2022

Speed Offset Attack Detection in Vehicular Ad-Hoc Networks (VANETs) Using Machine Learning

Bhuiyan Mustafa Tawheed
University of Windsor

Speed Offset Attack Detection in Vehicular Ad-Hoc Networks (VANETs) Using Machine Learning

By

Bhuiyan Mustafa Tawheed

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada

2022

Speed Offset Attack Detection in Vehicular Ad-Hoc Networks (VANETs) Using Machine Learning

by

Bhuiyan Mustafa Tawheed

APPROVED BY:

E. Abdel-Raheem
Department of Electrical & Computer Engineering

S. Samet
School of Computer Science

A. Jaekel, Advisor
School of Computer Science

August 31, 2022

DECLARATION OF CO-AUTHORSHIP

Co-Authorship

I hereby declare that this thesis incorporates material that is the result of research conducted under the supervision of Dr. Arunita Jaekel. In all cases, the key ideas, primary contribution, experimental designs, data analysis, and interpretation were performed by the author, and the contribution of the co-author was primarily through providing feedback and the proofreading of the published manuscripts. I am aware of the University of Windsor Senate Policy on Authorship, and I certify that I have properly acknowledged the contribution of other researchers to my thesis and have obtained written permission from each of the co-author(s) to include the above material(s) in my thesis. I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my work.

General

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication. I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore,

to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

An integral component of the Intelligent Transportation System (ITS) is the emerging technology called Vehicular ad-hoc network (VANET). VANET allows Vehicle to Vehicle (V2V) and Vehicle to Infrastructure (V2I) communication wirelessly to improve road safety, traffic congestion, and information dissemination. Communication of vehicles in a VANET network is vulnerable to various attacks. Commonly used cryptographic techniques alone are insufficient to ensure and protect vehicle message integrity and authentication from insider attacks. In such cases, additional measures are necessary to ensure the correctness of the transmitted data. Each vehicle in the network periodically broadcasts a basic safety message (BSM) that contains essential status information about a vehicle, such as its position, speed, and heading to other vehicles and Road Side Units (RSU) to report its status. A speed offset attack is where an attacker (misbehaving vehicle) misleads the network by adding an offset value to its actual speed data in each BSM. Such attacks can result in traffic congestion and road accidents; therefore, it is essential to accurately detect and identify such attackers to ensure safety in the network. This research proposes a novel data-centric approach for detecting speed offset attacks using Machine Learning (ML) and Deep Learning (DL) algorithms. Vehicular Reference Misbehavior (VeReMi) Extension Dataset is used for this research. Preliminary results indicate that the proposed model can detect malicious nodes in the network quickly and accurately.

DEDICATION

I would like to dedicate this thesis to my parents for their incredible love and support. This is to appreciate the selfless sacrifices they made and hardships they faced in supporting me throughout this journey and help me reach this stage.

Also, I dedicate it to my elder brother and his family, without their encouragement and unconditional support, nothing would have been easy. And to my entire family for their unconditional affection towards me.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor Dr. Arunita Jaekel and PhD student Muhammad Anwar Shahid, for guidance, support and encouragement throughout my research. I would like to thank my friends and classmates who supported me in boosting up my confidence whenever I needed one. Moreover, I would like to thank my internal reader Dr. Saeed Samet and my external reader Dr. Esam Abdel-Raheem for their support and feedback to improve my thesis. Finally, I would like to thank my parents for providing the confidence and strength to complete my research and for being my pillar of strength.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

VANET	Vehicular ad-hoc network
RSU	Road-side unit
OBU	On-board unit
PKI	Public key infrastructure
BSM	Basic safety message
VeReMi	Vehicular Reference Misbehavior Dataset
C-V2X	Cellular vehicle to everything
KNN	K-Nearest Neighbors
LR	Logistic Regression
AdaBoost	Adaptive Boost
XGBoost	Extreme Gradient Boost
RF	Random Forest
DT	Decision Tree
MLP	Multi Layer Perceptron

CHAPTER 1

Introduction

1.1 Vehicular ad-hoc networks

As technology advances and intelligent cities develop worldwide, Wireless Communication is becoming a new paradigm of efficient data transfer techniques. As a result of the vast range of options it offers, the Vehicular Ad-hoc Network (VANET) has attracted much attention and research. Building transport infrastructure and expanding it does not solve transportation issues. To mitigate these problems, Intelligent Transportation System (ITS) develops a system combining people, roads, and vehicles and applies advanced technologies in transportation. With its real-time, accurate, and efficient transportation system, ITS provides drivers with essential road information and services to alleviate traffic congestion, and accidents [1]. The global status report of 2018 by the World Health Organization (WHO) shows that road accidents cause 1.35 million people's deaths annually and are the 8th leading cause. Also, it is the 1st leading cause of death for young adults and children [2]. Studies show that around 60% of the accidents could be avoided if the driver could be warned of misbehavior in the network even half a second before the mishap [3]. VANET connects vehicles with reliable and vital information that helps ensure their safety on the road while providing a pleasant driving experience. As a result, VANET has become a crucial component of the Intelligent Transportation System. VANET differs from other ad-hoc networks as the network consists of components and infrastructures with more processing capacity, a larger storage limit in data storing infrastructures, a specific vehicle movement path, high dynamic topology, and a vast amount of energy

[4]. Within the vehicles in a VANET network is the On-Board Unit (OBU), which comprises of GPS, antennas, and processors. The Application Unit (AU) executes the program and assists OBUs in communicating. Roadside units (RSU)s placed on the side of roads help in node-to-node communication in the network. Moreover, the Central Authority or Authorization Party registers a node to the network and revokes it in case of misbehavior [5].

Dedicated Short-Range Communication (DSRC) is a technology that supports various applications on vehicular communication that allow medium to short-range high data transfer communication with low latency. These applications help to improve safety, for instance, collision avoidance, lane change warning, etc., and they run on data exchanged between vehicles and roadside infrastructures. According to a U.S. Department of Transportation study, vehicles with DSRC can prevent up to 82% of all accidents in the United States before they occur. DSRC standards and vehicular network communication are supported by Wireless Access in Vehicular Environment (WAVE) which is IEEE 1609 standard protocol [6].

Although DSRC supports various safety applications, it is not convenient for exchanging massive amounts of data. To bridge the gap, Cellular-V2X (C-V2X) is introduced. C-V2X allows faster connectivity and data transfer between vehicles, roadside units, central authority, and cloud-based services [7].

VANET applications are categorized as Comfort applications and Safety applications. Comfort applications give vehicle users a comfortable experience and ensure traffic efficiency. It includes map navigations price and locations of gas stations and restaurants, automatic toll collection, internet connectivity, weather updates, entertainment, and passenger health updates. On the other hand, Safety applications try to avoid accidents by delivering safety information to the desired receiver. There are three different kinds of messages that Safety applications give out: Information messages (IM) that consist of the toll collection point, speed limits, etc., Assistance messages (AM) that consist of critical information like cooperative collision avoidance and lane switching, lastly Warning messages (WM) that include information like lousy road condition and traffic signal ahead [8].

VANET communications communication can be of different types, such as Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), Infrastructure-to-Infrastructure (I2I), and Vehicle-to other devices (V2X). Communication over the VANET network includes sharing sensitive information with vehicles promptly. This makes the network prone to different types of attacks and misbehavior, which need to be detected and verified to ensure the network's correctness, integrity, authenticity, and confidentiality.

Moreover, the VANET network is susceptible to different types of attackers. Each of these attackers has other intentions. They also have different statuses and roles in the network; some are authorized members, and some are not. Chapter 2 will further discuss details on types of attackers.

This research addresses the detection of Speed Offset attacks. In this attack, the malicious vehicle adds an offset value and alters the actual speed data in each Basic Safety Message (BSM). This can lead to traffic congestion and even accidents. Cryptographic techniques can secure message integrity but not correctness; as a result, cryptographic methods are insufficient to ensure network security. Advanced detection methods are required to detect malicious vehicles delivering misleading information across the network.

1.2 Motivation

Due to the highly dynamic topology, infrastructure-less architecture, and time-sensitive information sharing, communication of nodes in the VANET network is crucial and prone to different security threats. Moreover, vehicles in the network are responsible for spreading safety-related information, so any alteration in this transmitted data by malicious nodes can cause grave harm to the network. Confidentiality, authenticity, and integrity of these data are essential to ensure soundness in the network. Defense mechanisms like Public Key Infrastructure, symmetric and hybrid methods, certificate revocation methods, and ID-Based Cryptography methods are used to ensure message authenticity [9]. Public Key Infrastructure (PKI) methods use cryptographic

techniques to authenticate vehicles in the network. Temporary digital certificates are issued by the Certification Authority (CA) to the communicating nodes. If misbehavior is detected, the certificate is revoked and added to the Certificate Revocation List (CRL) [10]. However, it is pretty challenging to maintain such lists due to high mobility. As these cryptographic techniques give message authenticity and cannot ensure message integrity and correctness, concrete misbehavior detection schemes must serve the cause.

