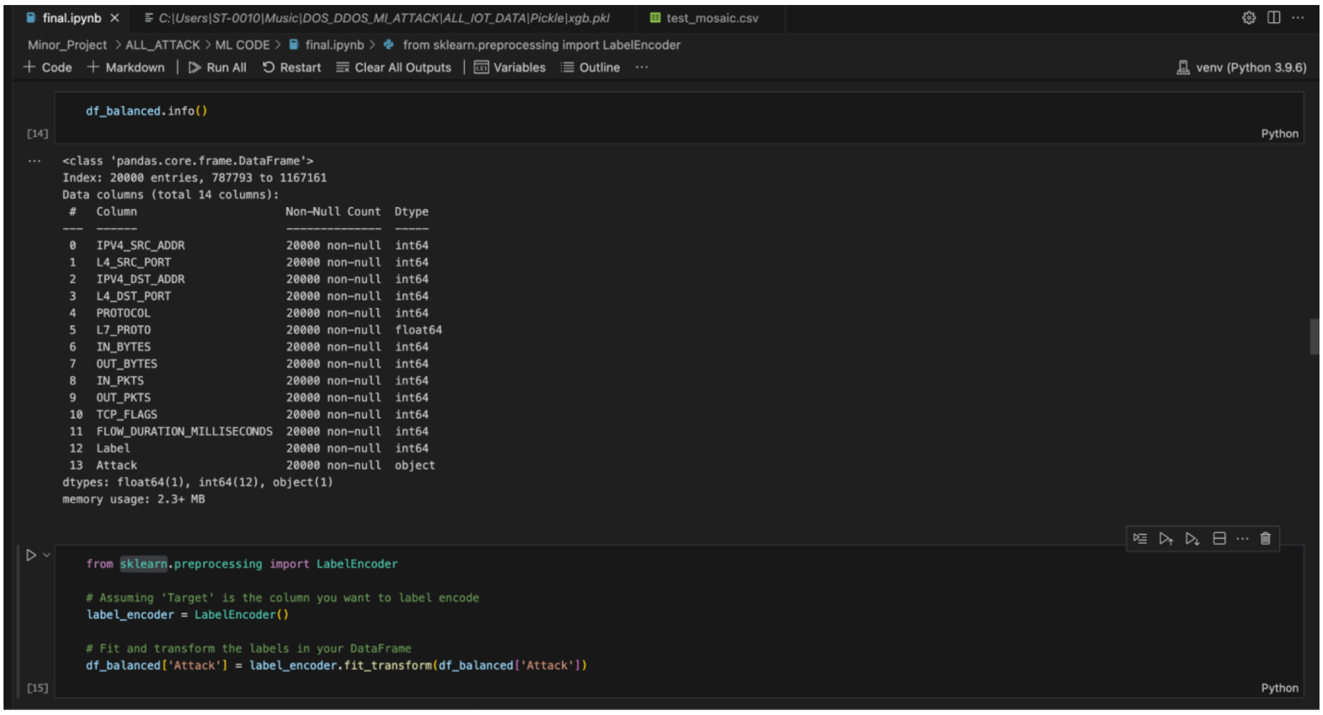
```
Attack
Benign    5000
dos       5000
mitm      5000
ddos      5000
Name: count, dtype: int64
```


final.ipynb
C:|Users|ST-0010|Music|DOS_DDOS_MI_ATTACK|ALL_IOT_DATA|Pickle|xgb.pkl
test_mosaic.csv
Minor_Project > ALL_ATTACK > ML CODE > final.ipynb > from sklearn.preprocessing import LabelEncoder
+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ...
venv (Python 3.9.6)
[13]
Python

```
sns.set(style="whitegrid")
plt.figure(figsize=(10, 5))
ax = sns.countplot(x="Attack", data=df_balanced)
plt.xticks()
plt.title("Attack count Status", {"fontname":"fantasy", "fontweight":"bold", "fontsize":"medium"})
plt.ylabel("count", {"fontname": "serif", "fontweight":"bold"})
plt.xlabel("Attack", {"fontname": "serif", "fontweight":"bold"})
```

```
Text(0.5, 0, 'Attack')
```

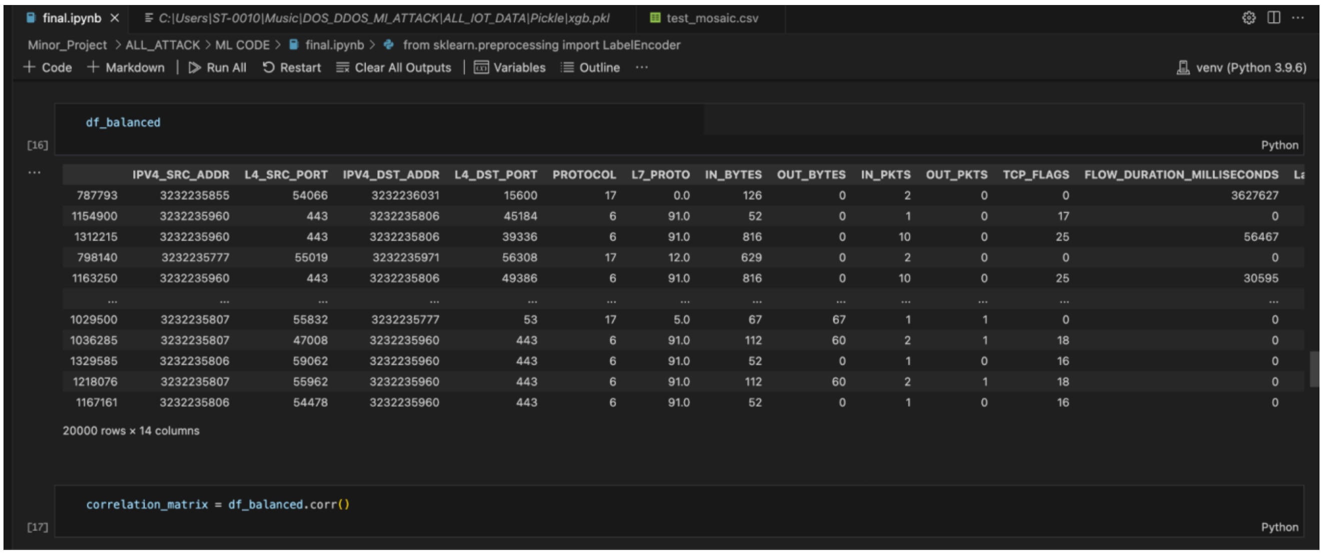
Attack count Status
5000
4000
3000
2000
1000
0
count
Benign
dos
mitm
ddos
Attack


final.ipynb
C:|Users|ST-0010|Music|DOS_DDOS_MI_ATTACK|ALL_IOT_DATA|Pickle|xgb.pkl
test_mosaic.csv
Minor_Project > ALL_ATTACK > ML CODE > final.ipynb > from sklearn.preprocessing import LabelEncoder
+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ...
venv (Python 3.9.6)
[14]
Python

```
df_balanced.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 20000 entries, 787793 to 1167161
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   IPV4_SRC_ADDR           20000 non-null  int64
 1   L4_SRC_PORT             20000 non-null  int64
 2   IPV4_DST_ADDR           20000 non-null  int64
 3   L4_DST_PORT             20000 non-null  int64
 4   PROTOCOL                20000 non-null  int64
 5   L7_PROTO                20000 non-null  float64
 6   IN_BYTES                20000 non-null  int64
 7   OUT_BYTES               20000 non-null  int64
 8   IN_PKTS                 20000 non-null  int64
 9   OUT_PKTS                20000 non-null  int64
 10  TCP_FLAGS               20000 non-null  int64
 11  FLOW_DURATION_MILLISECONDS  20000 non-null  int64
 12  Label                   20000 non-null  int64
 13  Attack                  20000 non-null  object
dtypes: float64(1), int64(12), object(1)
memory usage: 2.3+ MB
```

```
from sklearn.preprocessing import LabelEncoder

# Assuming 'Target' is the column you want to label encode
label_encoder = LabelEncoder()

# Fit and transform the labels in your DataFrame
df_balanced['Attack'] = label_encoder.fit_transform(df_balanced['Attack'])
```

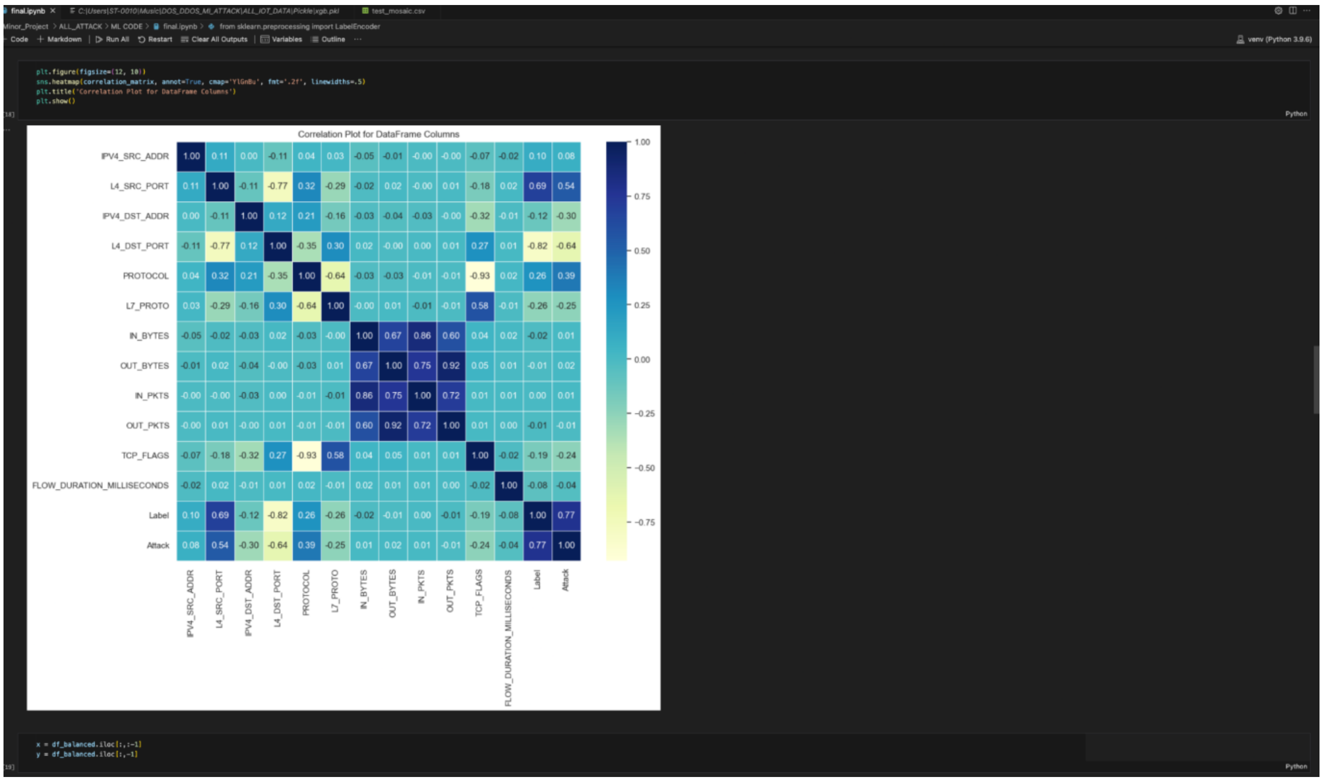

final.ipynb
C:|Users|ST-0010|Music|DOS_DDOS_MI_ATTACK|ALL_IOT_DATA|Pickle|xgb.pkl
test_mosaic.csv
Minor_Project > ALL_ATTACK > ML CODE > final.ipynb > from sklearn.preprocessing import LabelEncoder
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
venv (Python 3.9.6)

```python
df_balanced
```
[16] Python

