TOWARDS EXAMINING SUPERVISED AND UNSUPERVISED LEARNING FOR IOT ATTACK DETECTION

by

Nevetha Govindaraju

Submitted in partial fulfilment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
April 2023

Dalhousie University is located in Mi'kma'ki, the
ancestral and unceded territory of the Mi'kmaq.
We are all Treaty people.

The thesis is dedicated to my family and to all people who have given me immense support during these challenging times

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The Internet of Things (IoT) is the term used to describe the numerous physical objects/devices connected to the Internet and collecting and exchanging data globally. IoT devices are especially susceptible to network attacks, including but not limited to botnet attacks, spoofing attacks, and denial of service attacks. This thesis explores supervised and unsupervised learning approaches to compare two types of traffic flow exporters on different publicly available datasets. Evaluations and results show that it is possible to achieve high weighted average F1-scores for attack detection using off-the-shelf supervised learning algorithms and traffic flow features on IoT networks.

LIST OF ABBREVIATIONS USED

SL	Supervised Learning
USL	Unsupervised Learning
DT	Decision Tree
KNN	K-Nearest Neighbors
SVM	Support Vector Machines
LOF	Local Outlier Factor
1-SVM	One-class Support Vector Machines
IF	Isolation Forest
IoT	Internet of Things
HTTP	Hyper Text Transfer Protocol
DoS	Denial of Service
DDoS	Distributed Denial of Service
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
RTSP	Real Time Streaming Protocol
OS	Operating System
ROC	Receiver Operating Characteristic
AUC	Area Under ROC Curve
ARP	Address Resolution Protocol
ICMP	Internet Control Message Protocol
ESP	Encapsulating Security Payload
CSV	Comma Separated Values
RAM	Random Access Memory
PCAP	Packet Captures
AUPRC	Area Under Precision Recall Curve
MLP	Multi-Layer Perceptron
LSTM	Long Short Term Memory
DL	Deep Learning
XGBoost	Extreme Gradient Boosting
IDS	Intrusion Detection System
PCA	Principal Component Analysis
RNN	Recurrent Neural Network
LR	Logistic Regression

WAF1-Score	Weighted Average F1-Score
T2	Tranalyzer2

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor, Dr. Nur Zincir-Heywood and my mentor Dr. Marwa Elsayed for their immense support and guidance during my Master's Degree journey. I would also like to thank my readers Dr. Srinivas Sampalli and Dr. Malcolm Heywood for their timely review and constructive feedback on this work.

This thesis is the result of hard work and commitment from my side and my family and friends who have supported and encouraged me in this journey to study, develop, grow, and succeed.

CHAPTER 1 INTRODUCTION

The Internet of Things (IoT) is the network of physical items, or "things," embedded with sensors, software, and other technologies to communicate and exchange data with other devices and systems through the Internet. IoT device applications have grown so diverse that they now support more than half of the world's population. These devices help simplify people's lives more than ever. They can be monitored and managed from anywhere using smartphones and the Internet. Since these devices rely on the Internet, they generate significant network traffic that can occasionally cause disruptions in terms of network congestion. The capacity of online traffic surged as more IoT devices were linked to the Internet. Security vulnerabilities increase with the evolution of IoT devices as they lack the essential built-in security safeguards to counter threats, making them particularly vulnerable. These devices' limited environment and low computational capacity are the principal reasons, severely challenging their physical security. An IoT device's tampering, theft, or destruction are examples of physical attacks. IoT devices are especially susceptible to network attacks, including data theft, phishing, botnet attacks, spoofing, and denial of service attacks (DoS and DDoS attacks [13] [14]). These can result in significant data breaches and other cyber security risks like ransomware attacks. These make IoT devices more prone to threats to user security and privacy. Therefore, it is essential to safeguard user information by strengthening the network due to the rise in cyberattacks.

Cybercriminals can use device vulnerabilities as a launchpad for their attacks, emphasizing security's influence from the planning stage. Attackers may exploit vulnerabilities to target IoT devices to launch more sophisticated security breaches or distribute malware throughout the network. IoT botnets can highlight the impact of device vulnerabilities and how cybercriminals have evolved to use them. Mirai, [22] one of the most well-known types of IoT botnet malware, made headlines in 2016 by bringing down significant websites in a distributed denial of service (DDoS) campaign involving thousands of compromised household IoT devices. The threats IoT devices bring to large corporations and smart cities have been the subject of a comprehensive investigation. Due to the widespread use of IoT, its inherent mobility, and standardization restrictions, advanced technologies that can automatically detect suspicious activity on IoT devices connected to local networks are required.

Attack detection using conventional approaches and outdated data processing techniques has become ineffective because of the widespread use of IoT. Due to the increased size of network traffic, detecting attacks and malicious data in the preliminary stages is exceedingly challenging. Various datasets have been used in the literature, comprising different solutions to build detection strategies for malignant attacks on IoT devices.

One of the methodologies is the implementation of Machine Learning Algorithms to detect attacks on IoT devices. Machine learning techniques make detecting network attacks more efficient since these algorithms automatically develop prediction models. In addition, it enables analyzing vast amounts of data and revealing hidden data patterns, useful for attack detection.

This thesis aims to apply machine learning to contrast two types of machine learning algorithms: supervised and unsupervised, for attack detection using two different feature sets obtained from two flow extractor tools. The features are extracted from the CICIoT2022 and Bot-IoT datasets, comprising various malicious and normal traffic captures. Before implementing machine learning algorithms on these datasets, the packet captures (pcap files) are converted into flow features using two independent flow extractor tools, Argus and Tranalyzer2. These flow exporters transform the packet data into flow data with relevant feature information, enabling better interpretation of flow data. The benign and attack data are combined as a single entity for both datasets following the extraction of flow features. Finally, supervised, and unsupervised machine learning algorithms are implemented, and results from the implementations are recorded.

The proposed framework leverages three supervised learning algorithms and three unsupervised learning algorithms. The Supervised Learning algorithms chosen were Decision Trees, Support Vector Machines (SVM) and K-Nearest Neighbors (KNN), and the unsupervised learning algorithms were Local Outlier Factor (LOF), One-class SVM and Isolation Forest Algorithm. These proposed models were trained, tested, and evaluated on the datasets by calculating recall, precision, Receiver Operating Characteristic-Area Under Curve (ROC-AUC), and F1-score metrics. The findings are then compared and analyzed to see which machine learning algorithms detected and predicted the attacks the best. The main goal of this thesis is to detect the diverse types of attacks against IoT devices using ML algorithms and determine which algorithm is better for prediction.

My proposed framework using these datasets with distinct flow characteristics will serve as a benchmark for future research. The new contributions of my thesis are as follows:

- Contrasting supervised and unsupervised machine learning algorithms for attack detection
- Investigating two distinct feature sets extracted from Argus and Tranalyzer2 to evaluate the machine learning model.
- Benchmarking the Argus and Tranalyzer2 features on the two publicly available datasets, namely CIC IoT 2022 dataset and Bot-IoT dataset.

The rest of the thesis is organized as follows. The relevant literature is summarized in Chapter 2. The proposed approach is introduced in Chapter 3, along with a discussion of the research methodologies. In Chapter 4, the experiments are described in detail, along with the findings. Finally, in Chapter 5, conclusions are drawn, and future research is discussed.

CHAPTER 2 LITERATURE REVIEW

This chapter provides a comprehensive assessment of the literature. The main objective of developing attack detection strategies for IoT devices is to detect attacks at an early stage and safeguard the devices due to the growing risk of attacks on IoT devices. Various authors have proposed several approaches for developing attack detection frameworks. I proposed a machine learning framework to detect IoT attacks for the purpose of my research. The following is a review of the studies done by researchers who created IoT frameworks using ML algorithms.

2.1 Previous Works

Because of their characteristics, ensuring the security of IoT networks remains a challenge for researchers, attributing to a range of diverse technologies, some of which need to be updated. Due to the unique characteristics of IoT, these technologies have typical privacy and security issues with data that must be addressed. While many scientists are striving to address various IoT security concerns, the security level of IoT devices currently needs to catch up to what users require [14]. Denial of service (DoS), distributed denial of service, botnet attacks, information theft, and eavesdropping attacks are known cyberattacks that can affect IoT networks. Such attacks frequently harm organizations using IoT networks because they make it difficult to maintain the system correctly by preventing device functionality.

As the number of IoT devices grows, so does the threat level of cybersecurity vulnerabilities they expose. Because the Internet of Things devices are autonomous and do not need human intervention to operate effectively, early detection of vulnerabilities and threats on IoT devices is pivotal. As a result, suitable network security solutions for the IoT system must meet several criteria, such as fast detection of attacks and predicting results with high accuracy.

I propose adopting machine learning algorithms to identify attacks in IoT devices. Several authors suggested employing machine learning algorithms for IoT attack detection problems. In addition, numerous studies have suggested that machine learning techniques like K-Nearest Neighbors, Decision Trees, Support Vector Machines (SVM), and Random Forest (RF) can help with attack detection tasks. Though the underlying goal of the research was to detect attacks, the results varied depending on the algorithms and methods used to solve the problem.

Alsamiri et al. [1] proposed the use of Machine learning algorithms like K-Nearest Neighbours (KNN), AdaBoost, and Naïve Bayes (NB) to detect Cyber Attacks on IoT devices efficiently. They used the Bot-IoT dataset to detect attacks on IoT devices. Initially, they extracted the flow-based features from the raw packets using the CICFlowMeter network traffic flow extractor. Once they extracted the flow features, they performed data preprocessing and divided the dataset into 80% training and 20% testing for further implementations. The primary goal of their research is to evaluate the performance of machine learning algorithms in detecting IoT attacks. In the next step, they used the Random Forest regressor algorithm to select the features that can help find a lightweight security solution for IoT devices. They selected only seven features for implementing ML algorithms in the feature selection step among the 84 features extracted from CICFlowMeter. The implementation step was then organized into three phases: applying the proposed algorithms on each attack in the dataset separately, applying the algorithms on the entire dataset with a set of features combining the best features for each attack and applying the algorithms on the entire dataset with the seven best features obtained in the feature selection step. To evaluate the results of implementation, F-measure was selected. Their evaluations show that K-NN outperformed the other algorithms with a higher F-measure of 99%.

Zeeshan et al. [2] proposed a solution for detecting DoS and DDoS attacks using UNSW-NB15 and Bot-IoT Datasets. They propose a Protocol Based-Deep Intrusion Detection architecture (PB-DID), in which they created a dataset of packets from IoT network traffic by comparing features from the UNSWNB15 and Bot-IoT datasets based on flow and TCP. Their proposed architecture involves comparing features from the two datasets to determine similar features, features selection process, data pre-processing and training the model using an unsupervised Long-Short Term Memory (LSTM) deep learning model. As a result of determining similar features, they found twenty-nine features that are similar in both datasets. Further, for implementation, twenty-six features were selected from twenty-nine features that contain maximum information on the packets like ipaddress and portaddress. Furthermore, they performed binary and multi-class classification on the datasets for evaluation. The results of evaluations show that their model achieved an accuracy of over 96.3% by covering both datasets.

Dwibedi et al. [3] proposed a paper that focuses on the contribution of data by analyzing three different IDS datasets, namely, UNSW-NB15, Bot-IoT, and CSE-CIC-IDS2018, employing various Machine Learning algorithms like Random Forest, Support Vector Machines, Keras, Deep Learning, and XGBoost. Their paper compares the performance of an ML-based IDS model by training it with each of the algorithms and analyzing how the choice of a dataset can impact the model's performance. Confusion Matrix, Precision and Recall were chosen as the evaluation metrics. They divided the datasets into 75% training and 25% testing. They used two scenarios for implementation and evaluation. The first scenario was to assess each dataset by training the model with each of the selected algorithms. A class of attacks is separated in the second scenario, and the model is trained with the remaining class. The evaluations determine whether the model can identify the attacks that are not trained. This methodology helped them understand whether the trained Machine Learning models could identify new attacks. Their evaluation shows that Random

Forest and SVM performed better than the XGBoost algorithm with high precision and a recall score of 100%.

Das et al. [4] proposed a paper on a comprehensive analysis of the accuracies of Machine Learning algorithms for Network Intrusion Detection. The main goal of this thesis is to detect network intrusion with the highest accuracy and perform faster. Supervised machine-learning algorithms, namely Random Tree, Random Forest, Bayesian Networks, Naïve Bayes, K-Nearest Neighbors, C.45, Reduced Error Pruning Tree, Repeated incremental pruning to produce error reduction and Partial Decision Tree were implemented to analyze the Bot-IoT dataset and the UNSW-NB15 dataset. In addition, they conducted three experiments: Implementation of SL algorithms on the UNSW-NB15 dataset to detect the anomaly and the type of anomaly, and Supervised Learning on the Bot-IoT dataset to detect the type of anomaly. Accuracy was chosen as the evaluation metric to evaluate the results of experiments. The evaluation results show that the best algorithm that does not compromise on accuracy is the Random Tree algorithm, with an accuracy of 96%.

Leevy et al. [5] proposed an Easy-to-Classify Approach for the Bot-IoT Dataset. Their method uses a minimum number of dataset features with a simple ML algorithm for accurate classification. For this purpose, they used only 3 of the 29 features from the Bot-IoT dataset and the Decision Tree classifier as their classification algorithm. To carry out their research goal, the performance of one-feature, two-feature, and three-feature Decision Tree models are evaluated to determine the minimum number of features for the correct classification. They ran stratified five-fold cross-validation ten times for one-feature, two-feature, and three-feature Decision Tree models. With the F1 score as an additional metric, they used the AUC and Area Under Precision-Recall Curve (AUPRC) metrics to analyze the results. Their findings from the evaluation concluded that only the three-feature model has a higher AUC and AUPRC scores above 0.99. This model's F1 score is 100%.

Krishnan et al. [6] proposed a paper on Attack detection on IoT networks using supervised machine learning algorithms. They used three supervised machine learning classifiers for predicting malicious and benign data in the network traffic data of IoT devices. The algorithms they chose were Random Forest regressor, Support Vector Classifier and Xtreme Gradient Boosting. The dataset used for performing the analysis was the IoTID20 dataset. The dataset was split into 80% training and 20% testing. The feature selection methods applied were Sequential Backward Processing, Sequential Forward Processing and Recursive Feature Elimination. After pre-processing, a separate IoTID20 dataset with 33 features was chosen for the experiment. According to their analysis, all three supervised feature selection methods predicted normal and anomalous traffic with high accuracy and, therefore, can be used to predict attacks on IoT devices. Recursive Feature Elimination resulted in better accuracy results for all three algorithms.

Ahmad et al. [7] proposed Supervised Machine Learning Approaches for Attack Detection in the IoT Network. The NSL-KDD dataset detected unusual behaviours in IoT networks by applying various machine-learning algorithms. Training data consists of 125,973 data with 67,343 normal data and 58,630 anomalous data, and Testing data consists of 22,543 data with 9710 normal data and 12,833 anomalous data with 41 features for testing and training data. Logistic Regression, K-Nearest Neighbor, Linear Support Vector Machine, SVM with RBF kernel, SVM with the polynomial kernel, SVM and logistic regression with stochastic gradient descent (hinge loss, log loss, Huber loss, and modified Huber loss function), Gaussian Naive Bayes, Bernoulli Naive Bayes, decision tree, random forest, and multilayer perceptron (MLP) classifier were the types of Machine learning algorithms used. Accuracy was chosen as the evaluation metric to determine the evaluation results. From their evaluations, stochastic gradient descent with log loss function has significantly less training time with an AUC value of 0.92 and an accuracy of 77.16%. Thus, they concluded that the random forest classifier and Stochastic Gradient Descent with log loss function perform better than the other classifiers for detecting malicious traffic.

Rani et al. [8] proposed an efficient method using Random Forest Classifier for intrusion detection. Two datasets, namely NSL-KDD and KDDCUP99, were used to supply lightweight attack detection for IoT networks. The amount of data in KDDCUP was 494,020; in NSL-KDD, the total data was 148,517. The attack classes were Denial of Services, Probe, User to Root and Remote to Local. Among 41 features, only ten key features were selected for training and testing. Therefore, a random Forest Classifier is used for training the model. Their proposed method provided a higher accuracy rate of 99.9% and has the highest accuracy rate. In this approach, the features are manually chosen after analyzing different attacks and their characteristics depending upon the feature and the minimal features that were extracted.

Ahmad et al. [9] proposed a paper on Intrusion detection on the Internet of Things using supervised machine learning based on application and transport layer features. The dataset used was the UNSW-NB15 dataset. The Intrusion detection system is built to prevent malicious traffic from entering the network. For the same, they proposed feature clusters in terms of Flow, Message Queuing Telemetry Transport and Transmission Control Protocol with features in the UNSW-NB15 dataset. The machine learning algorithms they adapted were Random Forest, Support Vector Machines and Artificial Neural Networks. Issues like imbalanced data, over-fitting, and the curse of dimensionality of the dataset were eliminated by data pre-processing to provide a consistent dataset for experimentation. After the feature selection and extraction process, 49 features were further reduced to 37 features in Binary and Multi classification. Random Forest algorithm performed well, providing higher accuracy, followed by Support Vector Machines and Artificial Neural Networks. RF outperformed in cluster-based methodology by achieving 96.96% in flow features, 91.4% in TCP features and 97.54% in flow features and TCP clusters.

Saheed et al. [10] proposed a Machine Learning based Intrusion Detection System to detect attacks on IoT networks. The main aim of their project is to apply Machine Learning Supervised Algorithms based Intrusion Detection Systems for IoT networks. Feature scaling was done using the min-max concept. The dataset used was the UNSW-NB15 dataset. The six Machine Learning algorithms adopted were Xtreme Gradient Boosting, Cat Boost, K-Nearest Neighbors, Support Vector Machines, Quadratic Discriminant Analysis, and Naïve Bayes classifiers. In addition, an unsupervised method, namely Principal component Analysis, was chosen for feature selection. For building the models, 75% of the data was used as training data and 25% as testing data. The performance of their proposed model was evaluated in terms of accuracy, AUC, recall, precision, F1-score, kappa, and Mathew Correlation Coefficient. The accuracy of their proposed method PCA-XgBoost gave the highest accuracy of 99.99% and F1-score of 100% of all other proposed methods. The PCA-Cat Boost also outperformed other proposed methods with an accuracy of 99.99% and an F1-score of 99.99%.

Haji et al. [11] proposed a paper with a review of various machine learning techniques that are employed in attack detection and anomaly detection on IoT networks from 2019 to 2021. In the detailed review, researchers used machine learning algorithms to detect attacks and anomalies on IoT networks. to conduct the experiments, they used various datasets like Bot-IoT, UNSW-NB15, NSL-KDD, IoT-23, and DS2OS. The researchers used and compared different ML algorithms on these datasets like Logistic Regression, Decision Trees, Random Forest, K-Nearest Neighbor, Support Vector Machines and Naïve Bayes algorithms. The review shows that the Random Forest (RF) algorithm yielded the best results with 99.34%, 99.5%, 99.4%, 99.9%, 99%, 99.5%, and 99.9% accuracy compared to the rest of the algorithms. Furthermore, Decision Trees and KNN perform better than other algorithms.

Alasmary et al. [12] proposed a solution to detect Distributed Denial of Services attacks depending on the flow of traffic. The proposed solution consists of the IoT node detector, a classifier to monitor the outgoing traffic and the server detector, a classifier used by the IoT node if it is susceptible to a DDoS attack. To develop such a detector, they proposed the ShieldRNN approach for Recurrent Neural Network (RNN) and Long-Short Term Memory model (LSTM). The experiment was set to detect DDoS attacks like TCP SYN Flood, UDP Flood, TCP PSH and ACK flood, and ICMP flood. To launch attacks, they developed a tool that randomly launches attacks with random packets, with each attack with randomly generated IP addresses, source addresses and destination addresses. Wireshark is used to collect data for normal traffic and attack data. After data preprocessing, the dataset was divided into 90% training and 20% testing. To train the lightweight detector for the IoT node, they trained 12 different classifiers, four Feedforward Neural Networks classifiers (ANN) with a single hidden layer with sizes 3, 5,10, and 20 neurons, one Logistic Regression (LR) classifier, three Random Forest classifiers with different numbers of trees, three SVM classifiers with three different kernels, namely linear, RBF, Polynomial with the third degree, and one Naïve Bayes classifier. RandomForest classifiers achieved the best results compared to other classifiers when they were evaluated on the testing set with Accuracy and an F1-score of 100%. Additionally, they used two training and prediction strategies to train numerous RNN and LSTM models. Sequence-to-sequence training is the first technique in which the sequence model is trained to predict a label for each packet in the training sample. This methodology uses a majority voting method for prediction. Therefore, if most sequence packets are anticipated as attacks, the entire sequence will be predicted as attacks. If not, the final prediction is expected to be normal. As a result, they achieved the highest F1-score of values: 99.919%, 99.822%, 100%, 99.834%, 100%,100%, 100%, and 100% when they were tested on their dataset with the randomly generated list of sequence lengths: 26,57, 77, 212, 329, 597, 643, and 877, respectively. They also set a baseline result for DDoS detection using ShieldRNN on the CIC-IoT2022 dataset.