Each authenticated vehicle broadcasts Basic Safety Messages (BSMs) to the network, containing information about the vehicle's position coordinates, time, speed, etc. A speed offset attack is where a malicious node adds a fixed or random offset value to its speed data to alter the actual information passed in each BSM. This kind of attack causes an illusion for legitimate vehicles and can result in traffic congestion and accidents. The attackers are authenticated vehicles, so only cryptographic techniques are insufficient in identifying them.

Although several researchers in the past have adapted misbehavior detection using Machine learning approaches, very few notable works are there for identifying speed offset attacks in VANET. In this research, we propose efficient misbehavior detected model using Machine learning with an efficient and high detection rate.

1.3 Problem Statement

VANET networks are reliable and efficient information distribution and communication between network vehicles. Nevertheless, the network is highly likely to be attacked by malicious nodes who aim to damage it for their benefit. Speed offset attackers alter the actual speed of a vehicle and cause a discrepancy in the network. As these attackers are often authenticated users, cryptographic techniques fail to serve their purpose. The novelty of this work is that not much noticeable work is there that uses Machine learning approaches for speed offset misbehavior detection in VANET. Previous works mostly revolved around heuristic approaches like integrity and plausibility checks. Moreover, the earlier works relied on vehicles to store and share the

misbehavior detection system.

1.4 Solution Outline

In this research, to solve the abovementioned issue, we propose a Data-centric approach of using a machine learning-based misbehavior detection system to detect misbehaving nodes in the network with a high classification rate. Each vehicle sends out BSMs periodically. Based on the BSMs, a car is classified as malicious or legitimate by the model installed at RSU. An extension of the first public extensible dataset available in this field: VeReMi Extension Dataset (Vehicular Reference Misbehavior Extension Dataset) [11] is used in this research to train and evaluate the proposed approach. The model is trained on the dataset with low traffic densities. The first stage of the solution is data collection and preprocessing, where the correct information from the Ground Truth file is mapped together with the false information from the Log files. The resulting dataset is preprocessed for being fed to the model for classification. The second is the classification stage. The dataset is provided to multiple machine learning algorithms and a deep learning algorithm for detecting the vehicles as legitimate and malicious. Additionally, we propose implementing a time parameter that will assist in finding the best trade-off between the accuracy of a model and the time taken to run the model. This way, we can determine the most efficient algorithm for misbehavior classification. In our approach, we propose to create an efficient and time-sensitive misbehavior detection model for Speed offset attack detection in VANETs.

1.5 Thesis Organization

The remaining outline of this thesis is as follows: chapter 2 includes an overview of fundamental concepts of VANET and Speed offset attack, along with a literature review of related work in misbehavior detection using machine learning approaches. Chapter 3 contains an outline of the proposed methodology and a brief discussion of

the VeReMi dataset, followed by chapter 4, including an experimental setup and a discussion of the results. In the end, chapter 5 gives a conclusion followed by possible future work on the proposed methodology.

CHAPTER 2

Literature Survey

2.1 Overview of VANET

VANET supports various applications to improve traffic conditions through collaborative communication. These applications are divided into Safety and Comfort applications, as demonstrated in Figure 2.1.1.

Safety applications are further divided into Road safety applications and road efficiency applications, whereas Comfort applications consist of commercial and information services [12]. Road safety applications ensure to decrease in the chances of road accidents and damages caused by accidents. Moreover, these applications can provide drivers with warning messages about potential future accidents. This application type is further divided into three subcategories [13]. Firstly, when an RSU detects a probable collision between vehicles, it sends out warning messages to the OBU of vehicles through Collision avoidance applications. This safety application gives out intersection collision warnings, pre-collision warnings, lane change alerts, and alarms for dangerous locations to make drivers cautious. Secondly, the Traffic sign notification applications warn drivers about curve speed and traffic signal violations. Lastly, Incident management specializes in tackling the effect of an accident by giving out emergency vehicle alarms and post-collision warning messages.

The other division of Safety application is the Road Efficiency application which aims to improve traffic conditions by monitoring vehicles and road conditions. Road efficiency is divided into two branches. The first is a traffic management application, which consists of traffic flow information such as intelligent traffic control, traffic-

free tolls, speed control of vehicles, and route guidance for improved driving [14]. Secondly, Traffic monitoring applications notify the drivers of unusual situations by monitoring road conditions and tracking vehicles.

The Comfort applications provide drivers and passengers with entertainment and information services. Entertainment applications give the users on road Wifi access points and mobile cellular data networks for recreational services like internet surfing and online gaming. Whereas, Background information applications provide users with services like advertising business locations and information about local attractions and gas stations.

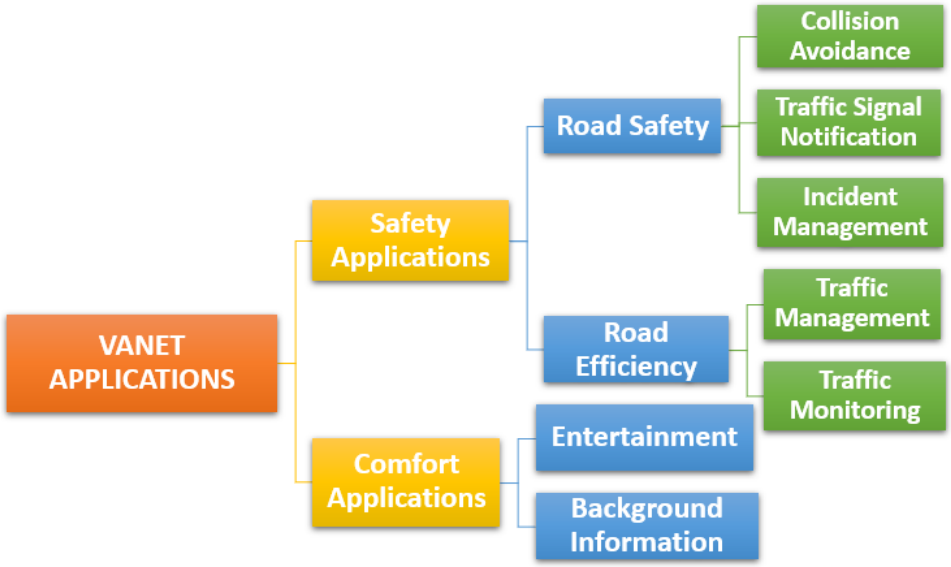

Fig. 2.1.1: Applications in VANET

Vehicles in the VANET network move freely with high speed, which results in its highly dynamic topology. Unlike MANET, the infrastructure-less architecture of VANET allows each node in the network to move and communicate independently [15]. Coverage of VANET network ranges from vast area network to local area network. Moreover, the VANET network does not have strict limitations on battery power and storage capabilities, which allows it to host a considerable number of vehicles in high traffic densities and also cover significant geographical areas [16]. The covered area consists of both uniform and non-uniform regions. The Uniform region is where the vehicles share their route, direction, and speed for a long time with all

other nodes in the network. The highway is an example of such a region, where the attributes of each vehicle stay nearly constant for a significant time. On the other hand, non-uniform areas are where the vehicles' paths, speed, and direction change frequently and are not shared throughout the network, such as local roads [17].

2.1.1 Types of Communication

Each vehicle in a VANET network acts as a node in charge of safely sending, receiving, and broadcasting data packets throughout the network. Vehicles in the network are equipped with On-Board Unit (OBU) that facilitates communication with other vehicles as well as roadside units (RSUs), and the Application Unit (AU) executes the OBU's computational programs [18]. Moreover, the vehicles contain a Global positioning system (GPS), allowing the OBUs to determine the current vehicle location. Road side unit (RSU) acts as the central point of communication for nodes in the network [19]. VANET facilitates different types of communication:

1. **Vehicle-to-Vehicle (V2V)**: This type of communication allows a vehicle to establish direct communication with another vehicle without the interference of any other infrastructure. This communication is used for safety, security, and dissemination applications and also allows a vehicle to broadcast data packets to all vehicles in its range [20].

2. **Vehicle-to-Infrastructure (V2I)**: This communication allows vehicles to communicate with nearby infrastructures such as RSUs and Central Authority to establish a connection and provide current status information. Infrastructures such as RSUs can also communicate with vehicles in the network, e.g., to broadcast safety messages and provide services across the network.

3. **Infrastructure-to-Infrastructure (I2I)**: Infrastructures communicate with each other to keep the nodes in the network within a range updated with the required information.

4. **Vehicle-to-everything (V2X)**: This is the communication between vehicles

and other devices in the network such as mobile phones and internet connectivity devices.