	IPV4_SRC_ADDR	L4_SRC_PORT	IPV4_DST_ADDR	L4_DST_PORT	PROTOCOL	L7_PROTO	IN_BYTES	OUT_BYTES	IN_PKTS	OUT_PKTS	TCP_FLAGS	FLOW_DURATION_MILLISECONDS	La
787793	3232235855	54066	3232236031	15600	17	0.0	126	0	2	0	0	3627627	
1154900	3232235960	443	3232235806	45184	6	91.0	52	0	1	0	17	0	
1312215	3232235960	443	3232235806	39336	6	91.0	816	0	10	0	25	56467	
798140	3232235777	55019	3232235971	56308	17	12.0	629	0	2	0	0	0	
1163250	3232235960	443	3232235806	49386	6	91.0	816	0	10	0	25	30595	
...	...	...	...	...	...	...	...	...	...	...	...	...	
1029500	3232235807	55832	3232235777	53	17	5.0	67	67	1	1	0	0	
1036285	3232235807	47008	3232235960	443	6	91.0	112	60	2	1	18	0	
1329585	3232235806	59062	3232235960	443	6	91.0	52	0	1	0	16	0	
1218076	3232235807	55962	3232235960	443	6	91.0	112	60	2	1	18	0	
1167161	3232235806	54478	3232235960	443	6	91.0	52	0	1	0	16	0	

20000 rows × 14 columns

```python
correlation_matrix = df_balanced.corr()
```
[17] Python

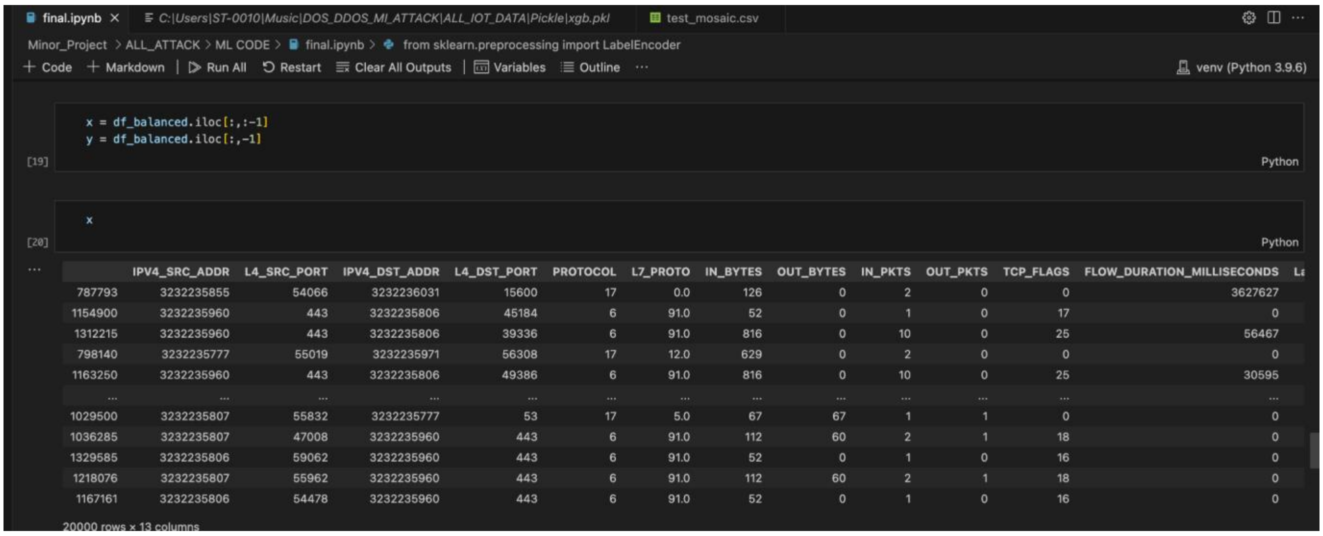


```python
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='YlGnBu', fmt='.2f', linewidths=.5)
plt.title('Correlation Plot for DataFrame Columns')
plt.show()
```
[18] Python

Correlation Plot for DataFrame Columns

	IPV4_SRC_ADDR	L4_SRC_PORT	IPV4_DST_ADDR	L4_DST_PORT	PROTOCOL	L7_PROTO	IN_BYTES	OUT_BYTES	IN_PKTS	OUT_PKTS	TCP_FLAGS	FLOW_DURATION_MILLISECONDS	Label	Attack
IPV4_SRC_ADDR	1.00	0.11	0.00	-0.11	0.04	0.03	-0.05	-0.01	-0.00	-0.00	-0.07	-0.02	0.10	0.08
L4_SRC_PORT	0.11	1.00	-0.11	-0.77	0.32	-0.29	-0.02	0.02	-0.00	0.01	-0.18	0.02	0.69	0.54
IPV4_DST_ADDR	0.00	-0.11	1.00	0.12	0.21	-0.16	-0.03	-0.04	-0.03	-0.00	-0.32	-0.01	-0.12	-0.30
L4_DST_PORT	-0.11	-0.77	0.12	1.00	-0.35	0.30	0.02	-0.00	0.00	0.01	0.27	0.01	-0.82	-0.64
PROTOCOL	0.04	0.32	0.21	-0.35	1.00	-0.64	-0.03	-0.03	-0.01	-0.01	-0.93	0.02	0.26	0.39
L7_PROTO	0.03	-0.29	-0.16	0.30	-0.64	1.00	-0.00	0.01	-0.01	-0.01	0.58	-0.01	-0.26	-0.25
IN_BYTES	-0.05	-0.02	-0.03	0.02	-0.03	-0.00	1.00	0.67	0.86	0.60	0.04	0.02	-0.02	0.01
OUT_BYTES	-0.01	0.02	-0.04	-0.00	-0.03	0.01	0.67	1.00	0.75	0.92	0.05	0.01	-0.01	0.02
IN_PKTS	-0.00	-0.00	-0.03	0.00	-0.01	-0.01	0.86	0.75	1.00	0.72	0.01	0.01	0.00	0.01
OUT_PKTS	-0.00	0.01	-0.00	0.01	-0.01	-0.01	0.60	0.92	0.72	1.00	0.01	0.00	-0.01	-0.01
TCP_FLAGS	-0.07	-0.18	-0.32	0.27	-0.93	0.58	0.04	0.05	0.01	0.01	1.00	-0.02	-0.19	-0.24
FLOW_DURATION_MILLISECONDS	-0.02	0.02	-0.01	0.01	0.02	-0.01	0.02	0.01	0.01	0.00	-0.02	1.00	-0.08	-0.04
Label	0.10	0.69	-0.12	-0.82	0.26	-0.26	-0.02	-0.01	0.00	-0.01	-0.19	-0.08	1.00	0.77
Attack	0.08	0.54	-0.30	-0.64	0.39	-0.25	0.01	0.02	0.01	-0.01	-0.24	-0.04	0.77	1.00

```python
x = df_balanced.iloc[:,:-1]
y = df_balanced.iloc[:,-1]
```

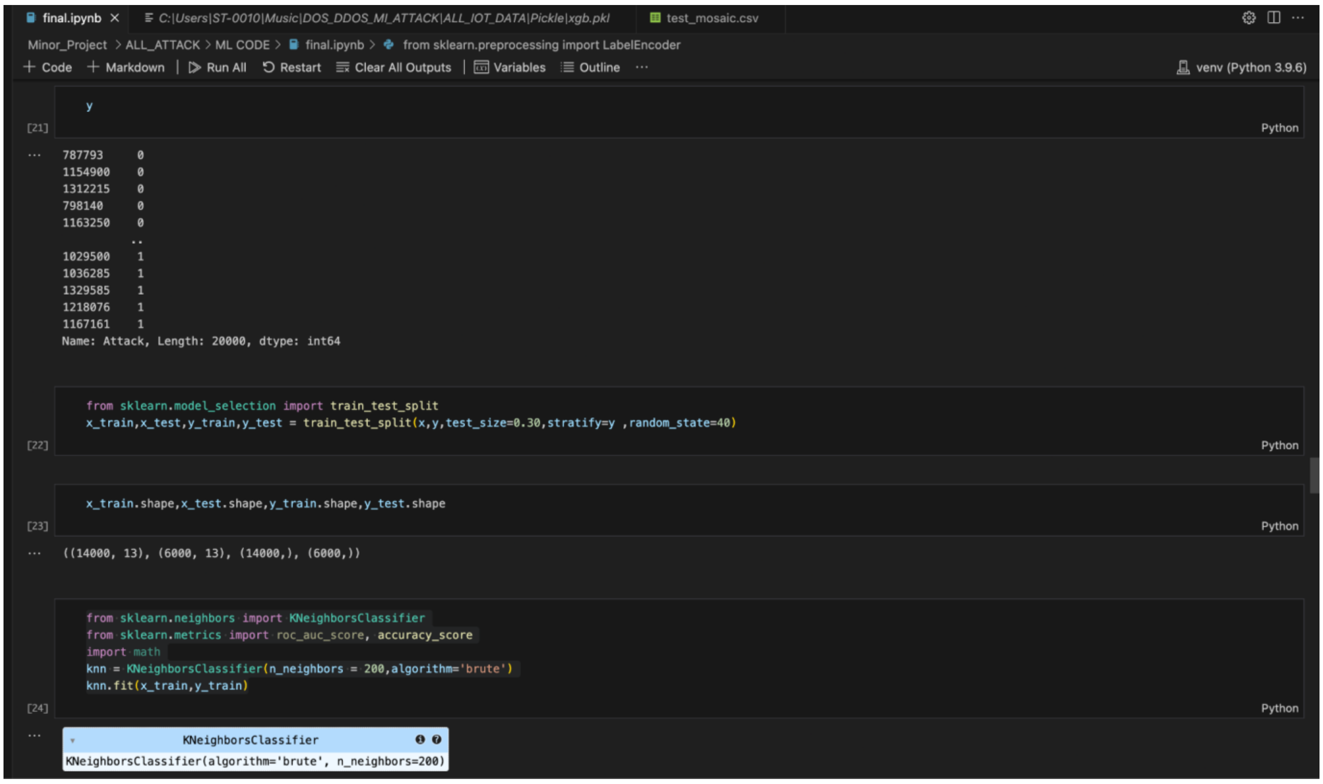

final.ipynb
C:|Users|ST-0010|Music|DOS_DDOS_MI_ATTACK|ALL_IOT_DATA|Pickle|xgb.pkl
test_mosaic.csv
Minor_Project > ALL_ATTACK > ML CODE > final.ipynb > from sklearn.preprocessing import LabelEncoder
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
venv (Python 3.9.6)

```python
x = df_balanced.iloc[:,:-1]
y = df_balanced.iloc[:,-1]
```
[19] Python

```python
x
```
[20] Python

	IPV4_SRC_ADDR	L4_SRC_PORT	IPV4_DST_ADDR	L4_DST_PORT	PROTOCOL	L7_PROTO	IN_BYTES	OUT_BYTES	IN_PKTS	OUT_PKTS	TCP_FLAGS	FLOW_DURATION_MILLISECONDS	La
787793	3232235855	54066	3232236031	15600	17	0.0	126	0	2	0	0	3627627	
1154900	3232235960	443	3232235806	45184	6	91.0	52	0	1	0	17	0	
1312215	3232235960	443	3232235806	39336	6	91.0	816	0	10	0	25	56467	
798140	3232235777	55019	3232235971	56308	17	12.0	629	0	2	0	0	0	
1163250	3232235960	443	3232235806	49386	6	91.0	816	0	10	0	25	30595	
...	...	...	...	...	...	...	...	...	...	...	...	...	
1029500	3232235807	55832	3232235777	53	17	5.0	67	67	1	1	0	0	
1036285	3232235807	47008	3232235960	443	6	91.0	112	60	2	1	18	0	
1329585	3232235806	59062	3232235960	443	6	91.0	52	0	1	0	16	0	
1218076	3232235807	55962	3232235960	443	6	91.0	112	60	2	1	18	0	
1167161	3232235806	54478	3232235960	443	6	91.0	52	0	1	0	16	0	

