



REPUBLIC OF TÜRKİYE

ALTINBAŞ UNIVERSITY

Institute of Graduate Studies

Electrical and Computer Engineering

**IMPROVING VANETS SYSTEM
SECURITY AGAINST DDOS ATTACKS BY
USING MACHINE LEARNING
ALGORITHM**

Naam Mudhafar Younus ALKADIRI

Master's Thesis

Supervisor

Asst. Prof. Dr. Muhammad ILYAS

Istanbul, 2023

IMPROVING VANETS SYSTEMS SECURITY AGAINST DDOS ATTACKS BY USING MACHINE-LEARNING ALGORITHM

Naam Mudhafar Younus ALKADIRI

Electrical and Computer Engineering

Master's Thesis

ALTINBAŞ UNIVERSITY

2023

The thesis titled IMPROVING VANETS SYSTEMS SECURITY AGAINST DDOS ATTACKS BY USING MACHINE-LEARNING ALGORITHM prepared by NAAM MUDHAFAR YOUNUS ALKADIRI and submitted on 12/04/2023 has been **accepted unanimously** for the degree of Master of Science in Electrical and Computer Engineering.

Asst. Prof. Dr. Muhammad ILYAS

Supervisor

Thesis Defense Committee Members:

Asst. Prof. Dr. Muhammad ILYAS	Department of Computer Engineering, Altınbaş University	_____
Asst. Prof. Dr. Abdullahi Abdu IBRAHIM	Department of Computer Engineering, Altınbaş University	_____
Asst. Prof. Dr. Jawad RASHEED	Department of Software Engineering, Nişantaşı University	_____

I hereby declare that this thesis meets all format and submission requirements of a Master's Thesis.

Submission date of the thesis to Institute of Graduate Studies: ____/____/____

I hereby declare that all information/data presented in this graduation project has been obtained in full accordance with academic rules and ethical conduct. I also declare all unoriginal materials and conclusions have been cited in the text and all references mentioned in the Reference List have been cited in the text, and vice versa as required by the abovementioned rules and conduct.

Naam Mudhafar Younus ALKADIRI

Signature



DEDICATION

I would like to present my big thanks to my supervisor Asst. Prof. Dr. Muhammad ILYAS for his guidance and support. Moreover, my parents, my husband Dr. Mahmoud Al-Layla, and my kids, for their patience and support.



ABSTRACT

IMPROVING VANETS SYSTEMS SECURITY AGAINST DDOS ATTACKS BY USING MACHINE-LEARNING ALGORITHM

ALKADIRI, Naam Mudhafar Younus

M.Sc., Electrical and Computer Engineering, Altınbaş University,

Supervisor: Asst. Prof. Dr. Muhammad

ILYAS Date: April / 2023

Pages: 58

With the fast and huge vehicular communication systems currently being developed, there is a clear and urgent need for advanced security to determine whether a vehicle has been attacked. Therefore, we developed a tool called the misbehavior detection system (MDS), which is intended to help the vehicle take action and minimize any potential harm from attackers. One of the most dangerous forms of attacks that threaten vehicular communication systems is distributed denial of service (DDoS) attacks. Increasing the security of VANETs against such attacks is a topic that a large number of researchers are now considering, to provide highly effective security capabilities, machine learning (ML) techniques were applied. NSL-KDD or KDD-CUP99 datasets form the basis for the greater part of the current research. Attacks on these datasets were outdated. Therefore, we used a new dataset generated by OMNeT++, Veins, and Sumo.

Seven different types of attack densities were conducted during this simulation, and the XGBoost classifier was used to evaluate and predict MDS systems.

The median F1-score for this XGBoost classifier was 99.70%, which represented a clear advantage over another ML method, where we used the Synthetic Minority Oversampling Technique (SMOTE) to class balance the datasets.

Keywords: Misbehavior Detection Systems (MDS), Machine Learning (ML), Vehicular Ad Hoc Networks (VANETs), Synthetic Minority Oversampling Technique (SMOTE), Synthetic Minority Oversampling Technique (SMOT).



TABLE OF CONTENTS

	<u>Pages</u>
ABSTRACT	vi
LIST OF TABLES.....	x
LIST OF FIGURES.....	xi
ABBREVIATIONS.....	xii
1. INTRODUCTION	1
1.1 INTRODUCTION AND PURPOSE	1
1.2 MACHINE LEARNING IMPORTANCE.....	2
1.3 DISTRIBUTED DENIAL OF SERVICE ATTACKS	3
1.4 THE SUGGESTED SOLUTION.....	3
1.5 THESIS OUTLINE.....	4
2. LITERATURE REVIEW	5
2.1 A SURVEY OF SECURITY IN-VEHICLE NETWORKS	5
2.2 SOFTWARE AND DATASETS FOR SIMULATING VEHICLE NETWORKS .	6
2.3 SIMULATORS FOR DETECTING MISBEHAVIOR IN-VEHICLE COMMUNICATION SYSTEMS	9
2.4 STATISTICAL METHODS FOR THE DETECTION OF MISBEHAVIOR IN VEHICULAR COMMUNICATION NETWORKS	10
2.5 OTHER TYPES OF MISBEHAVIOR DETECTION.....	16
2.6 STUDIES CONSTRAINTS.....	19
3. METHODOLOGY	22

3.1 MISBEHAVIOR DETECTION SYSTEM (MDS) IN VEHICULAR AD HOC NETWORK ARCHITECTURE	22
3.2 INTRODUCTION TO XGBOOST CLASSIFIER ARCHITECTURE	22
3.3 SYNTHETIC MINORITY OVERSAMPLING TECHNIQUE (SMOTE) FOR IMBALANCED CLASSIFICATION WITH PYTHON.....	24
3.4 SIMULATION TOOLS	24
3.5 EXPERIMENTAL METHODOLOGY	25
3.5.1 Simulation Settings	25
3.5.2 Data Preparation	27
3.5.3 Data Pre-Processing and Class Imbalance	28
3.5.4 Training Procedure	29
3.5.5 Classifier Performance	29
3.6 TYPES OF ERRORS	31
3.6.1 Log Loss (Binary Cross-Entropy Loss) Metric.....	31
3.6.2 Classification Errors	31
4. RESULTS	33
4.1 SIMULATION RESULTS.....	33
4.2 SIMULATION RESULTS AT DIFFERENT ATTACKS DENSITIES	34
4.2.1 Log Loss Error Curves	34
4.2.2 Classification Error Curves	39
5. CONCLUSION AND FUTURE WORK	44
REFERENCES	46

LIST OF TABLES

Table 3.1: Attack Times in Different Simulations.....	27
Table 4.1: The Accuracy of Different Attack Densities	33
Table 4.2: Different Machine-Learning Algorithms F1-Score Comparison.....	34



LIST OF FIGURES

	<u>Pages</u>
Figure 1.1: The Overall Design of Vehicle Ad Hoc Networks	2
Figure 3.1: A Screenshot of OMNeT++	26
Figure 3.2: The Imbalanced Dataset at 10% Attack Density	29
Figure 4.1: The 10% Attack Density	35
Figure 4.2: The 20% Attack Density	35
Figure 4.3: The 30% Attack Density	36
Figure 4.4: The 40% Attack Density	36
Figure 4.5: The 50% Attack Density	37
Figure 4.6: The 60% Attack Density	37
Figure 4.7: The 70% Attack Density	38
Figure 4.8: Classification Error at 10% Attack Density	39
Figure 4.9: Classification Error at 20% Attack Density	40
Figure 4.10: Classification Error at 30% Attack Density	40
Figure 4.11: Classification Error at 40% Attack Density	41
Figure 4.12: Classification Error at 50% Attack Density	42
Figure 4.13: Classification Error at 60% Attack Density	42
Figure 4.14: Classification Error at 70% Attack Density	43

ABBREVIATIONS

AI	:	Artificial Intelligence
WSM	:	Wave Short Message
IoT	:	Internet of Things
SMOTE	:	Synthetic Minority Oversampling Technique
VANETs	:	Vehicular Ad Hoc Network
MDS	:	Misbehavior Detection Systems
ML	:	Machine Learning
WSM	:	Wave Short Message
SUMO	:	Simulation of Urban Mobility
TTL	:	Time To Live
SVM	:	Support Vector Machine
TN	:	True Negative
TP	:	True Positive
V2V	:	Vehicle To Vehicle
V2X	:	Vehicle To Everything
ACK	:	Acknowledgment
ANN	:	Artificial Neural Network
CUSUM	:	Cumulative Sum
DDoS	:	Distributed Denial of Service
DoS	:	Denial of Service

FN	:	False Negative
FP	:	False Positive
GBDT	:	Gradient Boosting Decision Trees
GPT	:	Gradient Boosted Trees
GUI	:	Graphical User Interface
IDS	:	Intrusion Detection system
IP	:	Internet Protocol
LR	:	Logistic Regression
MANET	:	Mobile Ad Hoc network
MITM	:	Man In The Middle
NS2	:	Network Simulator 2
RSU	:	Road Side Unit
RF	:	Random Forest
OMNeT++	:	Objective modular Network Testbed In C++
VeReMi	:	Vehicular Reference Misbehavior Dataset
MLPC	:	Multilayer Perceptron Classifier
CO2	:	Carbon Dioxide
NED	:	Network Description
NN	:	Neural Network
R2LS	:	Remote To The Local System

1. INTRODUCTION

1.1 INTRODUCTION AND PURPOSE

Cities have undergone a change because of the recent spread of miniaturized autonomous driving technologies, which have made smart vehicles an attractive choice for daily commuting. Autonomous vehicles lighten the load on human drivers by carrying out sophisticated tasks like lane departure alerts, collision prevention, and traffic sign recognition. Autonomous driving technology has improved fuel economy by lowering emissions, and it may effectively control traffic flow and congestion and reduce crowding [1].

So cybersecurity and artificial intelligence are becoming important now as the number of self-driving cars on our roads increases. Communication links are crucial for self-driving vehicles and trucks to function securely and carry out essential tasks [1]. These links to the outside world enable vehicles to get over-the-air (OTA) software updates, real-time traffic data, navigational and road sign data, and data from other vehicles. Regrettably, this leaves them open to hacker infiltration [2]. Software on a basic USB stick may be compromised, and flaws in wireless vehicle-to-infrastructure (V2I) and vehicle-to-vehicle (V2V) connections to these systems may eventually put the safety of passengers and other road users in threat.

Network security intrusion is a term that has been used to describe the most typical DDoS attack [3]. In an effort to undermine the system's overall resilience, a DDoS attacker employs many vehicles to generate a lot of network traffic. In a DDoS attack, the targets of the attack are the cars themselves. DDoS attack machines are sometimes referred to as "zombies" because of this. In the worst-case scenario, stopping the connection between the vehicles might use a collision and death. It could be difficult to recognize this kind of attack when zombie vehicles are used in it. The major focus of this thesis is DDoS detection in vehicular communication networks by using the machine-learning algorithm. In (the figure1.1) below we can see the overall design of vehicle ad hoc networks [4].

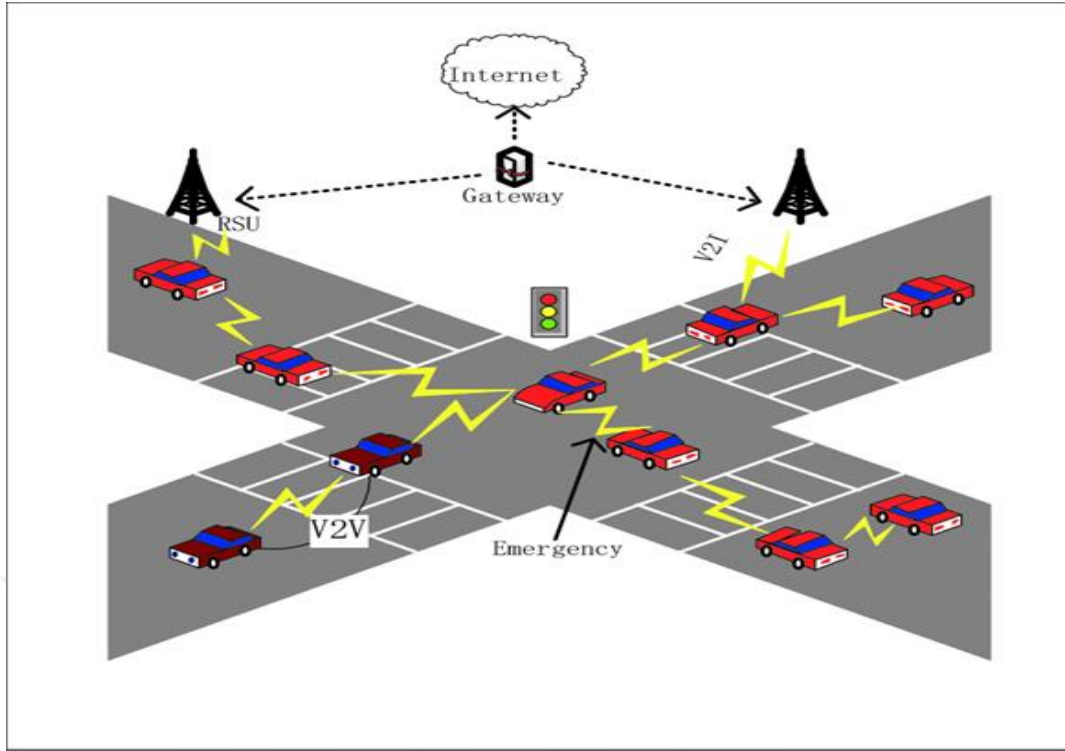


Figure 1.1: The Overall Design of Vehicle Ad Hoc Networks.

1.2 MACHINE LEARNING IMPORTANCE

These days, the importance of machine learning has grown up significantly in our lives, nowadays, we are seeing very important applications of artificial intelligence algorithms, and we will review some of them. As in [5] smart face mask that could detect whether someone was wearing a face mask or not which proved to be highly useful during the Covid-19 crisis, and in an intelligent adjunct framework built on machine learning (ML) to identify persons with COVID-19, persons with pneumonia, and healthy people, using X-ray pictures [6]. Furthermore, in forecasting air quality, the research in [7] was to determine how weather affects PM2.5 concentrations, Modern deep learning and machine learning models and recurrent neural networks were used for this forecasting. There are hundreds of studies examining the significance of AI in many fields of society, and these are just a handful that deals with applications of AI in our daily lives.