2.1.2 Challenges in VANET

VANET network deals with transmitting and generating real-time traffic information. The prime objective of functioning in this domain is ensuring road safety through continuously sharing information among vehicles. The disseminated information's lack of accuracy, efficiency, and reliability can have adverse effects. Ensuring smooth operation in this infrastructure-less and dynamic architecture can be pretty challenging. Several such challenges are listed below:

- **Security**: Ensuring security in the network is the most crucial challenge as VANET is a robust network providing safe and secure traffic analysis [21]. The most crucial security challenges of VANET are [22]:

    1. **Data consistency**: Inconsistency in data shared among vehicles from malicious and legitimate vehicles can result in accidents. Also, the network must be adaptive to instant errors in the transmitted data.
    2. **High mobility**: Vehicles in VANET are fast and continuously moving. Although it has high processing and storage capacity, a comparatively less complex algorithm is required to process data in VANET.
    3. **Key management**: Authentication of nodes depends on the key-based architecture. Ensuring the safe handling of these keys is very important in such networks.
    4. **Revocability**: Malicious vehicles should be revoked from the system on identification.

- **Scalability**: Nodes are open to leaving and joining the network at any time. The routing protocols must ensure service to all the vehicles functioning at different hours of the day and avoid any network congestion [15].

- **Efficient Routing and Network Management**: As VANET is a network with changing topology, deploying an efficient routing protocol that ensures minimum delay in packet sending, maximum system capacity, and less complexity for this changing topology is challenging [23]. Moreover, the network must use an adaptive routing protocol that can handle varying network densities in different scenarios [20].

- **Error Tolerance and Latency Control**: In a VANET network, the nodes are constantly sharing real-time data. The time taken in this communication is quite fast. These data packets can contain errors, so the network must prepare to deal with these errors without any delay. Moreover, the network must comply with data communication rapidly. Any latency in real-time data sharing between vehicles can result in grave accidents [21].

- **Dynamic Topology and Efficient Routing**: Vehicles in VANET are fast and continuously moving. RSUs and vehicles are considered network nodes, while one is stationary and the other can reach speeds of over 100 kilometers per hour. This extremity in the difference of node velocity causes a high mobility challenge for processing the joint wireless communication [16]. Moreover, the VANET network has a highly variable network density. The number of vehicles in a specific area per second can vary widely depending on the region and time. For instance, urban areas will have more density of nodes than rural areas, and the same region has highly variable node density based on time of the day [20].

  High mobility, high node velocity, and variable node density make VANET a dynamic network with changing topology. To address the challenges of changing topology VANET network must offer an adaptive Quality of Service that can continue to provide high connectivity time and low latency with the available bandwidth and vehicle density [5]. Additionally, it is challenging for a VANET network to implement an efficient routing protocol that will ensure minimum delay in packet sending, maximum system capacity, and less complexity for this changing topology [23].

2.1.3 Security Requirements in VANET

VANET provides its users with wireless services and information throughout the network. Ensuring the truthfulness and authenticity of these messages is crucial, as failing to do so can result in catastrophes. Moreover, the information shared through this infrastructure-less network is real-time and sensitive, making it lucrative to attackers. To keep the network safe from malicious attackers, security requirements that should be maintained are as follows [24]:

- **Authenticity**: The authentication process verifies and ensures that only legitimate vehicles are enrolled in the network and that the message they send is legit. Authentication also checks that the data is transferred between a legitimate sender and receiver pair [18]. Ensuring Authenticity in the network is one of the first steps toward network security. Moreover, Cryptographic techniques are adapted to secure the Authenticity of the network through unique key generation for the sender vehicles. Authentication is violated by critical attacks like Sybil Attack, Data Replay Attack, and Certificate Replication Attack. In the Sybil attack, the attacker creates ghost identities and disguises itself to mislead the legitimate vehicles by sending out false information [25]. In a Data Replay attack, the attacker alters the timestamp of a message, which confuses the network [26]. A certificate replication attack is where the attacker manages to get the private or public unique key of a legitimate vehicle and sends messages in the network by impersonating itself [27].

- **Integrity**: One of the essential security requirements is integrity. The integrity of a message is ensured when the content of a message from a sender to a receiver remains intact and unaltered. Incorrect messages can come from authenticated nodes, or the correct message can be altered by malicious nodes [28]. Attackers change the original message and replace it with false content to hamper the integrity of the network. Examples of integrity attacks are the Position falsification attack, Speed Offset attack, timing attacks, and Message alteration attacks. In a Position falsification attack, the malicious node sends

out false position information to the network to delude the legitimate nodes. Speed offset attack is similar, but the attacker plays around with speed information in this case. In a Message alteration/deletion attack, the attacker either alters or deletes an authentic message before reaching the receiver [29]. Lastly, in Timing attacks, the emergency messages are deleted to cause a hamper to the network [30].

- **Confidentiality**: Vehicles in the network send and receive messages both in the form of V2V and V2I communication. Therefore, the system must protect both the identity of an authorized vehicle and the authentic messages from them from outsiders. Access to the messages must be only in the hands of authenticated users. In confidentiality attacks, malicious nodes breach the network, obtain personal information about legitimate vehicles, and silently overhear and observe the network [30]. Eavesdropping attacks is a confidentiality attacks by mostly passive attackers.

- **Availability**: Facilities and operation of the network must be available to all authorized vehicles at all times. Malicious users attack the availability of the network by restricting legitimate users from accessing the network [4]. Availability attacks include Denial of Service (DoS), spamming, broadcasting, and jamming attacks. In a DoS attack, the attacker congests the communication medium by flooding unwanted information, making the resources unavailable to legitimate users [31]. A jamming attack occurs at the physical layer, similar to DoS, where the attacker jams the network with disruptive signals [32].

2.1.4 Attackers in VANET

The impact of each attack on the VANET network depends on the attacker's intention. An attacker is a node whose main motive is to hamper the functionality of the running network. A malicious node can disrupt the network for several reasons: to benefit from a network it is not part of, to eavesdrop on confidential information, or to disturb the ongoing functionalities. Based on their characteristics and potential, attackers are

divided as follows [4]:

1. **Internal Attackers**: They are the authorized network members who leave a more substantial impact on the network's functionality than the external attackers. Their motive is either personal gain or simply disrupting the network.

2. **External Attackers**: They are not part of the network but try to enter it to gain benefit by either pretending to be a legitimate node or through other attacks.

3. **Active Attackers**: These attackers try to alter the network through direct involvement. They send out malicious signals and data packets directly throughout the network. They impose a higher impact than the passive attackers.

4. **Passive Attackers**: They intend to observe the network without direct involvement for their benefit.

5. **Rational Attackers**: They attack the network for their interest; hence they are often easier to detect.

6. **Malicious Attackers**: On the contrary, they have no personal benefit; instead, they want to hinder the operation of a running network.

7. **Global Attackers**: These attackers have a wide-range reach and can manipulate the operation of all the nodes in the network.

8. **Local Attackers**: Their control range is limited, and they have lesser entities to manipulate.

2.2 Overview of Machine Learning

Artificial Intelligence introduced a branch called Machine learning which makes executing complex jobs faster and simpler using statistical learning [33]. Machine learning contributes to solving real-world problems in various fields. It is being extensively

implemented in essential domains like e-commerce, agriculture, natural language processing, healthcare, robotics, facial and image recognition, and many more [34]. Accordingly, the field of VANET is no exception and is utilizing Machine learning vastly for attacks, intrusion, and misbehavior detection in the network. Machine learning is categorized into four types [35]:

1. **Supervised Learning**: In this learning, the model is trained with labeled data. Later the model is expected to make predictions on the unlabeled data by learning from the labeled data. Supervised learning involves finding a relationship between a dataset's independent and dependant features.

2. **Unsupervised Learning**: In this case, the model learns from a set of unlabeled data. It uses its capability to learn from unlabeled data and find patterns to solve a problem.

3. **Semi-supervised Learning**: The model is trained using both labeled and unlabeled data. The model learns and predicts by using this combination of data.

4. **Reinforcement Learning**: Unlike the other learning techniques, the algorithm, in this case, learns, adapts, and predicts from its environment without providing any data. The model receives a prize for every successful learning with nothing to lose in case of failure.

2.2.1 Basic Machine Learning Concepts and Terminologies

- **Model**: A machine learning model is an algorithm that learns and solves given problems.

- **Dataset**: It is the information that the machine learning model learns from and predicts or executes its learning upon. Dataset is one of the most critical components for building a machine learning model.

- **Training and Testing Set**: The input dataset is divided into training and testing datasets. The training data is used to train the machine learning model, and the testing dataset is used to test the predictions or learnings of the model.

- **Features**: These are the attributes of the dataset. Usually, the features are represented in columns and are often known as data objects. They each represent a dataset property that contributes to solving the problem.

- **Data Pre-processing**: The raw version of the dataset might contain irrelevant information. There can also be duplicate entries and noise in the dataset, which will hinder the accuracy and quality of the prediction process of the machine learning model.

- **Cross-validation**: A statistical method where the dataset is divided into groups (folds) for training and evaluation of the machine learning model. In each iteration, a different subgroup is used to validate the results, while the remaining are used to train the model in the next iteration. The average of each prediction is used to calculate the overall prediction of the model using the provided dataset [36]. One of the most common forms of cross-validation is k-fold cross-validation, where 'k' denotes the number of divisions of the training dataset.