20000 rows × 13 columns


final.ipynb
C:|Users|ST-0010|Music|DOS_DDOS_MI_ATTACK|ALL_IOT_DATA|Pickle|xgb.pkl
test_mosaic.csv
Minor_Project > ALL_ATTACK > ML CODE > final.ipynb > from sklearn.preprocessing import LabelEncoder
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
venv (Python 3.9.6)

```
[21]
y
```

```
787793     0
1154900    0
1312215    0
798140     0
1163250    0
          ..
1029500    1
1036285    1
1329585    1
1218076    1
1167161    1
Name: Attack, Length: 20000, dtype: int64
```

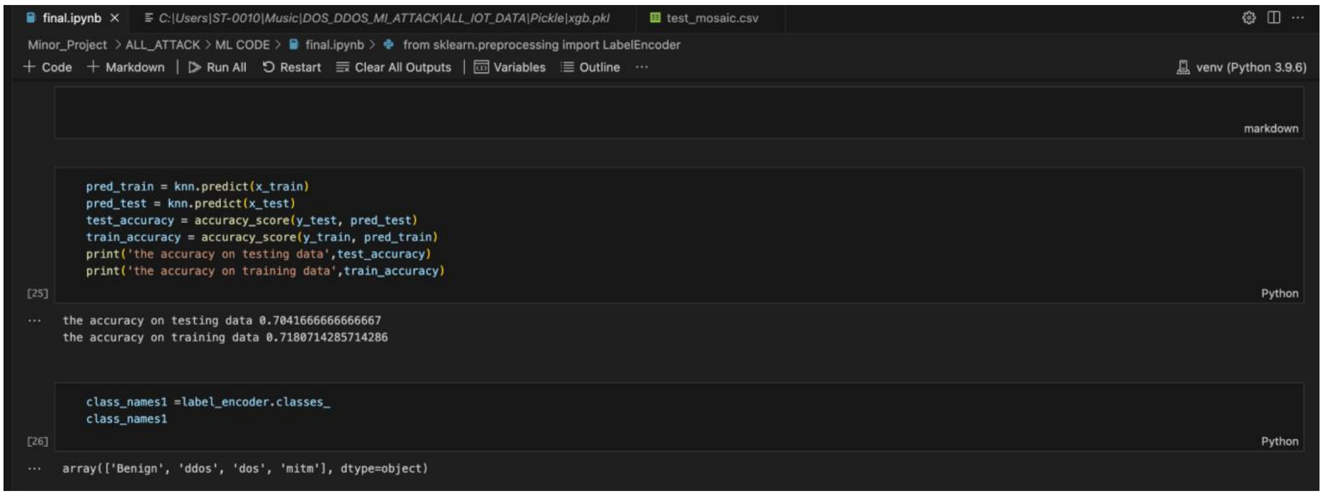
```
[22]
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.30,stratify=y ,random_state=40)
```

```
[23]
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

```
((14000, 13), (6000, 13), (14000,), (6000,))
```

```
[24]
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score, accuracy_score
import math
knn = KNeighborsClassifier(n_neighbors = 200,algorithm='brute')
knn.fit(x_train,y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier(algorithm='brute', n_neighbors=200)
```

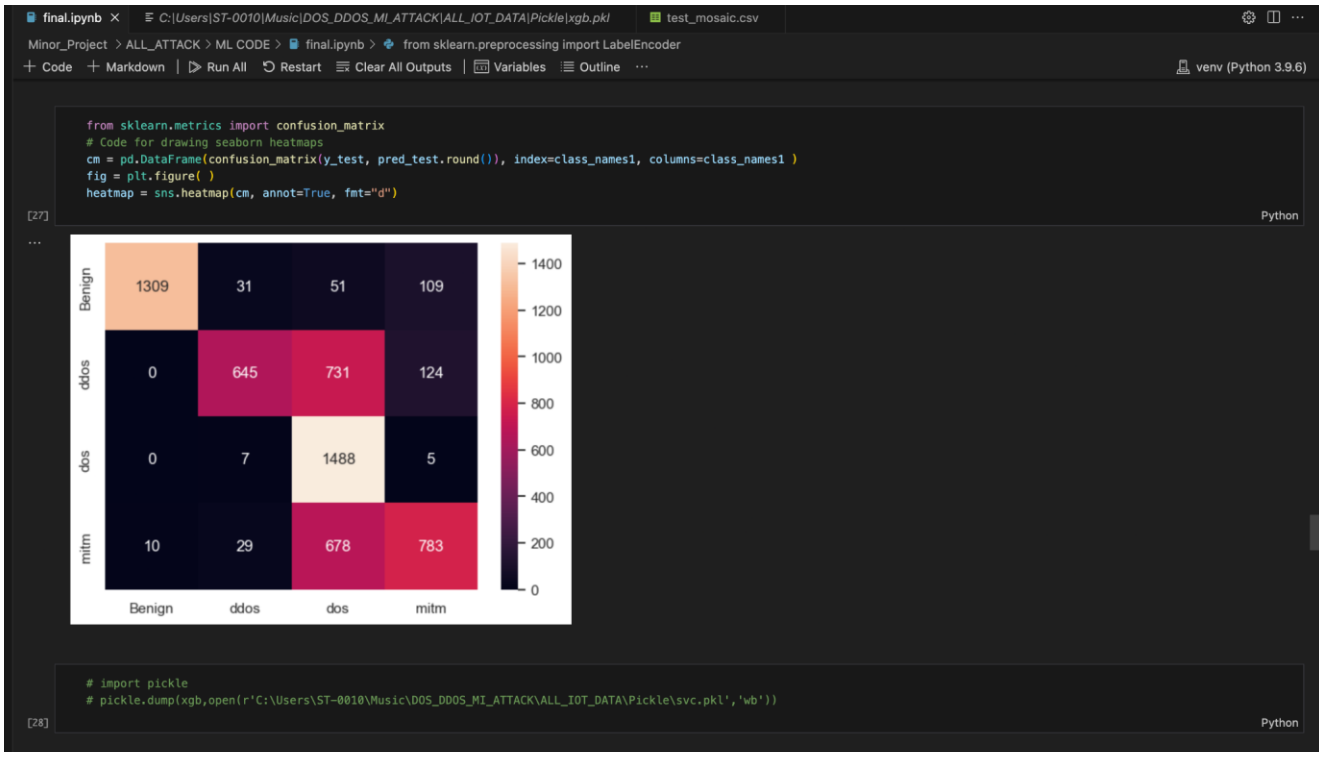

final.ipynb
C:|Users|ST-0010|Music|DOS_DDOS_MI_ATTACK|ALL_IOT_DATA|Pickle|xgb.pkl
test_mosaic.csv
Minor_Project > ALL_ATTACK > ML CODE > final.ipynb > from sklearn.preprocessing import LabelEncoder
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
venv (Python 3.9.6)
markdown

```
[25]
pred_train = knn.predict(x_train)
pred_test = knn.predict(x_test)
test_accuracy = accuracy_score(y_test, pred_test)
train_accuracy = accuracy_score(y_train, pred_train)
print('the accuracy on testing data',test_accuracy)
print('the accuracy on training data',train_accuracy)
```

```
the accuracy on testing data 0.7041666666666667
the accuracy on training data 0.7180714285714286
```

```
[26]
class_names1 =label_encoder.classes_
class_names1
```

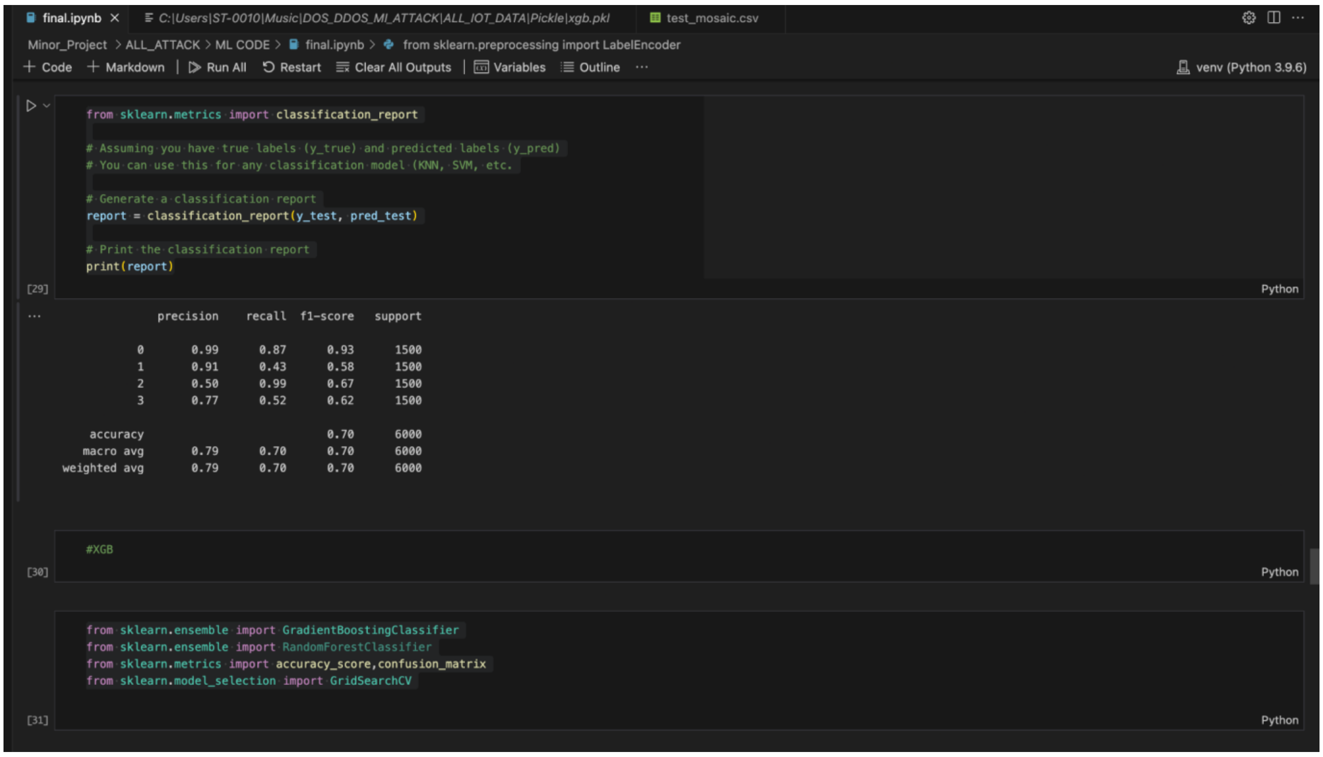
```
array(['Benign', 'ddos', 'dos', 'mitm'], dtype=object)
```


final.ipynb
C:\Users\ST-0010\Music\DOS_DDOS_MI_ATTACK\ALL_IOT_DATA\Pickle\xgb.pkl
test_mosaic.csv
Minor_Project > ALL_ATTACK > ML CODE > final.ipynb > from sklearn.preprocessing import LabelEncoder
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
venv (Python 3.9.6)
Python
Benign
ddos
dos
mitm
1400
1200
1000
800
600
400
200
0

```
from sklearn.metrics import confusion_matrix
# Code for drawing seaborn heatmaps
cm = pd.DataFrame(confusion_matrix(y_test, pred_test.round()), index=class_names1, columns=class_names1 )
fig = plt.figure( )
heatmap = sns.heatmap(cm, annot=True, fmt="d")
```
[27]

	Benign	ddos	dos	mitm
Benign	1309	31	51	109
ddos	0	645	731	124
dos	0	7	1488	5
mitm	10	29	678	783