2.2 IoT Attacks Studied

Based on the prior research examined, it is clear that various attacks were identified during the evaluation stages of the proposed architecture. DoS and DDoS attacks are the most detected types of attacks. In my thesis, to be comparable with the previous studies I have also employed their IoT datasets. The attacks that were included in those datasets are listed below:

- Denial of Services (DoS)
- Distributed Denial of Services (DDoS)
- Brute Force attack
- Scan attacks   and
- Theft attacks

2.3 IoT Vulnerabilities

The IoT devices are vulnerable to attacks like DoS, DDoS and theft attacks given the reasons such as usage of default passwords and insecure network services. The vulnerabilities that make the attackers exploit them and launch attacks on IoT devices and networks are analyzed here.

- **DoS and DDoS attacks** - The most common type of DDoS attacks is flooding the target system. In these attacks, input is received that exploits vulnerabilities in the target system and causes the system to crash or become very unstable, making it impossible to access the system.
- **Brute Force attacks** - A brute-force attack is when an attacker tries a number of different user credentials in an effort to guess the right ones. The majority of these attacks are automated and use word lists of usernames and passwords.
- **Scan attacks -** Automated tools are usually used in scanning attacks to look for open ports, vulnerabilities, and other bugs that could be used to obtain unlawful access or launch an attack.

- **Theft attacks –** This attack mainly includes gaining access to a user's information by stealing the password information or security codes. Keylogging is one of the categories under theft attacks analyzed in my research. Keyloggers, often known as keystroke loggers, are devices that capture what users enter on a keyboard. Every keystroke made on the victim's device is recorded by keylogger software during a keylogger attack and is sent to the attacker. In the Bot-IoT dataset, they used log-keys software to record the keystrokes on smartphones.

2.4 SUMMARY

The papers above discuss the works of various researchers who implemented Machine Learning algorithms for their research purposes. As mentioned in Chapter 1, my research goal is to detect attacks on IoT devices. For this purpose, I used two datasets comprising IoT attack packets, namely, the Bot-IoT dataset and the CIC-IoT2022 dataset, where the Bot-IoT dataset is used widely in literature, and the CIC-IoT2022 dataset, which is the most recently developed dataset. The papers [1],[2],[3],[4],[5], and [12] use the Bot-IoT dataset for the implementation of the author's proposed framework, and papers [1],[6], and [7] discuss the author's works on attack detection on IoT devices using different feature sets.

The author's works address using Supervised and Unsupervised Machine Learning algorithms to build their models. However, compared to Unsupervised Learning algorithms, several authors implement Supervised Learning algorithms [1],[3],[4],[5],[6],[7],[8],[9],[10],[11],[12] to build their models. However, none of their works discussed how implementing Supervised and Unsupervised ML algorithms on different datasets and feature sets results in the overall performance of their proposed frameworks. For implementing ML algorithms, the authors only used one feature set for a dataset. Also, their models only used part of the feature set by not removing the biased features, as features like ipaddress and portaddress [2] contain maximum information on the packets that may produce biased results in the evaluation. To evaluate their models, they used either F1- score or accuracy [2],[4],[6],[9],[7],[8],[12] as their evaluation metrics. However, accuracy is not a valid metric in imbalanced datasets since a valid metric must account for the proportion of data instances from various classes that are correctly categorized, and accuracy makes no distinction between the amount of correctly identified examples of different classes. Therefore, assessing the model's performance in terms of accuracy [38] could result in wrong conclusions and is considered a drawback.

I proposed an IoT attack detection framework to overcome the existing solutions' drawbacks and make new contributions to the literature. My proposed framework uses two feature sets for the CIC-IoT 2022 and Bot-IoT datasets. They are used to build Machine Learning models for IoT attack detection and contrast the results of Supervised and Unsupervised Machine learning algorithms. Also, to prevent the models from producing biased results, I have removed features like ipaddress, port address, mac address, and seq numbers which have maximum detail about the packets, while using the rest of the features extracted from the flow extractor tools. To assess the performance of my proposed framework, I used Weighted Average F1-score. The weighted average [37] is chosen if you have an imbalanced dataset but want to give more weight to classes with more examples. With weighted averaging, each class's contribution to the F1 average is weighted according to its size. The datasets used for my proposed framework have an unequal distribution of benign and attack packets making the datasets naturally imbalanced (a detailed description of the datasets is mentioned in Chapter 3). Therefore, I used Weighted Average F1-score to evaluate the algorithms and assess the model's performance.

In contrast to existing solutions, my new contributions to the literature are:

- Comparing the evaluations of flow extractors (Argus and Tranalyzer2).
- Using two feature sets to implement six different ML algorithms.
- Comparing the performance and evaluation results of Supervised and Unsupervised Learning algorithms.
- Using Precision, Recall, Area Under the Curve (AUC-ROC) and Weighted Average F1-score as an evaluation metric to evaluate my proposed framework.

The architecture of my proposed framework with the methodologies and the evaluation results of my experiments will be discussed further in Chapters 3 and 4.

CHAPTER 3 METHODOLOGY

This chapter gives detailed information on the proposed framework and methodologies implemented. The main research goal of this thesis is to detect attacks on IoT network traffic by implementing Supervised and Unsupervised machine learning algorithms on flow data using two different feature sets. The proposed framework is comprised of three phases. The network traffic data comprising benign and attack packets from the CICIoT2022 and the Bot-IoT dataset was collected in Phase I. The data are initially presented as packet dumps (pcap files). In Phase II, the pcap files collected from Phase I are converted into flow features using Argus and Tranalyzer2 flow extractor tools. Once the flows are extracted, in Phase III, supervised and unsupervised ML algorithms are implemented, and the results of the evaluations are recorded. Figure 3.1 represents the architecture of the proposed framework. The rest of the chapter discusses in detail the methodologies used in the proposed framework.

3.1 Phase I – Network Traffic

In phase 1, the network traffic data from the CICIoT2022 [17] and Bot-IoT datasets are collected. For this purpose, datasets that comprise network traffic data generated from IoT devices are required. The datasets used for my research incorporate benign and attack traffic data regarding packet dumps. However, the CICIoT2022 dataset has more benign data than attack data, and the Bot-IoT dataset has more attack data, making the datasets divergent.

3.1.1 CICIoT2022 dataset

The University of New Brunswick generated this dataset in 2022 by collecting network traffic from IoT devices like cameras, home automation, and sensors with various protocols like IEEE 802.11, Zigbee-based, and Z-wave protocols. The traffic was generated by analyzing the different behaviours of the devices in different scenarios comprising benign and attack packets. The generated data were recorded using Wireshark [23] – a network protocol analyzer, and the output packet captures (pcap) were saved.

In this dataset, the benign data were collected in five modes: Power, Idle, Interactions, Active and Scenarios. The attacks performed were Flooding and RTSP-Brute Force. The flooding attacks were further categorized as HTTP Flood, TCP Flood and UDP Flood. This imbalanced dataset has more benign flows of more than three million with attack flows of less than fifty thousand packets.

In Power mode, the network traffic data of the IoT devices were captured by powering on each device individually and capturing the data. In Idle mode, the network traffic data was captured at a specific time without human interactions or intervention. For Interactions, every relevant feature on IoT devices has been extracted, and the related network activity and transmitted packets have also been recorded. During Active mode, the network traffic data is recorded from human interactions with the IoT devices. Various experiments were conducted on IoT devices, and the relevant traffic data was recorded for scenarios.

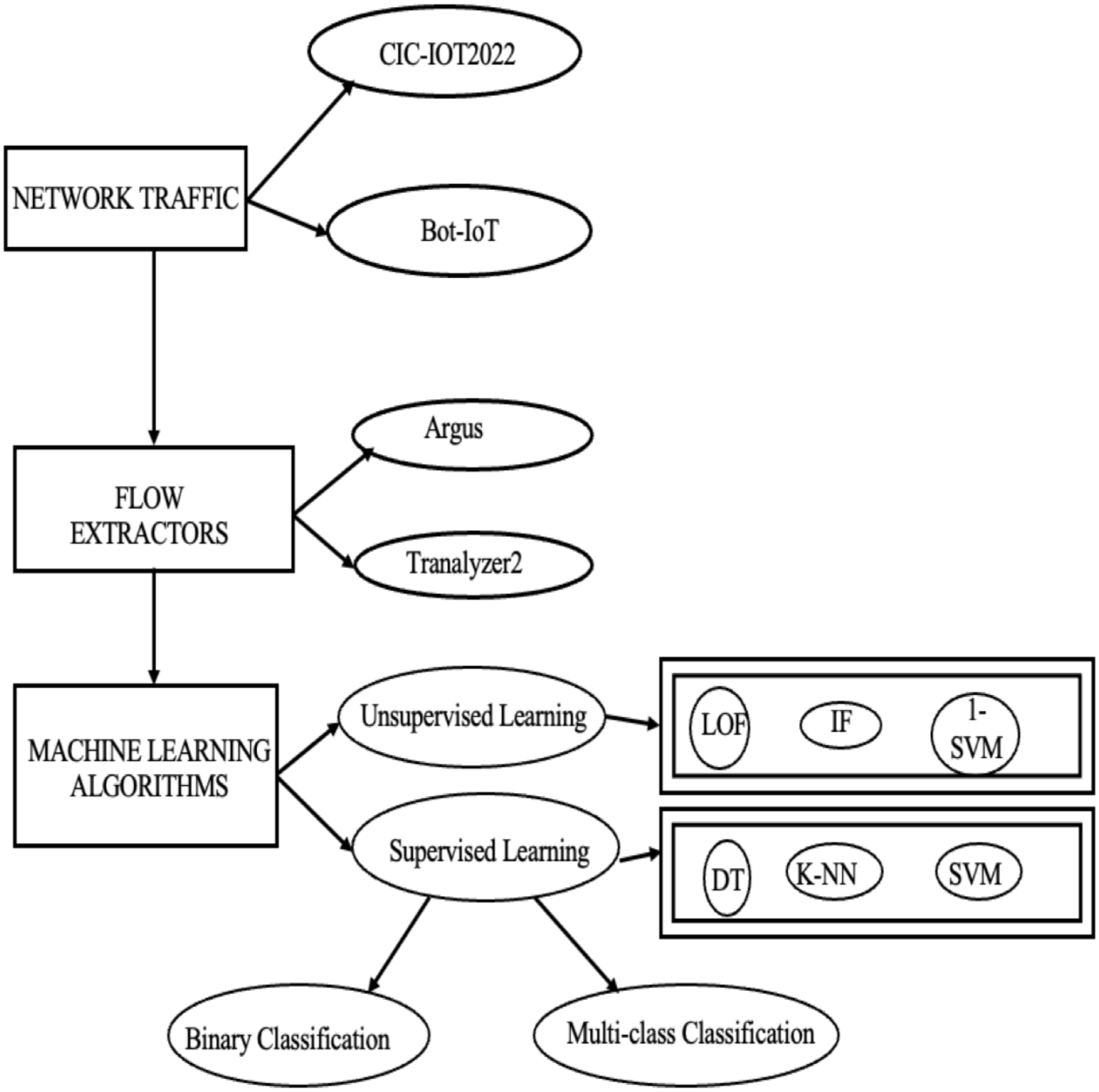

Figure 3.1 Architecture of the proposed framework

For collecting the Attack data, the researchers performed flood and brute force attacks on the IoT devices and captured the generated traffic. Flood attacks were classified as HTTP flood, TCP flood and UDP flood. In addition, Hydra and Nmap were used to perform a Brute Force attack on IoT devices. All these were analyzed with the help of a real-time network traffic analyzer tool, and all the packet data was saved. Figure 3.3 shows the graphical representation of the CICIoT2022 dataset.

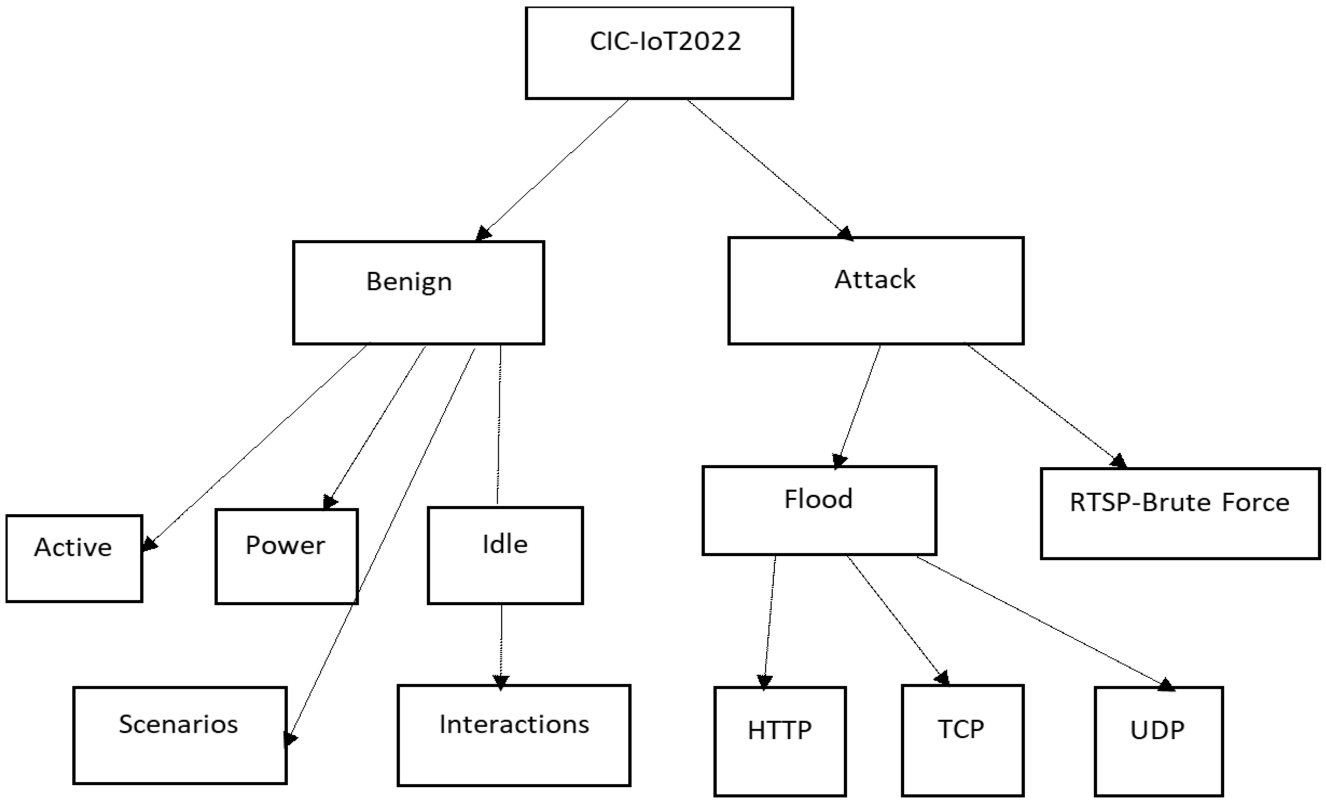

Figure 3.3 Graphical representation of the CIC-IoT2022 dataset

3.1.2 Bot-IoT dataset

In the Cyber Range Lab of UNSW Canberra Cyber, a network environment was established to create the Bot-IoT dataset. The environment simulated both botnet traffic and benign traffic. The source files for the dataset are available in several forms, such as the original pcap files with more than seventy-two million records with benign packets of less than ten thousand, the modified argus files, and CSV files. To make handling the dataset easier, they extracted 5% of it using certain MySQL queries. About 3 million records make up the extracted 5%. The files were divided based on attack category and subcategory. For example, the DDoS and DoS attacks are further classified according to the protocol employed in the dataset, including OS and Service Scan, Keylogging, and Data Exfiltration attacks. The Bot-IoT dataset's raw network packets (Pcap files) were collected using tshark [15], and Ostinato tool [18] and Node-red [19] were used to create simulated network traffic (for non-IoT and IoT respectively). Figure 3.4 shows a graphical representation of the Bot-IoT dataset.

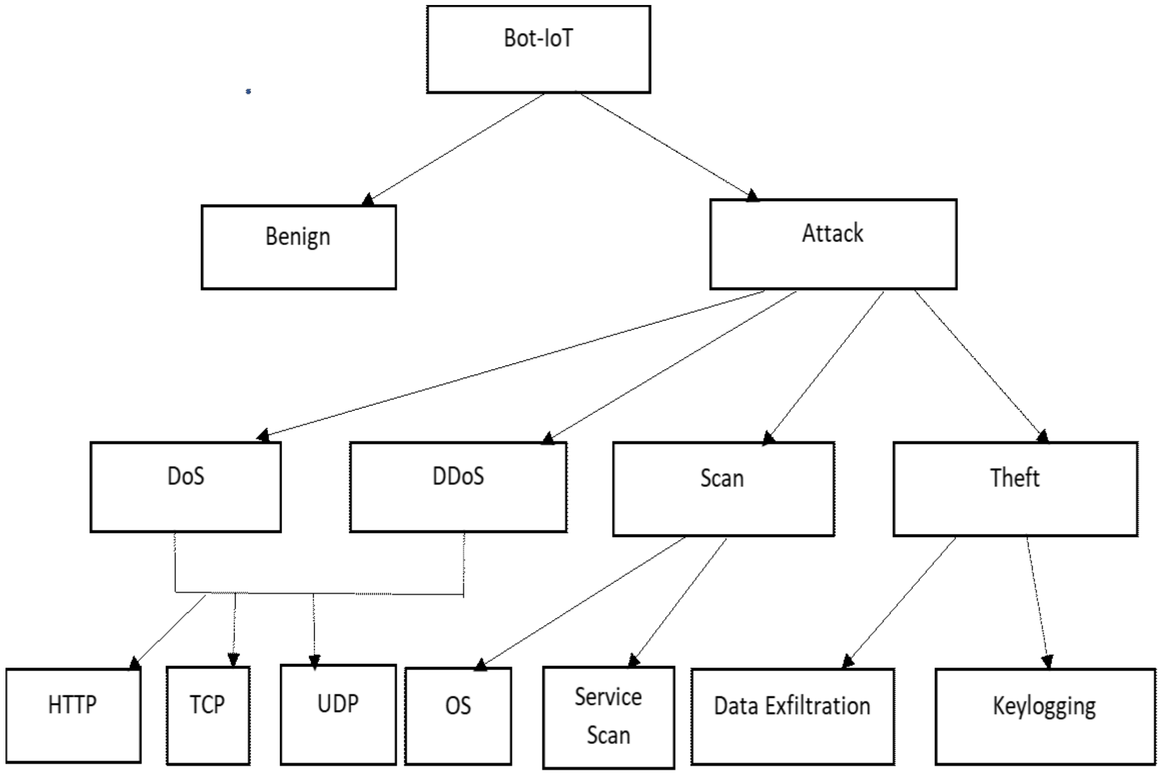

Figure 3.4 Graphical representation of Bot-IoT dataset

3.2 Phase II – Flow Extractors

Argus and Tranalyzer2, flow extractor tools, were used to convert the packet data collected from the former phase into flow data and generate features for further experimentation. These flow extractor tools help extract flows from the packets with relevant feature information, enabling better interpretation of flow data.

3.2.1 Argus

Argus is a system for monitoring network traffic flow in both directions [20]. It offers details about the state of the network flow. Argus, a network monitoring tool first released in 1993 and written in C, supported dispersed network architecture. Argus uses its binary format for flow extraction and supports many protocols, including TCP, ARP, ICMP, and ESP. However, Argus can only extract the flows represented in binary format. So, for Argus, the packet data is initially converted to binary format, and the flows are extracted. Then, the extracted data is converted from .txt to .csv to build and evaluate the machine learning algorithms. Thirteen flow features were taken from Argus for this study. After that, these features are utilized to examine how ML classifiers might employ them. A detailed description of the Argus flow features is mentioned in the Appendix section of this thesis.

3.2.2 Tranalyzer2

Tranalyzer2 [21] is a flow-based traffic analyzer developed on a flexible plugin-based architecture allowing efficient network traffic processing and analysis. It is a simple flow analyzer tool built in C that includes troubleshooting plug-ins. Tranalyzer2 extracts 109 features for each network flow. TShark [15] and Tranalyzer2 provide packet mode; however, Tranalyzer2 has a unique numerical ID connecting every packet to its flow. Tranalyzer2, based on the libpcap library, accepts not only IPv4/6 but also layer two and encapsulated packets such as MPLS, L2TP, and GRE from regular pcap files or live interfaces. With the help of a straightforward API, it is a memory-effective flow aggregator that makes it easier to create plug-ins. In addition, the output is accessible in text or binary format for future post-processing. A detailed description of the Tranalyzer2 flow features is mentioned in the Appendix section of this thesis.

3.2.3 Number of flows extracted

CIC-IoT2022 dataset – Binary classification of Argus and Tranalyzer2 flows

Class name	Number of flows extracted
Benign	2499316
Attack	207723

Table 3.1 Number of Argus and Tranalyzer2 flows extracted for binary classification from the CIC-IoT 2022 dataset.