2.2.2 Classification Algorithms

Classification is a category of supervised machine learning where the output data points are classified as members of different classes [37]. Classification is used to identify group memberships for data points [38]. Each data instance is accurately predicted and assigned to a target class in the classification process. Algorithms for classification tasks are classifiers that learn from the training data and predict categorical labels for each data point. A classification algorithm develops a model by learning from the labeled training data and predicts the class label for the unlabeled test dataset [39]. In VANET, machine learning can classify incoming vehicles as

legitimate and misbehaving based on the input Basic Safety Messages. There are mainly two types of classification [39]:

1. **Binary Classification**: The predicted output data can belong to either of the two classes in binary classification.

2. **Multiclass Classification**: In this classification task, more than two output classes are possible.

This research is based on supervised binary and multiclass classification using labeled and structured data. The predicted output will classify an input vehicle as either an honest and legitimate network member or a malicious attacker.

Following are brief descriptions of the classification algorithms used in this research. For our work, we implemented K-nearest neighbor, Logistic regression, Adaptive Boost, Decision tree, Random forest, Extreme Gradient Boost, and Multi-layer Perceptron classifiers for classification.

2.2.2.1 K-Nearest Neighbours Algorithm

K-Nearest Neighbour is a widely used supervised Machine learning classification algorithm that is a simple yet effective case-based learning model. KNN works well with balanced and imbalanced datasets [40]. The algorithm chooses a target point and determines the distance between that target variable and all other data points. It selects the 'K' number of nearest neighbors to the target point. It chooses the label for the target data point based on the similarity and popularity score of the labels of k nearest neighbors [41]. The query point gets assigned a label based on the majority voting technique of its neighbors [42]. This algorithm can use Euclidean, Manhattan, Mikowski, and Hamming distance functions to calculate the distance between data points [40].

2.2.2.2 Logistic Regression Algorithm

Logistic regression is a supervised statistical method widely used for binary classification in machine learning [43]. This algorithm classifies a particular class using

probability. The algorithm predicts the occurrence probability of binary class values using a logit function.

2.2.2.3 Adaptive Boost Algorithm

Adaptive Boost (AdaBoost) is a robust ensemble classifier built by combining several weaker classifiers. Ensemble classifiers are built upon multiple individual classifiers. AdaBoost generates member classifiers, and the output of the ensemble classifier depends on the diversity between the base classifier and each member classifier's performance. The algorithm is adaptive as it adjusts the weight of member classifiers to rectify incorrect predictions in each iteration. Our research uses a Decision tree with a single split [44].

2.2.2.4 Decision Tree Algorithm

The decision tree is a valuable and powerful supervised Machine learning algorithm for data mining of linearly inseparable data [45]. The algorithm uses the divide and conquer approach for classification. The dataset is transformed into a tree-like structure where the root node represents the best features of the dataset, and the branches carry out classification [46]. The two main components of a decision tree are the decision node representing the conditions the tree uses to operate and the leaf nodes representing the decision node's decision with a numerical value. Splitting, stopping, and pruning are the critical components for building a decision tree model [47]. Moreover, the decision tree is suitable for handling large, noisy, and complicated datasets and does not require pre-processing. The major drawback of this algorithm is that it can easily get overfitted.

2.2.2.5 Random Forest Algorithm

Random forest is a supervised machine learning algorithm suitable for classification and regression problems [48]. This algorithm consists of a collection of Decision trees that it trains to obtain the final prediction result. The best solution is determined using the ensemble method in which the algorithm returns the class getting the majority

vote count. Random forest is not only fast, scalable and robust but also overcomes the shortcoming of the Decision tree algorithm of being overfitted by large and complex datasets [45].

2.2.2.6 Extreme Gradient Boost (XGBoost) Algorithm

XGBoost is an ensemble approach of tree boosting, in which new models are added to improvise the errors of the existing models to make a combined final prediction. XGBoost is a library that implements the gradient boosting decision tree algorithm. XGBoost goes by the name of gradient boosting as it utilizes the gradient descent algorithm to minimize the loss of existing models. XGBoost is widely known for its unparallel execution speed and model performance. Moreover, it is one of the most scalable algorithms for both regression and classification for all purposes. This algorithm uses a novel tree learning algorithm for sparse data handling, and its quick learning and fast model exploration feature is due to its parallel and distributed computing system. Due to these properties, XGBoost is said to be a ten times speedier algorithm than its competitors. It can be implemented on datasets with billions of entities on a distributed system [49].

2.2.2.7 Multi-layer Perceptron classifier (MLP) Algorithm

Multi-layer perceptron (MLP) is a feed-forward neural network composed of input, output, and hidden layers. The input layer takes the required data to be processed, and the output layer performs the required classification and prediction tasks. The arbitrary number of hidden layers between the input and output layer increases the model's learning capability and enhances the overall model accuracy. The hidden layers are said to be actual algorithm engines [50]. Unlike a single-layer perceptron, a multi-layer perceptron solves non-linearly separable problems and can approximate any continuous function [51]. The MLP algorithm flows data from the input to the output layer. The neurons in each hidden layer are trained and tested using the backpropagation of the algorithm. The usefulness of MLP is in the domain of recognition, prediction, classification, and approximation.

2.3 VeReMi Extension Dataset

Both the European Telecommunications Standards Institute (ETSI) and Institute of Electrical and Electronics Engineers (IEEE) imposed vehicular Public Key Infrastructure (PKI) for digitally signing packets before sharing over vehicles in a Cooperative Intelligent Transport Systems (C-ITS). The aforementioned cryptographic technique ensures the sender's authenticity but not message correctness [11]. As a result, many published researchers have been developing a Misbehavior detection system for C-ITS. But there is a shortage of publicly available reliable datasets, where the VeReMi dataset bridged the gap. However, the original VeReMi dataset had shortcomings and room for improvement in the number of attacks and physical error model. The creators of the VeReMi extension dataset extend the original VeReMi dataset by making several additions. They added a set of more complex and dynamic attacks, implemented a realistic sensor error model to the vehicle's physical layer, and implemented a simple fusion detection technique with some local plausibility detectors. Moreover, to create a benchmark for future researchers, the creators of this dataset provided the results of running and verifying the new dataset on their misbehavior fusion mechanism.

The dataset was generated using the VEINS extension framework called Framework for Misbehavior Detection (F2MD). This framework is reproducible and detectable for multiple MBD use cases. VEINS [52], the open-source simulator for Inter-Vehicular Communication, is based on OMNET++ [53] and SUMO [54]. OMNeT++ is used for network simulator building and is a C++ simulation library. On the other hand, SUMO is known for simulating road traffic.

Moreover, this dataset is built upon the open-source synthetic traffic scenario Luxembourg SUMO Traffic (LuST) [55]. The LuST scenario is validated with actual traffic data from the VehicularLab of the University of Luxembourg. The subsection of the LuST network used in this research had a traffic size of 1.61 square kilometers and a maximum density of 67.4 vehicles per square kilometer.

To provide a more realistic dataset with real-world field tests, the creators of this

version of VeReMi add sensor error models to the primary data fields:

1. **Position Error**: Multiple position error detection systems have varying precision levels. Firstly, the Global Positioning System (GPS) has different precision limits for the open sky and urban areas [56]. However, in this work, the researchers implemented an internal correction system on the On-Board unit of every vehicle [57].

2. **Velocity Error**: Error from velocity is proportional to the speed of the vehicles. The error is calculated from the wheel spin of vehicles, having an average error of 0.05 meters per second [58].

3. **Acceleration Error**: The acceleration error model is inferred from the error readings of the vehicle's velocity.

4. **Heading Error**: The vehicle's heading can be calculated using a magnetic compass or through inference from successive positions. The magnetic compass's accuracy depends on the device's quality; conversely, inference from successive positions depends on the vehicle's velocity. Results show that when the vehicle has a higher speed, successive position heading is more accurate and vice versa.

For the misbehavior model, the researchers expanded the original VeReMi dataset by adding several threatening attacks to the model. Moreover, they distinguished between malfunctions occurring from faulty infrastructures and sensors and actual attacks, which are intended malicious behaviors of vehicles.

The JSON encoded generated dataset consists of the message's type, receive and send time, sender and receiver IDs, speed, position, acceleration, and heading information of a BSM shared by a vehicle. The dataset was divided into two parts based on hours of the day: The first being the Rush hour time (7 am to 9 am) and the second being the Low traffic hour time (2 pm to 4 pm). Both the subsets are from the Luxembourg network simulation and consist of ground truth files and files with received message data. Moreover, they set the misbehavior attacker penetration rate for all attacks to 30%.