```
# import pickle
# pickle.dump(xgb,open(r'C:\Users\ST-0010\Music\DOS_DDOS_MI_ATTACK\ALL_IOT_DATA\Pickle\svc.pkl','wb'))
```
[28]

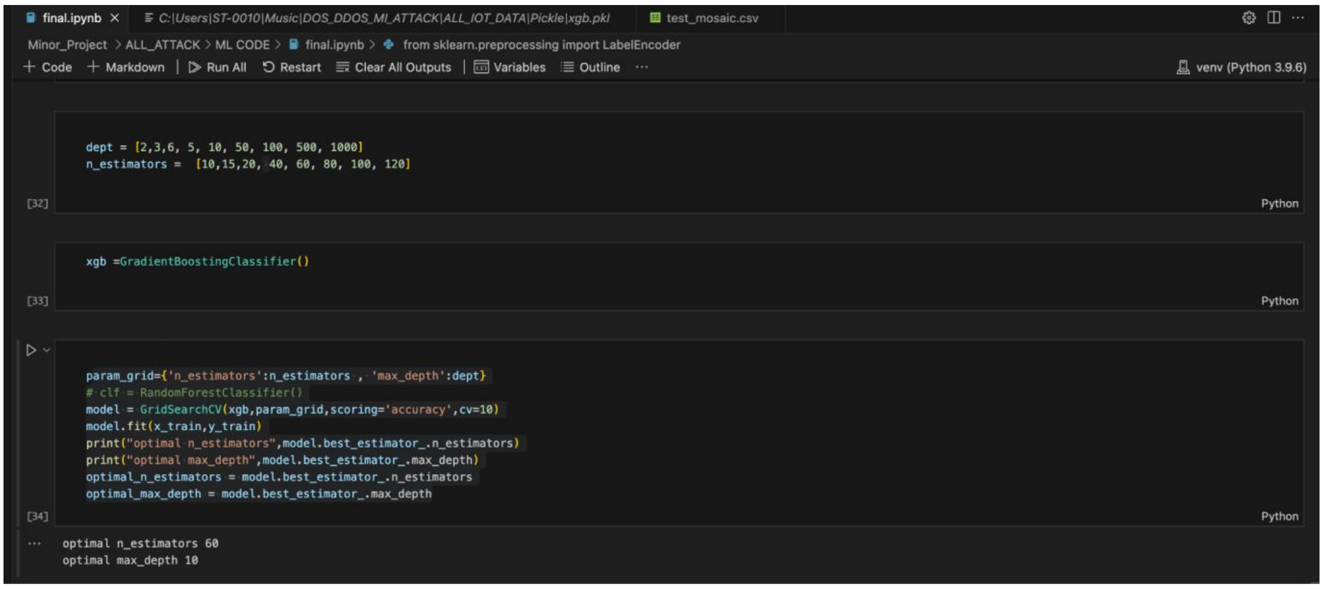

final.ipynb
C:\Users\ST-0010\Music\DOS_DDOS_MI_ATTACK\ALL_IOT_DATA\Pickle\xgb.pkl
test_mosaic.csv
Minor_Project > ALL_ATTACK > ML CODE > final.ipynb > from sklearn.preprocessing import LabelEncoder
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
venv (Python 3.9.6)
Python

```
from sklearn.metrics import classification_report

# Assuming you have true labels (y_true) and predicted labels (y_pred)
# You can use this for any classification model (KNN, SVM, etc.

# Generate a classification report
report = classification_report(y_test, pred_test)

# Print the classification report
print(report)
```
[29]

```
              precision    recall  f1-score   support

           0       0.99      0.87      0.93      1500
           1       0.91      0.43      0.58      1500
           2       0.50      0.99      0.67      1500
           3       0.77      0.52      0.62      1500

    accuracy                           0.70      6000
   macro avg       0.79      0.70      0.70      6000
weighted avg       0.79      0.70      0.70      6000
```

```
#XGB
```
[30]

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,confusion_matrix
from sklearn.model_selection import GridSearchCV
```
[31]

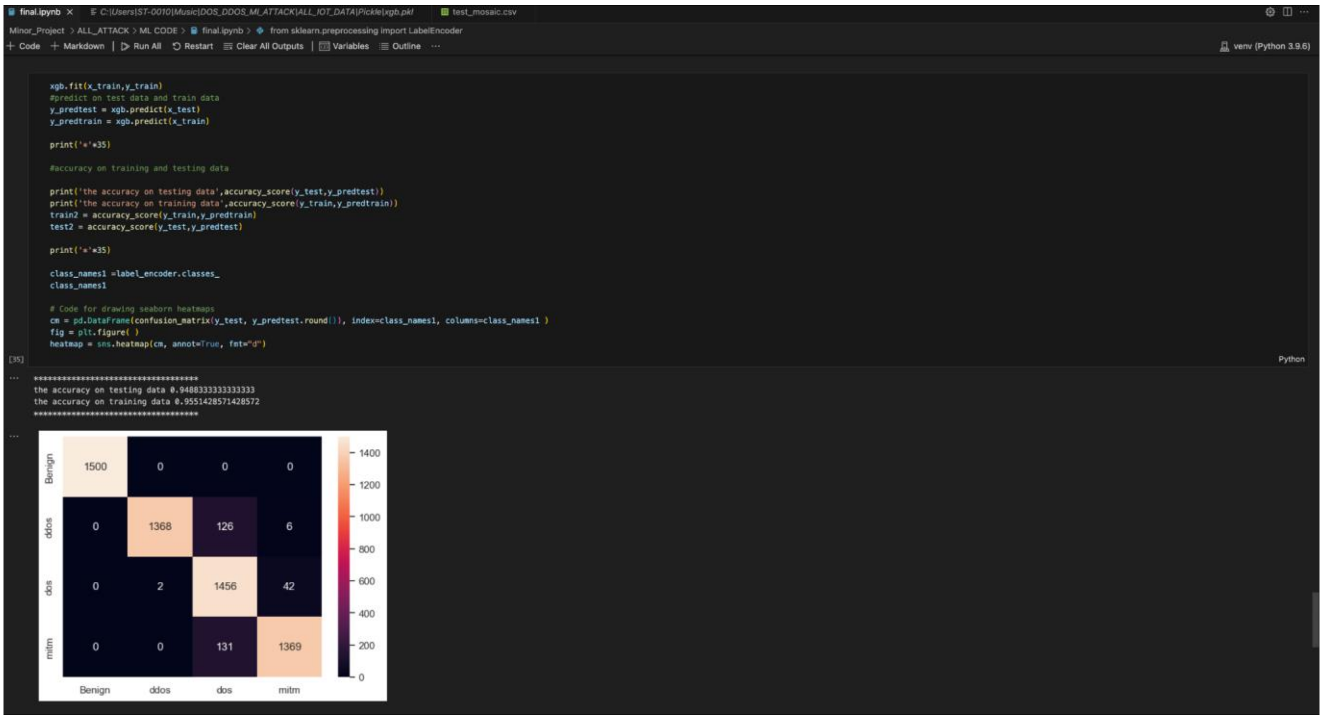

final.ipynb
C:|Users|ST-0010|Music|DOS_DDOS_MI_ATTACK|ALL_IOT_DATA|Pickle|xgb.pkl
test_mosaic.csv
Minor_Project > ALL_ATTACK > ML CODE > final.ipynb > from sklearn.preprocessing import LabelEncoder
+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline
venv (Python 3.9.6)

```
dept = [2,3,6, 5, 10, 50, 100, 500, 1000]
n_estimators =  [10,15,20, 40, 60, 80, 100, 120]
```
[32]

```
xgb =GradientBoostingClassifier()
```
[33]

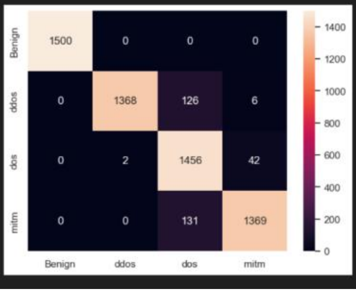
```
param_grid={'n_estimators':n_estimators , 'max_depth':dept}
# clf = RandomForestClassifier()
model = GridSearchCV(xgb,param_grid,scoring='accuracy',cv=10)
model.fit(x_train,y_train)
print("optimal n_estimators",model.best_estimator_.n_estimators)
print("optimal max_depth",model.best_estimator_.max_depth)
optimal_n_estimators = model.best_estimator_.n_estimators
optimal_max_depth = model.best_estimator_.max_depth
```
[34]

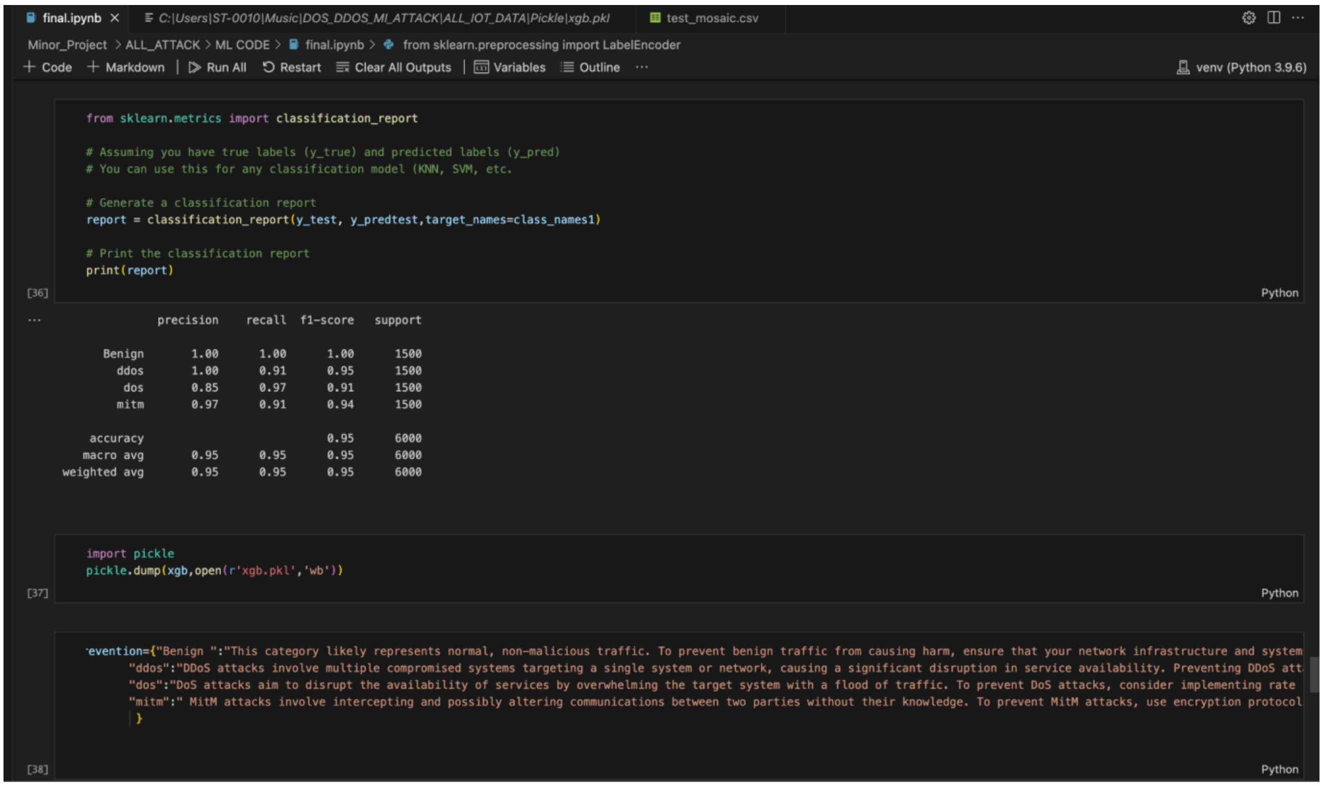
```
optimal n_estimators 60
optimal max_depth 10
```


final.ipynb
C:|Users|ST-0010|Music|DOS_DDOS_MI_ATTACK|ALL_IOT_DATA|Pickle|xgb.pkl
test_mosaic.csv
Minor_Project > ALL_ATTACK > ML CODE > final.ipynb > from sklearn.preprocessing import LabelEncoder
+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline
venv (Python 3.9.6)

```
xgb.fit(x_train,y_train)
#predict on test data and train data
y_predtest = xgb.predict(x_test)
y_predtrain = xgb.predict(x_train)

print('*'*35)

#accuracy on training and testing data

print('the accuracy on testing data',accuracy_score(y_test,y_predtest))
print('the accuracy on training data',accuracy_score(y_train,y_predtrain))
train2 = accuracy_score(y_train,y_predtrain)
test2 = accuracy_score(y_test,y_predtest)

print('*'*35)

class_names1 =label_encoder.classes_
class_names1

# Code for drawing seaborn heatmaps
cm = pd.DataFrame(confusion_matrix(y_test, y_predtest.round()), index=class_names1, columns=class_names1 )
fig = plt.figure( )
heatmap = sns.heatmap(cm, annot=True, fmt="d")
```
[35]

```
***********************************
the accuracy on testing data 0.9488333333333333
the accuracy on training data 0.9551428571428572
***********************************
```