CIC-IoT2022 dataset – Multi-classification of Argus and Tranalyzer2 flows

Class name	Number of flows extracted
Benign	2499316
HTTP	171809
TCP	15794

Class name	Number of flows extracted
UDP	10629
RTSP	9491

Table 3.2 Number of Argus flows extracted for multi-classification from the CIC-IoT 2022 dataset

Bot-IoT dataset – Binary classification of Argus flows

Class name	Number of flows extracted
Benign	477
Attack	15591796

Table 3.3 Number of Argus flows extracted for binary classification from the Bot-IoT dataset

Bot-IoT dataset – Multi-classification of Argus flows

Class name	Number of flows extracted
Benign	477
UDP-DoS	5578193
UDP-DDoS	4016098
TCP-DDoS	3115880
TCP-DoS	2245684
OS-Fingerprint	398816
OS-Service Scan	112378

Class name	**Number of flows extracted**
HTTP-DoS	69906
HTTP-DDoS	49856
Keylogging	3663
Theft	1322

Table 3.4 Number of Argus flows extracted for multi-classification from the Bot-IoT dataset.

Bot-IoT dataset – Multi-classification of Tranalyzer2 flows (Attack multi-classification)

Class name	**Number of flows extracted**
UDP-DoS	267063
UDP-DDoS	267441
TCP-DDoS	633998
TCP-DoS	558804
OS-Fingerprint	104028
OS-Service Scan	37679
HTTP-DoS	22189
HTTP-DDoS	20191
Keylogging	1574
Theft	411

Table 3.5 Number of Tranalyzer2 flows extracted for Attack multi-classification from the Bot-IoT dataset.

Following the flow features extraction, Phase III discusses the implementation of ML algorithms on the datasets using the flow features.

3.3 Phase III – Machine Learning Algorithms

Phase III is the essential phase of the proposed framework. Once the flow features are extracted, the next step is implementing machine learning algorithms [25] on the datasets and evaluating the models. Features, namely port address, IP address, mac address, and sequence numbers, are considered potential biases during implementations.

So, these features are removed from the flow features before building the model. Once the features are removed, the dataset is divided into 70% training and 30% testing. After completing the train test split, the next step is to fit the model using Machine Learning Algorithms.

My proposed framework includes implementing three supervised and three unsupervised learning algorithms: Decision Trees, Support Vector Machines, K-Nearest Neighbors, Local Outlier Factor, Isolation Forest, and One-class SVM.

3.3.1 Supervised Learning

Supervised learning [32],[41] is a machine learning approach for problems with labelled examples, where each data point has features and an associated label. Feature vectors (inputs) are converted to labels (outputs) through supervised learning algorithms, which learn a function based on sample input-output pairs. Training and testing are the two primary phases of supervised learning. Creating a classification model is called the training phase. The testing phase employs the classifier created in the initial phase to categorize examples not observed or trained. Supervised Learning algorithms use labelled training data, which consists of a collection of training samples, to determine a function. Each example in supervised learning consists of two elements: a desired output and an input. A supervised learning algorithm evaluates training data to provide an inferred function that may be used to map new data samples. The algorithm can accurately detect the class labels for instances not observed in the ideal situation, which requires the learning algorithm to generate reasonable conclusions from the training data to possible scenarios. The classifier for analyzing network traffic is built using supervised learning (SL) algorithms. I performed binary classification and multi-class classification in implementing the Supervised Learning algorithms.

Binary classification is when the classification is made between two classes: Benign and Attack classes. Multi-class classification is when classification is performed to predict different classes. For my research, I have performed multi-class classification among benign data and distinct kinds of attack data: DoS attacks, DDoS attacks, OS and Service Scan, Keylogging, Data Exfiltration attacks and Brute-Force attacks.

3.3.1.1 Decision Trees

The decision tree [26],[43] learning approach is most frequently employed in machine learning. The goal of a decision tree is to build a model that predicts the value of a target variable based on multiple input variables. Each inner (non-leaf) node is labelled with an input characteristic in a decision tree or classification tree. For example, each possible value of the target feature is labelled on the arcs that originate from a node with an input feature, or the arc may lead to a decision node on a different input feature. In addition, each leaf of the tree has a class or probability distribution across the classes labelled on it, indicating that the tree has classified the data set into a particular class or probability distribution. In simple terms, decision trees are just a collection of if-else statements. If the condition is satisfied, it determines whether to go on to the subsequent node associated with that choice. It offers a top-down structure tree with an iterative splitting of the training dataset.

The following chapter shows the visualization of the implemented decision tree model.

3.3.1.2 K-Nearest Neighbors

The K Nearest Neighbor algorithm [27],[44] stores all available scenarios and categorizes additional data or instances based on a similarity measure. A data point is often categorized using the classification of its neighbors. The outcome of the K-NN classification algorithm is a class identification. The class of an object is determined by a majority vote of its neighbors, with the object given to the class that is most prevalent among its k nearest neighbors (k is a positive integer). When K = 1, an instance is just put in the class of the nearest neighbor. In K-NN classification, all computation is postponed until after the function has been evaluated and the function is only locally approximated. Since this technique relies on distance for classification, normalizing the training data may improve its accuracy if the features reflect several tangible units or have significantly different scales. The neighbors are chosen from a collection of known-class items. Although no explicit training phase is needed, these known-class items can be viewed as the algorithm's training set. The training examples are vectors with class labels in a multidimensional feature space. Only the training samples' feature vectors and class labels are stored during the algorithm's training phase. As determined by a distance function, the K-Nearest Neighbors of the case allocated to the class share the most instances of it. Here, the number of neighbors' n' indicates the number of classes.

3.3.1.3 Support Vector Machines

In a high-dimensional space, SVM [28],[45] classifies data using a single or a collection of hyperplanes. SVM attempts to lower the possibility of generalization errors when creating the classifier. It is implemented when the primary hyperplane is chosen, which provides the most significant distance between the nearest instances of the classes in the training dataset. Support vector machines are supervised learning models with corresponding learning algorithms used in machine learning to classify data. An SVM training method creates a model that classifies new examples into one of two categories based on a series of training examples, each of which has been marked as belonging to one of the two categories. SVM assigns training samples to spatial

coordinates to maximize the distance between the two categories. Then, based on which side of the gap they fall, new samples are projected into that area and predicted to belong to a category.
There are several Kernels available for Support Vector Machines implementation. For my research, I have used Linear Kernel to evaluate the model.

A detailed description of the algorithms can be found in [24] and [41].

3.3.2 Unsupervised Learning

Unsupervised learning [33],[46] is an algorithm that discovers patterns from unlabeled data. An unsupervised learning algorithm attempts to resemble the input data during the training phase. It uses the error in the output of its simulation to resolve itself (correct its weights and biases). Without prior data training in contrast to supervised learning, the machine's objective in unsupervised learning is to categorize unsorted data according to similarities, patterns, and differences. Unlike supervised learning, unsupervised learning algorithms enable users to perform more complicated analysis tasks.

3.3.2.1 Local Outlier Factor

The Local Outlier Factor (LOF) [29],[47] is based on the idea of a local density, in which locality is determined by the distance between the nearest k neighbors, which is used to estimate the density. A region of similar density and points with a significantly lower density than its neighbors can be found by comparing the local densities of an object and its neighbors. It is the most popular and commonly applied unsupervised learning algorithm. The concept of nearest neighbors is used to calculate the anomaly or outlier score. The Local Outlier Factor algorithm calculates the local density deviation of a particular data point concerning its neighbors. The samples with a significantly lower density than their neighbors are regarded as outliers. LOF greater than one is considered an anomaly, and a LOF approximately equal to one is considered normal.

3.3.2.2 Isolation Forest

The Isolation Forest algorithm (IF) [31], [48] is based on the idea that erroneous data points can be easily distinguished from the rest of the sample. They are built using the decision trees algorithm. Additionally, this model is unsupervised because there are no predefined labels present. Instead, anomalies are defined as data points upon which Isolation Forests were developed. IF randomly chooses an attribute, then chooses a split value between the minimum and maximum values permitted for that characteristic to construct partitions on the sample that can be used to isolate a data point. Isolation Forest uses binary trees to find anomalies. The algorithm's minimal memory requirements and linear time complexity make it effective for handling large amounts of data. Data points that require fewer splits to be isolated are given higher anomaly scores by Isolation Forest, which divides the data space using orthogonal to the origin lines. An Isolation Forest processes data randomly sub-sampled in a tree structure using randomly chosen features. For example, if it took more branches to isolate the samples that travelled further into the tree, they are less likely to be anomalies.

Similarly, samples with shorter branches show anomalies because the tree found it easier to distinguish them from other data. An ensemble of many such trees may be used for better results because a single isolation tree has a lot of expected variability in the isolation depths that it will give to each observation. The final score is obtained by averaging the results (the isolation depths) from many such trees.

3.3.2.3 One-class SVM

One-Class SVM (1-SVM) [30],[49] is an unsupervised learning algorithm that distinguishes test samples from other classes. One-Class SVM and SVM are similar, but 1-SVM has just one class. As a result, a boundary is determined based on the given data. 1-SVM is based on determining the smallest hypersphere comprised of all the data points. The fundamental principle of 1-SVM is to minimize the hypersphere of the single class of examples in the training data and to treat all other samples as outliers or samples that do not fit the training data distribution. One-Class SVM, a class offered by SK-learn, internally accomplishes the mathematical modelling of hypersphere minimization by training on data samples. One-class SVMs do not require target labels throughout the model training process, in contrast to conventional supervised SVMs. Instead, it learns the boundary for the usual data points and recognizes the data outside the boundary as anomalies. In simple terms, any new data that deviates from the range is categorized as an outlier.

A detailed description of the algorithms can be found in [24] and [46].

SUMMARY

This chapter discusses in detail the methodologies used in my proposed framework. The proposed framework has three distinct phases. In phase I, the network traffic data is collected from the CIC-IoT2022 and the Bot-IoT datasets following the extraction of flow features of packet data in phase II. Once the dataset is processed, ML algorithms are implemented in them. Phase III discusses the ML algorithms used for implementation. I used SL and USL algorithms for implementing the data. In supervised learning algorithms, binary classification and multi-class classification techniques are performed. The next step after the implementation of the models is experimentation and evaluation. Chapter 4 discusses in detail the evaluation metrics used and the results of evaluations.

CHAPTER 4 EVALUATION AND RESULTS

Chapter 3 discussed the methodologies used to implement the proposed framework. This chapter discusses in detail the results of implementation. As mentioned in the previous chapter, six ML algorithms were used to evaluate the proposed framework using two feature sets obtained from Argus and Tranalyzer2. The results from the CICIoT2022 and Bot-IoT datasets are categorized based on the classification algorithms as Argus binary, Argus Multi, Tranalyzer2 binary and Tranalyzer2 multi. The rest of the chapter discusses the results and determines which algorithm outperformed with the highest evaluation score.

4.1 Evaluation Metrics and Results

All experiments are carried out on a Mac Book Air equipped with an M1 chip and 8GB of RAM. The Evaluations of the implemented ML models are done using SKlearn metrics [34] such as the F1-score [35], Precision, Recall, and Area under the Curve (ROC-AUC) [36].

Precision:

It measures the ratio of correctly predicted malicious flows to the total number of malicious flows.

Precision = True Positive / True Positive + False Positive

Recall:

It quantifies the ratio of attack instances correctly detected out of the total number of actual attacks.

Recall = True Positive / True Positive + False Negative

F1-score:

It is the harmonic mean of Precision and Recall

*F1-Score = 2 * (Recall * Precision) / Recall + Precision*

ROC-AUC:

It stands for the for the Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) curve between the true positive rate (i.e., recall) against the false positive rate (FPR). It tells how much the model can distinguish between classes.

Each experiment is evaluated using these metrics, and the predictions made are analyzed to determine which algorithm performed better. Weighted Average F1-Score [37] is used to determine the classifiers' overall performance. As mentioned in the previous chapter, I have used two different feature sets to evaluate the algorithms.

The results obtained from each experiment are as follows:

4.1.1 Results of CIC Dataset Binary Classification

Supervised Learning:

	Class Names	Decision Tree	K-Nearest Neighbor	Support Vector Machines
Precision	**Benign**	0.99	0.96	0.92
	Attack	0.90	0.83	1.00
Recall	**Benign**	0.99	0.99	1.00
	Attack	0.88	0.55	0.00
F1-score	**Benign**	0.99	0.98	0.96
	Attack	0.89	0.66	0.00
Weighted Average F1-score		**0.98**	0.95	0.89
ROC-AUC		0.93	0.77	0.50

Table 4.1 Results of Argus Binary classification

Table 4.1 displays the evaluation results of Argus Binary Classification for the CIC dataset. As mentioned in Chapter 3, in this dataset, the benign flows outnumber the attack flows, which resulted in the score variation between the classes. Therefore, I chose the Weighted Average F1-Score (WAF1-score) for an overall evaluation of the classifiers. According to the results of the evaluation, the Decision Tree algorithm outperforms KNN and SVM algorithms with the highest WAF1-score of 98%, and KNN outperforms SVM with a WAF1-score of 95% compared to SVM's 89% regardless of SVM having a higher precision score for attack class and recall score for benign class of 100%.

	Class Names	Decision Tree	K-Nearest Neighbor	Support Vector Machines
Precision	**Benign**	1.00	1.00	1.00
	Attack	1.00	1.00	1.00
Recall	**Benign**	1.00	1.00	1.00
	Attack	1.00	1.00	1.00
F1-score	**Benign**	1.00	1.00	1.00
	Attack	1.00	1.00	1.00
Weighted Average F1-score		**1.00**	**1.00**	**1.00**
ROC-AUC		0.99	0.99	0.99

Table 4.2 Results of Tranalyzer2 Binary classification

Table 4.2 displays the results of the binary classification of the CIC dataset using Tranalyzer2 flow features. Since tranalyzer2 features outnumber argus features (mentioned in Chapter 3), it resulted in better performance of the classifiers. As a result, all three classifiers performed better, with a higher average F1-score of 100%.

Compared to the binary classification of the CIC dataset regarding flow extractor performance, tranalyzer2 has better evaluation results than argus. However, the decision tree has a consistent score in both cases.

Unsupervised Learning:

	Class Names	**Local Outlier Factor**	**One-Class SVM**	**Isolation Forest**
Precision	**Benign**	0.92	0.92	0.93
	Attack	0.06	0.05	0.10
Recall	**Benign**	0.98	0.99	0.82
	Attack	0.02	0.01	0.23
F1-score	**Benign**	0.95	0.96	0.87
	Attack	0.02	0.01	0.14
Weighted Average F1-score		**0.88**	**0.88**	0.82
ROC-AUC		0.49	0.49	0.52

Table 4.3 Argus Binary Classification results

Table 4.3 displays the evaluation results obtained from unsupervised learning of the CIC dataset using argus flow features. The results show that LOF and 1-SVM performed better than IF, with a WAF1-score of 88%.

Table 4.4 displays the evaluation results of unsupervised learning of the CIC dataset using tranalyzer2 flow features. It is evident from the results that One-class SVM has the highest WAF1-score of 66% when compared to Local Outlier Factor and Isolation Forest algorithms with WAF1-scores of 62% and 65%, respectively. The Benign class from argus and tranalyzer2 flow features have consistent performance scores compared to the Attack class. In both cases, One-class SVM's performance is consistent with better performance regarding WAF1-score. Comparing the overall binary classification results in terms of the ML algorithms of the CIC dataset, SL algorithms have a better performance score than USL algorithms.

	Class Names	Local Outlier Factor	One-Class SVM	Isolation Forest
Precision	**Benign**	0.74	0.76	0.76
	Attack	0.06	0.13	0.17
Recall	**Benign**	0.88	0.99	0.92
	Attack	0.03	0.01	0.05
F1-score	**Benign**	0.81	0.86	0.83
	Attack	0.04	0.01	0.07
Weighted Average F1-score		0.62	**0.66**	0.65
ROC-AUC		0.45	0.49	0.48

Table 4.4 Tranalyzer2 Binary Classification results

Moreover, in terms of the flow extractor performance in the CIC dataset for binary classification, tranalyzer2 outperforms argus in supervised learning, and argus has a better weighted average F1-score than tranalyzer2.

4.1.2 Results of Bot-IoT Dataset Binary Classification

Supervised Learning:

	Class Names	Decision Tree	K-Nearest Neighbor	Support Vector Machines
Precision	**Benign**	0.50	0.37	1.00
	Attack	1.00	1.00	1.00
Recall	**Benign**	0.52	0.57	0.08
	Attack	1.00	1.00	1.00
F1-score	**Benign**	0.51	0.45	0.15
	Attack	1.00	1.00	1.00
Weighted Average F1-score		**1.00**	**1.00**	**1.00**
ROC-AUC		0.75	0.78	0.54

Table 4.5 Argus Binary Classification results for Supervised Learning

Results from Table 4.5 show that the weighted average F1-score for all three classifiers has a higher score of 100%, given that the F1-scores for the attack class in all the classifiers have a better performance score of 100% compared to the benign class. Their performance dropped as they struggled to detect the least representative, benign class.

Unsupervised Learning:

	Class Names	Local Outlier Factor	One-Class SVM	Isolation Forest
Precision	**Benign**	0.00	0.00	0.00
	Attack	1.00	1.00	1.00
Recall	**Benign**	0.64	0.18	0.23
	Attack	0.11	0.31	0.32
F1-score	**Benign**	0.00	0.00	0.00
	Attack	0.19	0.48	0.49
Weighted Average F1-score		0.19	0.48	**0.49**
ROC-AUC		0.37	0.24	0.27

Table 4.6 Argus Binary Classification results for Unsupervised Learning

Table 4.6 shows that Isolation Forest has better evaluation results than One-class SVM and Local Outlier Factor. LOF performs the worst with a low F1-score of 19%. However, all three classifiers have a higher precision score of 100% in the attack class.

When comparing the Binary classification results of the two datasets, Supervised Learning algorithms outperform Unsupervised Learning algorithms with a higher weighted average F1-score of 100%.

4.1.3 Results of CIC Dataset Multi Classification

Supervised Learning:

	Class Names	**Decision Tree**	**K-Nearest Neighbor**	**Support Vector Machines**
Precision	**Benign**	0.99	0.97	0.92
	HTTP	0.87	0.74	0.00
	TCP	0.63	0.29	1.00
	UDP	0.97	0.78	0.98
	RTSP	0.34	0.25	0.00
Recall	**Benign**	0.99	0.99	1.00
	HTTP	0.87	0.68	0.00
	TCP	0.51	0.04	0.01
	UDP	0.91	0.20	0.01
	RTSP	0.21	0.05	0.00
F1-score	**Benign**	0.99	0.98	0.96
	HTTP	0.87	0.71	0.00
	TCP	0.56	0.06	0.02
	UDP	0.94	0.32	0.03
	RTSP	0.26	0.08	0.00
Weighted Average F1-score		**0.98**	0.95	0.89
ROC-AUC		0.83	0.66	0.50

Table 4.7 Argus Multi-Classification results for Supervised Learning

Table 4.7 displays the multi-Class classification results of the argus flow feature from the CIC dataset. The metrics used to evaluate the models are the same for both binary and multi-classification. Given that the dataset is imbalanced (Chapter 3), the benign class has high precision and recall scores compared to each score of the attack classes. Compared to other classes, RTSP has a mediocre performance score. Regarding the WAF1-score, the decision tree performs better, with a score of 98%, compared to K-NN and SVM, with 95% and 88%, respectively. Nevertheless, when comparing KNN and SVM, K-NN has a consistent performance compared to SVM, which performs poorly in some classes.

	Class Names	Decision Tree	K-Nearest Neighbor	Support Vector Machines
Precision	**Benign**	1.00	1.00	1.00
	HTTP	0.99	0.99	0.98
	TCP	0.80	0.97	0.99
	UDP	0.99	0.99	0.99
	RTSP	1.00	1.00	1.00
Recall	**Benign**	1.00	1.00	1.00
	HTTP	0.99	1.00	1.00
	TCP	0.81	0.75	0.51
	UDP	0.99	0.88	0.98
	RTSP	1.00	0.99	0.99
F1-score	**Benign**	1.00	1.00	1.00
	HTTP	0.99	0.99	0.99
	TCP	0.81	0.84	0.68
	UDP	0.99	0.99	0.99
	RTSP	1.00	0.99	0.99
Weighted Average F1-score		**1.00**	**1.00**	0.99
ROC-AUC		0.98	0.97	0.95

Table 4.8 Tranalyzer2 Multi-Classification results for Supervised Learning

Table 4.8 displays the evaluation results of multi-classification using tranalyzer2 flow features. Regarding WAF1-score, DT and KNN have a higher score of 100% compared to SVM, with a score of 99%. However, all three classifiers consistently perform in terms of precision and recall of all the classes. From Tables 4.7 and 4.8, it is evident that the decision tree has a better performance compared to the other classifiers. SVM's performance was poor in some classes in argus multi-classification, but in tranalyzer2, all three classifiers' performance is better than argus. Decision tree and k-nearest neighbors consistently perform in terms of Precision and Recall in all the classes in both cases.