Lastly, to set a benchmark for future researchers, the creators of VeReMi ran their newly generated dataset on a simple detection algorithm based on plausibility and consistency checks of received messages. The checks are done on Position, Speed, Heading, and Acceleration data fields is comprised of 1) Absolute plausibility, 2) Temporal Consistency, 3) Relative consistency, 4) Consistency for Kalmar Filter, 5) Two vehicle's overlap, 6) Beacon Frequency compliance, (7) Sudden appearance plausibility and (8) Transmission range plausibility [59] [60].

Its plausibility check must reach a minimum threshold to be detected as a misbehaving vehicle. The output of the detection mechanism is represented using Accuracy, Precision, Recall, and F1-Score detection metrics. The results show that the outcome is widely dependent on the type of attack rather than the time of the day and vehicle density.

2.3.1 Attacks in VANET

The researchers of the VeReMi extension dataset incorporated and implemented their developed dataset on several different types of attacks and malfunctions [61], such as Position falsification attacks, Speed Offset attacks, Data Replay attacks, Denial of Service attacks, and others. In this thesis, we focus specifically on speed offset attacks, which are discussed in detail in section 2.3.1.1. Nonetheless, here we are listing some common attacks faced by the VANET network:

1. **Delayed Message Attack**: In this attack, the message contents remain unaltered, but the time of receipt of the message is delayed either intentionally by a misbehaving node or due to the malfunction of the network [62].

2. **Data Replay Attack**: In this attack, the malicious node replays an earlier message to create an unwanted panic in the network [63]. It is difficult to identify such attackers in cases like hit-and-run, especially if the attackers adapt the Sybil mode [64].

3. **Denial of Service Attack**: In this attack, the attacker creates congestion by flooding the network with messages exceeding the IEEE and ETSI standard

limit [11]. Consequently, the whole network starts misbehaving and denies services to legitimate users. Denial of service attacks is categorized as Network Mode attacks (Fabrication attacks, Data Alteration attacks, and Data Replay attacks) and Application Mode attacks (Sybil attacks and Message Suspension attacks) [65].

4. **Sybil Attack**: It is one of the most harmful attacks, where the attacker creates multiple pseudonyms and claims that each fake identity is a legitimate network node. Through impersonation, the malicious nodes create an illusion in the network by claiming to have different positions simultaneously [66].

5. **Position Falsification Attack**: It is one of the data alterations attacks where the attacker node misleads the network by altering the existing data and sending out incorrect position information. These attacks are generally generated by authenticated vehicles of the network, and hence cryptographic techniques are insufficient for detecting them [67].

2.3.1.1 Speed Attacks in VANET

This research primarily focuses on Speed Offset attacks in VANET. In the VANET network, each node shares its updated whereabouts with its neighboring nodes within a range. Nodes share this information in the form of Basic Safety Messages (BSMs), which are transmitted periodically. Authenticated users digitally sign these BSMs containing vehicle speed, position, direction, and transmission time. The authenticated senders digitally sign these BSMs in order to make sure that they have not been altered. However, the sender itself can send false messages in the BSM. Malicious nodes populate the BSMs with incorrect speed information that can mislead legitimate members and result in serious accidents. Speed attack is also considered a malfunction that can occur from an erroneous OBU or failure in a physical sensor [11]. Longitude and latitude fields of the BSMs are hampered due to such failures in infrastructure, which can result in legitimate nodes receiving false speed vector information. These attacks are known as Speed Attacks, in which incorrect speed

information is populated in BSMs.

Speed attacks cannot be detected using cryptographic techniques as these techniques cannot ensure message correctness. Additional misbehavior detection models are needed to detect such attacks. Speed attacks hamper the message's data integrity by altering the actual speed information contained in each BSM. Speed attacks are divided into four categories which are listed below:

1. **Constant Speed Attack**: In this attack, the malicious nodes send a constant speed value in their BSMs to the neighboring vehicles. By doing so, the attacker pretends to be moving with the same speed which misleads the neighboring vehicles in the network.

2. **Random Speed Attack**: Similar to Constant speed attacks, the attacker alters the BSM's actual speed data by entering random speed data simultaneously in the network. The legitimate nodes get confused as each BSM contains a new speed value.

3. **Constant Speed Offset Attack**: The malicious node adds a fixed offset value with the actual speed data and transmits it to the network. This type of attack is difficult to detect as a fixed amount and vehicle alter the speed data seems to be behaving normally.

4. **Random Speed Offset Attack**: Attacker vehicle adds a random offset value simultaneously to the actual speed information of each BSM. This type of attack is difficult to detect, as the constant speed offset attacks slightly alter the accurate speed information.

2.4 Literature Review

Several methods for anomaly-based intrusion detection were proposed in previous works, such as hashing pseudonyms to common values, RSSI-based analysis, fuzzy clustering, and Machine learning [68] [69]. Cryptographic techniques such as PKI are

widely used to digitally sign packets before sharing in a Cooperative Intelligent Transport Systems (C-ITS). Despite these cryptographic frameworks ensuring the message is authenticated, message correctness is vulnerable to being altered by attackers [70]. An additional misbehavior detection technique is required to solve the message legitimacy issue. Recently, many researchers have used machine learning for misbehavior detection in VANET. Machine learning is a data-centric approach that can optimize a network by detecting misbehaving vehicles in a dynamic network such as VANET [71]. Some implementation of machine learning approaches in detecting misbehavior detection is discussed in the following section.

2.4.1 Machine Learning Based Misbehaviour Detection in VANET

So et al. [72] proposed a model that extracted feature vectors from each BSM using plausibility checks and combined them with Supervised Machine learning for classification and detection of location spoofing attacks. The researchers aim to increase the overall precision of the misbehavior detection system by integrating Machine learning approaches with plausibility checks. Moreover, the researchers introduced a confusion matrix for representing the similar misbehavior of five position forging attacks using ML models. They addressed five different attacker types with three vehicle densities of location spoofing attacks' detection and classification problems using the VeReMi dataset. They considered KNN and Support Vector Machine (SVM) classification algorithms for misbehavior detection using Machine learning. They initiated the attack detection by creating sender-receiver pairs from the BSM data received containing packets sent at different times by the sender. The researchers considered six-dimension feature vectors, out of which the first two vectors are categorical features: the location plausibility check and movement plausibility check. The remaining four are quantitative data that describe the vehicle's behavior. The overall results of attack detection show that the proposed integrated model gave 20% better precision than the detection with only plausibility checks. This research provided a framework

with a better detection rate and classified the misbehaviors for better identification.

In their work, Ercan et al. [73] introduced two new features for classifying misbehaving vehicles in the network. Later the results of the proposed features are compared with the existing standard features. The research is focused on categorizing and detecting position falsification attacks from the VeReMi dataset. Two sets of features were proposed for misbehavior detection in two separate approaches. The researchers introduced two new features of the implemented features while the others were existing common features. The features utilized in this work were the received signal strength indicator (RSSI), the sender-receiver distance, the sender's position, the delta position, the angle of arrival (AoA), and the estimated distance by the path loss model. Their first proposal consisted of all six features, whereas the second approach did not contain the RSSI and delta position features. The authors also pointed out that recent works with anomaly detection techniques consisted of RSSI and distance between sender and receiver as the must-have features [74]. RSSI is useful in detecting position discrepancy as it denotes the strength of the radio signal that gets to the receiver. On the contrary, for keeping evidence of the signal sender's and receiver's position in the network, the calculated distance between them is used for detecting position falsification attacks. Another significant feature used in this research for detecting false attacker positions is the delta position which signifies the difference in the sender's position between the last two BSMs. The introduced features in this research are the angle of arrival (AoA) and the estimated distance using the path loss model. AoA is calculated with the help of utilizing the arctangent function on the sender and receiver distance for the x and y dimensions. On the other hand, the estimated distance using the path loss model was calculated with the assistance of the RSSI function from the dataset. After the features are prepared and gathered, they are provided to KNN and RF machine learning algorithms for misbehavior detection. The results obtained with the proposed feature sets were then compared with existing similar works. A comparison of the results showed that both the proposed feature sets outperformed the existing works in terms of KNN and RF models.

Similar to [73], Sharma et al. [75] in their work, proposed a data-centric misbehavior detection model for detecting and classifying misbehavior in an Internet of Vehicles (IoV) network by integrating supervised machine learning algorithms with plausibility checks. The proposed model detects and classifies Position Falsification attacks from the VeReMi dataset. Through this research, the authors aim to mitigate the gap in ensuring the security of V2V communication using traditional misbehavior detection techniques. The researchers propose implementing their model in the vehicles OBU to ensure real-time misbehavior detection. The proposed model integrates plausibility Location and Movement Plausibility checks. The location plausibility check compares the GPS location of the BSM received with the predicted range [76], and the Movement Plausibility check considers the velocities of received BSMs obtained throughout the journey to verify the movement of a vehicle [77]. The implemented model classifies and detects misbehavior using KNN, SVM, Naïve Bayes, Random Forest, Ensemble-Boosting, and Ensemble Voting classification algorithms. For validating the model, it is evaluated in two approaches: Firstly, the detection rate of each attack type for individual algorithms is evaluated. Secondly, the total detection rate for each plausibility check and attack for each ML algorithm integrated with plausibility checks are evaluated. Results of the proposed approach show that the overall performance increased after integrating plausibility checks with ML algorithms. Although after integrating plausibility checks, KNN, Naïve Bayes, and Random forest performance increased. Specifically, combining the ensemble learning algorithms with a location plausibility check gave the best recall and F1 score. The researchers calculated the classification loss model to add a dimension to model evaluation. The results of this evaluation demonstrated that Naïve Bayes had almost double the error as KNN, and both the ensemble methods showed the lowest classification error for their hyperparameter tuning. Overall, the integrated model presented 5% to 10% higher precision and 15% higher Area Under Curve (AOC) value for the attack types compared to the ML model alone.