In this thesis, we will focus on the importance of machine learning in predicting DDoS attacks for self-driving vehicles, because of its great importance in this matter, which poses a danger to the lives of individuals.

1.3 DISTRIBUTED DENIAL OF SERVICE ATTACKS

Because availability is a primary issue for network and cloud service providers, distributed denial of service, often known as DDoS, is among the most serious dangers to the safety of the Internet. In distributed denial of service attacks, the attackers take control of a number of machines that have security flaws. These systems are then used to send a huge number of packets to the servers or network links that are the targets of the attack.

It slows down or stops service altogether because the servers that are being targeted are unable to handle the volume of requests.

The hacked systems in a distributed denial of service attack (also known as a DDoS attack) are not aware that they are being used in the attack.

Because of this, the word "zombie" is frequently used to refer to the computers that are utilized in distributed denial of service attacks.

It is possible that lives will be lost because of prohibiting individual vehicles from being able to communicate with one another. It may be difficult to see this attack because it involves zombie cars driving about

1.4 THE SUGGESTED SOLUTION

Against attacks of this nature, we require a very robust defense mechanism. Because of this, we have produced and developed an MDS system that is based on the XGBoost ML algorithm. This system is able to predict DDoS attacks from the attacker's vehicle and take action against them. The simulation was run with the assistance of OMNET++, which is an advanced tool in network communication that enables the execution of mobile vehicles via SUMO and Veins.

The built simulations feature a wide variety of cars, which enables the MDS to make use of the data regarding location and emissions from each vehicle. In the actual world, knowledge such as this is both helpful and not difficult to acquire.

As a result, the purpose of this research was to test the hypothesis that an XGBoost would be an improved MDS approach, one that would beat both distributed neural networks and other machine learning algorithms in terms of accuracy.

1.5 THESIS OUTLINE

This thesis has four further chapters. General information on the research literature review on misbehavior detection in VANeTs systems and intrusion detection systems and others detection systems is introduced in Chapter 2. The efficacy of current statistical learning techniques is the system's primary concern and area of interest. The methodology, which is the suggested MDS, and a review of the detection techniques employed in the MDS, including an introduction to the XGBoost classifier, are presented in Chapter 3. The results and assessments utilizing the generated simulation models and discussions of the simulation findings are found in Chapter 4. Conclusion and future work are provided in Chapter 5.

2. LITERATURE REVIEW

2.1 A SURVEY OF SECURITY IN-VEHICLE NETWORKS

There has been a lot of interest in systems created to guard self-driving cars against DDoS attacks that hackers could be subjected to in recent years as they have become more prevalent on public roads.

Understanding the whole security architecture of automotive networks is crucial before one can appreciate the significance of misbehavior detection in these networks. The authors in [8] created a review of one-hundred fourteen research publications on vehicle ad hoc network (VANET) security. VANETs are vulnerable to a range of attacks, including hardware tampering, impersonation, eavesdropping, false information, and denial of service, as described in the earlier review.

Researchers have employed a variety of techniques, including mathematical techniques, techniques that may rely on machine learning and artificial intelligence, or even a combination of these techniques with earlier techniques. This section provides an overview of some current DDoS attack detection research. Other security measures, such as public key cryptography, RSA, and elliptic curve cryptography, play a crucial part in lowering the number of intrusions that reach the system, as discussed in [8].

Using a clustering algorithm, the authors propose an intrusion detection system (IDS). The writers included detailed descriptions of the benefits and drawbacks of IDS layouts. The isolated IDS architecture is the first of three suggested layouts. Individual intrusion detection systems (IDSs) with stand-alone designs are autonomous units. Collaborative architectures allow the nodes in the system's vicinity to share information. Finally but critically, Hierarchical IDS designs require grouping the nodes for use in detection. The precise designs of each of the architectures discussed here are detailed in [9]. In order to gauge how well their IDS architecture performed in relation to others, the authors did not do any statistical analysis.

2.2 SOFTWARE AND DATASETS FOR SIMULATING VEHICLE NETWORKS

The authors created the Vehicular Reference Misbehavior Dataset in [10] to aid academics in creating repeatable studies (VeReMi). The authors particularly addressed the obvious problem that many studies studying vehicular networks do not use replicable datasets and tools. VeReMi has message logs that may be used to create systems for detecting inappropriate conduct. This dataset's source code, which includes more than 255 simulations, is available to the public. Additionally, VeReMi offers a variety of attack kinds, attacker densities, and attack traffic densities. As a result, the user has a wide range of dataset design possibilities.

This dataset, according to Heijden et al., is the first of its type. Even if the dataset is not perfect, this is a fantastic accomplishment. VeReMi is exclusively intended for detecting techniques; users are unable to simultaneously simulate and study the impact of several attacks. The authors proceeded to utilize this information after describing VeReMi to analyze the Gini index. The dataset itself seems to have more of an influence than the Gini index values.

Many prominent intrusion detection datasets, such as DARPA [11], were developed at MIT Lincoln Lab's network simulator. When they do this, they use authentication techniques like scanning attacks.

Attacks from the user level all the way up to the root account (U2R) and from the remote to the local system (R2L). In addition, they use availability attacks like DoS attacks.

CAIDA DDos2007 [12] comprises around one hour of traffic traces from a DDoS attack (UDP flooding) aiming to prevent access to a server by utilizing its computer and network resources.

Using a modest testbed, AWID 2 [13][14] generates code to carry out a variety of well-known attacks against 802.11. Several forms of attacks against authentication, availability, and confidentiality are all a part of this category, for example, ARP injection, RTS beacon, and others (e.g, rogue AP).

R2L, probing Attack, DoS, and U2R are only a few of the 23 attacks against authentication

and availability found in the famous KDD CUP 99 [15] intrusion detection dataset.

By eliminating redundant information and developing more refined sub-datasets, NSL-KDD[16] [17] improved the KDD CUP 99. The same kinds of attacks seen in the KDD CUP 99 are present in NSL-KDD.

Another type of dataset is Kyoto [18], which is a honeypot that includes authentic packet-based traffic recast as sessions. Out of the 24 features used in each session, 14 are taken directly from the KDD CUP 99 dataset. The following 10 characteristics are related to the actual flow of data, and they include things like IP addresses, ports, and timestamps.

SUMO, or Simulation of Urban Mobility, is a simulation program that is often used in automotive research projects. A free traffic simulation program is called SUMO. SUMO is extensively utilized in the research on vehicle communications, as mentioned by [19]. Additionally, according to Bearish et al., this tool will often be used in conjunction with a different communication simulator, such as NS2 or NS3. The application of SUMO in conjunction with NS2 is demonstrated in [19].

OMNeT++ is the most popular simulation program in the research of vehicular communications. This tool makes it easy to create simulations that combine GUI, C++, and NED-based options. This software performed a distributed denial of service (DDoS) demonstration on vehicular networks [20].

Using this program, Mohd et al. demonstrated a distributed denial of service (DDoS) on vehicular networks [21].

Four simulated vehicles' communications served as the foundation for their analysis. Multiple attackers are necessary for a DDoS attack. In this investigation, groups of 3, 6, 9, 10, and 12 attacking vehicles were used to try to interfere with packet delivery from the four usual vehicles.

A DDoS attack requires several attackers. At 0.1 ms intervals, the attacking vehicles sent out packets of 1 Kb, 10 Kb, or 100 Kb. The efficacy of this implementation in OMNeT++ was demonstrated by the twelve-vehicle DDoS attack, which was able to cause a 13.98% loss of packets between the four vehicles when broadcasting 100 Kb packets.

Similar to this, Kaur et al. developed DDoS attacks against web servers using OMNet++ & ReaSE [22]. The zombie servers used to generate this DDoS attack were randomly dispersed. One hundred nineteen components were utilized in this simulation. This total comprised a number of web-based servers and three separate routers. The amount of bandwidth used by the web server rose as the DDoS attack's duration grew longer. The number of packets lost rose as the attack lasted longer. When buffer size is taken into account, the study found that smaller buffers actually increase the effect of the DDoS attack. Additionally, a number of academics have created the datasets needed to create misbehavior detection software using these modelling approaches [23][24].

OMNeT++ has established itself as a well-liked tool in contemporary network security research. The program was compared to a testbed setup by the authors in [25] utilizing a D-Link wireless access point and two wireless clients. In OMNeT++, the testbed was precisely duplicated.

RTS, CTS, and ACK DoS attacks were launched against the testbed and the simulation. Using a Wireshark network analyser, attacks on actual networks were conducted. The study's findings supported the usage of the network simulator OMNeT++. The OMNeT++ simulation was able to provide results that were within 1% of the actual network for each of the three attacks. Although the focus of this publication was not vehicular communication networks, OMNeT++ has been used in a significant amount of other research that has been validated by its findings.

Network Simulator 2 and 3 (NS2/NS3) are free, community-developed, event-driven simulators aimed at advancing the study of computer communication networks.

Veins4 is a free software environment modelling and simulating road traffic systems. Two widely used simulators, OMNeT++ and SUMO, serve their foundation.

It goes beyond these to provide a full range of models for simulating traffic in complex networks.

CTUns-5.0 [26] is a robust and customizable network simulator and emulator that can emulate a wide range of wired and wireless IP network protocols.

Wireless and cable network infrastructures may be simulated with the help of software like GloMoSim (Global Mobile Information System Simulator) [27]. Protocols for wireless networks are supported in GloMoSim.

Virtual hosts, switches, controllers, and connections may be generated in Mininet5, a network simulator. Mininet hosts use industry-standard Linux networking software, and its switches are Open Flow [28] -enabled, allowing for highly adaptable routing policies and SDN.

2.3 SIMULATORS FOR DETECTING MISBEHAVIOR IN-VEHICLE COMMUNICATION SYSTEMS

The authors in [29] investigated how several flooding techniques—pure flooding, multipoint relay flooding, and geographic aware flooding—affected vehicular communication systems. If the sender is in front of the recipient, GAF operates similarly to a pure flooding approach; however, if the sender is in the rear, GAF will discard packets. This is because it is assumed under Rule 12 that a vehicle that is following the receiver no longer poses a threat. Based on a system known as optimal link state routing, MPR (OLSR). This method relies on the notion that just a few neighbouring cars will get the message from each vehicle as it relays it. The message will then need to pass across each group of cars.

In a circumstance where safety is at stake, the authors suggested that flooding may be utilized to effectively compel communications between vehicles. In order to verify this theory, NS2 crash simulations were created. With each vehicle driving at a speed of 130 kph, each approach was investigated utilizing flooding rates of 5, 10, and 20 packets per second. Pure flooding had a higher packet delivery ratio while being more successful in delaying background packets. It was stated that compared to other flooding techniques, pure flooding did result in higher network traffic overhead.

Networks for vehicle communication are clearly vulnerable to MITM attacks. The hostile group needs to steal important information from a common vehicle in order to carry out this attack. Allowing the attacker to listen in, delay, drop, or change packets that are essential for vehicle safety.

Researchers in [30] have shown how to create these attacks utilizing OMNeT++, SUMO,

and VEINS. The default map provided in Veins was utilized for this simulation. For this simulation, 100 cars and 5 RSUs were employed together with 10%, 20%, 30%, 40%, or 50% malicious nodes. The research simulation is reasonably well documented and is based on OMNeT++, so recreating it if necessary for verification should not be too difficult.

The study examined dispersed and fleet-based attacks. The scattered nodes caused greater damage to the network because of the three aforementioned MITM attacks than the fleet-based approach. As the percentage of attackers reached 20% or higher, this became increasingly obvious.

2.4 STATISTICAL METHODS FOR THE DETECTION OF MISBEHAVIOR IN VEHICULAR COMMUNICATION NETWORKS

The researchers in [31] identified a use for neural networks. This study demonstrated fifteen artificial neural networks (ANN) designed specifically to detect Black Hole attacks in VANETs. A "Black Hole" attack is a special type of DoS attack in which the attacker discards crucial packet data. The dataset underwent a jamming mechanism before being turned into an ANN in order to help determine the most distinct border between each attribute. The normal and abnormal groups were properly identified in 99% of ANN-based MDS.

On the VANET communication system, the author of [32] created a multilayer perceptron classifier (MLPC). These simulations, which feature distributed denial of service attacks, were developed using OMNeT++, Veins, and SUMO.

The suggested MDS processed data and detected attacks using the Apache Spark distributed computing platform. This MDS's F1 score was 96%. A multilayer perceptron classifier (MLPC), although a very powerful prediction algorithm, might not be the ideal option for use in VANET systems to detect dangerous behaviour.

Statistical methods seem to have a lot of potential in MDS systems. As shown in [31] and [32], a variety of machine-learning models may be employed to detect Sybil and DoS attacks as well as false alarms in in-vehicle communication networks. The authors looked at the uses of the k-Nearest Neighbors, Logistic Regression, Decision Tree Classifier, Bagging, and Random Forest approaches in both of these works. These two researches employ F1-score,

recall, and accuracy as performance indicators

Neural networks were also taken into consideration during the creation of MDS. The creators of [33] employed a convolutional neural network (CNN) architecture to detect traffic anomalies. The suggested CNN consisted of eight hidden layers, half of which were convolutional and subsampling layers.

The authors compared their model to a system created using principal component analysis (PCA). Instead of using an open-source dataset, this study's authors investigated their network testbed. In terms of genuine positive discoveries, the authors discovered that the CNN model performed better than the PCA-based model. The CNN model was shown to have less bias than its PCA counterpart, according to the study did.