4.1.4 Results of Bot-IoT Dataset Multi Classification

Supervised Learning:

	Class Names	**Decision Tree**	**K-Nearest Neighbor**	**Support Vector Machines**
Precision	**Benign**	0.48	0.60	1.00
	Theft	0.64	0.26	0.00
	OS-Service Scan	0.85	0.89	0.88
	OS-Fingerprint Scan	0.86	0.87	0.56
	Keylogging	0.57	0.31	1.00
	HTTP-DDoS	0.53	0.45	0.00
	TCP-DDoS	0.73	0.73	0.88
	UDP-DDoS	0.99	0.97	0.00
	HTTP-DoS	0.61	0.59	0.51
	TCP-DoS	0.94	0.93	0.50
	UDP-DoS	0.92	0.91	0.58
Recall	**Benign**	0.51	0.40	0.07
	Theft	0.62	0.08	0.00
	OS-Service Scan	0.50	0.45	0.19
	OS-Fingerprint Scan	0.92	0.91	0.02
	Keylogging	0.40	0.08	0.01
	HTTP-DDoS	0.18	0.17	0.00
	TCP-DDoS	0.98	0.97	0.48
	UDP-DDoS	0.89	0.87	0.00
	HTTP-DoS	0.42	0.32	0.26
	TCP-DoS	0.53	0.53	0.93
	UDP-DoS	1.00	0.98	1.00
F1-score	**Benign**	0.50	0.48	0.14
	Theft	0.63	0.12	0.00

	Class Names	Decision Tree	K-Nearest Neighbor	Support Vector Machines
	OS-Service Scan	0.63	0.60	0.31
	OS-Fingerprint Scan	0.89	0.89	0.03
	Keylogging	0.47	0.13	0.02
	HTTP-DDoS	0.27	0.25	0.00
	TCP-DDoS	0.84	0.83	0.62
	UDP-DDoS	0.94	0.92	0.00
	HTTP-DoS	0.50	0.42	0.35
	TCP-DoS	0.68	0.68	0.65
	UDP-DoS	0.96	0.95	0.73
Weighted Average F1-score		**0.88**	0.87	0.48
ROC-AUC		0.81	0.75	0.60

Table 4.9 Argus Multi-Classification results for Supervised Learning

Table 4.9 displays the results of the Bot-IoT dataset's Argus Multi-class Classification. Compared to KNN and SVM results, decision tree performance is consistent in all classes. In both decision tree and KNN classifiers, the UDP-DDoS class has the highest precision and recall score, resulting in a higher F1 score. Compared to SVM, which performs poorly in some classes, KNN's performance is consistent. Regarding WAF1-score, the decision tree classifier performs better with a score of 88%.

	Class Names	Decision Tree	K-Nearest Neighbor	Support Vector Machines
Precision	**Theft**	0.92	0.65	0.00
	OS-Service Scan	0.98	0.86	0.43
	OS-Fingerprint Scan	0.99	0.94	0.89
	Keylogging	0.96	0.85	0.02
	HTTP-DDoS	1.00	0.96	0.96
	TCP-DDoS	1.00	0.61	0.55

	Class Names	Decision Tree	K-Nearest Neighbor	Support Vector Machines
	UDP-DDoS	1.00	0.56	0.86
	HTTP-DoS	1.00	0.87	0.56
	TCP-DoS	1.00	0.65	0.71
	UDP-DoS	1.00	0.60	0.96
Recall	**Theft**	0.94	0.58	0.08
	OS-Service Scan	0.97	0.84	0.57
	OS-Fingerprint Scan	0.99	0.94	0.41
	Keylogging	0.95	0.83	0.08
	HTTP-DDoS	1.00	0.88	0.28
	TCP-DDoS	1.00	0.81	0.92
	UDP-DDoS	1.00	0.71	1.00
	HTTP-DoS	1.00	0.98	0.19
	TCP-DoS	1.00	0.42	0.23
	UDP-DoS	1.00	0.43	0.83
F1-score	**Theft**	0.93	0.62	0.01
	OS-Service Scan	0.97	0.85	0.49
	OS-Fingerprint Scan	0.99	0.94	0.57
	Keylogging	0.95	0.84	0.04
	HTTP-DDoS	1.00	0.92	0.43
	TCP-DDoS	1.00	0.70	0.69
	UDP-DDoS	1.00	0.62	0.93
	HTTP-DoS	1.00	0.92	0.29
	TCP-DoS	1.00	0.51	0.35
	UDP-DoS	1.00	0.50	0.89
Weighted Average F1-score		**1.00**	0.63	0.63
ROC-AUC		0.99	0.85	0.71

Table 4.10 Tranalyzer2 Multi-Classification results for Supervised Learning

Table 4.10 displays the results from Tranalyzer2 Multi-class classification of distinct attacks. Compared to KNN and SVM classifiers, the decision tree classifier has higher precision, recall, and F1-score. The majority of classes have a score of 100% in the decision tree classifier and the highest weighted average F1 score of 100%.

The evaluation results from Tables 4.9 and 4.10 show that the decision tree classifier outperforms k-nearest neighbors and support vector machines. Regarding flow extractor performance, tranalyzer2 exceeds argus in the decision tree classifier and SVM. Visualizing decision trees with Mutual Information Gain provides a better insight into the evaluation results.

4.1.5 CICIoT2022 Binary and Multi Tranalyzer2 results of Decision Tree

	Class Names	**Decision Tree T2 Binary**
Precision	**Benign**	1.00
	Attack	1.00
Recall	**Benign**	1.00
	Attack	1.00
F1-score	**Benign**	1.00
	Attack	1.00
Weighted Average F1-score		**1.00**
ROC-AUC		0.99

Table 4.11 Tranalyzer2 Binary-Classification results for Supervised Learning.

Tables 4.11 and 4.12 display the binary and multi-classification results of the decision tree classifier using the tranalyzer2 flow features after removing two potentially biased features, namely DstPortClass and DstPortClassN. However, the results are similar to Tables 4.2 and 4.8, which present these features during evaluation.

	Class Names	**Decision Tree T2 Multi**
Precision	**Benign**	1.00
	HTTP	0.99
	TCP	0.80
	UDP	0.99
	RTSP	1.00
Recall	**Benign**	1.00
	HTTP	0.99
	TCP	0.81
	UDP	0.99
	RTSP	1.00
F1-score	**Benign**	1.00
	HTTP	0.99
	TCP	0.81
	UDP	0.99
	RTSP	1.00
Weighted Average F1-score		**1.00**
ROC-AUC		0.98

Table 4.12 Tranalyzer2 Multi-Classification results for Supervised Learning

4.2 Visualization of Decision Trees:

The evaluation results show that Decision trees outperform the other implemented ML algorithms with a better performance. So, I have visualized the Decision Tree to understand the algorithm better, like which feature has a more significant contribution in identifying attacks with the help of Mutual Information [42]. Here are the results of the visualization.

Binary classification results of the Argus flow features - CIC-IoT 2022 dataset:

Figure 4.1 represents the Mutual information (MI) of the Binary Argus Algorithm. The features Avg_duration, Min_duration and Max_duration have the higher MI score when compared to other features. Figure 4.2 is a Tree representation of the Binary Argus classification. The Maximum depth of the tree is 82, and There are a total of 25447 leaf nodes. Given that the depth of the tree is longer, for better visualization, I have set the Maximum depth of the tree as 5. Class 0 represents the Benign class, and Class 1 represents the Attack class.

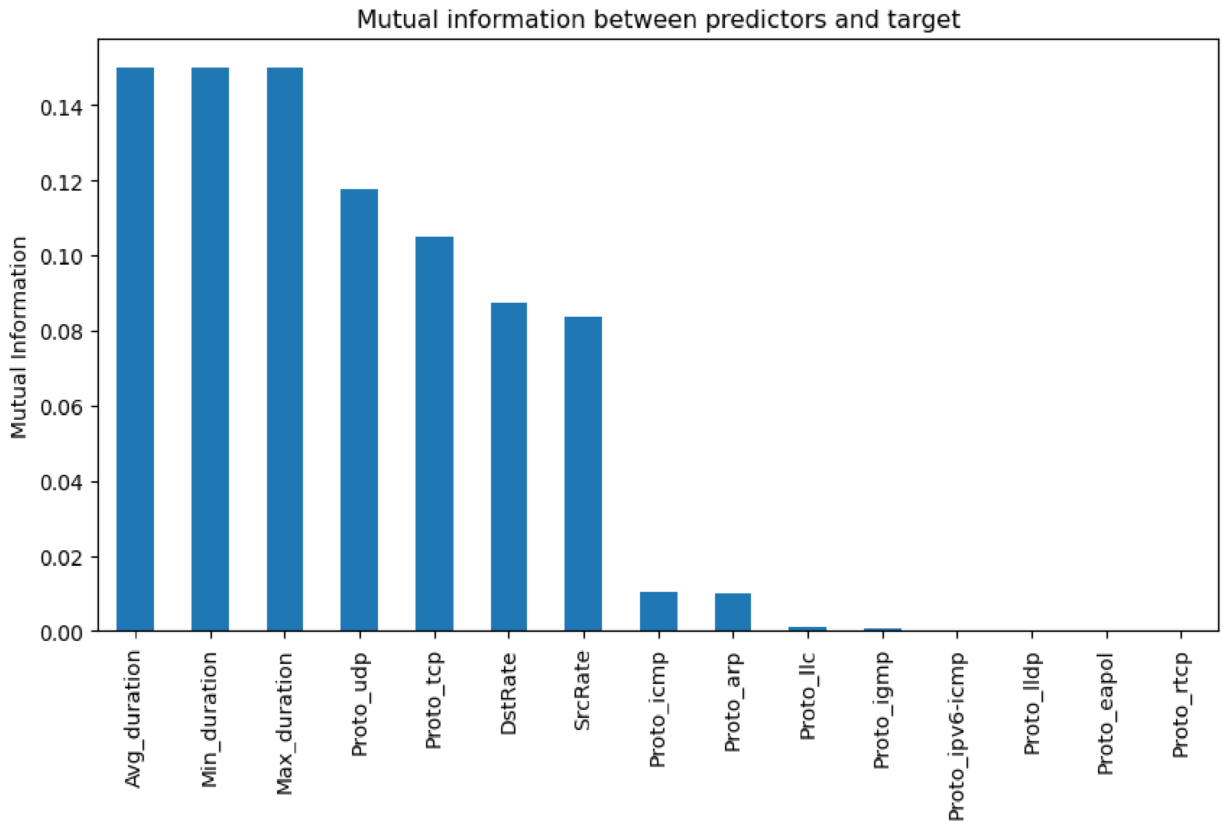

Figure4.1 Mutual Information on Binary Argus Decision Tree Classification

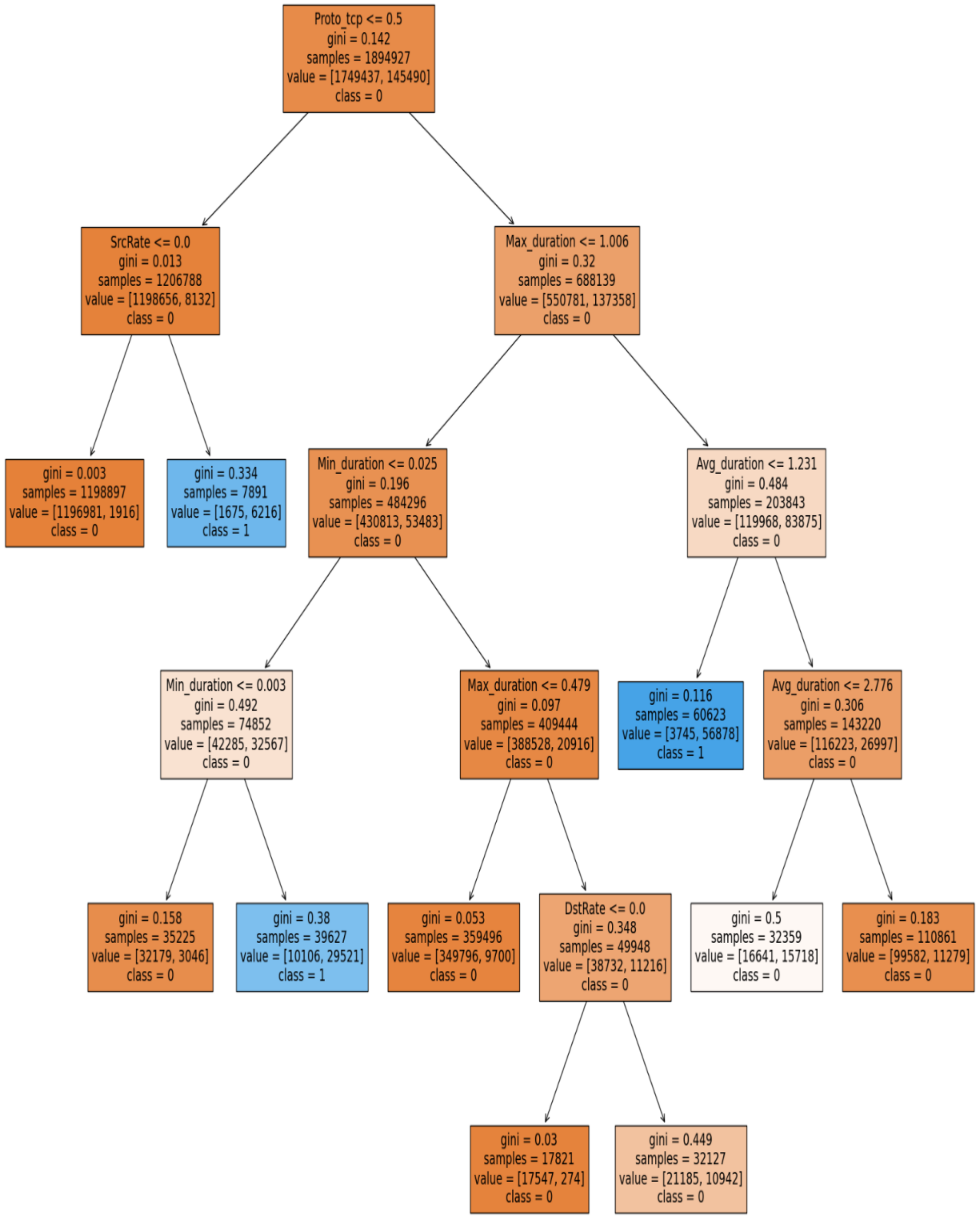

Figure 4.2 Decision Tree visualization of Argus Binary Classification

Binary classification results of the Tranalyzer2 flow features - CIC-IoT 2022 dataset: Figure 4.3 represents the MI for Decision Tree Classification for Binary Tranalyzer2. The total number of features in tranalyzer2 exceeds 50, so I have selected the top 25 features for representing Mutual Information. DstPortClass and DstPortClassN represent the traffic classification based on port names and numbers. Therefore, they have a higher MI value compared to other features. Figure 4.4 represents a decision tree visualization of Binary Tranalyzer2. This tree has a Maximum Depth of 34 with 176 leaf nodes. Class 0 and Class 1 represent the Benign and Attack classes, respectively.

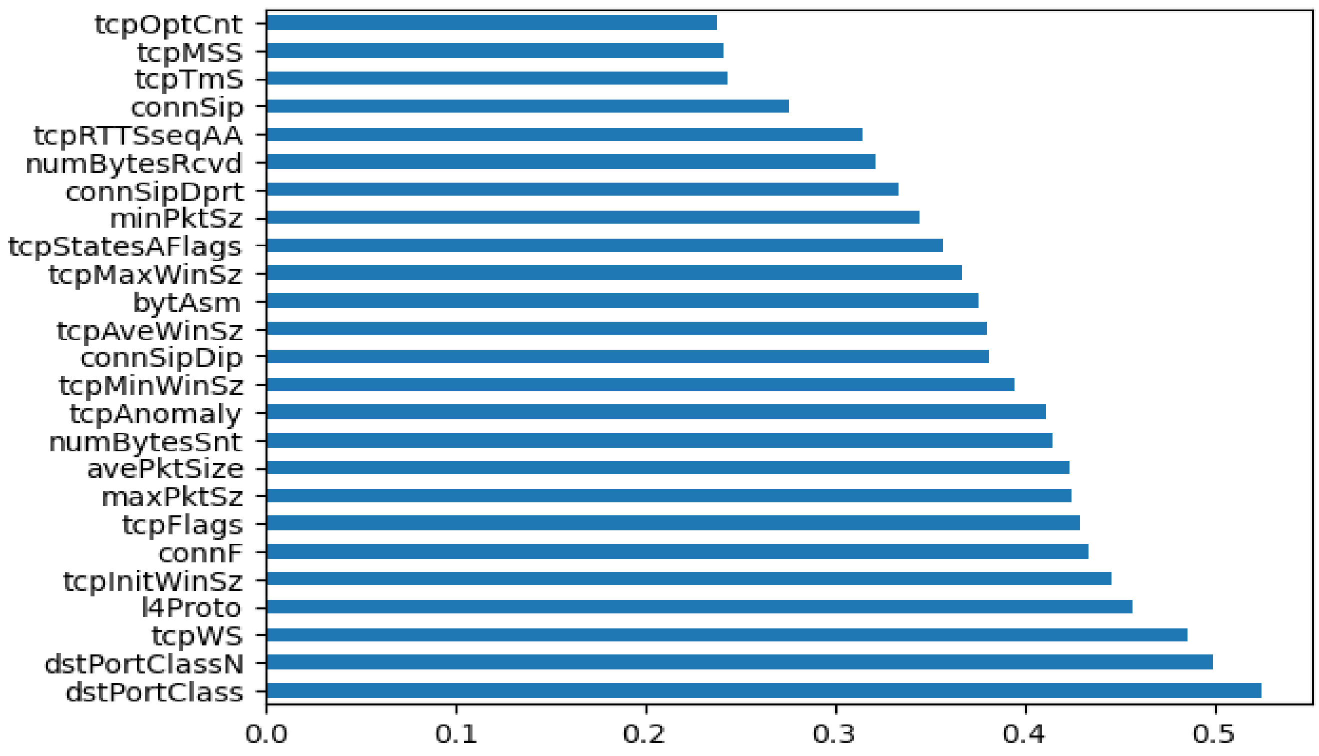

Figure 4.3 Mutual Information on Binary Tranalyzer2 Decision Tree Classification

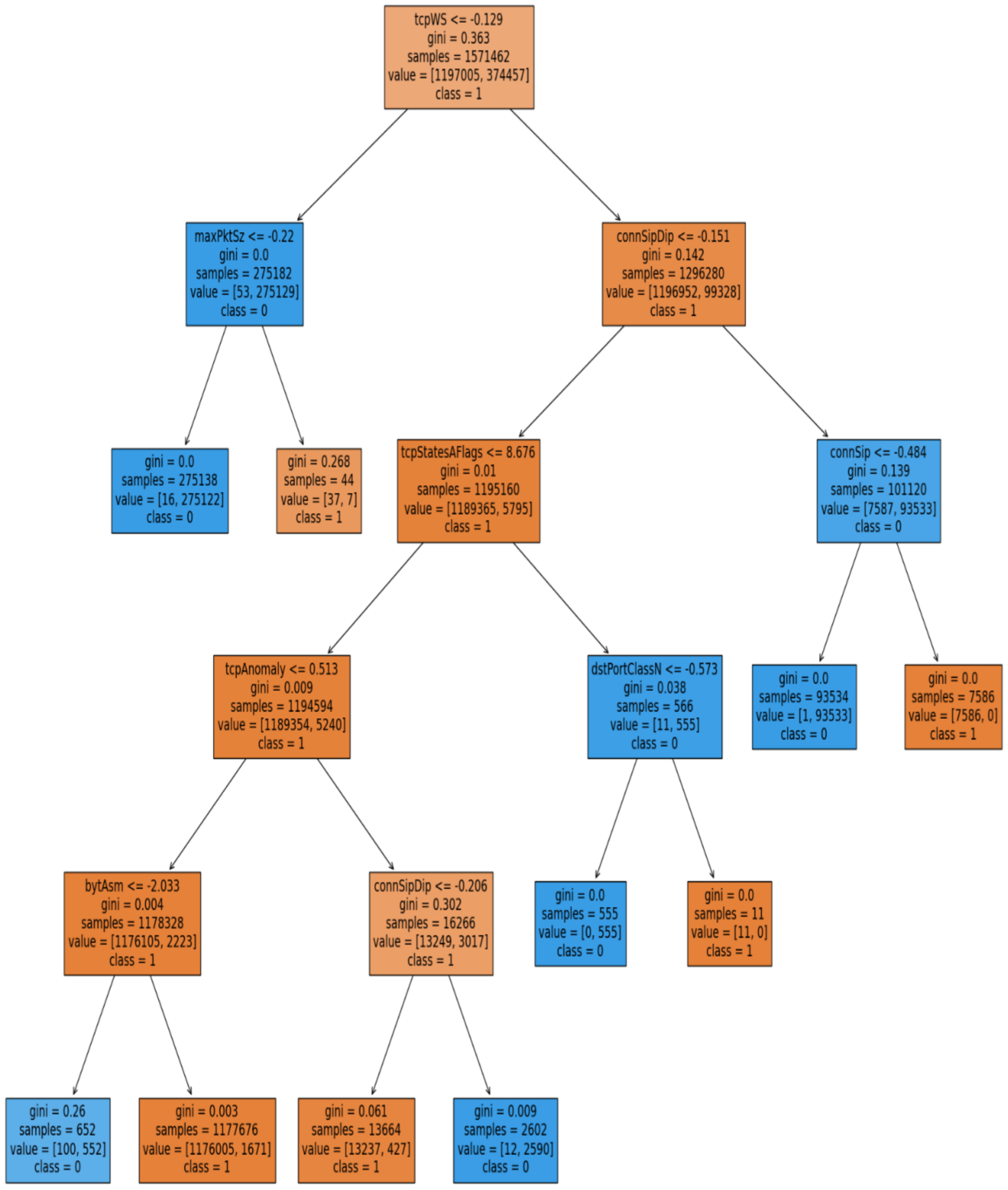

Figure 4.4 Decision Tree visualization of Tranalyzer2 Binary Classification

Binary classification results of the Argus flow features - Bot-IoT dataset:

Figure 4.5 is the MI representation of the Binary Argus classification of the Bot-IoT dataset. Again, TCP protocol has the highest Information gain, with ARP protocol having the lowest value. Figure 4.5 represents the DT visualization with a maximum depth of 35 with 307 leaf nodes, with Benign and Attack classes represented as Class 0 and Class 1.