In their work, Singh et al. [78] proposed a misbehavior detection model using Machine learning for detecting Wormhole attacks in VANET's multi-hop communi-

cation. The research focused on unicast transmission, a multi-hop communication system for data transfer between sender and receiver nodes. The wormhole attacks detected are the attacks by the insider attackers on the topology-based AODV routing protocol of the network. This work implemented the misbehavior detection model on their traffic scenario data generated by SUMO [79] traffic simulator and NS3 network simulator [80]. Trace files from Manhattan mobility generated by SUMO are provided to the NS3 simulator, where the network is simulated numerous times with variable attacking pair nodes. KNN and SVM algorithms are individually fed with the generated dataset to determine the detection accuracy. The final results demonstrate high accuracy of around 99% for both the classifiers.

2.4.2 Detecting Speed Offset Attacks

Hsu et al. [61] proposed a misbehavior detection model integrating a Convolutional Neural Network (CNN), Long short-term memory (LSTM) for reconstructing a position feature, and SVM for classifying and detecting misbehaving vehicles. The proposed model uses the Vehicle to everything important safety messages (BSMs) from the VeReMi extension dataset and evaluates the model's performance on 19 faults included in the dataset, along with Speed offset attacks. The researchers aim to utilize the proposed model on a vehicle's OBU to get notified about a misbehaving sender and inform the nearby RSU to take preventive measures. The authors pointed out that integrating ML algorithms with Deep learning algorithms increases the overall classification performance. Using a position reconstruction model, the authors generate a new feature that will specify the magnitude of behavioral difference between the tested vehicles and the standard pattern. For better quality of extracted features, CNN is implemented, and LSTM is applied for time series analysis of sequential BSM data. This approach is that it uses the legitimate Vehicle's information for the training of the reconstruction model. This phase of the model is trained at the RSU. Once the reconstructed position has been developed, the model calculates the target features needed for the next stage's classification using Mean Absolute Error (MAE). An SVM-based binary classifier is used for the classification phase with

11 features of the following four classes: behavioral deviation, location plausibility checks, velocity information, and comprehensive information. The authors evaluate their model's result with a few other works. The first is a CNN-4LSTM model with a threshold classification algorithm [81], and the second is the SVM-based classifier with features generated with plausibility checks [72]. Results show that the proposed model demonstrates the best performance among the works mentioned above, with an average misbehavior detection F1 score of 96.11%. Overall, the proposed misbehavior detection model detects over 95.35% of vehicles causing various attacks.

In a similar approach to [61], Alladi et al. [81] proposed a misbehavior detection scheme based on a CNN-LSTM combined Deep neural network based on sequence reconstruction. Further, they implemented a Thresholding algorithm for the final classification of the attacks. According to the authors of this work, a deep neural network provides the best performance for detecting a wide range of attacks and over the shortcomings of traditional data security methods. The proposed model uses 19 different anomaly types, including Speed Offset attacks and one normal vehicle type of the VeReMi Extension dataset. They used only the normal type data for training, and the remaining 19 attack types were used for sequence creation and testing of the model. Specifically, X and Y coordinates of position and speed data were considered for developing the sequence reconstruction-based misbehavior detection scheme. The model implemented CNN-LSTM and a stacked LSTM model and compared the results obtained by their respective sequence reconstruction scheme. The CNN-LSTM model was trained on time sequences generated by regular vehicles. After the training phase, the potential thresholds of each attack were calculated using the thresholding algorithm. The thresholds obtained using Mean Absolute Error (MAE) assisted in getting the recall scores along with other evaluation metrics, which presented the final comparison of the model's performance. Results show that the proposed CNN-LSTM model had an accuracy of 98%, which was better than the accuracy of 95.4% of the stacked LSTM model. Not only in terms of accuracy but also for the other evaluation metrics, the proposed CNN-LSTM model performed the best. Moreover, the authors presented a reconstruction graph to show the detection performance of their model.

The higher deviation indicated more MAE, signifying a better attack classification.

A comparative analysis of the researches discussed in this section is presented in the table 2.4.1.

No.	Paper	Machine Learning Model?	Dataset Used	Attacks Detected	Approach
1	So et al [72].	Yes	VeReMi	Position Falsification Attack	Plausibility checks combined with KNN and SVM Machine Learning algorithms for classification
2	Ercan et al. [73]	Yes	VeReMi	Position Falsification Attack	Introduced features: Estimated arrival angle and distance using path loss model. Classified using KNN and RF Machine Learning models
3	Sharma et. al [75]	Yes	VeReMi	Position Falsification Attack	Plausibility checks integrated with KNN, SVM, Naïve Bayes, RF, Ensemble-Boosting, and Ensemble Voting classification algorithms
4	Singh et al. [78]	Yes	Own dataset generated using SUMO and NS3 simulator	Wormhole Attack	KNN and SVM machine learning algorithms for classification
5	Hsu et al. [61]	Yes	VeReMi Extension	Speed Offset and many other attacks	Deep learning based approach with a combination of LSTM and CNN with SVM classifier
6	Alladi et al. [81]	Yes	VeReMi Extension	Speed Offset and many other attacks	Deep learning based approach with a combination of LSTM and CNN on the time sequence data.
7	**Proposed Model**	Yes	VeReMi Extension	Speed Offset attacks	Machine learning approach with KNN, LR, AdaBoost, XGBoost, Decision Tree, RF and MLP classifiers with a time metric for efficiency evaluation.

Table 2.4.1: Comparison table of Literature Review

CHAPTER 3

BSM Based Classification of Speed Attacks in VANET

3.1 Introduction

VANET is a highly dynamic network that is prone to various types of attacks. Multiple techniques are employed for identifying these attacks on a VANET network using a misbehavior detection method. This study proposes developing a misbehavior detection model using machine learning techniques to detect Speed offset attacks in VANET. BSMs transmitted by the vehicles are crucial in identifying and forecasting an attacker's behaviors in the network, as the BSMs contain the network member's current status. The related information in the BSM includes the sender id, speed, time, position, receiver id, and message-id of a vehicle. A vehicle's BSM is transmitted periodically to all the nearby vehicles and infrastructures of the network to inform the network about its whereabouts. Moreover, it is essential to gather BSM-related vehicle data from the network to attain the goal of this research. The proposed methodology uses the VeReMi Extension dataset comprising a collection of network BSMs under different traffic scenarios and attacker densities. In the proposed method, we aim to:

- Provide a framework for Speed offset attacks detection.
- Provide an efficient machine learning-based approach for classifying network vehicles as malicious or legitimate.

3.2 Proposed Architecture

Authenticated nodes send out BSMs periodically to the network's neighboring vehicles, RSUs, and infrastructures by digitally signing them. The receivers can use the public key to check the message's authenticity. During registration to the network, the vehicles are designated private and public keys by the Central authority. The vehicles later use these keys for signing the messages transferred using the Digital Signature algorithms. Although these cryptographic techniques preserve the authenticity of the network by ensuring only registered vehicles take place in the BSM transfer process, they cannot guarantee the message's integrity. As demonstrated in figure 3.2.1, in this research, we propose a misbehavior detection model that will work as an additional layer on top of the cryptographic techniques for ensuring message integrity. The RSUs within the network range receive the periodic BSMs, which are classified in the next phase using the model. The proposed framework can either be installed in the RSUs or the vehicle's OBU to detect and classify attackers.

As each vehicle is assigned a unique sender ID using registration, the RSUs will verify each message received and mark the vehicle based on the unique ID. After each BSM makes its way into the RSU, the proposed model applies machine learning classifiers in the architecture to classify the incoming node as either malicious or legitimate. If a sender is identified as malicious, the nearby RSUs and other nodes can be aware of the misbehaving vehicle.

The proposed model will impose different operations on the infrastructures in the network. Operations at the vehicles are as follows: The authorization authority registers the vehicles before entering the network. Next, the vehicle's OBUs send periodic BSMs to the network and use Digital signatures to ensure no alterations of the messages either on the way or after reaching the receiver. Finally, when an RSU broadcasts a message about a malicious node, a vehicle must listen to the message and act accordingly. On the contrary, the RSU carries out a different set of operations: The periodic BSMs sent by vehicles are received by the RSUs, and used to classify the sender as legitimate or malicious. Lastly, whenever an attacker vehicle is classified,

it broadcasts the information to other vehicles and infrastructures in the network.