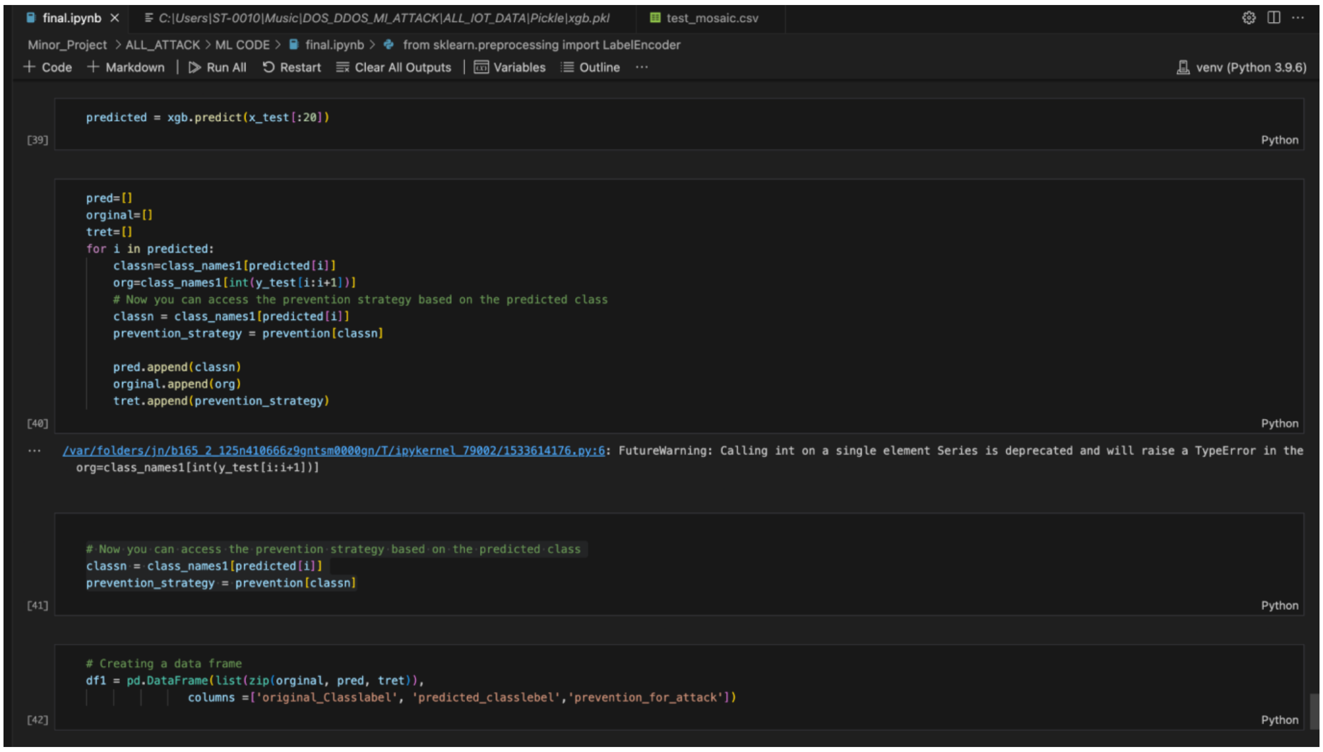


```
from sklearn.metrics import classification_report

# Assuming you have true labels (y_true) and predicted labels (y_pred)
# You can use this for any classification model (KNN, SVM, etc.

# Generate a classification report
report = classification_report(y_test, y_predtest,target_names=class_names1)

# Print the classification report
print(report)
```

```
              precision    recall  f1-score   support

      Benign       1.00      1.00      1.00      1500
        ddos       1.00      0.91      0.95      1500
         dos       0.85      0.97      0.91      1500
        mitm       0.97      0.91      0.94      1500

    accuracy                           0.95      6000
   macro avg       0.95      0.95      0.95      6000
weighted avg       0.95      0.95      0.95      6000
```

```
import pickle
pickle.dump(xgb,open(r'xgb.pkl','wb'))
```

```
evention={"Benign ":"This category likely represents normal, non-malicious traffic. To prevent benign traffic from causing harm, ensure that your network infrastructure and system
      "ddos":"DDoS attacks involve multiple compromised systems targeting a single system or network, causing a significant disruption in service availability. Preventing DDoS att
      "dos":"DoS attacks aim to disrupt the availability of services by overwhelming the target system with a flood of traffic. To prevent DoS attacks, consider implementing rate
      "mitm":" MitM attacks involve intercepting and possibly altering communications between two parties without their knowledge. To prevent MitM attacks, use encryption protocol
      }
```



```
predicted = xgb.predict(x_test[:20])
```

```
pred=[]
orginal=[]
tret=[]
for i in predicted:
    classn=class_names1[predicted[i]]
    org=class_names1[int(y_test[i:i+1])]
    # Now you can access the prevention strategy based on the predicted class
    classn = class_names1[predicted[i]]
    prevention_strategy = prevention[classn]

    pred.append(classn)
    orginal.append(org)
    tret.append(prevention_strategy)
```

```
/var/folders/jn/b165_2_125n410666z9gntsm0000gn/T/ipykernel_79002/1533614176.py:6: FutureWarning: Calling int on a single element Series is deprecated and will raise a TypeError in the
  org=class_names1[int(y_test[i:i+1])]
```

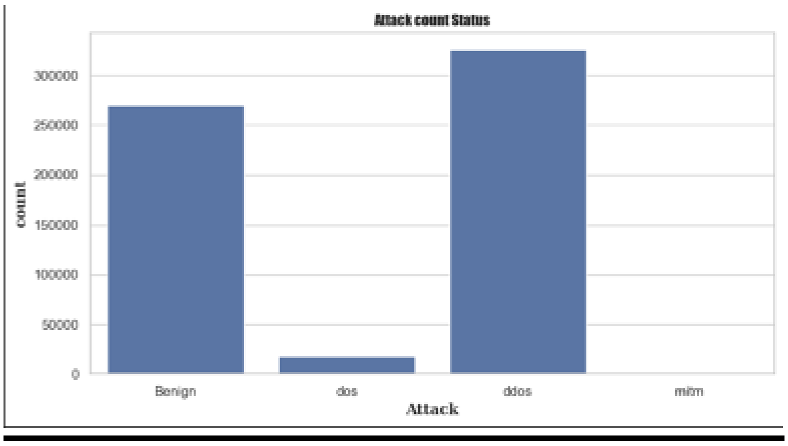
```
# Now you can access the prevention strategy based on the predicted class
classn = class_names1[predicted[i]]
prevention_strategy = prevention[classn]
```

```
# Creating a data frame
df1 = pd.DataFrame(list(zip(orginal, pred, tret)),
               columns =['original_Classlabel', 'predicted_classlebel','prevention_for_attack'])
```

CHAPTER 8

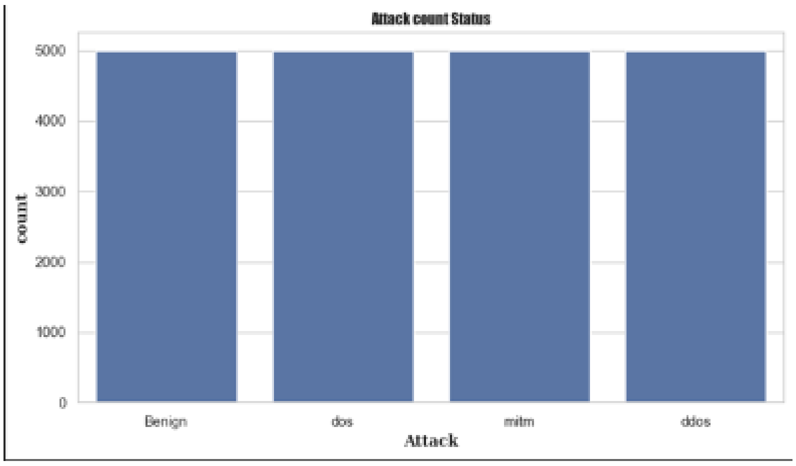
RESULTS AND ANALYSIS

The bar graph depicts the number of distinct attack categories present in the collected data. Most occurrences belong to the category of benign events, which amount to over 250,000. This indicates that there are a lot of benign or non-attack events. Similarly, DDOS and its variants also show a high count, that is more than 300,000, which is a major worry in the case of this type of attack. In contrast, DOS shown attacks appear to be the most infrequent, indicating their occurrence is quite rare. But strikingly, the attacks that are well-known as MITM, do not present any count at all. To sum up, the graph clearly presents an uneven distribution of the attacks throughout the timeline, with most of them being non-hostile and DDOS attacks instances.



Graph. 1. - Attack count Status

The bar graph clearly indicates that all four attack types are identical with a frequency of occurrences of 5000. The uniformity of the height of the bars indicates that there is no skew in the dataset that is, every attack type like Benign, DDOS, DOS, and MITM, occurs with the same number of instances. Such balanced representation reinforces that no one attack type is predominant and therefore provides better ease in analysing the distribution of attacks in the data set.



Graph. 2. - Attack count Status

The correlation matrix illustrates how different network parameters- with respect to IPs, ports, packet sizes, bytes, and flags- are related to each other and is primarily aimed at observing trends that can help recognize certain types of assaults. The set of correlation coefficients ranges from -1 to 1, with 1 being the best negative correlation, -1 being the best positive correlation and 0 meaning that there is no correlation whatsoever. To illustrate, **TCP_FLAGS** is strongly negatively correlated with **PROTOCOL** (-0.93), suggesting these features are inversely related which could appear as abnormal traffic. However, **IN_BYTES** and **OUT_BYTES** are positively correlated (0.67) suggesting that there is usually an increase in **OUT_BYTES** when there is an increase in the **IN_BYTES** count. In addition, with the help of correlation analysis it was noted that **L4_DST_PORT** parameter is negatively correlated to the Attack variable (-0.64), assuring that certain attack trends might not avoid certain destination ports. Hence, the figure explains some of the relationships between the variables which may be useful in predicting certain types of attacks like DDOS or MITM attack.