Those investigating MDS have also made use of deep learning techniques. Researchers at Ewha W. University developed a deep neural network (DNN) architecture to monitor a car's controller area network (CAN) for signs of malicious activity. The DNN model was trained to look for inaccurate information about the tire pressure warning light in the automobile. Open Car Testbed and Network Experiments (OCTANE) compiled the dataset used to train and test the DNN model. All three, all five, and all seven of the model's layers were examined. All three approaches may have some success in predicting malicious and benign packets. Even though identification performance did not improve beyond level seven, regular packet detection did improve by 4%. The length of time needed for both training and testing naturally increased along with the number of layers [34].

Researchers at the Dalian University of Technology used the driver's emotional state algorithm (EAD) as the foundation for their investigation of anomalies [35].

In order to evaluate if the driver is being cautious, normal, or aggressive, the resulting algorithm EAD makes use of the emotional recognition coefficient Rde.

Rde, acceleration, velocity, and distance are all used as inputs in a Gaussian mixed model (GMM) to locate anomalies.

The NGSIM dataset was used for the assessment of the authors' new approach. The United States Department of Transportation (DOT) is responsible for this open data collection.

Using accuracy, recall, and F1 scores, we compared the proposed system to the hierarchical temporal memory (HTM) technique. It was found that the EAD method outperformed the HTM algorithm in both circumstances. When comparing HTM's accuracy and recall, the latter was consistently worse for HTM. Therefore, F1's performance was not meeting expectations set by EAD. Using the same dataset, one may verify that the models developed in [15] and [19] do not perform comparably to these findings.

The authors of [36] presented IDS for VANETs based on the network ToN-IoT dataset. There are two particular problems with the network records of this dataset: class imbalances and missing values. The Chi-squared and SMOTE techniques were used to tackle this problem. The ToN-IoT dataset is far more vulnerable to attack than earlier datasets like NSL-KDD, KDD-CUP99, and UNSW-NB15 [17][37]. The outcomes outperformed all other ML techniques when they employed the XGBoost model as a machine-learning algorithm.

The authors in [24] utilized hypothesis testing to detect rogue nodes in VANET systems. A simulation utilizing OMNET++, SUMO, and VACaMobil generated the data. For traffic flow modelling the green shield model was used. False Information and Sybil attacks were both considered and tested in this IDS simulation. If the test statistic is not within a 99% confidence interval, the parameter will be rejected. The proposed methods true positive and false positive values were compared to data-centric misbehavior detection (DCMD) and efficient and lightweight intrusion detection mechanism for the vehicular network (ELIDV). The suggested IDS was able to obtain just as many true positives as the other two models, if not more, even when the number of attackers rose by up to 40%. Even though ELIDV was highly well-scored, ELIDV performed only slightly better in the false positives test when the attack rate exceeded 30%.

[38] Has examined the efficiency of linear chain conditional random field (Linear-CRF) algorithms for misbehavior detection systems. The in-vehicle CAN communication system's missing and erroneous alarm data will be found using this anomaly detection methodology. The CAN bus network data examined in this report was created using the CANoe simulation software on a Win7 computer. The authors manually supplemented this data with anomaly data that was randomly injected. Two models were included in the suggested methodology:

a normal model built using a Linear-CRF and an anomalous model built using Viterbi's approach. When employing any independent model, the identification rate was above 90% and increased as the number of states increased. Similar declines were seen in the misreporting rate. The authors found that when combined, detection and misreporting rates were generally higher. They should have compared their model to more widely used models to properly illustrate the rise in detection.

The authors in [39] conducted a study to evaluate the performance of several machine-learning approaches for misbehavior detection. With NCTUns-5.0, the network simulation was created. In this simulation, there was 4528 samples total, 1427 of which were malicious. In the testing sample, the authors applied a variety of attacks, including packet retention, suppression, and identity fording. Weka, a data-mining program, was utilized to identify these attacks using several decision tree, random forest, K-means, and naive Bayes variants.

Random forest and a decision tree version were the most effective methods for spotting attacks. To demonstrate how successful each statistical approach was for each attack, the authors did not divide the findings. The reader is solely provided with the results of all attacks.

Modern misbehavior detection systems have found use in the big data method. Research on the effectiveness of distributed strategies in conjunction with machine learning to stop DDoS attacks in vehicular communication systems was published in 2019 [40].

Spark was utilized in the described technique to turn a communications dataset into a reliable distributed dataset (RDD). After the data has been loaded into the RDD, a decision tree will be trained to employ each of the databases after it has been constructed. Following the completion of the training phase, the individual trees that make up the random forest are brought together by means of the bootstrap sampling process, which is a traditional method. Both the NSL-KDD dataset and the UNSW-NB15 dataset were employed so that the suggested methods could be trained on both. Despite the fact that NSL-KDD was developed from KDDCUP99, the format is still exhibiting signs of its advanced age in several ways. Because of this, the UNSW-BN15 dataset, which is kept up to date by the Australian Centre for Cyber Security, was utilized. Additionally, this dataset was utilized. In a series of head-to-head tests, the produced Random Forest approach competed against SVM, XGBoost,

GBDT, and Naive Bayes. The results of these tests were compared below. In spite of the fact that both XGBoost and gradient-boosting decision trees (GBDT) performed rather admirably, the author's variant of boosted random forests ended up achieving the greatest score of any of the other methods [37][40][17].

The application of ML techniques to solve IDS problems in a variety of networks has seen widespread adoption. For instance, in past studies [41], they used the RF method to automatically build infiltration patterns. This was accomplished by using a radio frequency. After that, intrusions were located by contrasting the activity on the network with the patterns that were generated. In order to evaluate the effectiveness of the model, the writers utilized the knowledge they possessed. The authors made use of the dataset known as KDD-CUP99. In order to address the problem of unbalanced data, we employed procedures that included both down-sampling and over-sampling. The experiment was then run in the WEKA environment, using 34% of the samples as test data and 66% of the samples as train data. According to the trial's findings, 94.7% of samples were detected, and 2% of them were falsely positive.

In this study [42], the authors presented a one-of-a-kind way to check and evaluate the performance of TMs in the VANET environment. Their system combines a threat model and a risk assessment, both of which are used to detect possible attack vectors, and their system includes a vulnerability scanner. While the adversarial nodes are still active, these simulations of the enemy may be employed to put TMs through their paces.

They demonstrated the adaptability of their framework with examples such as the entity-oriented trust model, the data-oriented trust model, and the hybrid trust model. A huge number of simulations were carried out in order to investigate the responses that TMs have to different kinds of surroundings and foes. An unexpected discovery was made by TEAM, an organization that questions the widely held belief that hybrid trust models function more effectively in VANET due to the fact that they need concurrently evaluate the reliability of vehicles and data. On the other hand, their hypothesis predicts that data-oriented and hybrid TMs are inferior to entity-oriented TMs. This is due to the fact that the network has nodes that have a significant amount of reputation and experience, which assists in the dissemination of only valid information.

The authors in [43] make use of privacy-preserving collaborative MDS, which allows joint detection to be carried out without compromising the car's confidentiality. A vehicle in the proposed approach communicates encrypted weighted input to the LA about the target vehicle's behavior. In order to avoid being tracked by the LA or any other entity, the feedback is mixed at the LA using additive homomorphic characteristics.

Through meticulous security research, they have shown that the proposed method is safe from a variety of attacks and can identify attacks that use spoofed trust values. In addition, they have conducted an analysis of the suggested scheme's performance and shown that, in comparison to other schemes, the computation time of the proposed scheme is superior. High detection scores are also achieved using the suggested method. However, they did not investigate and examine how privacy-preserving MDS aids in spotting DoS and Sybil attacks in mobile communication systems.

The authors in [44] investigated the feasibility of using entropy and opportunistic selective sampling to identify malicious activity on VANETs. Using practical data, the authors compare and contrast the effects of selective and packet sampling.

Modelling the WAVE environment. It was found that a small number of flows, in comparison to packets, might contain a significant amount of data useful for detecting anomalies and unusual behavior. The results of calculating the entropy change showed that under typical network traffic circumstances, the values do not fluctuate much. With this work, they show that a linear classification model may be utilized to identify protocol disobedience and reduce the resulting broadcast storm.

As the size of the dataset grew in terms of flow samples and model tweaking, the performance of a binary classification model built on top of the TensorFlow framework improved in terms of loss, accuracy, and recall. With the growing magnitude and proximity of the obtained data compared to other sources, they think that the categorization of broadcast misbehavior flows may be improved by including more image and text machine-learning datasets in the model and metrics.

In this work [45], the authors describe and assess their most recent findings on using AI to solve the issue of jamming in real-time in V2X safety-critical circumstances. Their analysis

of the causes of their losses in VANETs is made more difficult by the presence of random jitter during the generation of CAMs. However, they conclude that their hybrid jamming detector, which combines statistical network traffic analysis with data mining techniques, may still achieve acceptable performance under these circumstances. The study's use case is a complex platooning C-ITS application in which V2X-enabled cars travel at highway speeds in close proximity to one another with narrow separation distances between them.

In [46] they suggested a system whereby a support vector machine is used to identify whether a jammer is active in a given network. The training data is based on the combined statistics of packet-drop probability that they collect. As shown by the findings, the suggested detection method is very efficient.

2.5 OTHER TYPES OF MISBEHAVIOR DETECTION

The authors in [47] proposed a modified version of the Reference Broadcast Method (RBS) to mitigate DoS attacks in-vehicle communications. RBS is built on the concept of an extra reference node acting as an intermediate between the two nodes in the issue. The IP-CHOCK technique provided here combines the RBS model with clock synchronization. The real IP-CHOCK implementation was done in C++. This article made use of detailed pseudocode. To execute the simulations, the Network Simulator (NS-2.34) and the tool command language (tcl) were utilized.

A GPS-based technique called Greedy Geographic Routing is described in [48]. This technique updates a trust routing database using GPS data (TRT). Each neighbour's trust value and progress value are stored in the TRT and may be utilized to identify misconduct.

For indicating when ACK messages are received from the destination node, this database must be updated and checked often. A node packet will be discarded if it is not from the source or the destination. OMNeT++, SUMO, and VaCAMobil data that the authors created were used to evaluate this methodology. It would have been preferable if this research had included additional descriptive statistics, such as accuracy and false positives. Without this information, it is difficult to compare the effectiveness of the procedure to the other methods mentioned.

Vehicle communication systems are susceptible to jamming attempts, just as wired systems are. This type of DoS uses a malicious signal to obstruct radio frequency transmission in a typical way. In other words, attackers would muddy a functioning vehicular communication system with made-up noise.

The authors in [49] introduced a technique of anomaly detection designed to identify DDoS attacks on roadside units (RSU). The suggested approach is founded on the online discrepancy test (ODIT), which is founded on the Cumulative Sum (CUSUM) and Geometric Entropy Minimization (GEM). In testing, the proposed model was contrasted with the Generalized Cumulative Sum (G-CUSUM), a realistic rendition of CUSUM. The simulation data needed to analyze the detection methods was produced using OMNET++, SUMO, and Veins. The paper includes simulation parameters.

The authors used IEEE 802.11p to mimic traffic on a roadway on the University of South Florida's campus. When low-rate DDoS attacks were used against this dataset, it was revealed that the proposed approach had a lower detection delay vs. false alarm probability than G-CUSUM-based techniques.

By employing a technique based on k-means, the authors of [50] offered a unique way to identify these attacks. Interference, Smart Attack, and Constant Attack Scenarios were used to evaluate this approach.

The authors utilized an altogether unique simulation; despite the fact that their data visualizations show that, their unsupervised learning strategy was effective. If they employed cutting-edge simulation technology or made their simulation code available for evaluation by other academics, the work would be far more credible.

The objective of the writers in [51] was the utilization of the RSSI for detecting misbehavior. When GPS is not accessible, their INTERLOC method is specially intended to identify Sybil's attempts. INTERLOC does not need any RSUs to work because it can get the RSSI values from nearby cars.

A circle is drawn around each participating vehicle in INTERLOC. By aligning these circles, one may create a localisation polygon that includes the target vehicle. The risk of a Sybil attacker increases if the vehicle is outside of this range. The simulation of this method was

performed using OMNeT++, Veins, and SUMO. The number of cars in the simulations varied from 900 to 7200. A car was randomly selected to be the attacker. Over time, the preferred vehicle would be swapped out on occasion.

FRIIS and the developed misbehavior detection system created by [52] were defeated by INTERLOC, according to the simulation findings. Sadly, the authors did not go into much detail about the simulation program. The study's replication and validation are made more challenging to complete due to the absence of information.

[53] Put out a misbehavior detection technique called sentinel. To defend against flooding attacks, this strategy was specially created for software-defined vehicular networks (SDVN). This method's two main components are detection and mitigation. In order to assess whether a flooding assault has a place, the detection step examines the new packet and flows. By examining their effects on detection and false positive rates, the cut-offs for misbehavior detection were manually established. In the quickly evolving world of automobile communication networks, this might be problematic.

A tree-building algorithm will search through all of the principal node's neighbours to find the malicious node if the algorithm determines that an attack was discovered.

The technique will advise the victim to discard any packets addressed by the attacker if a malicious node is discovered. The authors ran simulations in OMNeT++, Veins, and SUMO to evaluate the approach. In situations with less than 200 cars, the algorithm was shown to perform rather well at attack mitigation, but as the number of vehicles rose, its performance suffered.

Another method for DDoS detection in automotive communications systems was put out by [54] Multivariant stream analysis, or MVSA, is a technique for generating rules by gathering information from the flow of traffic. Every new packet will be compared to the rules the algorithm has created. As the main determinants of multivariate stream weight, average payload, hop count, time to live (TTL) value and packet frequency are all taken into consideration. Either the packet will be categorized as a regular packet or an attacker once the multivariate stream weight has been determined using the technique. Network Simulator 2.34 simulations were used to evaluate this approach. For each simulation, the AODV

routing protocol was used to perform the DDoS attacks.

MVSA scored well in terms of throughput, accuracy, detection time, and delivery ratio compared to the baseline methods utilized. MVSA scored well in terms of throughput, accuracy, detection time, and delivery ratio. Trilateral trust, Multi filter, and Hybrid Intrusion Detection Systems (H-IDS) served as the baseline algorithms for this comparison. It would have been ideal to include a statistical benchmark like logistic regression in the baseline to make it easier to compare the procedure to alternative strategies. To make it easier to compare the procedure to alternative strategies, it would have been ideal to include a statistical benchmark like logistic regression in the baseline.