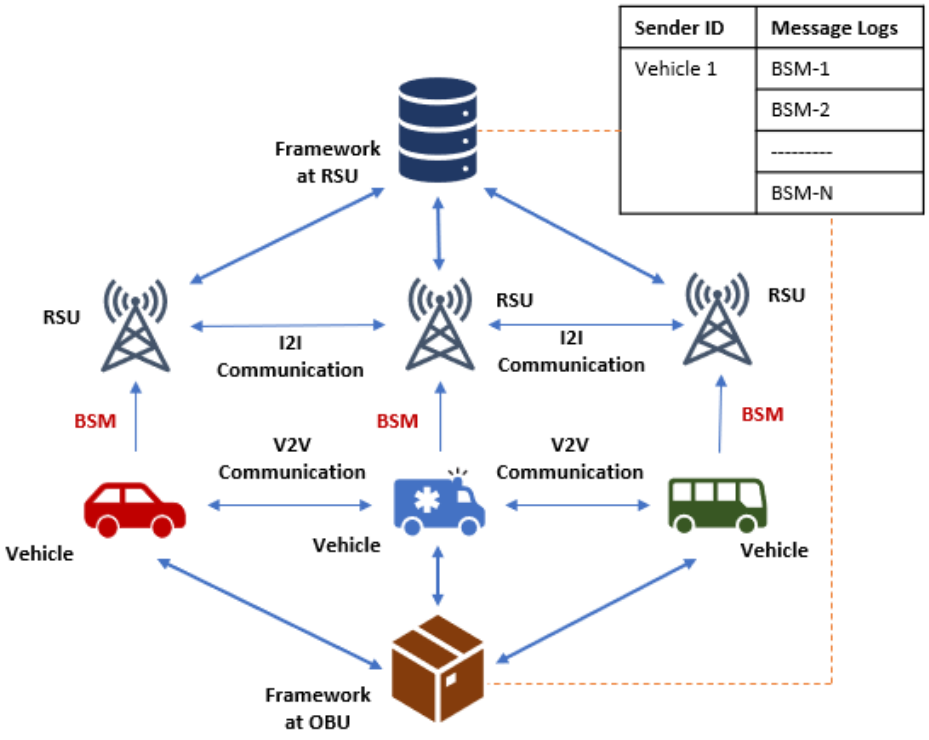

Fig. 3.2.1: Proposed Architecture

3.3 High-level Outline of Proposed Approach

The methodology comprises three levels: firstly, data extraction, followed by preparation of the collected data, and finally, utilizing the preprocessed data for classification of the attacks. Details of these stages are as follows:

3.3.1 Data Extraction

Our work uses the VeReMi extension dataset, which is created by modifying the VeReMi dataset. The extended dataset addresses 19 different types of attacks, which resulted in 39 datasets. In each simulation, two subsets are created: one for low-traffic hours and one for rush hours. These subsets consist of a Ground Truth file and multiple log files. Every simulation has only one ground truth file containing infor-

mation about the type of attacker, which assists in distinguishing between malicious and legitimate nodes. On the contrary, every vehicle participating in a simulation contributes with one log file containing the BSMs received from neighboring nodes. The ground truth files work as a reference with original information for comparison. In contrast, the log files can have incorrect data inserted by attackers, such as false information about the speed parameters in the speed offset attacks. The logs derived from the dataset are JavaScript Object Notation (JSON) files merged and combined into a comma-separated values (CSV) file. The combined log file is labeled using the unique sender ID to proceed toward the preprocessing stage.

3.3.2 Data Preparation

Data preparation is the second stage in which the data is prepared for fitting into the classifiers. Data within the combined log files are cleaned to remove non-contributing features and repetitive data. Each vehicle generates a log file, due to which the same BSM might be generated from multiple vehicles, resulting in redundant BSM data. Therefore, the first pre-processing stage was removing the redundant data points for better classification of attacks. The next step involved determining the more minor and non-contributing features and removing them from the dataset. Through this process, we decided the contribution capacity of each element in the classification process. Moreover, keeping features with lesser contributions will result in a reduction in model efficiency. At this level, we used our knowledge and experience in this domain to select the essential features and remove the unnecessary ones. We eliminated the noise features from our domain knowledge as their relevance to our goal is insignificant. Moreover, we removed all the 'Z' coordinate features as they contained a value of '0'. The message type and density level feature repeated the same value for all the entries; thus, we removed them.

Out of 34 features presented in the Speed offset attack dataset, we finalized the 15 most important and impactful features (including the dependant feature) for our classification model, as shown in Table 3.3.1. The finalized feature set's independent variables included the following: speed, acceleration, heading, and position vectors,

send and receive time of a message, ID and Pseudonym value of a sender, receiver ID, and lastly, the ID of the message being transferred. On the other hand, the dependent variable was the parameter of attack detection.

Next, we determined the final features' importance scores using the decision tree approach [82] to determine the most impactful features. Figure 3.3.1 and figure 3.3.2 graphically demonstrate the feature importance scores of each feature for the low traffic scenario for the Constant Speed offset and Random Speed offset datasets, respectively. The results showed that the sender ID, sender Pseudo and Message ID features had a minor contribution to our model. Moreover, the feature importance scores demonstrated that Speed in the x-direction and Heading in the x-direction was the most impactful feature for both datasets. After selecting the final features, the dataset went through standardization and normalization, which will be discussed in chapter 4.

From previous similar works on other attacks, we learned that sender ID, sender Pseudo and message ID significantly influence the overall model detection for certain attacks, e.g., replay attacks. But, this did not seem to be the case for speed offset attacks, as shown in the feature importance graphs. However, to eliminate the possibility of overfitting on these parameters, we created three versions of the prepared dataset for the classification stage as demonstrated in Figure 3.3.3:

1. **Scenario 1**: All 14 independent features.
2. **Scenario 2**: All independent features except Receive Time, Send Time, Receiver ID, sender ID, Sender Pseudo, and Message ID.
3. **Scenario 3**: All independent features except Send Time, Receiver ID, sender ID, Sender Pseudo, and Message ID.

3.3.3 Hyperparameter Tuning for Classification

Hyperparameters are parameters that are responsible for controlling the learning of a model. Before the learning process of a machine learning algorithm begins,

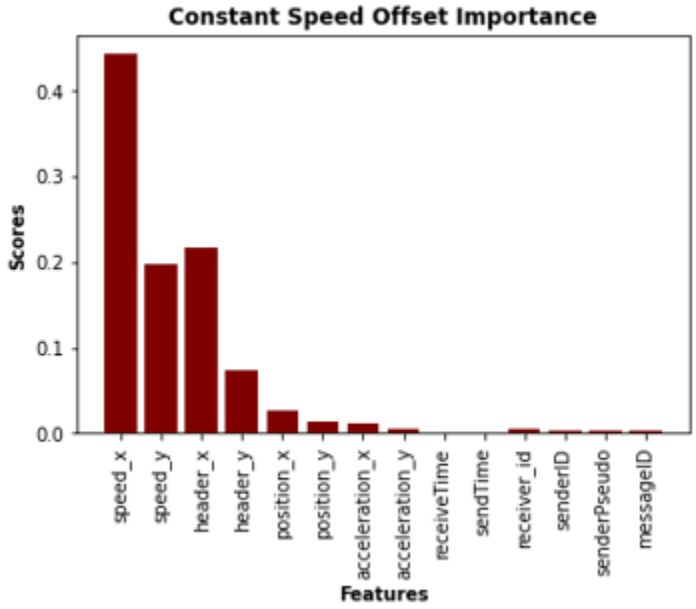

Fig. 3.3.1: Constant Speed Offset Feature Importance Score

its hyperparameters must be customized according to the dataset. Each dataset adapts and performs best with different combinations of hyperparameters. Before the model starts learning, different combinations of hyperparameters are tried on the model. The group of parameters giving the best results is determined as the hyperparameters for the particular dataset. Hyperparameter tuning is the process of customizing and determining a model's most suitable set of hyperparameters for getting the best results. Moreover, hyperparameter tuning is a way of preventing the model from over-fitting and under-fitting. Several techniques can be used to find the optimal set of hyperparameters for a problem. In this research, we used Grid Search CV, which evaluates the algorithm for a range of hyperparameters defined in a grid and determines the best one [83]. We implemented hyperparameter tuning on the following algorithms:

K-Nearest Neighbor: For KNN, we implemented hyperparameter tuning on various parameters. The parameter grid for hyperparameter tuning consisted of the range of 'n-neighbours': [1, 3, 5, 7, 9, 11, 13, 15]. The 'weights' parameter had options of either 'uniform' or 'distance', and finally, the choice of 'metric' was between 'Minkowski', 'euclidean', and 'manhattan'. The best set of