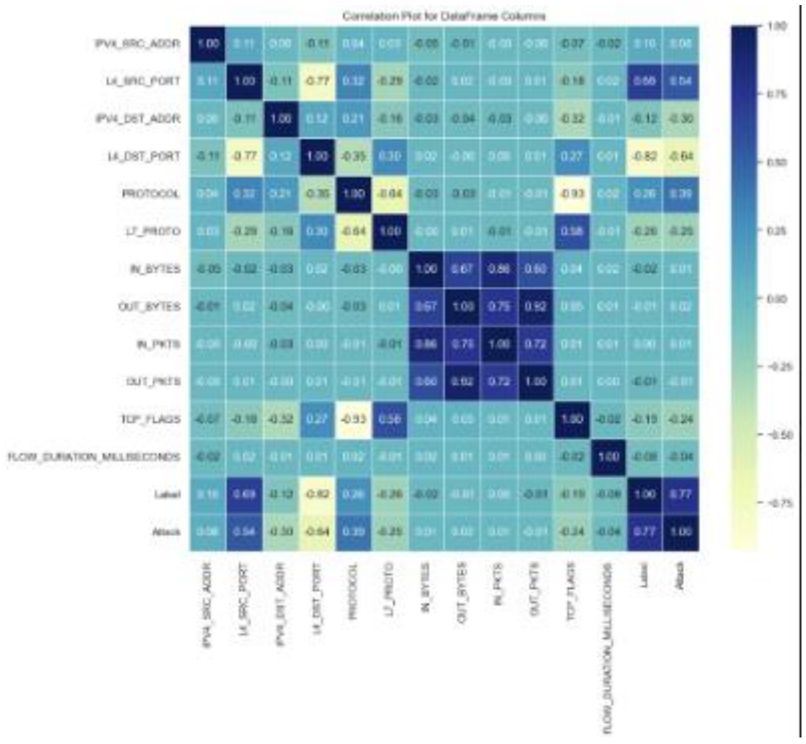
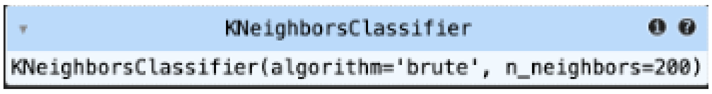


Fig. 1. - Correlation Plot for Benign, DDOS, DOS, MITM DataFrame Columns – Heatmap

The KNN classifier algorithm makes use of brute-force techniques as this methodology involves computing the distance between every single data point to all the points of the training data to get the nearest neighbors. In this case of setting n_neighbors=200, new point obtained by the algorithm is compared with its nearest neighbors in dense 200 vicinity, and thus trying to classify the new point. Employing 200 neighbours is a substantial quantity, which may render the model less responsive to minor fluctuations in the data, perhaps resulting in underfitting (where the model is excessively simplistic to identify significant patterns). The brute-force method is less efficient than certain other methods; yet, it maintains accuracy, particularly with small to medium-sized datasets.



```
KNeighborsClassifier
KNeighborsClassifier(algorithm='brute', n_neighbors=200)
```

Fig. 2. – KneighborsClassifier

The confusion matrix illustrates the extent to which the KNN model was able to classify types of network traffic into Benign, DDoS, DoS, and MitM attacks. The model performs excellently in predicting the Benign and DoS traffic where most of the instances were classified correctly. However, it finds it challenging to tell DDoS attacks apart from DoS as they are often interchanged – for instance, as many as 731 DDoS were misclassified as DoS. In the same way, DoS is also often confused with MiTM traffic. In general, this model can tell Benign and DoS attacks apart adequately but does not perform so well while classifying between attacks of DDoS, MiTM and DoS.

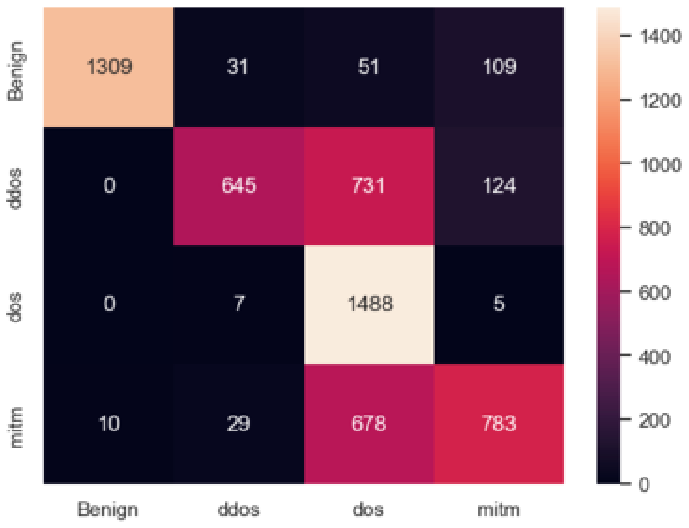


Fig. 3. – KNN Confusion Matrix

The metrics derived from the confusion matrix reveal that the XGBoost Models perform remarkably well with respect to classifying various classes of network traffic. It does not make any errors in misclassifications while detecting all Benign traffic. For DDoS classes, judgements are made confidently and correctly to most instances but indicate some called DoS. Likewise, many DoS and MiTM instances are correctly distinguished by the model; however, it does misclassify some of the DoS as MiTM as well as some of the MiTMs as DoS. In general, the model exhibits a good level of performance; however, classifications of certain types of attacks remain problematic (more so between DoS and MiTM, DDoS and DoS).

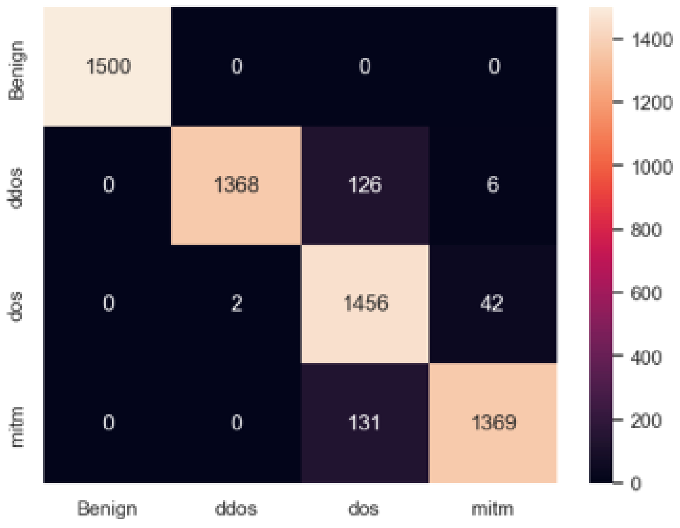


Fig. 4. – XGBoost Confusion Matrix

CHAPTER 9

CONCLUSION AND FUTURE DEVELOPMENTS

Therefore, the research has made an unprecedented contribution to the network security field by developing a solution based on machine learning for detection and mitigation of several kinds of cyber threats in the form of DOS, MITM, and DDOS attacks. Cybersecurity is a highly discussed topic due to growth and development in networks, which have more data traffic and sophisticated attack patterns. This research has overcome the above challenges by combining the potential of SDN with that of machine learning to provide a robust and adaptive detection mechanism.

This project applied the centralized control and flexible architecture of SDN to enhance the accuracy of threat detection using special SDN features, including flow statistics, to identify patterns in network traffic that were anomalous. This was an integration of SDN that would allow unmatched packets to be redirected to the controller, which acts as a focal point for anomaly detection. With the processing of flow data at the controller level, the system can really monitor the real-time traffic pattern and pinpoint possible threats before they can escalate.

The very important thing of this study is a flow-based intrusion detection system backed by pre-trained machine learning. A model designed in the recognition of suspicious flow of traffic produced a level of detection accuracy of close to 98-99% in controlled environments. However, since the machine learning models are fundamentally flawed, false positives are highly common and occur even more frequently in the dynamic network environment, more so when normal patterns might sometimes be like the attack patterns. This latest research raised an entropy-based computation method to be used as a layer on the flowIDS: The entropy-based techniques were sensitive to fine changes in the traffic patterns and thus, proved to be a pretty good complement to the applied machine learning technique. In this way, accuracy has been increased while bringing the false positives down many folds.

When tested with real-time network data, the system had an impressive 99% detection accuracy, thus further highlighting the soundness and reliability of the model. Although the detection accuracy is very high, a 14% false positive rate was seen in this case. This needs to improve further if a better system is sought. A false positive is harmless, but in the network operations, this can be a headache, as it may initiate unnecessary alarms, and resources are thus diverted from actual threats by attending to them. Hence, with such a reduction in future versions, the system could be highly effective and would be very practical for mass-scale network deployment.

With the hybrid methodology applied in this research, there are several broader implications. It shows how techniques may be used together to address the weakness of single methods: machine learning is extremely powerful but very imprecise in noisily polluted environments and entropy-based methods can be nonspecific when working with highly complex landscapes of threats. Together, however, they form a complementary system of both precision and adaptability. This approach also puts the needs of SDN across as architecture that can be extremely flexible to accommodate modern advanced security solutions, which provides an interesting platform for future research and development in this field of cybersecurity.

This work is of high importance to the organisations that manage large-scale networks, such as data centers, telecommunications providers, and enterprise environments. Advanced attack vectors cannot be addressed by traditional security solutions because they are evolving continuously. Therefore, the proposed system offers a scalable and adaptive solution fine-tuned to the needs of any network environment to provide security beyond static rule-based systems.

On a practical note, it proves the importance of continuous learning and adaptation in cybersecurity. The more data fed into the machine learning models over time, the better they might be at prediction, and thereby the system can "learn" to adapt to emerging types of threats. After all, attackers are working day and night to get around existing defenses. It means the system produced by marring machine learning with SDN would be dynamic as it is adaptive to changing and evolving based on shifts in the nature of the threats.

Future Improvement

Following are some areas that might further enhance the developed system: Deep learning or reinforcement learning techniques might be promising because they were listed among those techniques that are thought to have the ability to solve complex pattern recognition problems. Another possible improvement would be optimization of entropy computation to reduce the false positive rate. The last thing would be automatic response initiated into the SDN controller for faster recovery operations based on detected threats, hence making the whole system efficient.

In summary, this work forms a basis for further development of much more complex, dynamic security solutions leveraging machine learning techniques and the capabilities of SDN. The system developed here will address strategic issues related to detection accuracy with minimized false positives, hence marking a very important development in the field of network security. Future extensions based on this work will doubtlessly go a long way toward better protection of networks from evolving cyber threats, bringing in a safer digital future.

REFERENCES

[1] Bindra, N.; Sood, M. Detecting DDoS attacks using machine learning techniques andcontemporary intrusion detection dataset. Autom. Control. Comput. Sci. 2019, 53,419–428

[2] Usha, G., Narang, M., Kumar, A. (2021). Detection and Classification of DistributedDoS Attacks Using Machine Learning. In: Smys, S., Palanisamy, R., Rocha, Á.,Beligiannis, G.N. (eds) Computer Networks and Inventive CommunicationTechnologies. Lecture Notes on Data Engineering and CommunicationsTechnologies, vol 58. Springer, Singapore.

[3] Saini, P.S.; Behal, S.; Bhatia, S. Detection of DDoS attacks using machine learningalgorithms. In Proceedings of the 2020 7th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 12–14 March2020; IEEE: Piscataway, NJ, USA, 2020;

[4] Sambangi, S.; Gondi, L. A Machine Learning Approach for DDoS (Distributed Denial
of Service) Attack Detection Using Multiple Linear Regression. Proceedings 2020,63, 51.

[5] Fadil, A.; Riadi, I.; Aji, S. Review of Detection DDOS Attack Detection Using NaiveBayes Classifier for Network Forensics. Bull. Electr. Eng. Inform. 2017, 6, 140–148

[6] S.T Zargar, J. Joshi and D Tipper, "A survey of defence mechanisms againstdistributed denial of service flooding attacks", IEEE communication surveys and
Tutorials, vol. 15, no. 4, pp. 2046-2049, 2013.

[7] Al-issa AI, Al-Akhras M, ALsahli MS, Alawadhi M (2019) Using machine learning
to detect DoS attacks in wireless sensor networks. In: 2019 IEEE jordan international
joint conference on electrical engineering and information technology (JEEIT),
Amman, Jordan, pp 107–112

[8] McCullough, E.; Iqbal, R.; Katangur, A. Analysis of Machine Learning Techniques for Lightweight DDoS Attack Detection on IoT Networks. In Forthcoming Network sand Sustainability in the IoT Era. FoNeS-IoT 2020.

[9] Suresh, M., Anitha, R. (2019). Evaluating Machine Learning Algorithms for DetectingDDoS Attacks. In: Wyld, D.C., Wozniak, M., Chaki, N., Meghanathan, N.,
Nagamalai, D. (eds) Advances in Network Security and Applications. CNSA 2011.Communications in Computer and Information Science, vol 196. Springer, Berlin,Heidelberg.

[10] S. T. Zargar, J. Joshi, and D. Tipper, “A survey of defense mechanisms against distributed denial of service (DDOS) flooding attacks,” IEEE Communications
Surveys and Tutorials, vol. 15, no. 4, pp. 2046–2069, 2018.

[11] Sharafaldin I, Lashkari AH, Hakak S, Ghorbani AA (2019) Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy. In: IEEE 53rd
international carnahan conference on security technology, Chennai, India

[12] Kumari, K., Mrunalini, M. Detecting Denial of Service attacks using machine learning
algorithms. J Big Data 9, 56 (2022).