2.6 STUDIES CONSTRAINTS

Mizukoshi and colleagues have undertaken more research on distributed system-based misbehavior detection systems. For the purpose of misbehavior detection, Apache Spark and genetic algorithms were coupled [55]. An entropy-based detection algorithm was used since attackers frequently use phony IP addresses. The authors compared results from the publicly accessible datasets KAREN's WITZ and the DARPA Intrusion Detection Evaluation. These datasets included conversational information from DDoS attacks. Sixteen nodes from the Hokkaido University Academic Cloud were used for this investigation.

On the DARPA and KAREN datasets, detection rates remained over 95% with false positive and false negative rates below 0.02 percentage. Regret, the authors did not include more trustworthy statistics like F1 scores, the ideal comparison for this approach would have been with naive Bayes, logistic regression, or random forests, among other statistical techniques.

Combining machine-learning techniques with conventional signal-processing methods has been used to identify DDoS attacks on web servers. Real attack data from SURF net was transformed using discrete Fourier transform (DFT) and discrete wavelet transform (DWT) by the authors of [56].

The training and testing datasets were transformed, and then the data was input into naïve bayes.

Fouladi et al. were able to detect attacks with an accuracy of 95% by combining the DFT and DWT. When just one transformation was used, DWT performed better than DFT in terms of its ROC curve but not in terms of the precision of misbehavior detection.

Unfortunately, the authors did not include data like the F1 score. Applying DFT before executing statistical algorithms in vehicular communication systems may eliminate the need for more complex computing techniques like neural networks. These approaches would need to be carefully examined in comparison to approaches employing statistics like false positives, false negatives, and F1 scores. Although the following study does not specifically address security or misbehavior detection.

The application of graph theory and deep reinforcement theory in communication systems is illustrated in [57] in an innovative way. Utilizing the proposed system by the author will increase the V2V system's overall throughput. This system's two phases are as follows. The initial one is based on a bipartite graph of mobile phone users. The users' channel allocation is then done using Max N-cut. A deep reinforcement learning (DRL) system, which is employed for power control, makes up the system's second half. In comparison to the random and hybrid models, the suggested resource allocation strategy exhibits encouraging results. For the sake of replication, simulation settings are supplied in this research. Although the authors note that training time is an issue for this strategy, it was not a problem when it was being used.

3. METHODOLOGY

3.1 MISBEHAVIOR DETECTION SYSTEM (MDS) IN VEHICULAR AD HOC NETWORK ARCHITECTURE

As we will describe in this chapter, we introduce a misbehavior detection system (MDS) for VANETs based on different data preparation and data pre - processing strategies. We will use machine-learning algorithms to detect internal attacks on VANETs system, as we have noted in previous studies, they discussed the important topic about attacks detection and prevention on VANETs systems. Attacks such as DDoS, Sybil, grey holes, man-in-the-middle (MITM) and others are a threat for the VANET system. While various machine learning methods, neural networks, and deep learning produced a number of findings and good percentages, the vast majority of studies used well-known traditional and public datasets, as described in the second chapter. Here we will use a realistic dataset generated by OMNeT++, Veins, and Sumo, the simulations consist of a group of normal cars and attacking cars with DDoS attacks. After that, we examine the obtained dataset using the supervised machine-learning XGBoost classifier. We performed some pre-processing, configuration, and arrangement of the training dataset to solve the bias problem toward the majority. We get the best results, which outperformed all the past studies.

3.2 INTRODUCTION TO XGBOOST CLASSIFIER ARCHITECTURE

Gradient-boosted decision trees provide the foundation for the creation of XGBoost [58][59]. XGBoost is an efficient and fast method when compared to other Machine learning algorithms. It is a technique for boosting machinery or applying boosts to machines. Extreme Gradient Boosting, or XGBoost, assists in maximizing memory and hardware resources for tree-boosting techniques. It has the advantages of enhancing the algorithm and modifying the model. To improve the accuracy of the results, it performs a Taylor expansion for the cost function by the second derivative. The XGBoost method uses an additive training approach to optimize the objective function, which implies that the outcome of the earlier stage affects the optimization process later on. The t -th objective function of the model can be stated as in equation (3.1):

$$obj(t) = \sum_{i=1}^n l(y_i, y_i^{\wedge t-1}) + f_{t(x_i)} + \Omega_{(f_t)} + \mathcal{C} \quad (3.1)$$

Where l is the t -th round's loss term, Ω is the regularization term as in equation (3.2) and C is a constant.

$$\Omega_{(f_t)} = \gamma T_t + 0.5\lambda \sum_{j=1}^T w_j^2 \quad (3.2)$$

Where the customization parameters are γ and λ . The more significant these two values, in general, the more fundamental the structure of the tree. It is also possible to correctly treat overfitting problems.

As we see in equation (3.3) a second-order Taylor expansion:

$$obj(t) = \sum_{i=1}^n [l(y_i, y_i^{\wedge t-1}) + g_i f_t(x_i) + 0.5 h_i f_t^2(x_i)] + \Omega_{(f_t)} + C \quad (3.3)$$

Where the first and second derivatives, respectively, are g and h . we can define g as in equation (3.4) and h as in equation (3.5):

$$g_i = \partial_{y_i^{\wedge t-1}} l(y_i, y_i^{\wedge t-1}) \quad (3.4)$$

$$h_i = \partial_{y_i^{\wedge t-1}}^2 l(y_i, y_i^{\wedge t-1}) \quad (3.5)$$

By obtaining the derivative and substituting the earlier XGBoost equations (3.2) and (3.4) and (3.5) into equation (3.3). Then refers in equation (3.6) to the solution of weights.

$$w_j^* = - \frac{\sum g_i}{\sum h_i + \lambda} \quad (3.6)$$

Where the loss function score is shown by obj^* in equation (3.7).

$$obj^* = -0.5 \sum_{j=1}^T \frac{(\sum g_i)^2}{\sum h_i + \lambda} + \gamma \cdot T \quad (3.7)$$

The quality of the tree's structure increases with decreasing object size. The weights' solution is referred to as w^* [57].

3.3 SYNTHETIC MINORITY OVERSAMPLING TECHNIQUE (SMOTE) FOR IMBALANCED CLASSIFICATION WITH PYTHON

On classification datasets with significant class imbalances, predictive models with

imbalanced classification are created. Working with unbalanced datasets is a problem since most machine learning algorithms would underperform on the minority class, even though this class' performance is often of the utmost importance. One approach to addressing imbalanced datasets is to oversample the minority class.

Duplicating instances from the fewer class is the easiest method, however, it should be emphasized that these examples do not provide any new insight into the model. Instead, fresh instances may be created by synthesizing the old ones. So (SMOTE) is a method of data augmentation for fewer classes [60] [61].

3.4 SIMULATION TOOLS

We have provided a summary of the OMNeT++[62] [20]discrete event simulation platform in this paper. This platform was created to allow the modeling of communication networks and other parallel and distributed systems. The NS-2 (and NS-3) [63] project goal is to construct a network simulator, whereas OMNeT++ aims to offer a rich simulation platform and defers to separate research groups for the creation of simulation models. This is a key difference between the two projects' approaches. Since OMNeT++ has been developed to handle complex communication simulations, it served as the foundation for all simulations.

OMNeT++ provides a component architecture for models. C++ is used to write the code for the individual components, which are subsequently integrated into bigger components and models with the help of a high-level language (NED). The option to reuse models is provided at no additional cost. OMNeT++ has comprehensive support for graphical user interfaces, and its modular architecture enables the simulation kernel (as well as models) to be incorporated into your applications with relative ease [64][65].

Veins and SUMO must be utilized in order for these simulations to support mobile vehicles. SUMO [19][66][67] is a traffic simulation program that creates typical cars and other forms of vehicular movement. In order to create vehicular communication systems, Veins is then utilized as a bridge to connect OMNeT++ and SUMO.

3.5 EXPERIMENTAL METHODOLOGY

3.5.1 Simulation Settings

On an Ubuntu virtual machine, every simulation of the vehicular network was built. Upon installation, Veins includes a built-in map and a simulation of a mobile vehicular communication system; in this generic simulation, 194 vehicles are moving in the same direction when, 73 seconds into the run, a collision occurs. This collision lasts for a total of 50 seconds. The vehicles respond to this by warning other cars of the impending traffic jam. There is only one roadside unit in the whole 200-second scenario.

We used this simulation to implement the DDoS simulation after making certain changes to meet the requirements of our study.

Each of the following five vehicle numbers—15, 20, 25, 30, and 35—had its own simulation. In addition to the vehicles previously indicated, four parked cars were added to the list of vehicles previously mentioned that functioned as the bases from which to execute the DDoS attacks against the other vehicles and roadside installations. Despite being used in the attack, these vehicles would be regarded as the same as mobile vehicles in all other circumstances. Since the attacker employed these vehicles as zombies, they could be contacted normally before the attack.

The accident that was introduced into the default program at 73 seconds remained unaltered.

The 1-hop broadcasting between the vehicles and the roadside devices is used in the OMNeT++ simulations created for this thesis. Each message will therefore be sent to every node within the sender's range.

A screenshot of a simulation running in OMNeT++ is shown in Figure 3.1 below the attackers are identified in this graphic as "hackers," whereas the typical cars are identified as "nodes".

Table 3.1: Attack Times in Different Simulation.

Attack%	First Attack Density Time	Second Attack Density Time
10%	50-74	210-224
20%	50-74	210-262
30%	50-112	210-262
40%	50-150	210-262
50%	50-150	210-300
60%	50-150	210-337
70%	50-150	174-340

Short messages (WSM) delivered 25,000 WAVE to each vehicle during this time. The communication code present in traCIDemo11p.cc of the Veins demo served as the basis for the code used to produce the WSM. The handlePositionUpdate () function included the harmful modifications. This algorithm repeatedly generates WSM, which are subsequently sent to all cars around and contain the current road id of the moving vehicles.

3.5.2 Data Preparation

The Veins simulation included in OMNeT++ automatically recorded the x- and y-coordinates, as well as the speed, acceleration, and CO2 emissions of each individual car. During the experiment, a new dataset was collected for each of the aforementioned variables at a rate of one per second. After that, the data was converted to an Excel file so that it could be arranged and investigated in a Jupyter Notebook. Before the data could be used to train models, the column names had to be scrubbed properly to remove any instances of commas, periods, or number signs. In order to make the information in the OMNeT++ generated time column easier to understand, the "t" was replaced with the words "Time in Seconds." The "if-then" feature of Excel was then used to generate an additional column in the spreadsheet that was titled "Attack Bool."

If there were an attack within this time, the Excel formula would return the value one; but, if there were not an attack, the result would be zero. For this column, "Attack Bool" was the term that was used in every simulation. In the form of a comma-separated values (CSV) file, the dataset was uploaded to Google Colab by utilizing Google Drive. Students are able to study Python without having to install anything on their computers because Google Colab is a web-based Integrated Development Environment (IDE) for Python that runs on a web browser [22]. All that is required to use Google Colab is a web browser and a Google account. The authors give their strong recommendation to use Google Colab. After that, the sanitized CSV was imported into a Jupyter Notebook.

3.5.3 Data Pre-processing and Class Imbalance

Before feeding the data into ML algorithms, cleaning and preparation are crucial to ensuring high performance. One problem with the dataset that we are using for our studies is its class imbalance. With oversampling, under-sampling, and hybrid approaches, the imbalanced problem was resolved. Some academics have employed the duplication of the minority class points through oversampling under similar circumstances.

However, this technique's drawback is that it overfits these points. The dominating class loses some points when is used by others. This approach presents a challenge since some properties that are deleted may be essential to an accurate portrayal of the class. Therefore, we use a hybrid approach where some majority class points are removed and some minority class points are duplicated. By providing synthetic minority class samples, the synthetic minority oversampling method (SMOTE) enhances basic random oversampling and tackles the overfitting problem that could occur with simple random oversampling. SMOTE creates, newer ones Instead of reproducing already-existing data points. To generate additional minority data points, two comparable minority samples are linearly combined. New feature values are continuously interpolated between the minority sample and its closest neighbours.

In (figure 3.2) below, we can see the distribution of the attack bool datasets, which are divided into two class attacks and no attacks. Before applying the SMOTE technique, we can see how the dataset is imbalanced and the biasing is towards the majority, which is here the no attacks class because the attack density here is only 10% percentage.

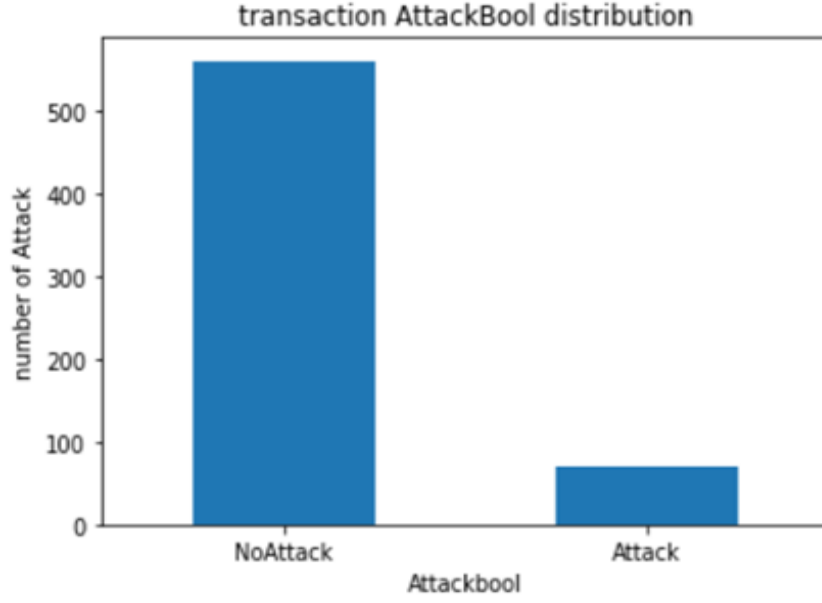


Figure 3.2: The Imbalanced Dataset at 10% Attack Density.