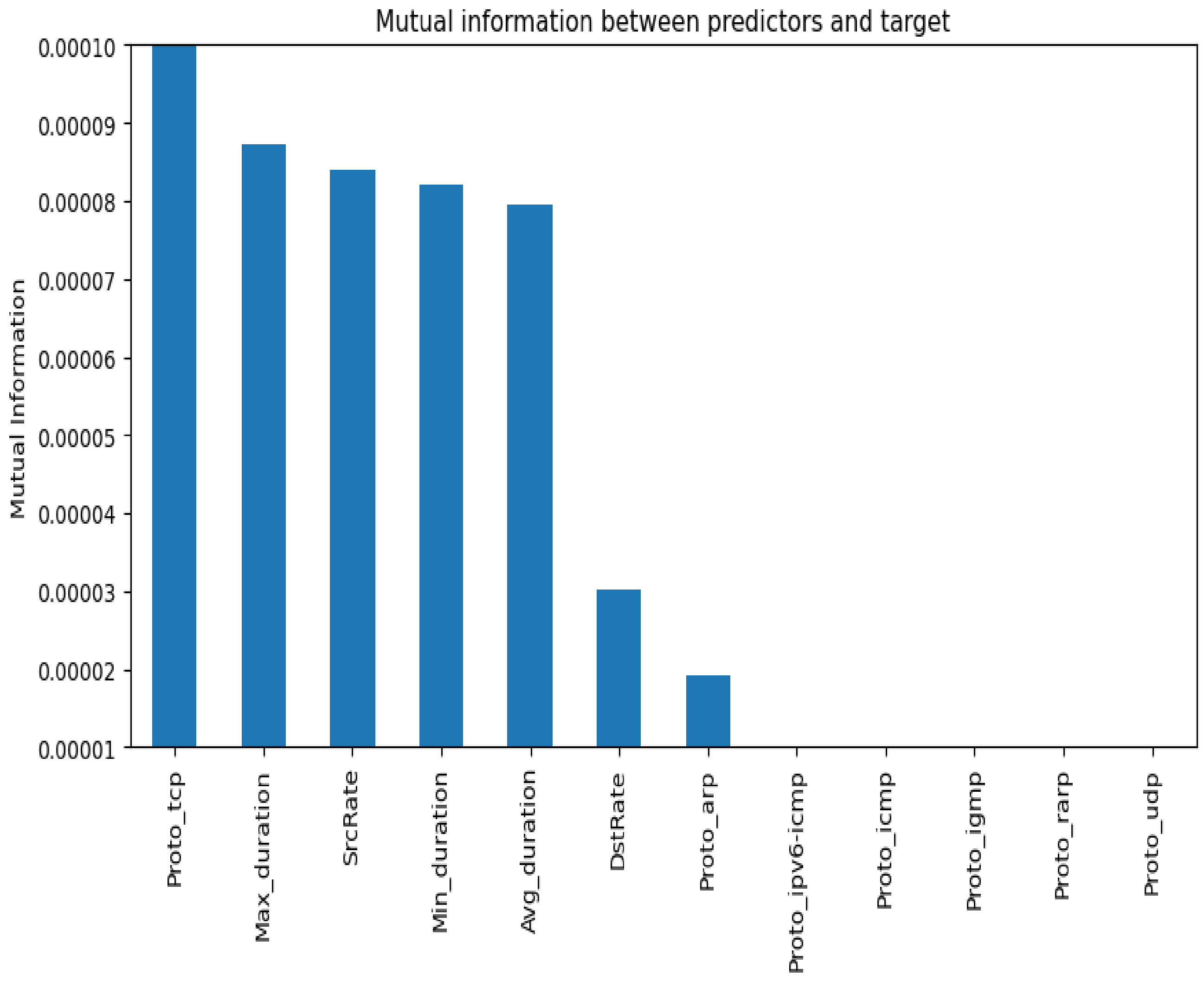

Figure 4.5 Mutual Information on Binary Argus Decision Tree Classification

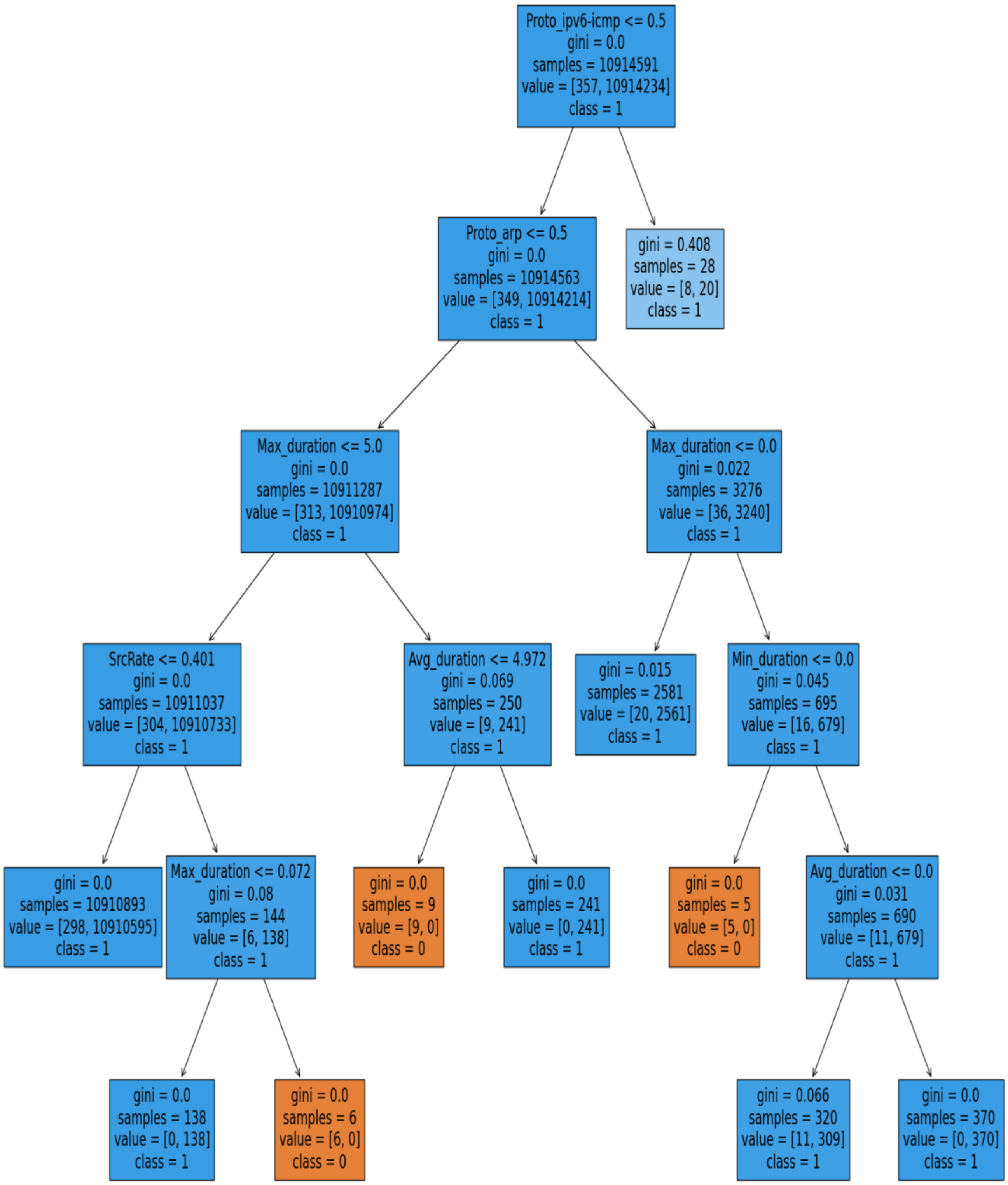

Figure 4.6 Decision Tree visualization of Argus Binary Classification

Multi-classification results of the Argus flow features - CIC-IoT dataset:

Figure 4.7 represents the mutual information on the Multi Argus Classification of the CICIoT2022 dataset. The results are similar to the Binary Argus Classification of the CIC dataset, as represented in Figure 4.1. The tree has a maximum depth of 34, with 31678 leaf nodes. Figure 4.8 represents the DT visualization where Class 0 represents the Benign class, Class 1 represents the HTTP attack and Class 3 represents the UDP Attack.

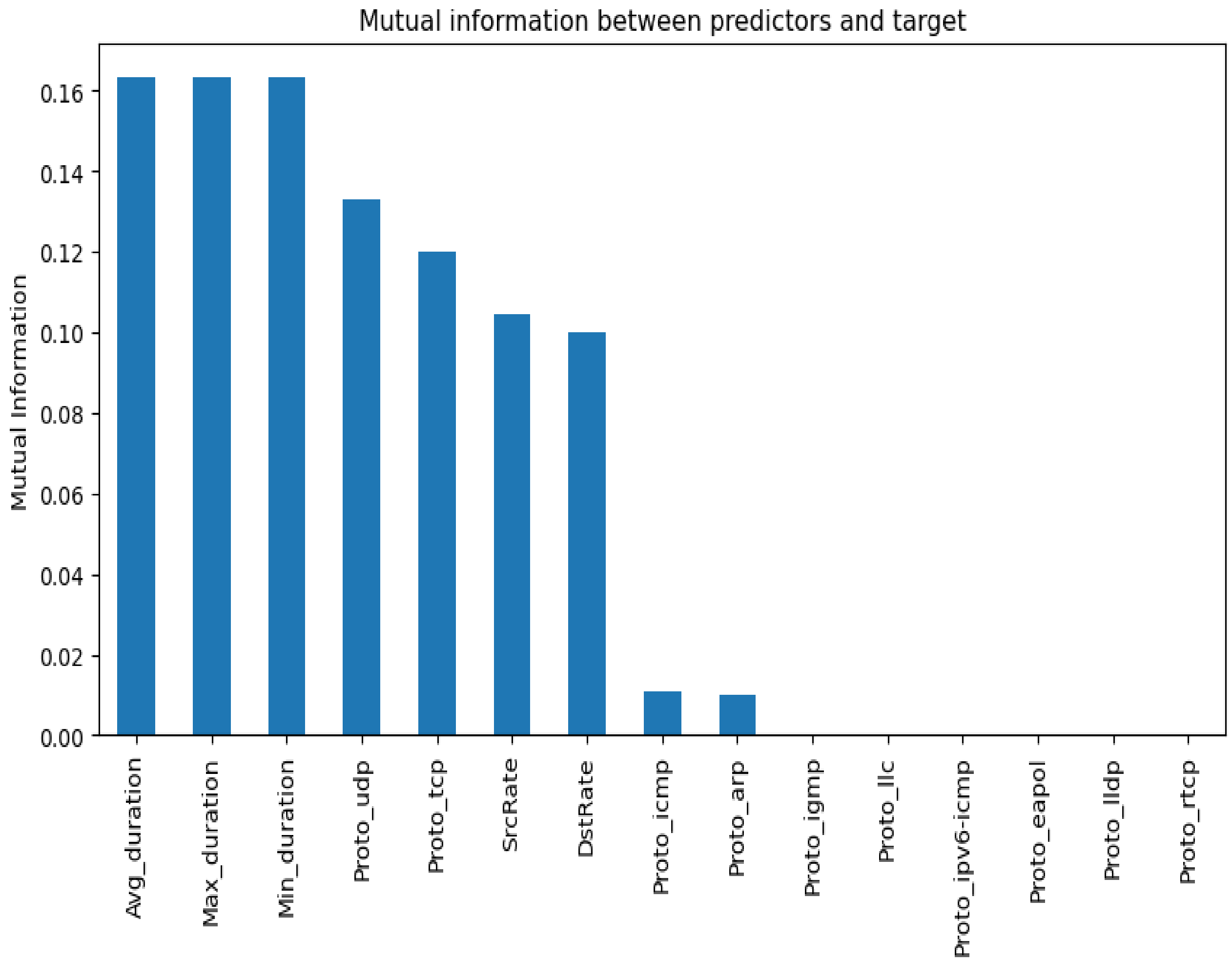

Figure 4.7 Mutual Information on Multi Argus Decision Tree Classification

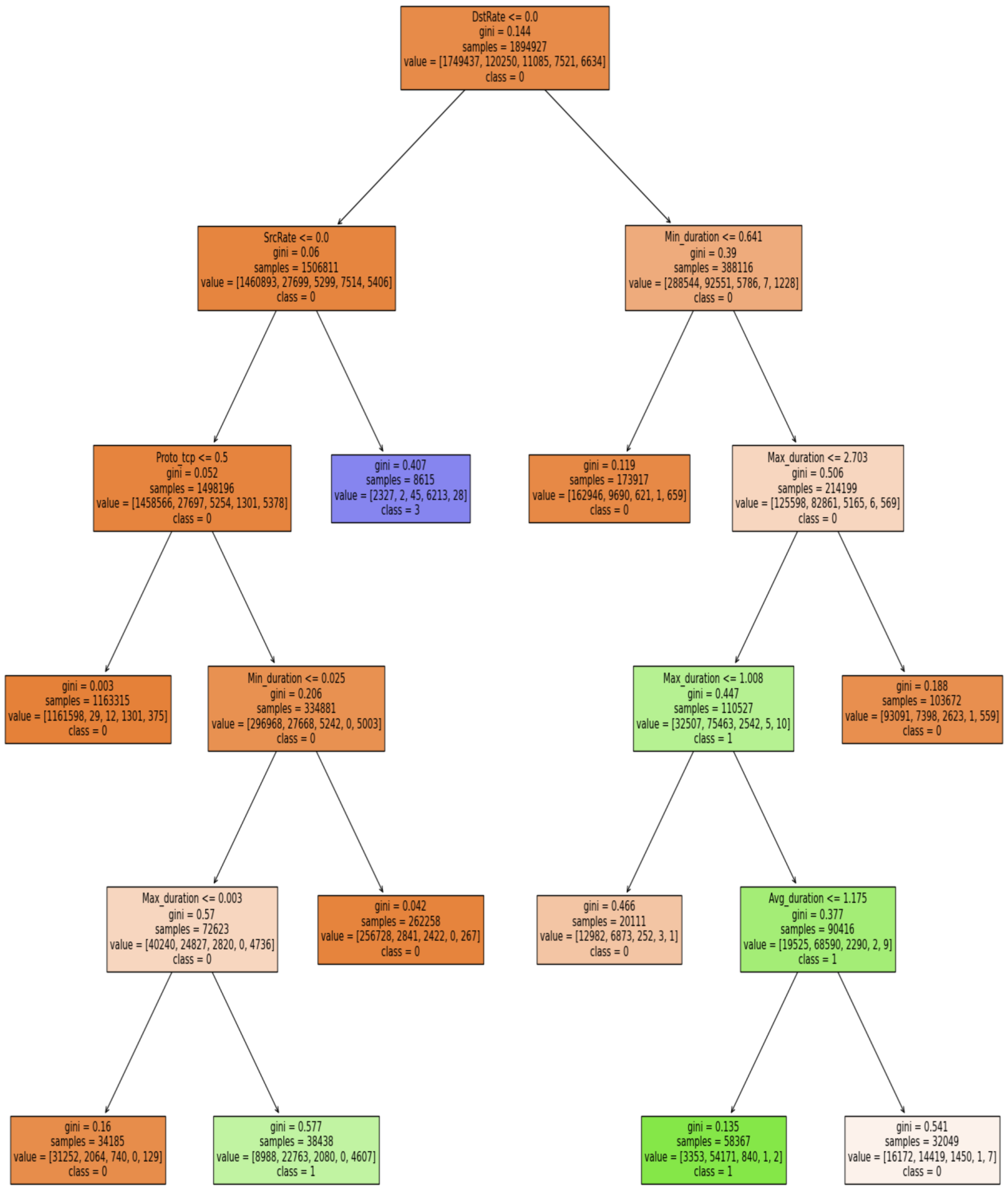

Figure 4.8 Decision Tree visualization of Argus Multi Classification

Multi-classification results of the Tranalyzer2 flow features - CIC-IoT 2022 dataset: Figure 4.9 is the MI representation of Multiclass classification on Tranalyzer2 features of the CICIoT2022 dataset. The MI results of Binary and Multi classification for Tranalyzer2 on the CIC dataset are similar to DstPortClass, having the highest value among the features. The tree has a maximum depth of 47, and the number of leaf nodes is 6172. Figure 4.10 represents the decision tree visualization where Class 0, 1, 2,3 and 4 represent the Benign class, HTTP attack, TCP attack, UDP attack, and RTSP attack, respectively.