[13] V. D. M. Rios, P. R. M. Inácio, D. Magoni and M. M. Freire, "Detection and Mitigation of Low-Rate Denial-of-Service Attacks: A Survey," in IEEE Access, vol. 10, pp. 76648-76668, 2022

[14] M. Sathishkumar and Y. -C. Liu, “Resilient Adaptive Event-Triggered Control for Singular Networked Cascade Control Systems Under DoS Attacks,” in IEEE Access, vol. 10, pp. 89197-89210, 2022

[15] S. Khodadadi, T. K. Tasooji and H. J. Marquez, "Observer-Based Secure Control for Vehicular Platooning Under DoS Attacks," in IEEE Access, vol. 11, pp. 20542-20552, 2023

APPENDICES

APPENDIX A - CODING

```
import numpy  as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv(r"C:\Users\skart\Downloads\project\ALL_ATTACK\data\NF-ToN-IoT.csv")
df
df.sample(10)
df.info()
import ipaddress

# Convert IPV4_SRC_ADDR to integer representation
df['IPV4_SRC_ADDR'] = df['IPV4_SRC_ADDR'].apply(lambda x:
int(ipaddress.IPv4Address(x)))

# Convert IPV4_DST_ADDR to integer representation
df['IPV4_DST_ADDR'] = df['IPV4_DST_ADDR'].apply(lambda x:
int(ipaddress.IPv4Address(x)))
df.Attack.value_counts()
fi_df = df[df['Attack'].isin(['ddos', 'dos', 'Benign', 'mitm'])]

fi_df
fi_df.Attack.value_counts()
fi_df.describe()
sns.set(style="whitegrid")
plt.figure(figsize=(10, 5))
ax = sns.countplot(x="Attack", data=fi_df)
plt.xticks()
plt.title("Attack count Status", {"fontname":"fantasy", "fontweight":"bold",
"fontsize":"medium"})
plt.ylabel("count", {"fontname": "serif", "fontweight":"bold"})
plt.xlabel("Attack", {"fontname": "serif", "fontweight":"bold"})
from sklearn.utils import resample

# Separate the majority and minority classes
df_majority = df[df['Attack'] == "ddos"]
df_minority1 = df[df['Attack'] == "Benign"]
df_minority2 = df[df['Attack'] == "dos"]
df_minority3 = df[df['Attack'] == "mitm"]


# Downsample majority class and upsample the minority classes
df_minority1_upsampled = resample(df_minority1, replace=True, n_samples=5000,
random_state=100)
df_minority2_upsampled = resample(df_minority2, replace=True, n_samples=5000,
random_state=100)
df_minority3_upsampled = resample(df_minority3, replace=True, n_samples=5000,
random_state=100)

df_majority_downsampled = resample(df_majority, replace=False, n_samples=5000,
random_state=100)

# Combine minority classes with downsampled majority class
df_balanced = pd.concat([df_minority1_upsampled, df_minority2_upsampled,
df_minority3_upsampled, df_majority_downsampled])

# Display new class counts
df_balanced['Attack'].value_counts()
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Plot settings
sns.set(style="whitegrid")
plt.figure(figsize=(10, 5))

# Create the count plot
ax = sns.countplot(x="Attack", data=df_balanced)

# Customize the plot
plt.title("Attack count Status", {"fontname":"fantasy", "fontweight":"bold",
"fontsize":"medium"})
plt.ylabel("count", {"fontname": "serif", "fontweight":"bold"})
plt.xlabel("Attack", {"fontname": "serif", "fontweight":"bold"})

# Show plot
plt.show()
df_balanced.info()
from sklearn.preprocessing import LabelEncoder

# Assuming 'Target' is the column you want to label encode
label_encoder = LabelEncoder()

# Fit and transform the labels in your DataFrame
df_balanced['Attack'] = label_encoder.fit_transform(df_balanced['Attack'])
Df_balanced
correlation_matrix = df_balanced.corr()
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='YlGnBu', fmt='.2f', linewidths=.5)
plt.title('Correlation Plot for DataFrame Columns')
plt.show()
x = df_balanced.iloc[:,:-1]
y = df_balanced.iloc[:,-1]
X
```

```
Y
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.30,stratify=y ,random_state=40)
X_train.shape,x_test.shape,y_train.shape,y_test.shape
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score, accuracy_score
import math
knn = KNeighborsClassifier(n_neighbors = 200,algorithm='brute')
knn.fit(x_train,y_train)
pred_train = knn.predict(x_train)
pred_test = knn.predict(x_test)
test_accuracy = accuracy_score(y_test, pred_test)
train_accuracy = accuracy_score(y_train, pred_train)
print('the accuracy on testing data',test_accuracy)
print('the accuracy on training data',train_accuracy)
class_names1 =label_encoder.classes_
Class_names1
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

# Pure function to get unique class names
def get_class_names(y_test):
    return sorted(list(set(y_test)))

# Pure function to create a confusion matrix DataFrame
def create_confusion_matrix(y_test, pred_test):
    cm_array = confusion_matrix(y_test, pred_test.round())
    class_names = get_class_names(y_test)
    return pd.DataFrame(cm_array, index=class_names, columns=class_names)

# Pure function to plot the heatmap
def plot_confusion_matrix(cm):
    plt.figure(figsize=(8, 6))
    heatmap = sns.heatmap(cm, annot=True, fmt="d", cmap='Blues')
    plt.ylabel('True Label')
    plt.xlabel('Predicted Label')
    plt.show()

# Assuming y_test and pred_test are already defined
cm = create_confusion_matrix(y_test, pred_test)
plot_confusion_matrix(cm)
# import pickle
# pickle.dump(xgb,open(r'C:\Users\ST-
0010\Music\DOS_DDOS_MI_ATTACK\ALL_IOT_DATA\Pickle\svc.pkl','wb'))
from sklearn.metrics import classification_report

# Assuming you have true labels (y_true) and predicted labels (y_pred)
# You can use this for any classification model (KNN, SVM, etc.

# Generate a classification report
report = classification_report(y_test, pred_test)

# Print the classification report
print(report)
!pip install xgboost
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,confusion_matrix
from sklearn.model_selection import GridSearchCV

dept = [2,3,6, 5, 10, 50, 100, 500, 1000]
n_estimators =  [10,15,20, 40, 60, 80, 100, 120]
xgb =XGBClassifier()

param_grid={'n_estimators':n_estimators , 'max_depth':dept}
# clf = RandomForestClassifier()
model = GridSearchCV(xgb,param_grid,scoring='accuracy',cv=10)
model.fit(x_train,y_train)
print("optimal n_estimators",model.best_estimator_.n_estimators)
print("optimal max_depth",model.best_estimator_.max_depth)
optimal_n_estimators = model.best_estimator_.n_estimators
optimal_max_depth = model.best_estimator_.max_depth
xgb.fit(x_train,y_train)
#predict on test data and train data
y_predtest = xgb.predict(x_test)
y_predtrain = xgb.predict(x_train)

print('*'*35)

#accuracy on training and testing data

print('the accuracy on testing data',accuracy_score(y_test,y_predtest))
print('the accuracy on training data',accuracy_score(y_train,y_predtrain))
train2 = accuracy_score(y_train,y_predtrain)
test2 = accuracy_score(y_test,y_predtest)

print('*'*35)

class_names1 =label_encoder.classes_
class_names1

# Code for drawing seaborn heatmaps
cm = pd.DataFrame(confusion_matrix(y_test, y_predtest.round()), index=class_names1,
columns=class_names1 )
fig = plt.figure( )
heatmap = sns.heatmap(cm, annot=True, fmt="d")
```

```
from sklearn.metrics import classification_report

# Assuming you have true labels (y_true) and predicted labels (y_pred)
# You can use this for any classification model (KNN, SVM, etc.

# Generate a classification report
report = classification_report(y_test, y_predtest,target_names=class_names1)

# Print the classification report
print(report)
import pickle
pickle.dump(xgb,open(r'C:\Users\skart\Downloads\project\ALL_ATTACK\Pickle\xgb.pkl','w
b'))
prevention={"Benign ":"This category likely represents normal, non-malicious traffic. To
prevent benign traffic from causing harm, ensure that your network infrastructure and systems
are properly configured and updated. Implement robust security measures such as firewalls,
intrusion detection systems (IDS), and antivirus software to protect against potential threats.",
        "ddos":"DDoS attacks involve multiple compromised systems targeting a single system or
network, causing a significant disruption in service availability. Preventing DDoS attacks
requires a combination of network infrastructure hardening, traffic monitoring, and DDoS
mitigation solutions. Implement robust network security measures, including ingress and
egress filtering, to block malicious traffic at the network perimeter. Consider using DDoS
mitigation services or appliances that can detect and filter out DDoS traffic before it reaches
your network.",
        "dos":"DoS attacks aim to disrupt the availability of services by overwhelming the target
system with a flood of traffic. To prevent DoS attacks, consider implementing rate limiting,
access controls, and traffic filtering mechanisms to mitigate the impact of excessive traffic.
Additionally, deploy DoS protection solutions or services that can detect and block malicious
traffic in real-time.",
        "mitm":" MitM attacks involve intercepting and possibly altering communications
between two parties without their knowledge. To prevent MitM attacks, use encryption
protocols such as HTTPS for web traffic and VPNs for secure network connections. Employ
certificate-based authentication and regularly update cryptographic keys to ensure secure
communication channels.",
        }
predicted = xgb.predict(x_test[:20])
pred=[]
orginal=[]
tret=[]
for i in predicted:
    classn=class_names1[predicted[i]]
    org=class_names1[int(y_test[i:i+1])]
    # Now you can access the prevention strategy based on the predicted class
    classn = class_names1[predicted[i]]
    prevention_strategy = prevention[classn]

    pred.append(classn)
    orginal.append(org)
    tret.append(prevention_strategy)
```

```
# Now you can access the prevention strategy based on the predicted class
classn = class_names1[predicted[i]]
prevention_strategy = prevention[classn]
# Creating a data frame
df1 = pd.DataFrame(list(zip(orginal, pred, tret)),
            columns =['original_Classlabel', 'predicted_classlebel','prevention_for_attack'])

df1
```

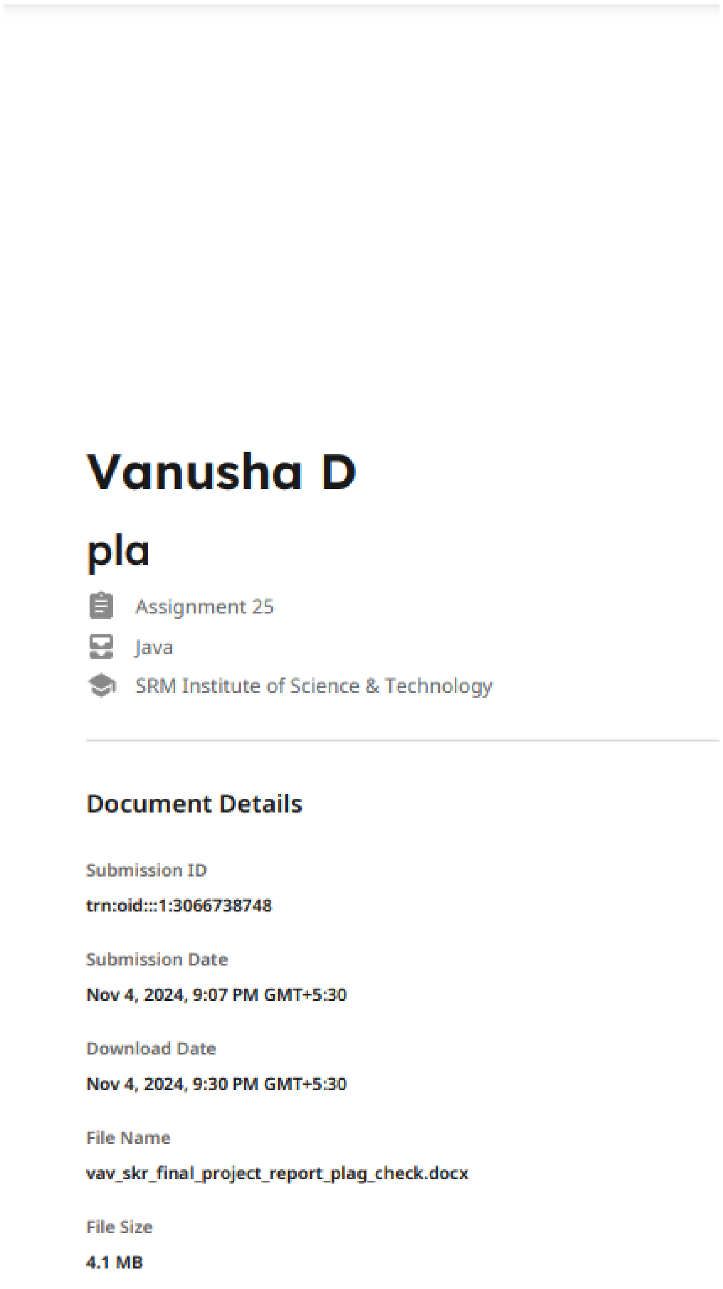
APPENDIX B

CONFERENCE PRESENTATION

Our paper on **Identification and Counteraction of Benign, DDOS, DOS, and MITM utilsing AI calculations** will be presented at 3rd International Conference on Automation, Computing and Renewable Systems. Our paper is expected to be accepted with a plagiarism check of just 5 %