3.5.4 Training Procedure

After implementing the cleaning and pre-processing procedures and solving the class imbalance problem, we set the dataset to training and testing procedures. First, we divided the dataset into two sets, the first containing training, and validation data comprising 70% of the total dataset, and the second containing the unseen test dataset, representing 30% of the sample, to evaluate the effectiveness of the selected ML algorithms. All datasets were in CSV format.

3.5.5 Classifier Performance

Several measures were utilized to determine whether the XGBoost ML algorithm was effective. Because they provide a comprehensive explanation of the outcomes of ML-based inappropriate behavior prediction, the particular assessment methods that were used were chosen for this project [30]. The accuracy in equation (3.8) is the first measure that analyses the general efficacy of a technique by measuring the proportion of correctly classified instances as an attack or a common occurrence. This is done by evaluating the total number of correctly classified instances.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (3.8)$$

The second metric, precision in equation (3.9) depicts the percentage of accurately diagnosed attacks among all detected attacks.

$$Precision = \frac{Tp}{TP+FP} \quad (3.9)$$

Recall, the final metric in equation (3.10), gives the proportion of properly identified attacks to all attacks in the test dataset.

$$Recall = \frac{Tp}{TP+FN} \quad (3.10)$$

The F1-score in equation (3.11), which calculates the weighted average of accuracy and recall, is the fourth metric [19].

$$F1 - score = 2 \times \frac{Precision * Recall}{Precision + Recall} \quad (3.11)$$

Where true positive (TP) is the total number of legitimately classed attacks that are real attacks. The total number of actual normal recordings that are correctly labelled as normal behavior is known as the true negative (TN). The total number of genuine attack samples that were mistakenly identified as normal are known as false negatives (FN). The total number of genuine normal samples that are incorrectly labeled, as attacks are known as false positives (FP).

3.6 TYPES OF ERRORS

3.6.1 Log Loss (Binary Cross-Entropy Loss) Metric

Like the majority of other gradient boosting algorithms, XGBoost makes use of the well-known measurement known as "log loss." The effectiveness of a classification model is evaluated using this probability-based metric. Cross-entropy loss, also known as log loss, assesses how well a classification model performs when producing a probability value between zero and one. The most significant classification metric based on probabilities is called log loss.

Log-loss is a useful tool for comparing models, despite the difficulty in interpreting raw log-loss numbers. A lower log-loss number shows a better prediction for any given situation. As we can see in equation (3.12)

$$LogLoss = -\frac{1}{N} \sum_{i=1}^N [y_i \log p_i + (1 - y_i) \log(1 - p_i)] \quad (3.12)$$

Where N stands for the number of samples or instances in equation 3.12 above. The result of the i-th occurrence would be "y_i." Let us presume that an instance can assume two possible results, such as 0 and 1. Since 'y_i' in the equation above would be 1, '1-y_i' is equal to 0. The probability of the i-th occurrence adopting the value "y_i" is indicated by the symbol "p_i." Log loss, then, adds up the likelihood that a sample will assume both states 0 and 1, multiplied by the total number of cases. The equation's basic premise is that the probabilistic factor for the true output (y_i) is -log (probability of true output), and the probabilistic factor for the other output is -log (1-probability of true output) [68][69].

3.6.2 Classification Errors

Classification error [70] commonly referred to as zero-one loss, measures how well a classification model performs when the expected output is a discrete class label. It is the number of incorrect classifications (or the percentage of incorrect classifications) relative to the overall number of cases as we can see in equation (3.13).

$$Classification\ error = \frac{errors}{total} = \frac{FP+FN}{TP+TN+FP+FN} \quad (3.13)$$

In conclusion, classification error measures how well a hard classifier performs, while log loss measures how well a probability-based classifier performs.

4. RESULTS

4.1 SIMULATION RESULTS

The XGBoost classifier shows excellent results based on the SMOTE technique applied to the dataset before training and testing.

By applying different attack densities, we can see from table 4.1 below that the accuracy of XGBoost is extremely good for all densities at different times throughout the 380 seconds simulation time on OMNeT++.

Table 4.1: The Accuracy of Different Attack Densities.

Attack density %	The accuracy of XGBoost classifier %
10%	99.17
20%	99.70
30%	99.30
40%	99.60
50%	98.57
60%	99.60
70%	99.32

In addition, the precision and Recall and F1-score result are 99 % (approximately).

Table 4.2 compares XGBoost when the attack density is 10% and the other ML algorithms from the previous study according to the F1-score evaluation metric described in chapter three.

Table 4.2: Different Machine-Learning Algorithms F1-Score Comparison.

Machine-Learning Algorithms	F1-Score Value
Support vector machine (SVM)	0.88
Logistic Regression (LR)	0.89
Multi-layer perceptron classifier (MLPC)	0.96
Decision Tree Classifier (GBT)	0.97
Random forest (RF)	0.97
XGBoost classifier	0.99

4.2 SIMULATION RESULTS AT DIFFERENT ATTACKS DENSITIES

4.2.1 Log Loss Error Curves

In the figures below, we can see the (log loss) curves of our model XGBoost binary classification for different attack densities applied to the vehicle simulations, which start from 10 % attack density and then increase by 10 % until reaches 70 %. We have two duration scenarios for the attacks as shown in table 3.1 in chapter 3. The first duration attack starts at 50 seconds of the simulation time for all the seven-version attacks, the second attack starts at 210 seconds of the simulation time for the first sex version except for 70 % attack density starts at 174 seconds of the simulation time. We will describe detail about attack times in the figures below.

In the first (figure 4.1) for 10% attack density, which has two scenarios of attacks, the first attack was between 50-74 seconds while the second attack was between 210-224 seconds.

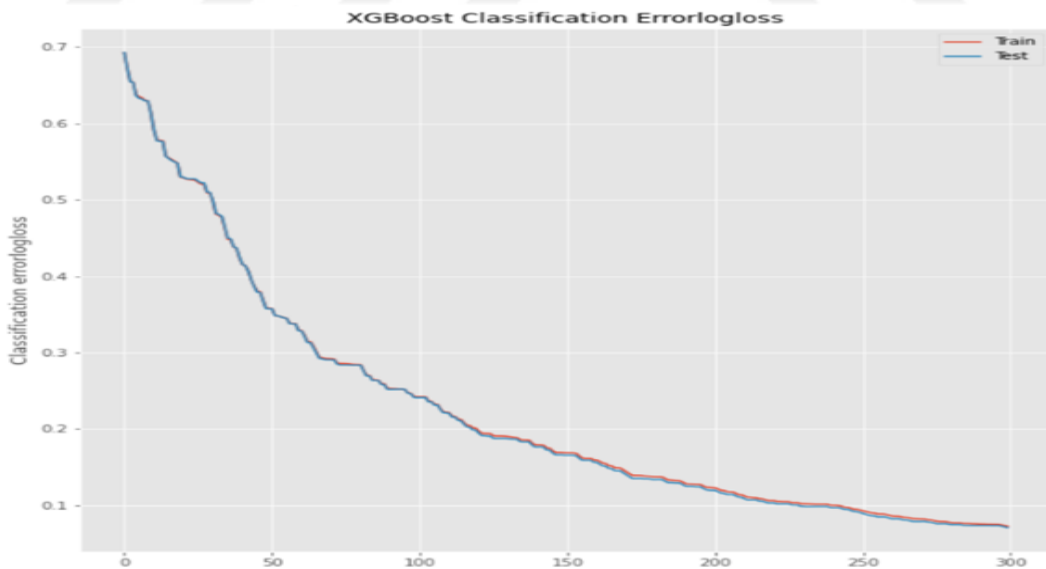


Figure 4.1: The 10% Attack Density.

(Figure 4.2) below is for 20% attack density this attack's first duration was between 50-74 seconds and the second attack duration was between 210-262 seconds of the hall simulation time.

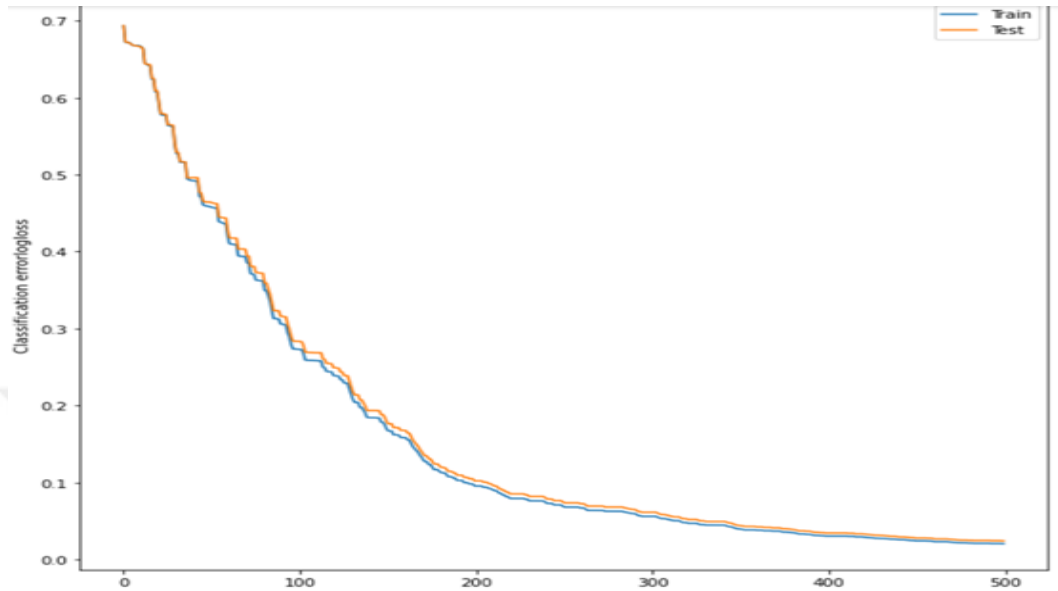


Figure 4.2: The 20% Attack Density.

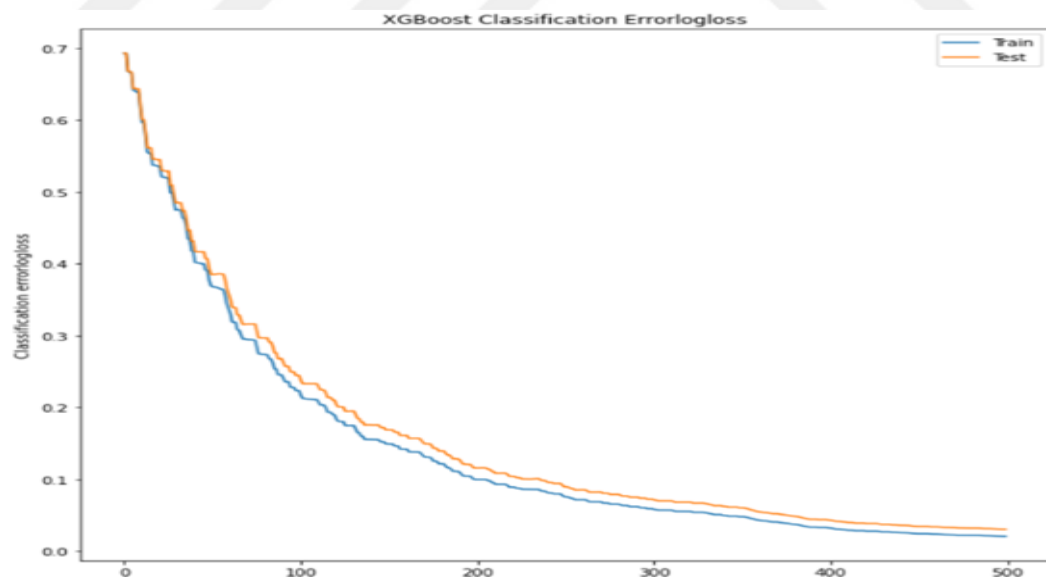


Figure 4.3: The 30% Attack Density.

(Figure 4.3) above shows the log loss curve for 30 % attack density, also it has two scenarios for attacks the first is from 50-112 seconds and the second one is from 210-262 seconds.

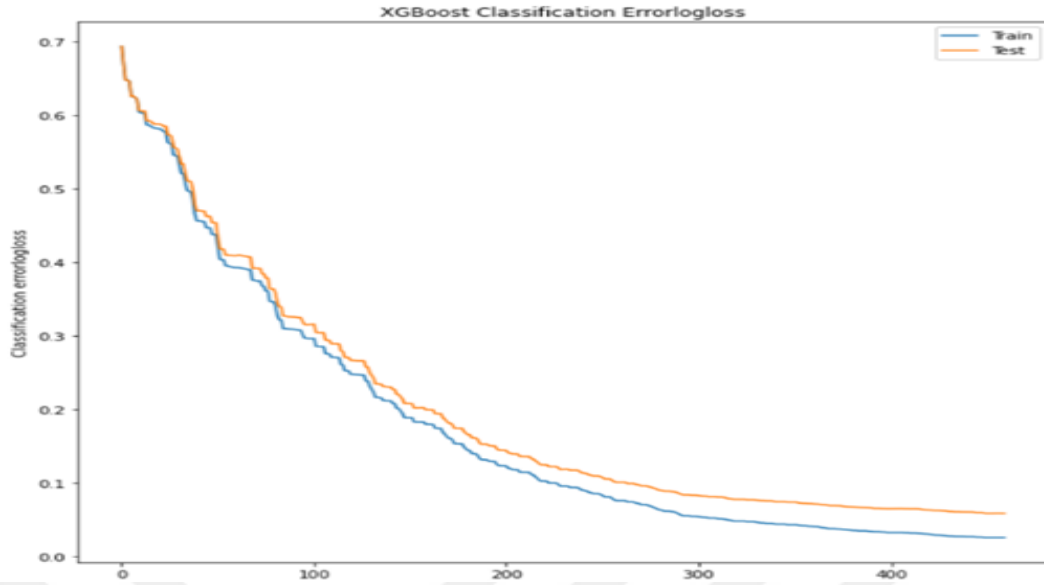


Figure 4.4: The 40% Attack Density.