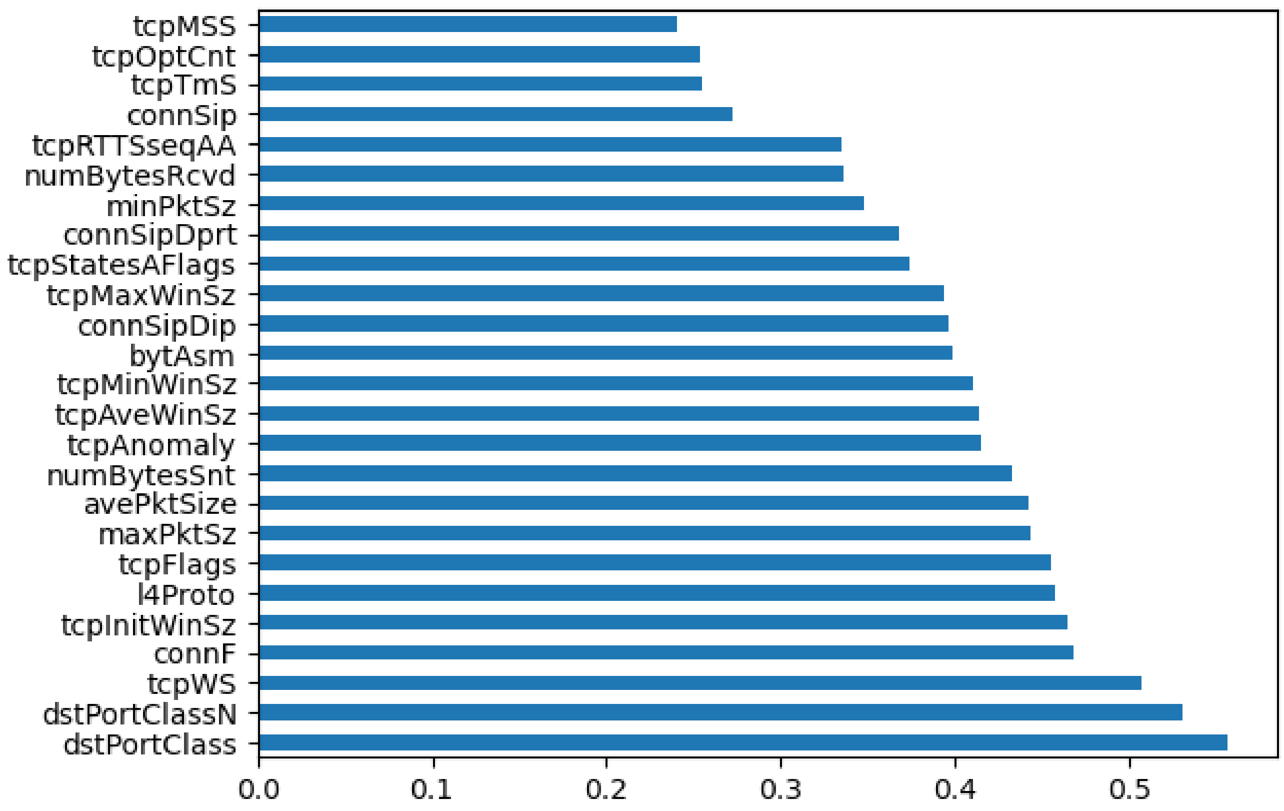

Figure 4.9 Mutual Information on Multi Tranalyzer2 Decision Tree Classification

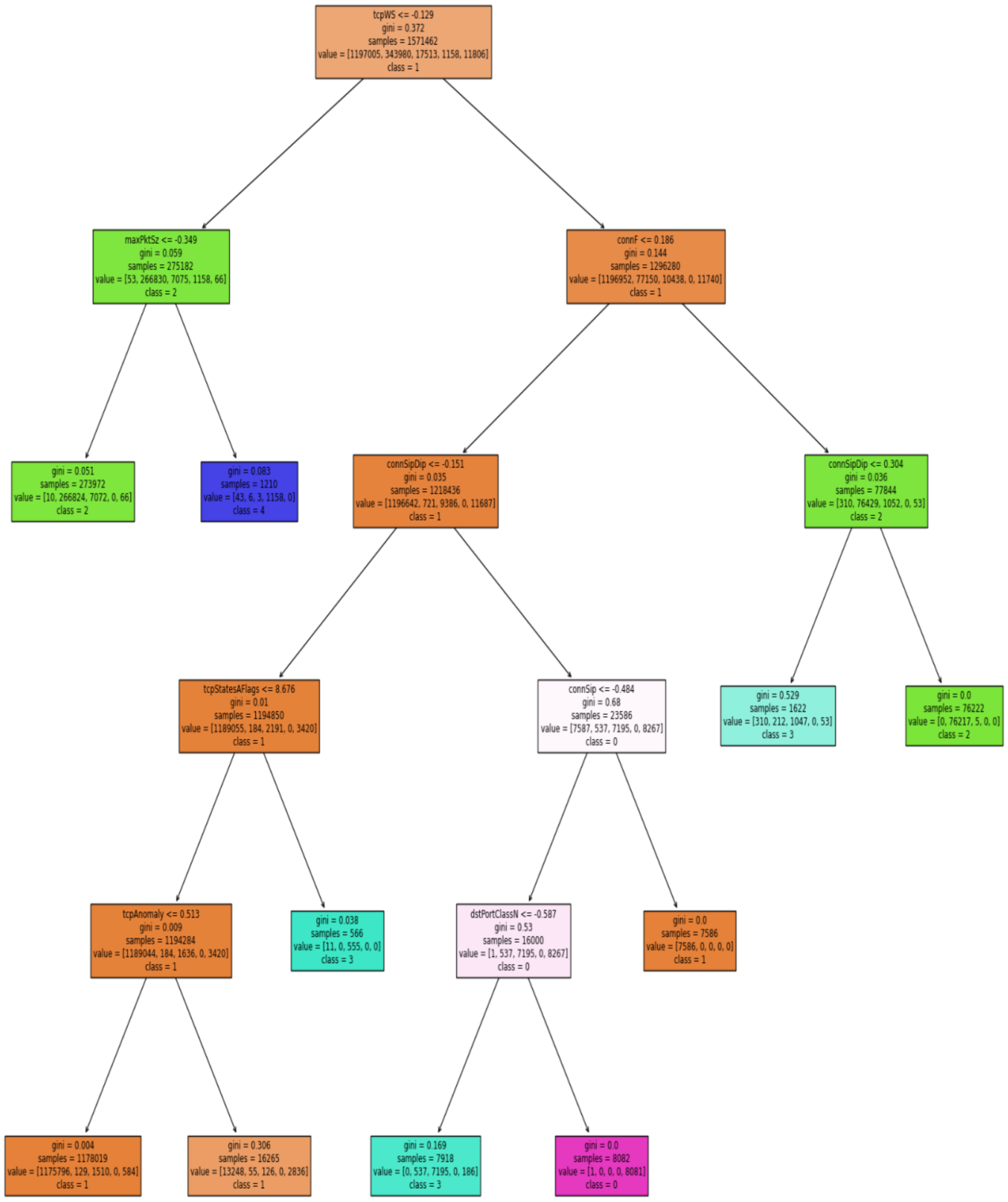

Figure4.10 Decision Tree visualization of Tranalyzer2 Multi Classification

Multi-classification results of the Argus flow features - Bot-IoT dataset:

Figure 4.11 is the MI representation of Multiclass classification on Argus features of the Bot-IoT dataset. The MI results of Binary and Multi classification for Argus on the CIC dataset are similar to having Min_duration, Avg_duration and Max_duration having the highest value among the features. The tree has a maximum depth of 77, and the number of leaf nodes is 114312. Figure 4.12 represents the decision tree visualization where Class 2, 4, 7,8 and 10 represent OS-Service Scan, Keylogging, UDP-DDoS, HTTP-DoS, and UDP-DoS attacks, respectively.

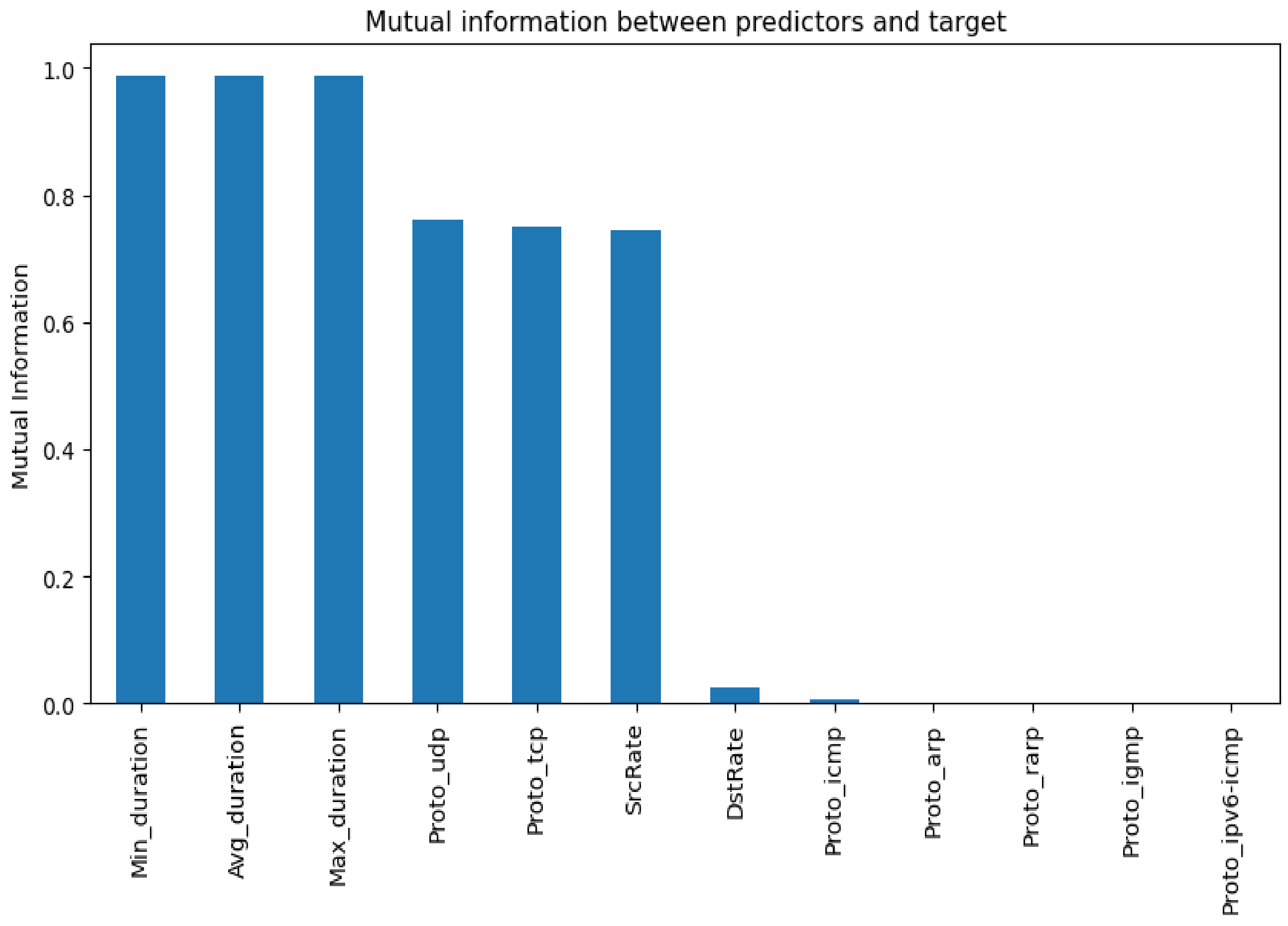

Figure 4.11 Mutual Information on Multi Argus Decision Tree Classification

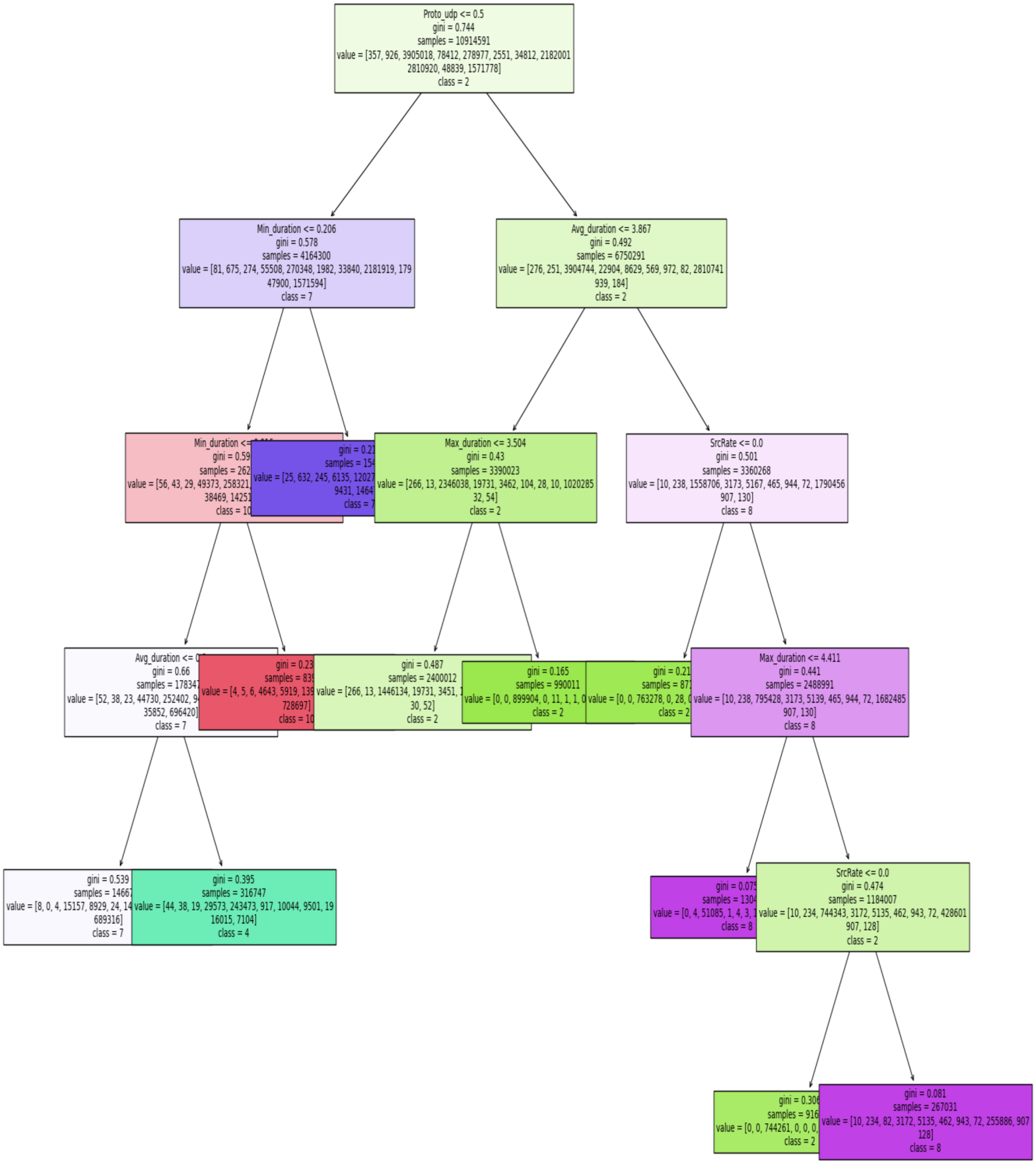

Figure 4.12 Decision tree Visualization of Multi Argus

Multi-classification results of the Tranalyzer2 flow features - Bot-IoT dataset:
Figure 4.13 is the MI representation of the Multi Tranalyzer2 classification of the Bot-IoT dataset. The SrcMac_DstMac_numP feature has the highest Information gain. Figure 4.14 represents the DT visualization. It has a maximum depth of 44 with 1439 leaf nodes. Class 2, 3, 5, 6, 7, 8, and 9 represent OS-Service Scan, OS-Fingerprint Scan, HTTP-DDoS, TCP-DDoS, UDP-DDoS, HTTP-DoS and TCP-DoS Attacks, respectively.

Mutual Information:

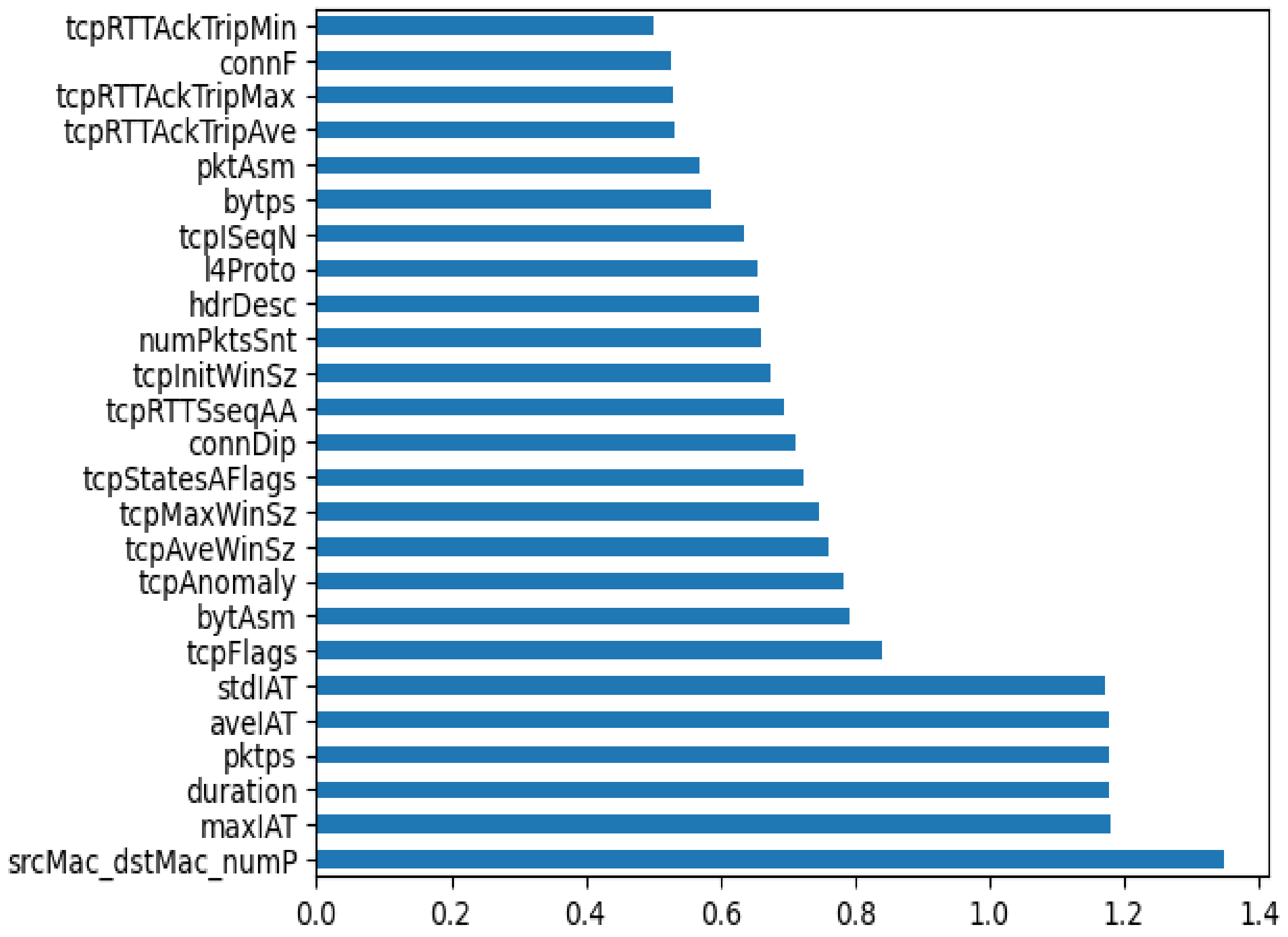

Figure 4.13 Mutual Information on Multi Tranalyzer2 Decision Tree Classification

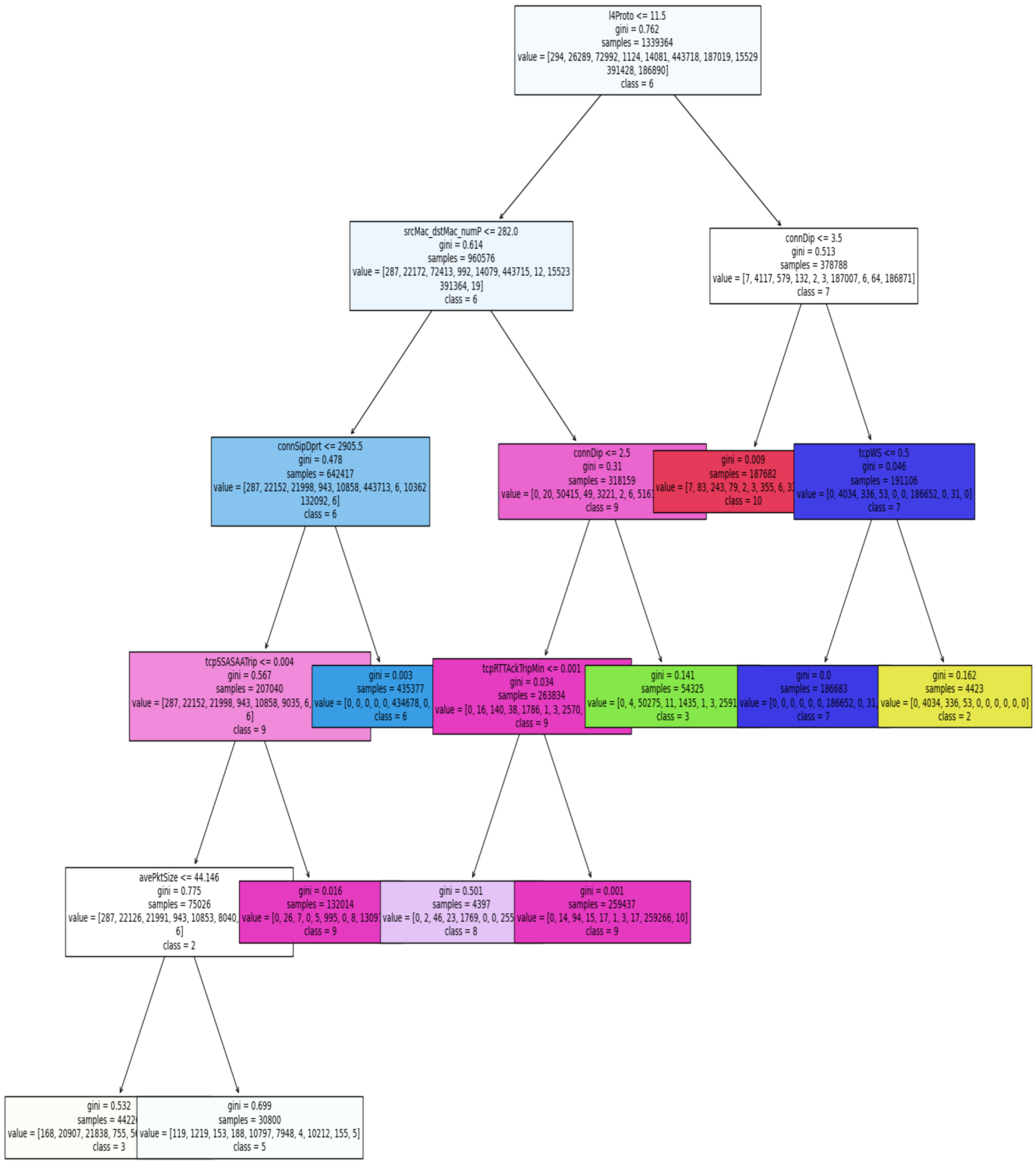
l4Proto <= 11.5
gini = 0.762
samples = 1339364
value = [294, 26289, 72992, 1124, 14081, 443718, 187019, 15529
391428, 186890]
class = 6
srcMac_dstMac_numP <= 282.0
gini = 0.614
samples = 960576
value = [287, 22172, 72413, 992, 14079, 443715, 12, 15523
391364, 19]
class = 6
connDip <= 3.5
gini = 0.513
samples = 378788
value = [7, 4117, 579, 132, 2, 3, 187007, 6, 64, 186871]
class = 7
connSipDprt <= 2905.5
gini = 0.478
samples = 642417
value = [287, 22152, 21998, 943, 10858, 443713, 6, 10362
132092, 6]
class = 6
connDip <= 2.5
gini = 0.31
samples = 318159
value = [0, 20, 50415, 49, 3221, 2, 6, 5161
class = 9
gini = 0.009
samples = 187682
value = [7, 83, 243, 79, 2, 3, 355, 6, 3
class = 10
tcpWS <= 0.5
gini = 0.046
samples = 191106
value = [0, 4034, 336, 53, 0, 0, 186652, 0, 31, 0]
class = 7
tcpSSASAATrip <= 0.004
gini = 0.567
samples = 207040
value = [287, 22152, 21998, 943, 10858, 9035, 6,
6]
class = 9
gini = 0.003
samples = 435377
value = [0, 0, 0, 0, 0, 434678, 0,
class = 6
tcpRTTAckTripMin <= 0.001
gini = 0.034
samples = 263834
value = [0, 16, 140, 38, 1786, 1, 3, 2570,
class = 9
gini = 0.141
samples = 54325
value = [0, 4, 50275, 11, 1435, 1, 3, 2591
class = 3
gini = 0.0
samples = 186683
value = [0, 0, 0, 0, 0, 0, 186652, 0, 31,
class = 7
gini = 0.162
samples = 4423
value = [0, 4034, 336, 53, 0, 0, 0, 0, 0, 0]
class = 2
avePktSize <= 44.146
gini = 0.775
samples = 75026
value = [287, 22126, 21991, 943, 10853, 8040,
6]
class = 2
gini = 0.016
samples = 132014
value = [0, 26, 7, 0, 5, 995, 0, 8, 1309
class = 9
gini = 0.501
samples = 4397
value = [0, 2, 46, 23, 1769, 0, 0, 255
class = 8
gini = 0.001
samples = 259437
value = [0, 14, 94, 15, 17, 1, 3, 17, 259266, 10]
class = 9
gini = 0.532
samples = 4422
value = [168, 20907, 21838, 755, 5
class = 3
gini = 0.699
samples = 30800
value = [119, 1219, 153, 188, 10797, 7948, 4, 10212, 155, 5]
class = 5