APPENDIX C

PLAGIARISM REPORT



Vanusha D

pla

Assignment 25

Java

SRM Institute of Science & Technology

Document Details

Submission ID

trn:oid:::1:3066738748

Submission Date

Nov 4, 2024, 9:07 PM GMT+5:30

Download Date

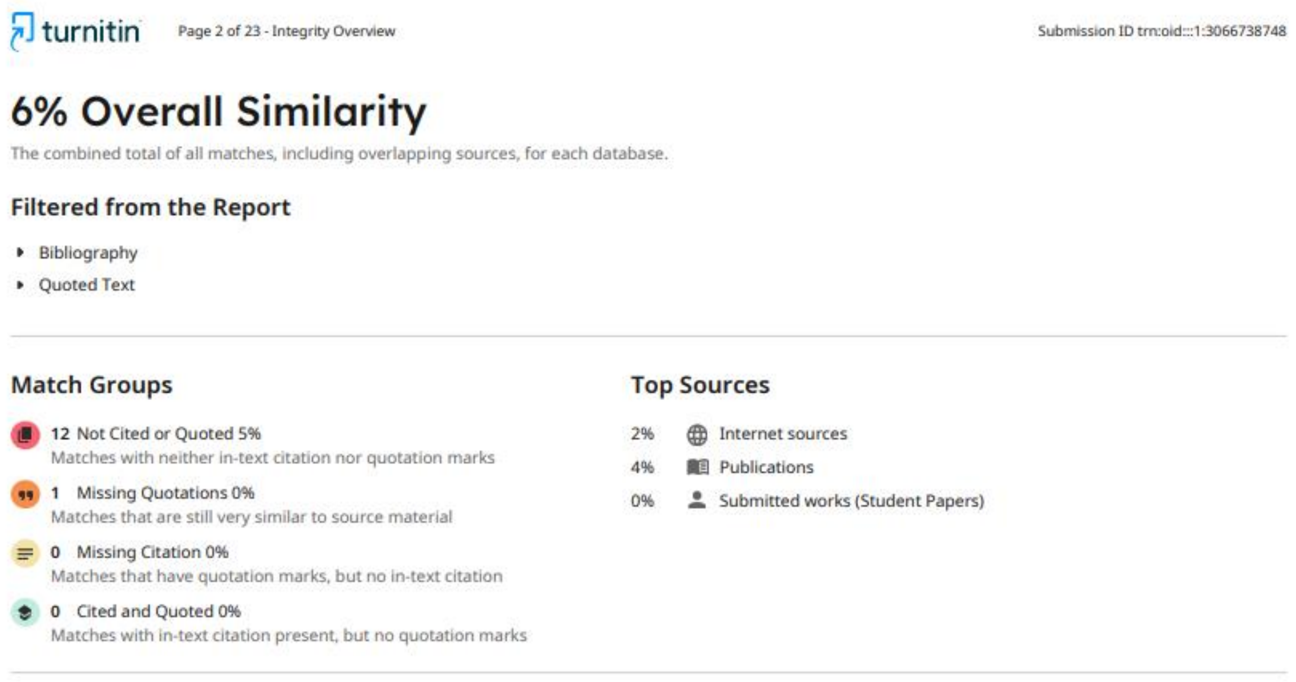
Nov 4, 2024, 9:30 PM GMT+5:30

File Name

vav_skr_final_project_report_plag_check.docx

File Size

4.1 MB



turnitin Page 2 of 23 - Integrity Overview Submission ID trn:oid:::1:3066738748

6% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text

Match Groups

12 Not Cited or Quoted 5%
Matches with neither in-text citation nor quotation marks

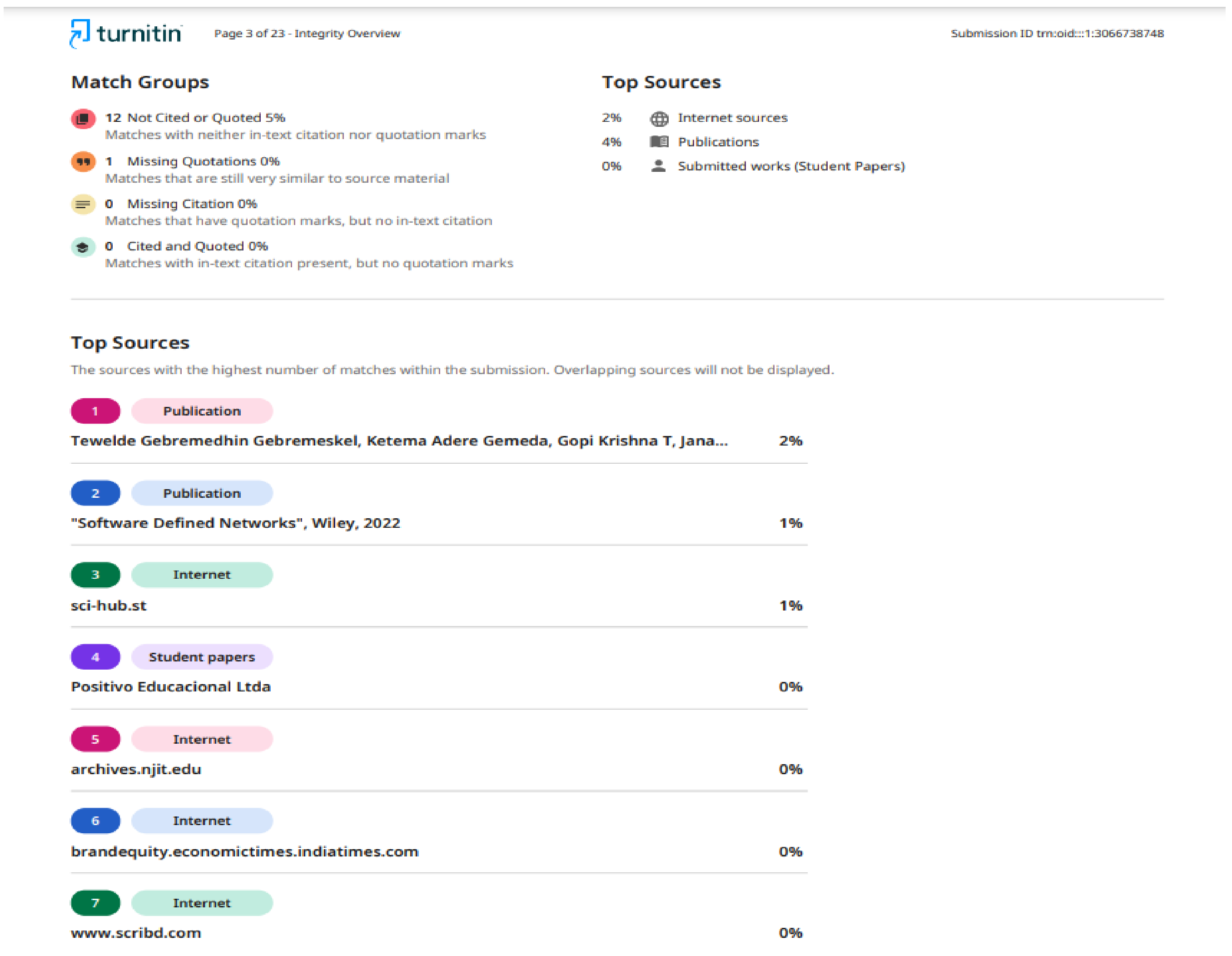
1 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

2%	Internet sources
4%	Publications
0%	Submitted works (Student Papers)



turnitin Page 3 of 23 - Integrity Overview Submission ID trn:oid:::1:3066738748

Match Groups

12 Not Cited or Quoted 5%
Matches with neither in-text citation nor quotation marks

1 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

2%	Internet sources
4%	Publications
0%	Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Publication	Tewelde Gebremedhin Gebremeskel, Ketema Adere Gemeda, Gopi Krishna T, Jana...	2%
2	Publication	"Software Defined Networks", Wiley, 2022	1%
3	Internet	sci-hub.st	1%
4	Student papers	Positivo Educacional Ltda	0%
5	Internet	archives.njit.edu	0%
6	Internet	brandequity.economictimes.indiatimes.com	0%
7	Internet	www.scribd.com	0%

Format - I

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University u/s 3 of UGC Act, 1956)

Office of Controller of Examinations

REPORT FOR PLAGIARISM CHECK ON THE DISSERTATION/PROJECT REPORTS FOR UG/PG PROGRAMMES

(To be attached in the dissertation/ project report)

1	Name of the Candidate **(IN BLOCK LETTERS)**	VELLANKI VENKAT ADITYA
2	Address of the Candidate	SRM Institute of Science and Technology
3	Registration Number	RA2111003011799
4	Date of Birth	23-06-2003 18-02-2004
5	Department	Computer Science and Engineering
6	Faculty	Engineering and Technology, School of Computing
7	Title of the Dissertation/Project	IDENTIFICATION OF BENIGN, DDOS, DOS, MITM UTILISING AI CALCULATIONS
8	Whether the above project /dissertation is done by	~~Individual~~ or group GROUP (Strike whichever is not applicable) a) If the project/ dissertation is done in group, then how many students together completed the project 2 b) Mention the Name & Register number of other candidates SUDARAKUNTALA KARTHIKEYA RA2111003011812
9	Name and address of the Supervisor / Guide	Dr. S. Nikkath Bushra SRM Institute of Science & Technology **Mail ID:** nikkaths@srmist.edu.in **Mobile Number:** 8939938999
10	Name and address of Co-Supervisor / Co- Guide (if any)	**Mail ID:** **Mobile Number:**

11	Software Used	Vs Code		
12	Date of Verification	07-11-2024		
13	Plagiarism Details: (to attach the final report from the software)			
Chapter	Title of the Chapter	Percentage of similarity index (including self citation)	Percentage of similarity index (Excluding self-citation)	% of plagiarism after excluding Quotes, Bibliography, etc.,
1	INTRODUCTION	6%	0%	0%
2	PROBLEM STATEMENT	3%	2%	2%
3	PROPOSED MODEL	6%	0%	0%
4	SYSTEM ARCHITECTURE	0%	0%	0%
5	CODING AND TESTING	0%	0%	0%
6	RESULTS AND ANALYSIS	4%	0%	0%
7	CONCLUSION	0%	3%	3%
8				
9				
10				
	Appendices			

I We declare that the above information have been verified and found true to the best of my / our knowledge.

[illegible] Signature of the Candidate	Dr. S. Niicalli Bushra [illegible] Name & Signature of the Staff (Who uses the plagiarism check software)
Dr. S. Niicalli Bushra [illegible] Name & Signature of the Supervisor/ Guide	Name & Signature of the Co-Supervisor/Co-Guide

[illegible]

Name & Signature of the HOD