(Figure 4.5) below also shows the log loss curves for the training and testing dataset, we also have two scenarios for the attacks the first starts at 50-150 and the second attack is between 210-300.

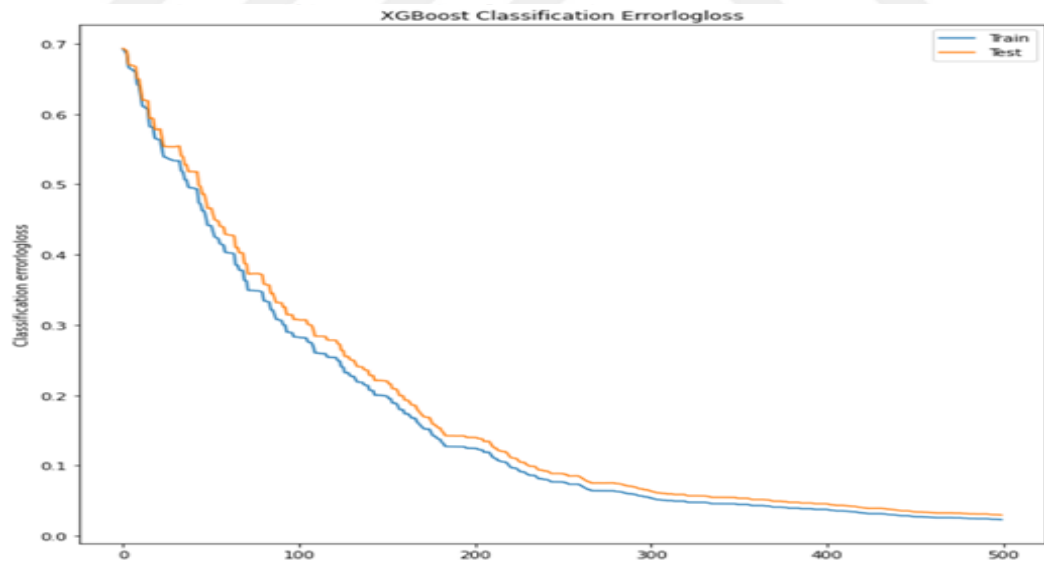


Figure 4.5: The 50% Attack Density.

In (figure 4.6) below we can see the log loss for 60% attack density, the first attack time was between 50-150s when the second attack time was between 210-337s.

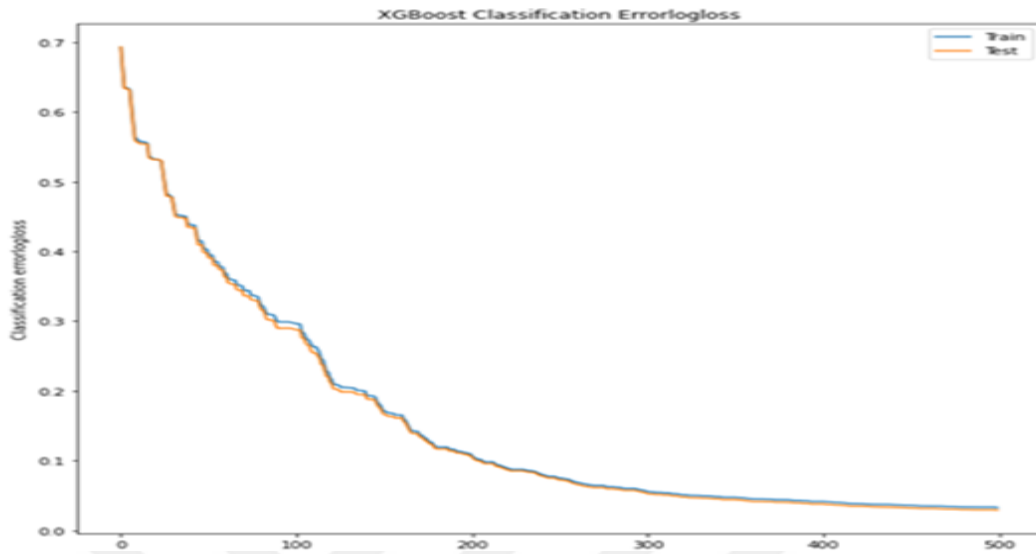


Figure 4.6: The 60% Attack Density.

The last (figure 4.7) was for 70% density, which represents 228s of the 380s simulation times, which is the large duration of the vehicles under attack. The first attack duration is between 50-150, while the second attack duration was between (174-340) which is the only one from the previous attack density starting at 174s of the simulation time.

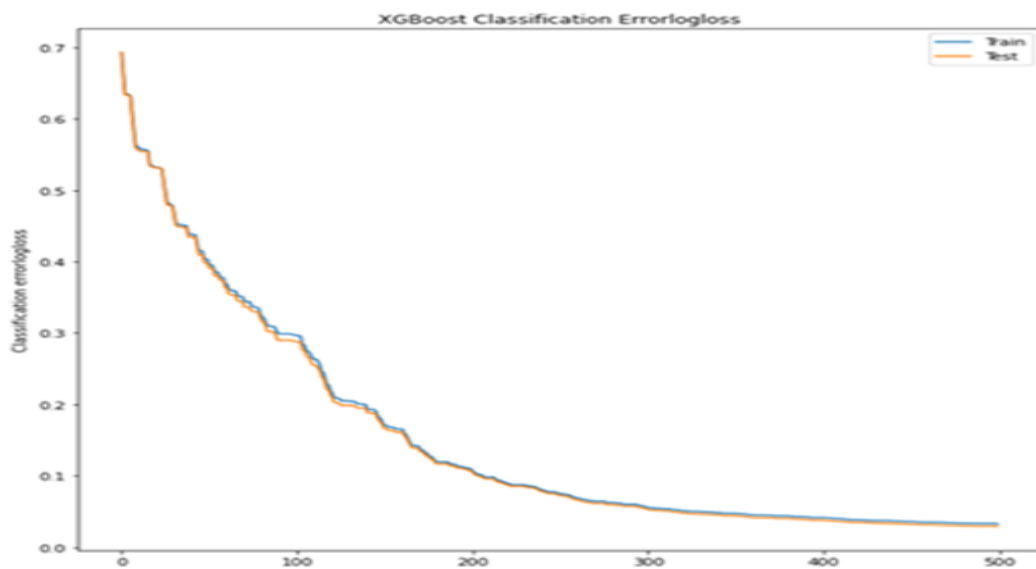


Figure 4.7: The 70% Attack Density.

We can see from the seven figures above that the logarithmic loss for the XGBoost classification (log loss) is approximately equal for all densities, and the error starts from 0.7 and decreases until reaches 0.05 and which is a very good result. As we know when the log loss error decreases the accuracy increase, which is our goal in this study. The exception is only for 40% attack density when the error is a little bit increased and reaches 0.075 and is still constant to the end of the 300 n-estimator on the x-axis.

We choose the default values for the XGBoost hyper parameter tuning, first the learning rate, which is the learning rate of boosting equal to 0.01, and we choose the max-depth equals to four, we experimented with changing the learning rate to improve the model. In XGBoost, the default-learning rate is set at 0.01. We experimented with n_estimator=400 and a learning rate of 0.05. The model's construction provided moving slowly toward performance goals, similar to the usual learning pace. With N-estimators equal to 400, we employed learning rates of 0.1 and 0.01. The model provides the same efficiency but at a different pace, according to this. As a result, performance metrics for different learning rates are unaffected at N-estimator = 400.

The default hyper parameter selected in our XGBoost classifier:

First, our model is binary classification so we choose: Objective='binary: logistic',

The Learning-rate=0.05 as we describe above, and N-estimators=400, Max-depth=four, Subsample=0.5, and Colsample-bytree=0.25.

The early stopping rounds selected for the model is 10, which means the model is not improved after 10 rounds through testing dataset and still has the same accuracy, this is a distinguished way in the XGBoost classifier to prevent overfitting problems.

4.2.2 Classification Error Curves

In the figures below from (figure 4.8) to (figure 4.14), we can see the classification error curves for our XGBoost model.

We reduce the n_estimators to 100 only instead of 300 n-estimators as we do with the log-loss errors, so we can see how the errors changed more clearly.

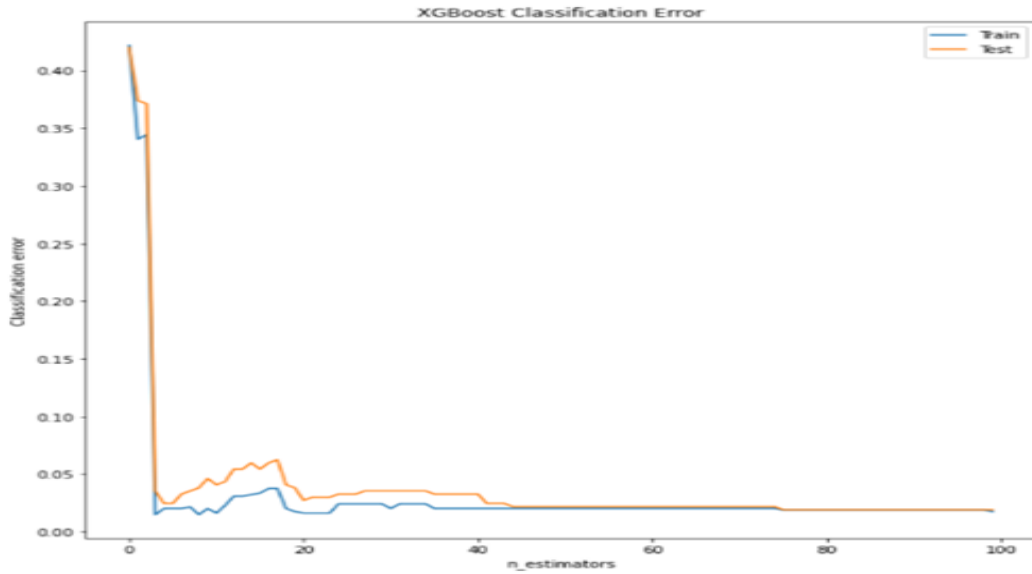


Figure 4.8: Classification Error at 10% Attack Density.

In Figure 4.8 above at 10% attack density, we see that at 19 estimators the error reaches 0.07 maximum value then decreases and reaches 0.02 value, and remains constant to the end.

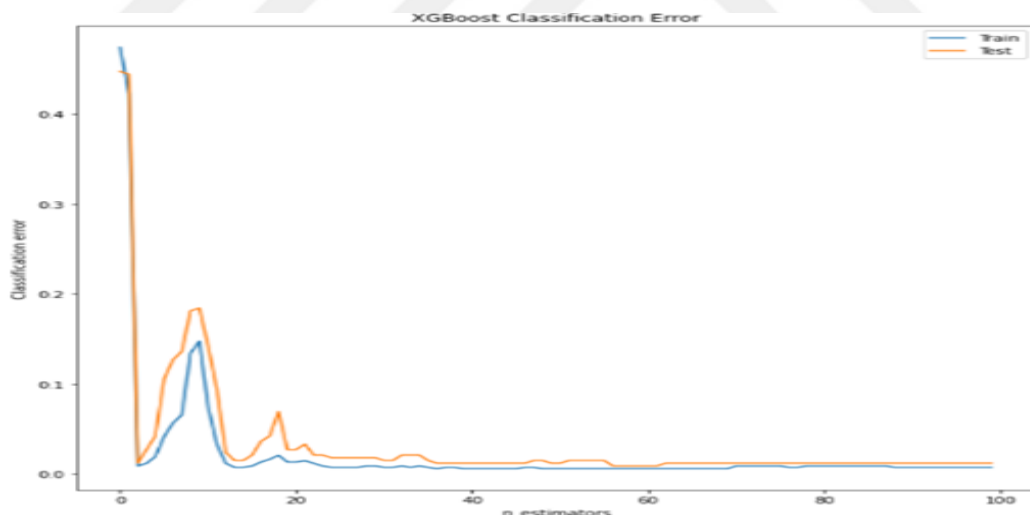


Figure 4.9: Classification Error at 20% Attack Density.

In figure 4.9 above at 20% attack density, we can see the error reaches 0.18 and then decrease, reaches 0.02, and is still constant to the end.

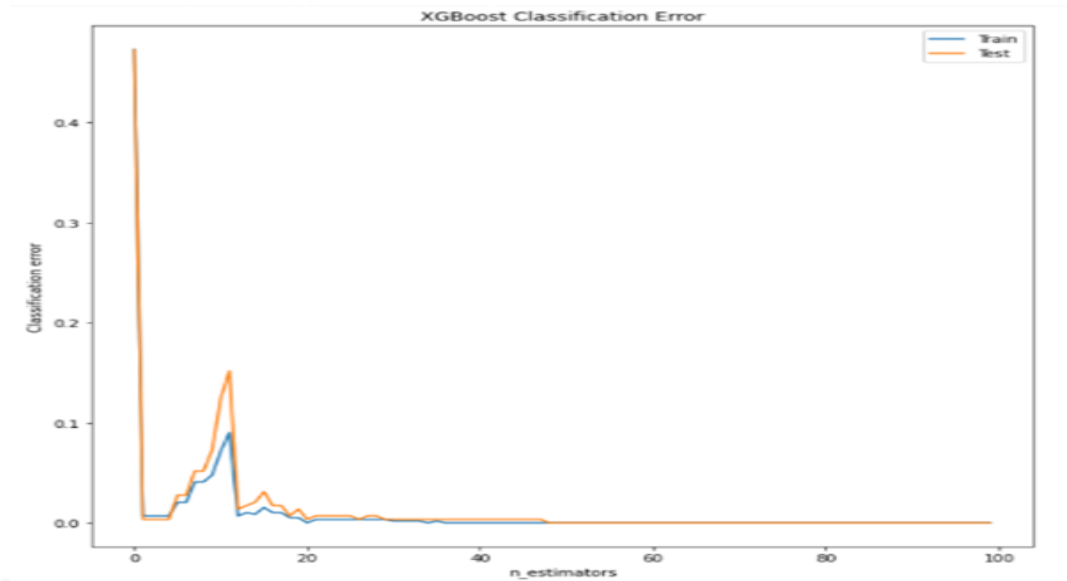


Figure 4.10: Classification Error at 30% Attack Density.