Figure 4.14 Decision tree Visualization of Multi Tranalyzer2

Multi-classification results of the Tranalyzer2 flow features – CIC-IoT 2022 dataset after removing the potentially biased features:

Figure 4.15 is the MI representation of Multiclass classification on Tranalyzer2 features of the CICIoT2022 dataset after removing dstPortClass and dstPortclassN. The MI results of Binary and Multi classification for Tranalyzer2 on the CIC dataset are similar (Figure 4.17) to tcpWS, having the highest value among the features. The tree has a maximum depth of 47, and the number of leaf nodes is 6172. Figure 4.16 represents the decision tree visualization where Class 0, 1, 2,3 and 4 represent the Benign class, HTTP attack, TCP attack, UDP attack, and RTSP attack, respectively.

Mutual Information:

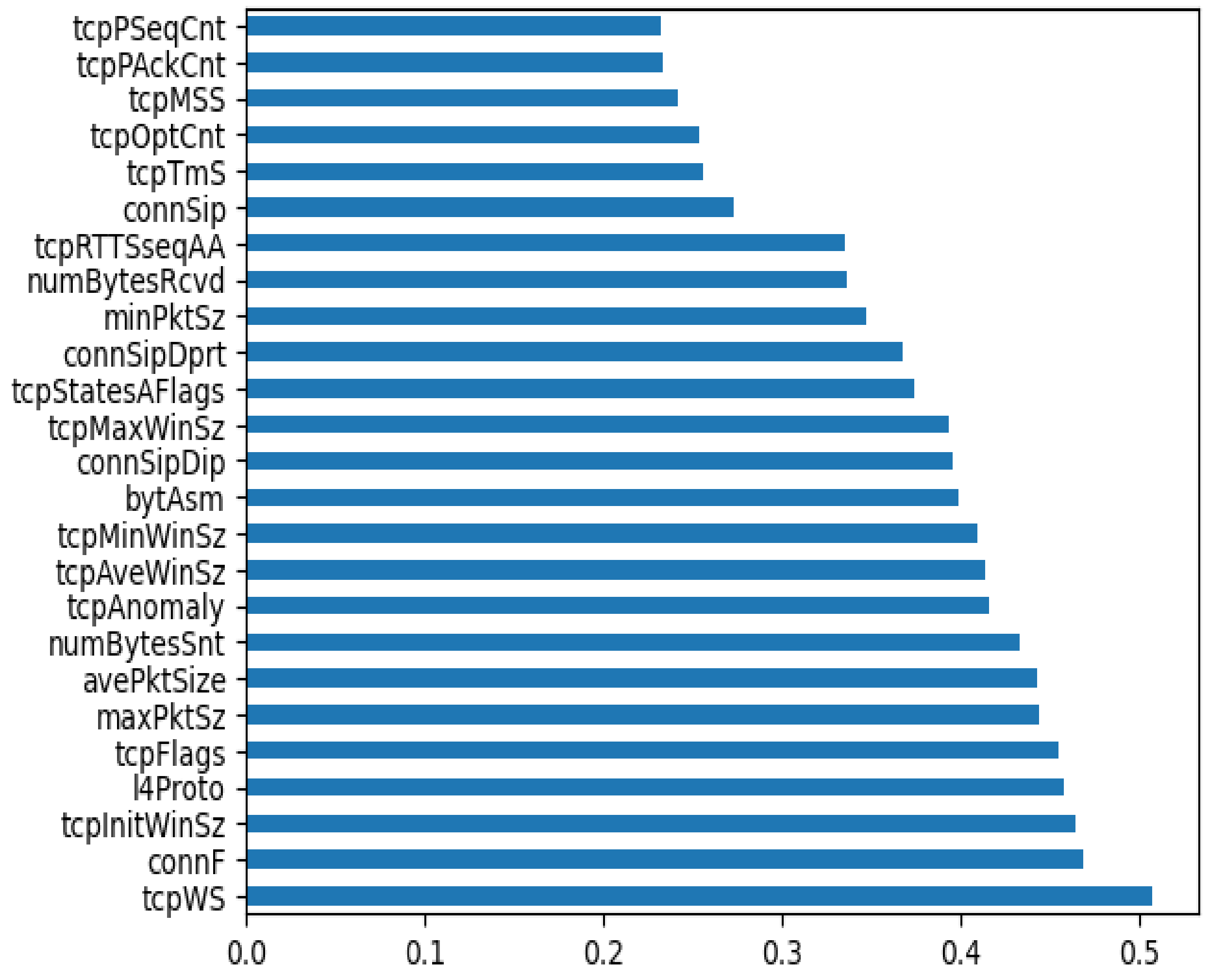

Figure 4.15 Mutual Information on Multi Tranalyzer2 Decision Tree Classification

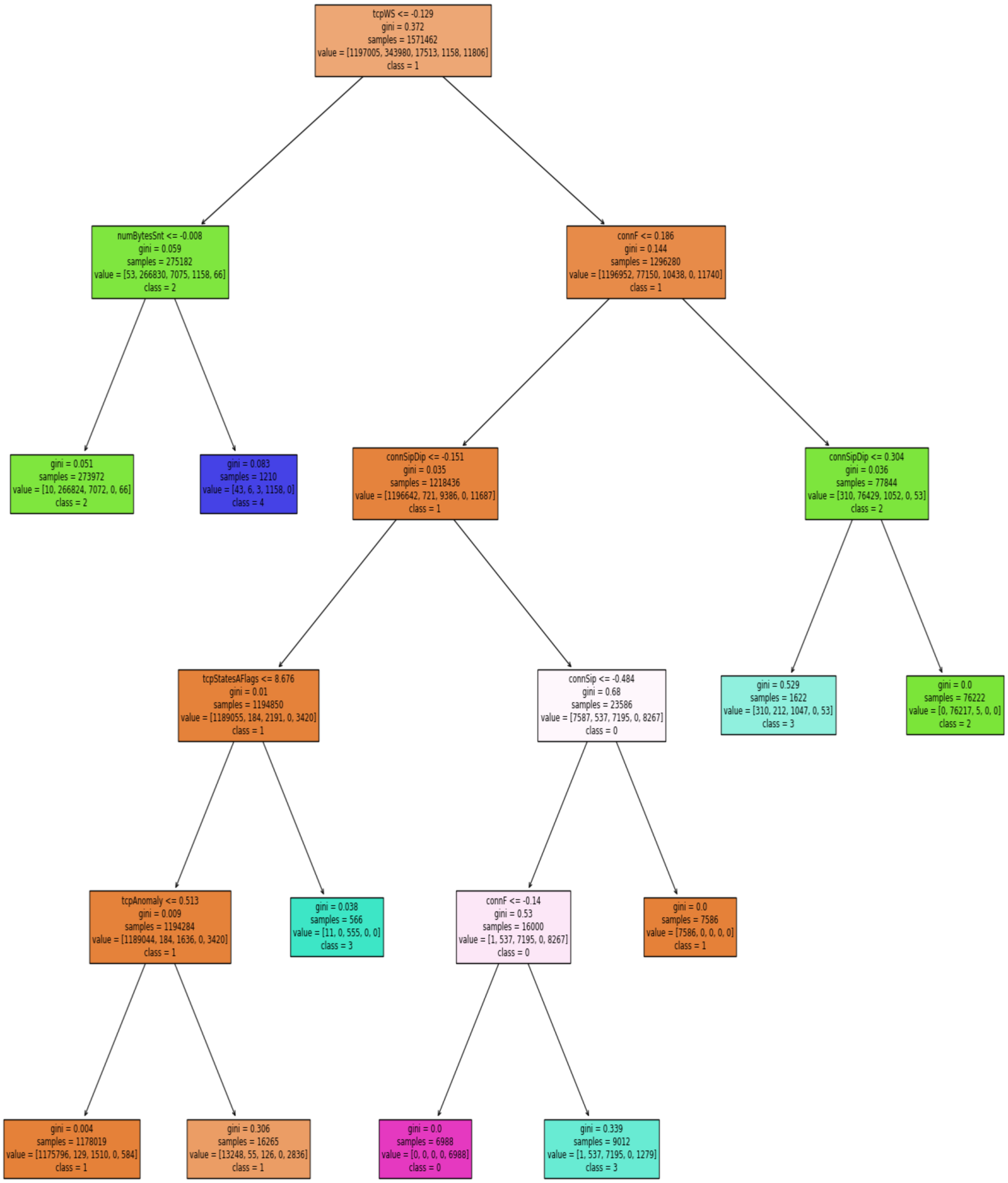

Figure 4.16 Decision tree Visualization of Multi Tranalyzer2

Binary classification results of the Tranalyzer2 flow features - CIC-IoT 2022 dataset after removing the biased features:

Figure 4.17 represents the MI for Decision Tree Classification for Binary Tranalyzer2. The total number of features in tranalyzer2 exceeds 50, so I have selected the top 25 features for representing Mutual Information. tcpWS has a higher MI value compared to other features. Figure 4.18 represents a decision tree visualization of Binary Tranalyzer2. This tree has a Maximum Depth of 34 with 176 leaf nodes. Class 0 and Class 1 represent the Benign and Attack classes, respectively.

Mutual Information:

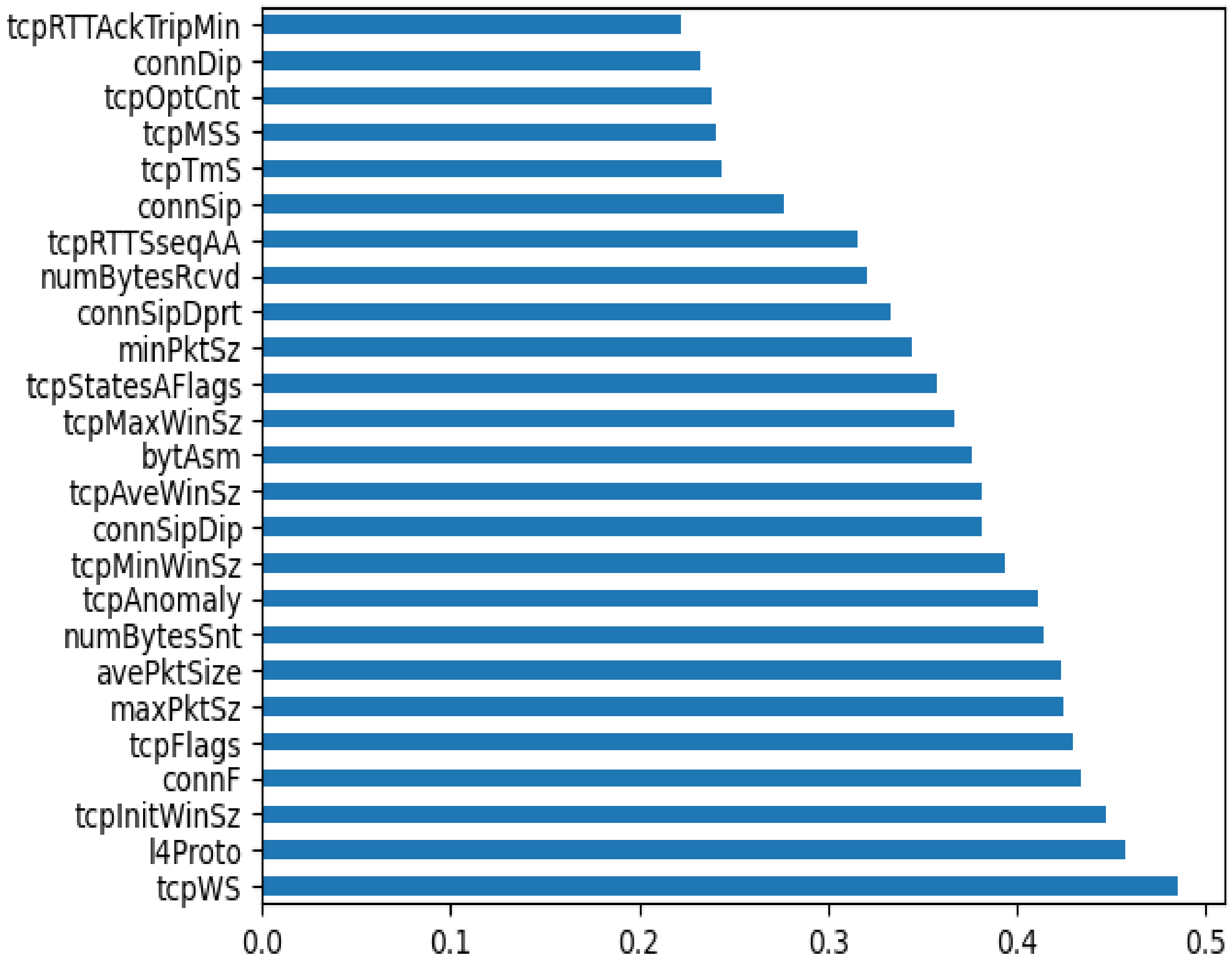

Figure 4.17 Mutual Information on Binary Tranalyzer2 Decision Tree Classification

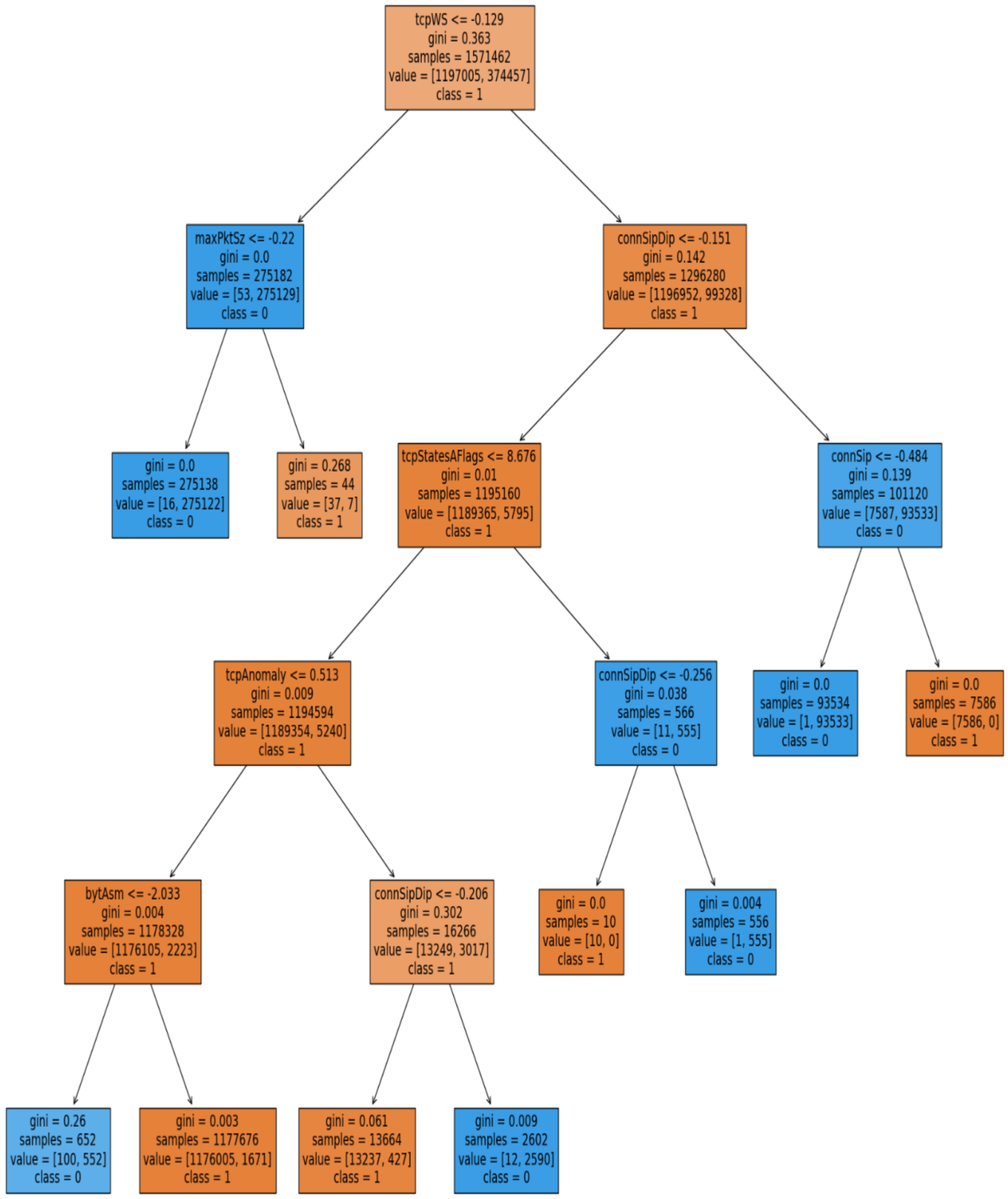

Figure 4.18 Decision tree Visualization of Binary Tranalyzer2

4.3 Results and Discussion

I used the Weighted Average F1-score metric to evaluate the results of implementation. Tables 4.1, 4.2, 4.7, and 4.8 display the Argus and Tranalyzer2 results of Supervised Learning (binary and multi-classification) of the CICIoT2022 dataset. The DT algorithm outperformed K-NN and SVM in all cases with the highest Weighted Average F1-score. Furthermore, when comparing the results of Argus and Tranalyzer2, in the binary and multi-classification of the models, Tranalyzer2 has the highest Precision and Recall scores in both benign and attack classes. However, in Argus classification results, SVM performed poorly, with a score of 0% in the attack classes. This indicates that SVM cannot identify the true positives of the attack class because SVM will perform poorly when there are more training data samples than features for each data point. In the CIC dataset, benign flows outnumber attack flows significantly (Chapter 3), contributing to the model's poor performance.

Tables 4.5, 4.9 and 4.10 display the Argus and Tranalyzer2 results of Supervised Learning (binary and multi-classification) of the Bot-Iot dataset. The results show that the DT algorithm outperforms K-NN and SVM with the highest evaluation scores. However, compared to the CIC dataset, attack flows outnumber benign flows in the Bot-IoT dataset (Chapter 3), contributing to the model's poor performance in the benign class.

Tables 4.3, 4.4 and 4.6 discuss Argus and Tranalyzer2 results of Unsupervised Learning of the CICIoT2022 and Bot-IoT datasets. The results of the algorithms vary in each category. For example, one-class SVM and Local Outlier Factor performed well in the CIC dataset, and Isolation Forest outperformed LOF and one-class SVM in the Bot-IoT dataset. However, the performance of 1-SVM is consistent with the highest score when using Argus and Tranalyzer2 flow features in the CICIoT2022 dataset.

From the results of the visualization of decision trees, in most cases, Argus features like Avg_duration, Min_duration, and Max_duration that contain information about the duration of aggregated packets have the highest Mutual Information score compared to other features. While in tranalyzer2, tcpWS (TCP Window Scale) has the highest Mutual Information score contributing to identifying attacks when implementing ML algorithms.

Here is the summary of the results based on the Weighted Average F1-score for all classifiers.

SUMMARY:

CICIoT2022 – Weighted Average F1-Score

(DT tranalyzer2 results of binary and multi-classification after excluding the dstportclass and dstportclassN from the flow features)

	Argus	**Tranalyzer2**
DT-B	**0.98**	**1.00**
KNN-B	0.95	**1.00**
SVM-B	0.89	**1.00**
DT-M	**0.98**	**1.00**
KNN-M	0.95	**1.00**
SVM-M	0.89	0.99
LOF	**0.88**	0.62
IF	0.82	0.65
1-SVM	**0.88**	**0.66**

Table 4.13 Weighted Average F1-Scores for the CICIoT2022 dataset

Table 4.13 displays the weighted average F1-scores results of binary and multi-classification of the CIC dataset. Although, in both cases (binary and multi-class), decision trees outperform the other classifiers in SL with a higher score of 98% in argus binary and multi-classification and 100% when using tranalyzer2 flow features, I-SVM and LOF outperform the Isolation Forest algorithm with a higher evaluation score of 88% and when using argus flow features 1-SVM outperforms LOF and IF with the highest score of 66% when applied tranalyzer2 flow features. Regarding flow extractor performance, all three supervised learning algorithms achieved a higher evaluation score of 100% when using tranalyzer2 flow features. Comparatively, in unsupervised learning algorithms, the argus flow feature dataset has a higher evaluation score of 88% than the tranalyzer2.

Bot-IoT – Weighted Average F1-Score

	Argus	**Tranalyzer2**
DT-B	**1.00**	-
KNN-B	**1.00**	-
SVM-B	**1.00**	-
DT-M	**0.88**	**1.00**
KNN-M	0.87	0.63
SVM-M	0.48	0.63
LOF	0.19	-
IF	**0.49**	-
1-SVM	0.48	-

Table 4.14 Weighted Average F1-Scores for the Bot-IoT dataset

Table 4.14 displays the Weighted Average F1-Scores for the Supervised and Unsupervised Learning algorithms evaluated using the Bot-IoT dataset. The source files for the dataset are available in various formats, including the original pcap files, the modified argus files, and CSV files, as mentioned in Chapter 3. The modified argus files contain benign and attack traffic data, whereas the original pcap files are divided into DoS and DDoS attack classes. This situation prevents the tranalyzer2 flow extractor from being able to extract benign features from the original pcap data. As a result, Tranalyzer2 is not used to implement binary classification of the Bot-IoT dataset. However, using multi-classification, evaluations are carried out by applying multi-classification to attack classes.

However, I used the modified argus files already available in binary format to implement the binary classification of argus. The results suggest that all three classifiers attain a higher performance score of 100% for binary classification employing argus flow characteristics. However, DT outperformed the other classifiers in both cases for multi-classification (using Argus and Tranalyzer2 flow features). Regarding Unsupervised Learning, the Isolation Forest algorithm has better results than LOF and 1-SVM.

When comparing the overall performance of the ML Algorithms, the Decision Tree Algorithm outperformed the K-Nearest Neighbors and Support Vector Machines Algorithms and Unsupervised Learning algorithms.

The next chapter discusses the overall summary of this thesis and future works.

CONCLUSION AND FUTURE WORK

In my thesis, I used the approach of implementing Machine Learning Algorithms to detect IoT device attacks. Finding publicly available datasets on IoT networks is the first step. For this purpose, I used CIC-IoT2022 and Bot-IoT datasets. The next major step is to extract the flow features for implementation from the datasets. Recent research has suggested that Argus and Tranalyzer2 be used to extract flow features as both are recently developed tools with extensive features for a straightforward interpretation of flow data. Further, implementations are carried out using Supervised and Unsupervised Machine Learning algorithms on the datasets. Decision tree, KNN, SVM, LOF, Isolation Forest, and One-class SVM algorithms were used for implementation. My proposed framework using these datasets with distinct flow characteristics will serve as a benchmark for future research. From this lens, my new contributions are: (i) Employing supervised and unsupervised machine learning algorithms with unique features for attack detection; (ii) Using two distinct feature sets extracted from Argus and Tranalyzer2 to evaluate the machine learning model; and (iii) Benchmarking the Argus and Tranalyzer2 features on the two publicly available datasets, namely CIC-IoT 2022 dataset and Bot-IoT datasets.

The implementation results show that Decision Trees outperforms the KNN and SVM algorithms in all cases, with the Decision Trees' average F1 score being 100% for binary classification and multi-classification using Tranalyzer2 on both datasets. However, the performance of One-Class SVM, Isolation Forest and the Local Outlier Factor algorithms varied on the datasets used in this thesis and did not reach as high scores as the supervised learning algorithms. Having a lower performance with unsupervised learning algorithms is expected given that label information is not used while training these algorithms. Overall results show that Decision Tree algorithms perform better than the others on all evaluations including all datasets, binary, multi-class, Argus and Tranalyzer2 feature sets in this research.

It should be noted there that all the algorithms were implemented off-the-shelf, I.e., using default hyperparameter values provided in the machine learning libraries, in this thesis. But it is known that hyper-parameter optimization could play a significant role in improving performance. This might be one of the reasons why unsupervised learning algorithms did not have a consistent performance on different evaluations. Future research will involve hyperparameter tuning to examine the impact on the different machine learning models' performance and studying further the complexity of the Decision Trees.

From the results of implementation, Supervised learning algorithms had higher evaluation scores using T2 features, whereas the Unsupervised learning algorithms performed better with Argus features.

Future research will involve investigating the results of the performance while integrating the features of Argus and Tranalyzer2 flow exporters and studying the generalizability of the solutions further. Moreover, to improve the proposed framework's performance, further research would be beneficial in terms of different feature sets and other learning algorithms such as Deep Neural Network, ensemble learning algorithms and studying explainable AI models. Finally, the framework will be tested on other publicly available datasets, including further attack and benign data.

APPENDIX A

Argus Flow description

Feature #	**Feature Names**	**Feature Description**
1	Proto	Transaction protocols present in network flow
2	SrcAddr	source IP address
3	Sport	source port number
4	DstAddr	destination IP address
5	Dport	destination port number
6	Seq	argus sequence number
7	StdDev	standard deviation of aggregated records
8	Min	minimum duration of aggregated records
9	Mean	average duration of aggregated records
10	DstRate	destination to source packets per second
11	SrcRate	source to destination packets per second
12	Max	maximum duration of aggregated records

Tranalyzer2 Flow description

Feature #	**Feature Names**	**Feature Description**
1	dir	Flow direction
2	flowInd	Flow index
3	flowStat	Flow status and warnings
4	timeFirst	Date time of first packet
5	timeLast	Date time of last packet
6	duration	Flow duration
7	numHdrDesc	Number of different headers descriptions
8	numHdrs	Number of headers (depth) in hdrDesc
9	hdrDesc	Headers description
10	srcMac	Mac source
11	dstMac	Mac destination
12	ethType	Ethernet type
13	ethVlanID	VLAN IDs

Feature #	Feature Names	Feature Description
14	srcIP	Source IP
15	srcIPCC	Source IP country
16	srcIPOrg	Source IP organisation
17	srcPort	Source port
18	dstIP	Destination IP address
19	dstIPCC	Destination IP country
20	dstIPOrg	Destination IP organization
21	dstPort	Destination port
22	l4Proto	Layer 4 protocol
23	macStat	macRecorder status
24	macPairs	Number of distinct source/destination MAC addresses pairs
25	srcMac_dstMac_numP	Source/destination MAC address, number of packets of MAC address combination
26	srcMacLbl_dstMacLbl	Source/destination MAC label
27	dstPortClassN	Port based classification of the destination port number
28	dstPortClass	Port based classification of the destination port name
29	numPktsSnt	Number of transmitted packets
30	numPktsRcvd	Number of received packets
31	numBytesSnt	Number of transmitted bytes
32	numBytesRcvd	Number of received bytes
33	minPktSz	Minimum layer 3 packet size
34	maxPktSz	Maximum layer 3 packet size
35	avePktSize	Average layer 3 packet size
36	stdPktSize	Standard deviation layer 3 packet size
37	minIAT	Minimum IAT
38	maxIAT	Maximum IAT
39	aveIAT	Average IAT
40	stdIAT	Standard deviation IAT
41	pktps	Sent packets per second

Feature #	Feature Names	Feature Description
42	bytps	Sent bytes per second
43	pktAsm	Packet stream asymmetry
44	bytAsm	Byte stream asymmetry
45	tcpFStat	tcpFlags status
46	ipMindIPID	IP minimum delta IP ID
47	ipMaxdIPID	IP maximum delta IP ID
48	ipMinTTL	IP minimum TTL
49	ipMaxTTL	IP maximum TTL
50	ipTTLChg	IP TTL change count
51	ipToS	IP Type of Service hex
52	ipFlags	IP aggregated flags
53	ipOptCnt	IP options count
54	ipOptCpCl_Num	IP aggregated options, copy-class and number
55	ip6OptCntHH_D	IPv6 Hop-by-Hop destination option counts
56	ip6OptHH_D	IPv6 aggregated Hop-by-Hop destination options
57	tcpISeqN	TCP initial sequence number
58	tcpPSeqCnt	TCP packet seq count
59	tcpSeqSntBytes	TCP sent seq diff bytes
60	tcpSeqFaultCnt	TCP sequence number fault count
61	tcpPAckCnt	TCP packet ACK count
62	tcpFlwLssAckRcvdBytes	TCP flawless ACK received bytes
63	tcpAckFaultCnt	TCP ACK number fault count
64	tcpBFlgtMx	TCP Bytes in Flight MAX
65	tcpInitWinSz	TCP initial effective window size
66	tcpAveWinSz	TCP average effective window size
67	tcpMinWinSz	TCP minimum effective window size
68	tcpMaxWinSz	TCP maximum effective window size
69	tcpWinSzDwnCnt	TCP effective window size change down count
70	tcpWinSzUpCnt	TCP effective window size change up count
71	tcpWinSzChgDirCnt	TCP effective window size direction change count

Feature #	Feature Names	Feature Description
72	tcpWinSzThRt	TCP packet count ratio below window size WINMIN threshold
73	tcpFlags	TCP aggregated protocol flags (FINACK, SYNACK, RSTACK, CWR, ECE, URG, ACK, PSH, RST, SYN, FIN)
74	tcpAnomaly	TCP aggregated header anomaly flags
75	tcpOptPktCnt	TCP options packet count
76	tcpOptCnt	TCP options count
77	tcpOptions	TCP aggregated options
78	tcpMSS	TCP maximum segment size
79	tcpWS	TCP window scale
80	tcpMPTBF	TCP MPTCP type bitfield
81	tcpMPF	TCP MPTCP flags
82	tcpMPAID	TCP MPTCP address ID
83	tcpMPdssF	TCP MPTCP DSS flags
84	tcpTmS	TCP time stamp
85	tcpTmER	TCP time echo reply
86	tcpEcI	TCP estimated counter increment
87	tcpUtm	TCP estimated up time
88	tcpBtm	TCP estimated boot time
89	tcpSSASAATrip	TCP trip time (A: SYN, SYN-ACK, B: SYN-ACK, ACK)
90	tcpRTTAckTripMin	TCP ACK trip min
91	tcpRTTAckTripMax	TCP ACK trip max
92	tcpRTTAckTripAve	TCP ACK trip average
93	tcpRTTAckTripJitAve	TCP ACK trip jitter average
94	tcpRTTSseqAA	TCP round trip time (A: SYN, SYN-ACK, ACK, B: ACK-ACK)
95	tcpRTTAckJitAve	TCP ACK round trip average jitter
96	tcpStatesAFlags	TCP state machine anomalies

Feature #	Feature Names	Feature Description
97	icmpStat	ICMP Status
98	icmpTCcnt	ICMP type code count
99	icmpBFTypH_TypL_Code	ICMP Aggregated type H (>128), L (<32) & code bit field
100	icmpTmGtw	ICMP time/gateway
101	icmpEchoSuccRatio	ICMP Echo reply/request success ratio
102	icmpPFindex	ICMP parent flowIndex
103	connSip	Number of unique source IPs
104	connDip	Number of unique destination IPs
105	connSipDip	Number of connections between source and destination IP
106	connSipDprt	Number of connections between source IP and destination port
107	connF	The f-number: connSipDprt / connSip [EXPERIMENTAL]

BIBLIOGRAPHY

[1] M. Anwer, S. Mahmood Khan, M. Umer Farooq and N. Waseemullah. Attack Detection in IoT using Machine Learning. *Engineering, Technology and Applied Science Research Journal, Vol.11, No.3,* June 2021*,* pages 7273-7278.

[2] M. Zeeshan, Q. Riaz, M. A. Bilal, M. K. Shahzad, H. Jabeen, S. A. Hyder and A. Rahim. Protocol-Based Deep Intrusion Detection for DoS and DDoS Attacks Using UNSW-NB15 and Bot-IoT Datasets. *IEEE Access*, vol. 10. 2022, pages 2269-2283.

[3] S. Dwibedi, M. Pujari and W. Sun. A Comparative Study on Contemporary Intrusion Detection Datasets for Machine Learning Research, *IEEE International Conference on Intelligence and Security Informatics (ISI)*, Arlington, VA, USA, 2020, pages 1-6.

[4] A. Das, S.A. Ajila and C. H. Lung. A Comprehensive Analysis of Accuracies of Machine Learning Algorithms for Network Intrusion Detection. *International Conference on Machine Learning for Networking,* 2019, pages 40-57.

[5] J. L. Leevy, J. Hancock, T. M. Khoshgoftaar and J. M. Peterson. An Easy-to-Classify Approach for the Bot-IoT Dataset. *2021 IEEE Third International Conference on Cognitive Machine Intelligence (CogMI)*, Atlanta, GA, USA, 2021, pages 172-179.

[6] S. Krishnan, A. Neyaz and Q.Liu. IoT Network Attack Detection using Supervised Machine Learning. *International Journal of Artificial Intelligence and Expert Systems, 10(2),* 2021, pages: 18-32.

[7] M. S. Ahmad and S. H. Shah. Supervised Machine Learning approaches for Attack Detection in the IoT Network. Keshav Dahal, Debasis Giri, Dr. Sarmistha Neogy, Dr. Subrata Dutta and Sanjay Kumar, editors, *Internet of Things and Its Applications. Lecture Notes in Electrical Engineering,* vol 825. Springer, Singapore, 2022.

[8] D. Rani and N. C. Kaushal. Supervised Machine Learning Based Network Intrusion Detection System for Internet of Things. *11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, Kharagpur, India, 2020, pages 1-7.

[9] M. Ahmad, Q. Riaz, M. Zeeshan and H. Tahir. Intrusion Detection in Internet of Things using Supervised Machine Learning based on Application and Transport Layer features using UNSW-NB15 dataset. *J Wireless Com Network,* 2021.

[10] Y. K. Saheed, A. I. Abiodun, S. Misra, M. K. Holone, R. C. Palacios. A Machine Learning-based Intrusion Detection for detecting Internet of Things network attacks. *Alexandria Engineering Journal,* vol 61, Issue 12, 2022, pages 9395-9409.

[11] H. Tyagi and R. Kumar. Attack and Anomaly detection in IoT networks using Supervised Machine Learning approaches. *Revue d'Intelligence Artificielle,* vol. 35, No. 1, 2021, pages 11-21.

[12] F. Alasmary, S. Alraddadi, S. Al-Ahmadi and J. Al-Muhtadi. ShieldRNN: A Distributed Flow-Based DDoS Detection Solution for IoT Using Sequence Majority Voting. *IEEE Access*, vol. 10, 2022, pages 88263-88275.

[13] Denial of Service Attack. https://en.wikipedia.org/wiki/Denial-of-service_attack, 2023, Accessed: Jan-2023.

[14] Distributed Denial of Service Attack. https://www.geeksforgeeks.org/what-is-ddosdistributed-denial-of-service/, 2023, Accessed: Jan-2023.

[15] Tshark Tutorial. https://hackertarget.com/tshark-tutorial-and-filter-examples/, 2023, Accessed: Jan-2023.

[16] Bot-IoT Dataset. https://research.unsw.edu.au/projects/bot-iot-dataset, 2022, Accessed: July-2022.

[17] CIC IoT Dataset 2022. https://www.unb.ca/cic/datasets/iotdataset-2022.html, 2022, Accessed: May-2022.

[18] Ostinato Tool. https://ostinato.org/, 2022, Accessed: July-2022.

[19] Node-RED tool. https://nodered.org/, 2022, Accessed: July-2022.

[20] Argus. https://openargus.org/, 2022, Accessed: June-2022.

[21] Tranalyzer2. https://tranalyzer.com/doc/gettingstarted, 2022, Accessed: August-2022.

[22] Mirai botnet attack. https://en.wikipedia.org/wiki/Mirai_(malware), 2023, Accessed: March-2023.

[23] Wireshark. https://www.wireshark.org/docs/wsug_html_chunked/, 2022, Accessed: May-2022.

[24] Machine Learning. https://www.goodreads.com/book/show/32505087-machine-learning, 2023, Accessed March-2023.

[25] Machine Learning Algorithms. https://www.geeksforgeeks.org/machine-learning/, 2023, Accessed: Feb-2023.

[26] Decision Tree Algorithm. https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm, 2022, Accessed: Sep-2022.

[27] K-Nearest Neighbors Algorithm. https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761, 2022, Accessed: Sep-2022.

[28] Support Vector Machines Algorithm. https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/, 2022, Accessed: Sep-2022.

[29] Local Outlier Factor Algorithm. https://medium.com/@pramodch/understanding-lof-local-outlier-factor-for-implementation-1f6d4ff13ab9, 2022, Accessed: Nov-2022.

[30] One-class SVM Algorithm. http://rvlasveld.github.io/blog/2013/07/12/introduction-to-one-class-support-vector-machines/, 2022, Accessed: Nov-2022.

[31] Isolation Forest Algorithm. https://towardsdatascience.com/outlier-detection-with-isolation-forest-3d190448d45e, 2022, Accessed: Nov-2022.

[32] Supervised Machine Learning Algorithms. https://www.ibm.com/topics/supervised-learning, 2023, Accessed: Feb-2023.

[33] Unsupervised Machine Learning Algorithms. https://www.javatpoint.com/unsupervised-machine-learning, 2023, Accessed: Feb-2023.

[34] Sklearn Metrics. https://scikit-learn.org/stable/modules/model_evaluation.html, 2022, Accessed: Aug-2022.

[35] Precison, Recall and F1-score. https://towardsdatascience.com/precision-recall-and-f1-score-of-multiclass-classification-learn-in-depth-6c194b217629, 2022, Accessed: Aug-2022.

[36] ROC-AUC Curve. https://www.scikit-yb.org/en/latest/api/classifier/rocauc.html, 2022, Accessed: Oct-2022.

[37] Weighted Average F1-score. https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f, 2022, Accessed: Sep-2022.

[38] Accuracy. https://medium.com/@KrishnaRaj_Parthasarathy/ml-classification-why-accuracy-is-not-a-best-measure-for-assessing-ceeb964ae47c, 2023, Accessed: Mar-2022.

[39] Internet of Things. https://www.techtarget.com/iotagenda/definition/IoT-device, 2023, Accessed: Jan-2023.

[40] Attack Detection in IoT devices using ML. https://www.hornetsecurity.com/en/security-information/the-importance-of-machine-learning-in-cyber-security/, 2022, Accessed: June-2022.

[41] Supervised Learning. https://en.wikipedia.org/wiki/Supervised_learning, 2023, Accessed: Feb-2023.

[42] Mutual Information. https://machinelearningmastery.com/feature-selection-with-numerical-input-data/, 2023, Accessed: Feb-2023.

[43] Decision Trees. https://en.wikipedia.org/wiki/Decision_tree_learning, 2023, Accessed: Feb-2023.

[44] K-Nearest Neighbors. https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm, 2023, Accessed: Feb-2023.

[45] Support Vector Machines. https://en.wikipedia.org/wiki/Support_vector_machine, 2023, Accessed: Feb-2023.

[46] Unsupervised Learning. https://en.wikipedia.org/wiki/Unsupervised_learning, 2023, Accessed: Feb-2023.

[47] Local Outlier Factor. https://en.wikipedia.org/wiki/Local_outlier_factor, 2023, Accessed: Feb-2023.

[48] Isolation Forest. https://en.wikipedia.org/wiki/Isolation_forest, 2023, Accessed: Feb-2023.

[49] One-class SVM. https://en.wikipedia.org/wiki/One-class_classification, 2023, Accessed: Feb-2023.