In addition, at 30% attack density in (figure 4.10) above, the error reaches 0.15 and then decreases and reaches to 0.01 and this is an excellent value.

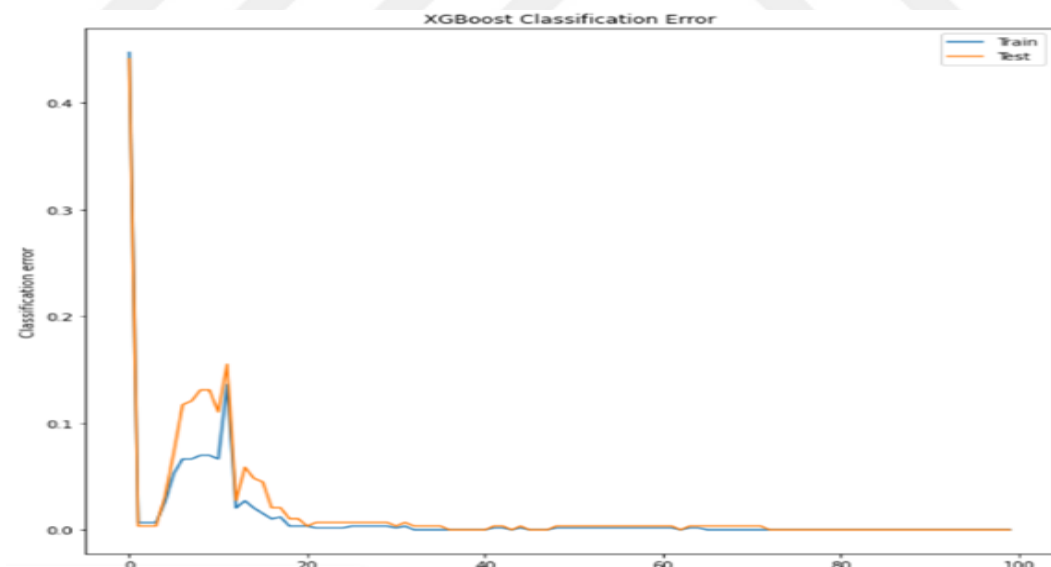


Figure 4.11: Classification Error at 40% Attack Density.

At a 40% attack density, we can see in Figure 4.11 above that the error reaches a maximum of 0.15 before falling to a minimum of 0.01 and remaining constant the entire time.

As can be seen in figure 4.12 below, the error level increases to 0.12 when there is a 50% attack density, then decreases to 0.01 and remains the same throughout the time.

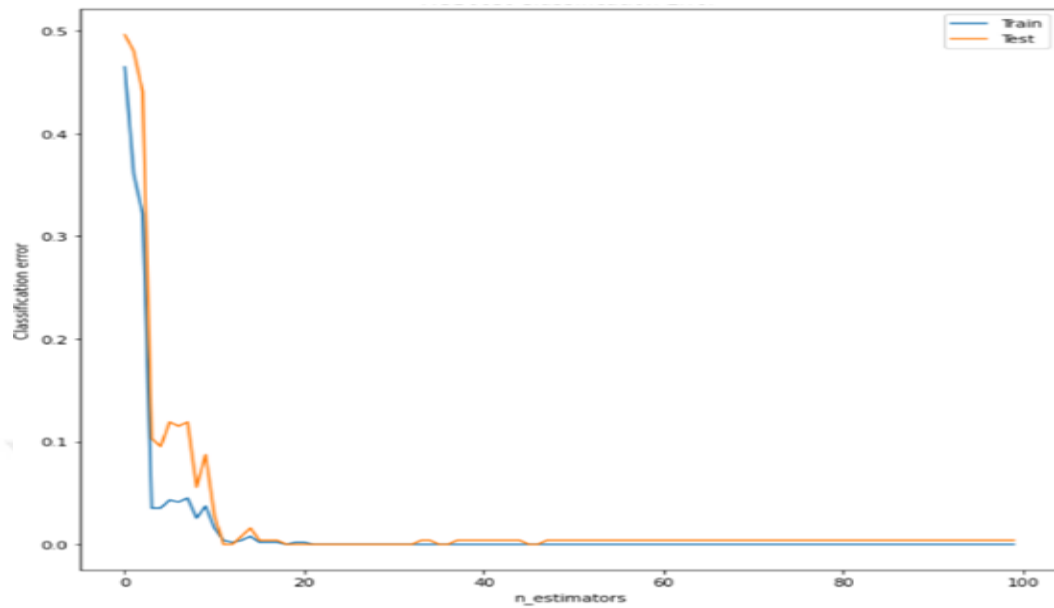


Figure 4.12: Classification Error at 50 % attack density.

As can be observed in figure 4.13 below the level of error rises to 0.12 while there is a 60% attack density, after which it drops to 0.01 and then remains the same during the whole time.

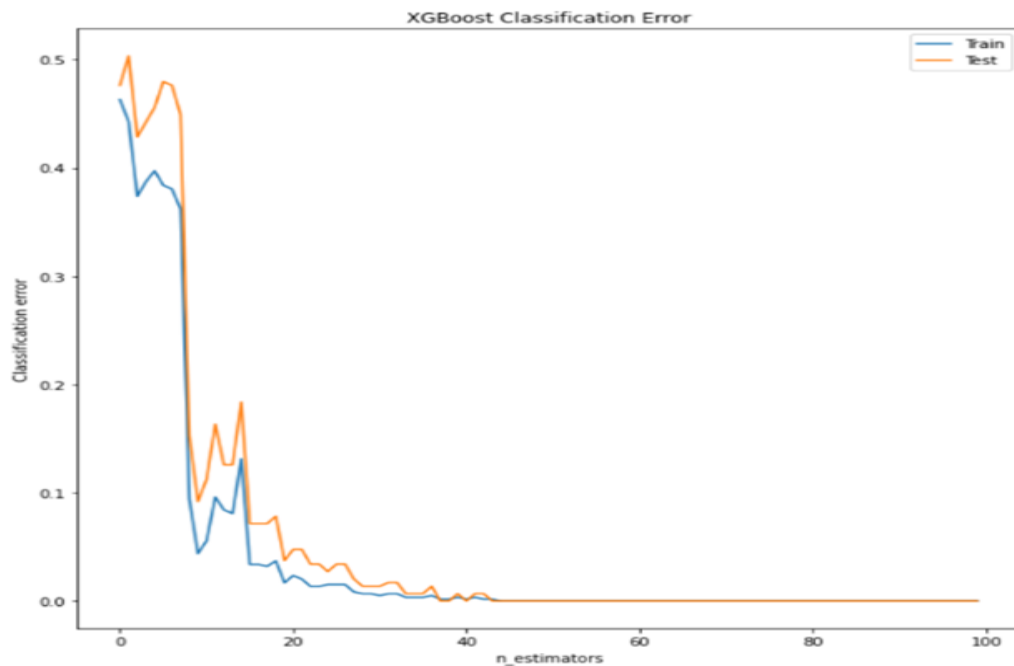


Figure 4.13: Classification Error at 60% Attack Density.

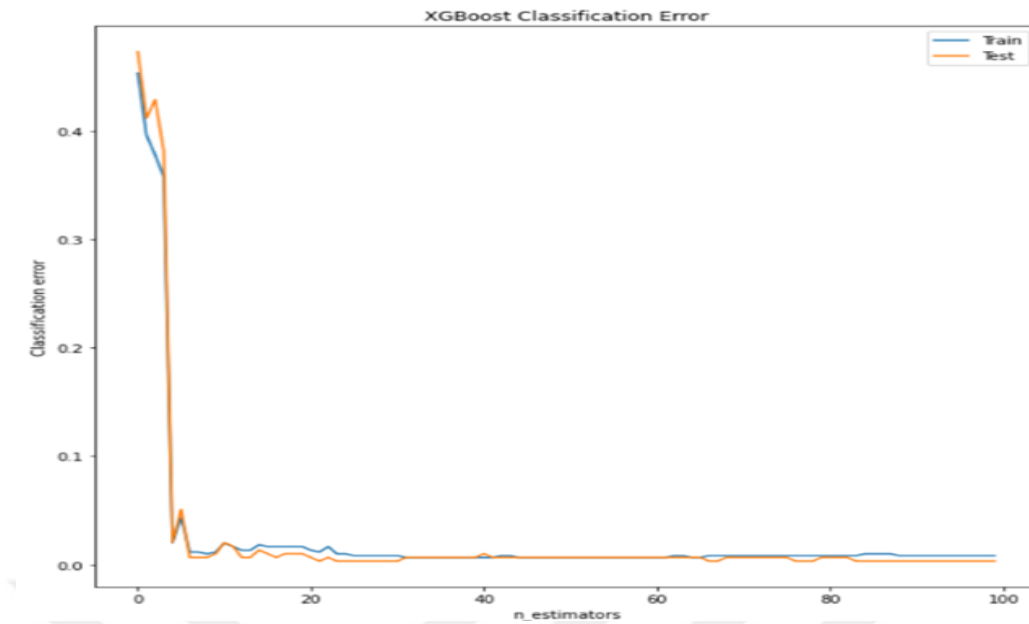


Figure 4.14: Classification Error at 70% Attack Density.

In the end, as can be seen in figure 4.14 which is presented above, the amount of error increases to 0.16 when there is a 70% attack density; after this, it decreases to 0.01 and then stays the same for the whole time period.

We can see from the figures above the error approximately converges to zero value, which is what we need to get the best accuracy according to the vice-versa relationship between the error and accuracy. When the error is low then the accuracy of the classifier is high, and this is what we obtained from our model, which is approximately equal to 99.70 at 20% attack density.

5. CONCLUSION AND FUTURE WORK

The use of misbehavior detection systems is going to become increasingly vital as technological advancements in vehicle communication networks continue to progress. MDS is a necessary and important tool to continue the evaluation of vehicular communication networks. If an MDS is not installed, then any attacks, including DDoS and others, might be launched against vehicles that are linked to the network. These kinds of attacks can prevent communication from taking place by flooding the network with data packets, which can result in mayhem and potentially even accidents.

Following the use of machine learning strategies that were fruitful in MDS, there has been a resurgence of interest in this particular topic. Such as the utilization of neural networks and distributed systems. Recent studies in misbehaviour detection systems (MDS) have concentrated on using machine learning techniques like Random forest (RF), Support vector machine (SVM), Logistic Regression (LR), and Decision Tree Classifier (GBT), which achieved very good results in terms of accuracy, recall, precision, and f1-scores. Studies on Neural networks in [71] which include Multi-layer perceptron (MLPC) and distributed computing in [32], have proven to be beneficial.

The speed in MDS systems is a very important factor in time-critical systems to prevent accidents so there are many MDS systems that depend on distributed systems as in [55] and [40] and they achieved very good results.

In this thesis, we presented Misbehavior Detection System for VANETs based on realistic datasets generated by OMNeT++, Veins, and SUMO. We applied seven different attacks version in the built-in simulation after making some modifications to meet our research needs. Distributed denial of service attacks (DDoS) is specifically examined in these simulations by implementing some of the zombie-attacking cars in the simulation. The dataset generated has a class imbalance problem, but by using SMOTE technique, we were able to overcome this problem, avoiding any bias toward the majority, reducing overfitting, and allowing for the best possible performance. Our approach could cover more attacks using these datasets than it could with older ones. Then we made some of the necessary pre-processing and cleaning operations to our datasets before training and testing. After training and testing the datasets, we achieved excellent results. The log loss error and the

classification error were very small and diverged to zero more than the f1-score, precision, recall, and accuracy were 99.70% percentages that outperformed other ML learning and neural network algorithms.

However, by using only x- and y-coordinates, speed, acceleration, and CO2 emission measurements for every vehicle and running the simulation for only 380 seconds, our XGBoost model could predict and prevent DDoS attacks with 99.70% accuracy.

If cars are able to monitor and convey this information in real-time, then it is possible that they will not be required to employ big data techniques.

Large cities that have a significant quantity of traffic from vehicles might be an exception to this rule. It is possible that it will be essential and helpful to compare the results of a simulation of a huge city with those of smaller ones, such as our study.

This result outperformed all other machine-learning algorithms, however, in future work we intend to develop a model that produces the most accurate metrics in Apache Spark and Cluster systems, to decrease the time delay in misbehavior-detection systems. Even so, a tenth of a second might save a person's life.

REFERENCES

- [1] J. Cui, L. S. Liew, G. Sabaliauskaite, and F. Zhou, “A review on safety failures, security attacks, and available countermeasures for autonomous vehicles,” *Ad Hoc Networks*, vol. 90, Jul. 2019, doi: 10.1016/j.adhoc.2018.12.006.
- [2] A. Alsarhan, A. Y. Al-Dubai, G. Min, A. Y. Zomaya, and M. Bsoul, “A new spectrum management scheme for road safety in smart cities,” *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 11, pp. 3496–3506, 2018.
- [3] A. Alshammari, M. A. Zohdy, D. Debnath, and G. Corser, “Classification Approach for Intrusion Detection in Vehicle Systems,” *Wirel. Eng. Technol.*, vol. 9, no. 04, p. 79, 2018.
- [4] L. Wang, Z. Chen, and J. Wu, “An opportunistic routing for data forwarding based on vehicle mobility association in vehicular ad hoc networks,” *Inf.*, vol. 8, no. 4, Nov. 2017, doi: 10.3390/info8040140.
- [5] S. Waziry, A. B. Wardak, J. Rasheed, R. M. Shubair, and A. Yahyaoui, “Intelligent Facemask Coverage Detector in a World of Chaos,” *Processes*, vol. 10, no. 9, p. 1710, 2022.
- [6] J. Rasheed and R. M. Shubair, “Screening Lung Diseases Using Cascaded Feature Generation and Selection Strategies,” in *Healthcare*, 2022, vol. 10, no. 7, p. 1313.
- [7] K. H. Waseem *et al.*, “Forecasting of Air Quality Using an Optimized Recurrent Neural Network,” *Processes*, vol. 10, no. 10, p. 2117, 2022.
- [8] F. Qu, Z. Wu, F.-Y. Wang, and W. Cho, “A security and privacy review of VANETs,” *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 6, pp. 2985–2996, 2015.
- [9] S. Chadli, M. Emharraf, M. Saber, and A. Ziyyat, “The design of an IDS architecture for MANET based on multi-agent,” in *2014 Third IEEE International Colloquium in Information Science and Technology (CIST)*, 2014, pp. 122–128.

- [10] R. W. van der Heijden, T. Lukaseder, and F. Kargl, “VeReMi: A dataset for comparable evaluation of misbehavior detection in VANETs,” in *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, 2018, vol. 254, pp. 318–337. doi: 10.1007/978-3-030-01701-9_18.
- [11] M. Zissman, “Darpa intrusion detection evaluation.” MIT Lincoln Laboratory, <http://www.ll.mit.edu/IST/ideval/index.html>, 1999.
- [12] “The CAIDA ‘DDoS Attack 2007’ Dataset.” https://www.caida.org/catalog/datasets/ddos-20070804_dataset
- [13] “The AWID2 Dataset.” <https://icsdweb.aegean.gr/awid/awid2> (accessed Oct. 12, 2022).
- [14] C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis, “Intrusion detection in 802.11 networks: empirical evaluation of threats and a public dataset,” *IEEE Commun. Surv. Tutorials*, vol. 18, no. 1, pp. 184–208, 2015.
- [15] “kdd.ics.uci.edu/databases/kddcup99/kddcup99.html”, [Online]. Available: <https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>
- [16] “NSL-KDD dataset”, [Online]. Available: <https://www.unb.ca/cic/datasets/nsl.html>
- [17] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the KDD CUP 99 data set,” in *2009 IEEE symposium on computational intelligence for security and defense applications*, 2009, pp. 1–6.
- [18] J. Song, H. Takakura, and Y. Okabe, “Description of kyoto university benchmark data,” Available link http://www.Tak.com/Kyoto_data/BenchmarkData-Description-v5.pdf [Accessed 15 March 2016], 2006.
- [19] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, “SUMO-Simulation of Urban MObility An Overview.”
- [20] A. Varga and R. Hornig, “An overview of the OMNeT++ simulation environment,” 2008. doi: 10.4108/ICST.SIMUTOOLS2008.3027.

- [21] T. K. Mohd, S. Majumdar, A. Mathur, and A. Y. Javaid, "Simulation and Analysis of DDoS Attack on Connected Autonomous Vehicular Network using OMNET++," in *2018 9th IEEE Annual Ubiquitous Computing, Electronics and Mobile Communication Conference, UEMCON 2018*, Nov. 2018, pp. 502–508. doi: 10.1109/UEMCON.2018.8796717.
- [22] R. Kaur, A. L. Sangal, and K. Kumar, "Modeling and simulation of DDoS attack using omnet++," *2014 International Conference on Signal Processing and Integrated Networks, SPIN 2014*. pp. 220–225, 2014.
- [23] S. Gyawali, Y. Qian, and R. Q. Hu, "Machine Learning and Reputation Based Misbehavior Detection in Vehicular Communication Networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 8871–8885, Aug. 2020, doi: 10.1109/TVT.2020.2996620.
- [24] K. Zaidi, M. B. Milojevic, V. Rakocevic, A. Nallanathan, and M. Rajarajan, "Host-based intrusion detection for VANETs: A statistical approach to rogue node detection," *IEEE Trans. Veh. Technol.*, vol. 65, no. 8, pp. 6703–6714, 2015.
- [25] M. Malekzadeh, A. A. A. Ghani, S. Subramaniam, and J. M. Desa, "Validating Reliability of OMNeT++ in Wireless Networks DoS Attacks: Simulation vs. Testbed.," *Int. J. Netw. Secur.*, vol. 13, no. 1, pp. 13–21, 2011.
- [26] "Ctuns", [Online]. Available: <http://nsl.cs.nctu.edu.tw/NSL/nctuns.html>
- [27] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: a library for parallel simulation of large-scale wireless networks," in *Proceedings. Twelfth Workshop on Parallel and Distributed Simulation PADS'98 (Cat. No. 98TB100233)*, 1998, pp. 154–161.
- [28] N. McKeown *et al.*, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.
- [29] P. Muhlethaler, A. Laouiti, and Y. Toor, "Comparison of flooding techniques for safety applications in VANETs," in *2007 7th International Conference on ITS Telecommunications*, 2007, pp. 1–6.

- [30] F. Ahmad, A. Adnane, V. N. L. Franqueira, F. Kurugollu, and L. Liu, "Man-in-the-middle attacks in vehicular ad-hoc networks: Evaluating the impact of attackers' strategies," *Sensors*, vol. 18, no. 11, p. 4040, 2018.
- [31] K. M. Ali Alheeti, A. Gruebler, and K. McDonald-Maier, "Intelligent intrusion detection of grey hole and rushing attacks in self-driving vehicular networks," *Computers*, vol. 5, no. 3, p. 16, 2016.
- [32] N. Jatou, "DISTRIBUTED NEURAL NETWORK BASED ARCHITECTURE FOR DDoS DETECTION IN VEHICULAR COMMUNICATION SYSTEMS," p. 114, 2021.
- [33] L. Nie, Y. Li, and X. Kong, "Spatio-Temporal Network Traffic Estimation and Anomaly Detection Based on Convolutional Neural Network in Vehicular Ad-Hoc Networks," *IEEE Access*, vol. 6, pp. 40168–40176, Jul. 2018, doi: 10.1109/ACCESS.2018.2854842.
- [34] M.-J. Kang and J.-W. Kang, "A novel intrusion detection method using deep neural network for in-vehicle network security," in *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*, 2016, pp. 1–5.
- [35] N. Ding, H. Ma, C. Zhao, Y. Ma, and H. Ge, "Driver's emotional state-based data anomaly detection for vehicular Ad Hoc networks," in *Proceedings - 2019 IEEE International Conference on Smart Internet of Things, SmartIoT 2019*, Aug. 2019, pp. 121–126. doi: 10.1109/SmartIoT.2019.00027.
- [36] A. R. Gad, A. A. Nashat, and T. M. Barkat, "Intrusion Detection System Using Machine Learning for Vehicular Ad Hoc Networks Based on ToN-IoT Dataset," *IEEE Access*, vol. 9, pp. 142206–142217, 2021, doi: 10.1109/ACCESS.2021.3120626.
- [37] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," Dec. 2015. doi: 10.1109/MilCIS.2015.7348942.

- [38] T. Kai, L. Zhongwei, J. Wenqi, G. Yadong, and T. Weiming, "In-vehicle CAN bus anomaly detection algorithm based on linear chain condition random field," in *2019 IEEE 19th International Conference on Communication Technology (ICCT)*, 2019, pp. 1153–1159.
- [39] J. Grover, N. K. Prajapati, V. Laxmi, and M. S. Gaur, "Machine learning approach for multiple misbehavior detection in VANET," in *International conference on advances in computing and communications*, 2011, pp. 644–653.
- [40] Y. Gao, H. Wu, B. Song, Y. Jin, X. Luo, and X. Zeng, "A distributed network intrusion detection system for distributed denial of service attacks in vehicular ad hoc network," *IEEE Access*, vol. 7, pp. 154560–154571, 2019.
- [41] H. Zhang, S. Dai, Y. Li, and W. Zhang, "Real-time distributed-random-forest-based network intrusion detection system using Apache spark," in *2018 IEEE 37th international performance computing and communications conference (IPCCC)*, 2018, pp. 1–7.
- [42] F. Ahmad, V. N. L. Franqueira, and A. Adnane, "TEAM: A trust evaluation and management framework in context-enabled vehicular ad-hoc networks," *IEEE Access*, vol. 6, pp. 28643–28660, 2018.
- [43] S. Gyawali, Y. Qian, and R. Q. Hu, "A privacy-preserving misbehavior detection system in vehicular communication networks," *IEEE Trans. Veh. Technol.*, vol. 70, no. 6, pp. 6147–6158, 2021.
- [44] K. Sharshembiev, S.-M. Yoo, and E. Elmahdi, "Protocol misbehavior detection framework using machine learning classification in vehicular Ad Hoc networks," *Wirel. Networks*, vol. 27, no. 3, pp. 2103–2118, 2021.
- [45] N. Lyamin, D. Kleyko, Q. Delooz, and A. Vinel, "AI-based malicious network traffic detection in VANETs," *IEEE Netw.*, vol. 32, no. 6, pp. 15–21, 2018.
- [46] N. V. Abhishek and M. Gurusamy, "JaDe: Low Power Jamming Detection Using Machine Learning in Vehicular Networks," *IEEE Wirel. Commun. Lett.*, vol. 10, no. 10, pp. 2210–2214, 2021.

- [47] K. Verma, H. Hasbullah, and H. K. Saini, "Reference broadcast synchronization-based prevention to DoS attacks in VANET," in *2014 Seventh International Conference on Contemporary Computing (IC3)*, 2014, pp. 270–275.
- [48] H. Almutairi, S. Chelloug, H. Alqarni, R. Aljaber, A. Alshehri, and D. Alotaish, "A new black hole detection scheme for VANETs," in *Proceedings of the 6th International Conference on Management of Emergent Digital EcoSystems*, 2014, pp. 133–138.
- [49] A. Haydari and Y. Yilmaz, "Real-time detection and mitigation of DDoS attacks in intelligent transportation systems," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 157–163.
- [50] D. Karagiannis and A. Argyriou, "Jamming attack detection in a pair of RF communicating vehicles using unsupervised machine learning," *Veh. Commun.*, vol. 13, pp. 56–63, 2018.
- [51] M. T. Garip, P. H. Kim, P. Reiher, and M. Gerla, "INTERLOC: An interference-aware RSSI-based localization and Sybil attack detection mechanism for vehicular ad hoc networks," in *2017 14th IEEE Annual Consumer communications & networking conference (CCNC)*, 2017, pp. 1–6.
- [52] B. Xiao, B. Yu, and C. Gao, "Detection and localization of sybil nodes in VANETs," in *Proceedings of the 2006 workshop on Dependability issues in wireless ad hoc networks and sensor networks*, 2006, pp. 1–8.
- [53] G. de Biasi, L. F. M. Vieira, and A. A. F. Loureiro, "Sentinel: defense mechanism against DDoS flooding attack in software defined vehicular network," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6.
- [54] R. Kolandaisamy *et al.*, "A multivariant stream analysis approach to detect and mitigate DDoS attacks in vehicular ad hoc networks," *Wirel. Commun. Mob. Comput.*, vol. 2018, 2018.

- [55] M. Mizukoshi and M. Munetomo, "Distributed denial of services attack protection system with genetic algorithms on Hadoop cluster computing framework," in *2015 IEEE congress on evolutionary computation (CEC)*, 2015, pp. 1575–1580.
- [56] R. F. Fouladi, C. E. Kayatas, and E. Anarim, "Frequency based DDoS attack detection approach using naive Bayes classification," in *2016 39th International Conference on Telecommunications and Signal Processing (TSP)*, 2016, pp. 104–107.
- [57] S. Gyawali, Y. Qian, and R. Q. Hu, "Resource allocation in vehicular communications using graph and deep reinforcement learning," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.
- [58] H. Jiang, Z. He, G. Ye, and H. Zhang, "Network Intrusion Detection Based on PSO-Xgboost Model," *IEEE Access*, vol. 8, pp. 58392–58401, 2020, doi: 10.1109/ACCESS.2020.2982418.
- [59] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.
- [60] J. Wang, M. Xu, H. Wang, and J. Zhang, "Classification of imbalanced data by using the SMOTE algorithm and locally linear embedding," in *2006 8th international Conference on Signal Processing*, 2006, vol. 3.
- [61] S. Bagui and K. Li, "Resampling imbalanced data for network intrusion detection datasets," *J. Big Data*, vol. 8, no. 1, Dec. 2021, doi: 10.1186/s40537-020-00390-x.
- [62] A. Köpke *et al.*, *Simulating Wireless and Mobile Networks in OMNeT++ The MiXiM Vision*. 2008.
- [63] S. Bajaj *et al.*, "Improving simulation for network research," Technical Report 99-702b, University of Southern California, 1999.
- [64] "What is OMNeT++?," 2018. <https://omnetpp.org/intro/>
- [65] A. Varga, "Discrete event simulation system," in *Proc. of the European Simulation Multiconference (ESM'2001)*, 2001, pp. 1–7.

- [66] D. Krajzewicz, “Traffic simulation with SUMO—simulation of urban mobility,” *Fundam. traffic Simul.*, pp. 269–293, 2010.
- [67] J. E. D. Krajzewicz and M. Behrisch, L. Bieker, “SUMO homepage.”, [Online]. Available: <https://sourceforge.net/projects/sumo/>
- [68] D. Zhang and Y. Gong, “The comparison of LightGBM and XGBoost coupling factor analysis and prediagnosis of acute liver failure,” *IEEE Access*, vol. 8, pp. 220990–221003, 2020.
- [69] M. Z. Naser and A. H. Alavi, “Error metrics and performance fitness indicators for artificial intelligence and machine learning in engineering and sciences,” *Archit. Struct. Constr.*, pp. 1–19, 2021.
- [70] M. Z. Naser and A. Alavi, “Insights into performance fitness and error metrics for machine learning,” *arXiv Prepr. arXiv2006.00887*, 2020.
- [71] K. M. A. Alheeti, A. Gruebler, and K. McDonald-Maier, “Intelligent intrusion detection of grey hole and rushing attacks in self-driving vehicular networks,” *Computers*, vol. 5, no. 3, Sep. 2016, doi: 10.3390/computers5030016.