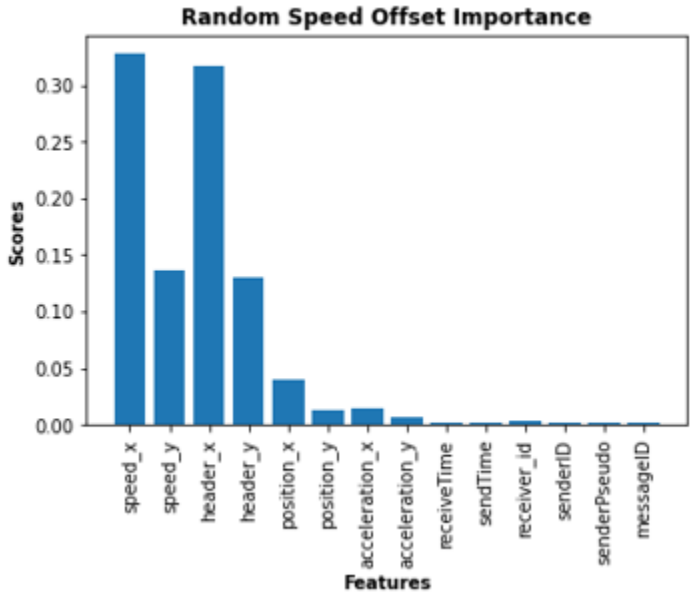

Fig. 3.3.2: Random Speed Offset Feature Importance Score

hyperparameters for the dataset of this research were: 'n-neighbours' = 3, 'weights' = 'uniform', and finally the 'metric' selected was 'Minkowski'. To further test our model, we tried implementing it with higher neighbors' values, but it was seen that 'k=3' gave the best result throughout.

Multi-layer Perceptron: After a rigorous hyperparameter tuning for MLP, out of the option of one, two, and three hidden layers, our model selected three hidden layers with 45, 30, and 15 neurons, respectively. Out of the 'tanh' and 'relu', we found 'relu' to be a more suitable 'activation function', and finally, out of '128' and '256', '128' was the more desirable 'batch size' for classification.

XGBoost: For this algorithm, the 'booster' parameter was chosen as 'gbtree' and 'objective' was 'multi: softprob'.

Random Forest: The estimator is one of the most crucial parameters for a Random Forest tree. Hence, we fitted our dataset for the following range of estimators: 10, 30, 50, 60, 80, 100, 130, 150, 170, and 200. The results showed that the highest accuracy obtained was for the estimator of 80. We kept the 'max-depth' at '6,' and the 'criterion' was set to 'entropy'.

Decision Tree: Before starting the learning process, we determined the maximum

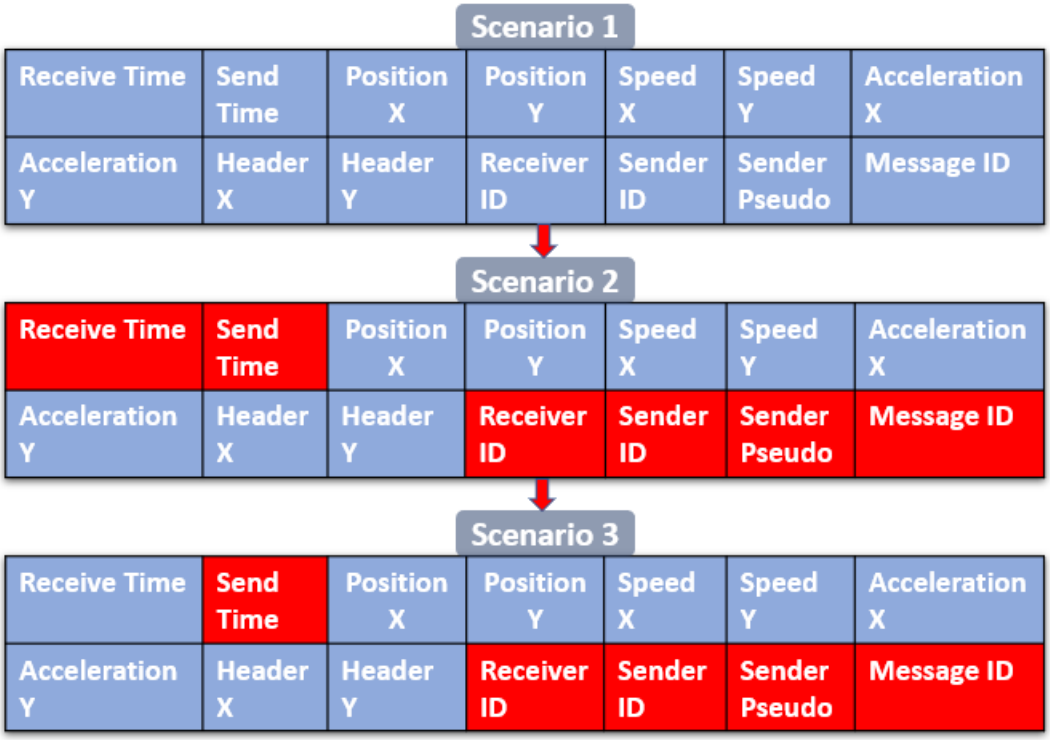

Fig. 3.3.3: Scenarios of Datasets

depth for each dataset's scenarios. The depth value for binary classification varied between '40' and '41' for the Random speed offset dataset, and for the Constant Speed offset dataset, it was between '29' and '31'. And for the multiclass classification, the maximum depth value was between '43' to '45'.

3.3.4 Cross Validation for Classification

K-fold cross-validation is a process of resampling the given data and splitting it in K folds for evaluating a machine learning model. The dataset is divided into K fragments, where one sample is the validation or test set, and the remaining 'K-1' fragments work for training the model. In the next iteration, the testing group and the training group change. In the end, the average evaluation score of each group is considered to evaluate the model's performance [83]. Unlike train-test split, K-fold cross-validation generates a less biased result. Moreover, it prevents the model from over-fitting and measures the accuracy efficiently. However, this process requires

more operation time than the train-test split but yields more optimistic results. For this research, we selected a value of 'k=5' for generating classification results.

3.4 Classification

The final stage of the methodology is applying the machine learning algorithms to the prepared dataset for classification. In this stage, the attacker vehicles of the network and identified. We propose detecting the attackers in two separate datasets of two attack types: Constant speed offset and Random Speed Offset. In our work, we will implement both binary and multiclass classification. The two types of attacks will be identified separately for the binary classification. And for the multiclass attacks, the dataset will be combined, and classification will be performed to determine the attacks together. For classifying attackers performing Speed offset attacks, we used the following machine learning algorithms: K-Nearest Neighbor, Logistic Regression, Adaptive Boost, XGBoost, Decision Tree, Random Forest, and the deep-learning classifier: Multi-layer Perceptron. We implemented the classifiers for the three dataset versions mentioned in section 3.3.2. We implemented three different versions to compare the concerning features' contribution to the overall classification performance. Before the final classification stage, the dataset will be divided into the train (80%), and test (20%) sets. Furthermore, the training set will be cross-validated to strengthen the ground of the results. Our misbehavior detection model will finally use the classifier with the best performance.

3.4.1 Binary Classification

The binary classification works with individual datasets of positive or negative classes. The classification is based on predefined classes [84]. For our thesis, we plan to implement binary classification on two datasets separately: Constant Speed offset attacks and Random speed offset attacks. Each of these two datasets contains labels of legitimate and malicious nodes. In binary classification, the model learns patterns from individual training data and predicts for one dataset at a time.

3.4.2 Multiclass Classification

In multiclass classification, each test datapoint can indicate any of the multiple classes [85]. Multiclass classification is performed in a combined dataset with two or more categories. For our work, we combined the two types of speed offset attacks and implemented multiclass classification on them. Through this approach, the model could train and learn the two attack types and classify them accordingly. After the model is trained for such classification, one giving an input data point, the model will first detect if it's an attacker datapoint. Next, it will determine the class of speed offset attack this data point belongs to. Compared to binary classification, multiclass classification is efficient and versatile for its ability to learn and detect multiple attacks simultaneously.

SL No.	All features	Eliminated Features	Selected Features for classification
1	Message Type	Message Type	
2	Receive Time		Receive Time
3	Send Time		Send Time
4	Sender ID		Sender ID
5	Sender Pseudo		Sender Pseudo
6	Receiver ID		Receiver ID
7	Message ID		Message ID
8	File Name	File Name	
9	Density Level	Density Level	
10	Position X		Position X
11	Position Y		Position Y
12	Position Z	Position Z	
13	Position Noise X	Position Noise X	
14	Position Noise Y	Position Noise Y	
15	Position Noise Z	Position Noise Z	
16	Speed X		Speed X
17	Speed Y		Speed Y
18	Speed Z	Speed Z	
19	Speed Noise X	Speed Noise X	
20	Speed Noise Y	Speed Noise Y	
21	Speed Noise Z	Speed Noise Z	
22	Acceleration X		Acceleration X
23	Acceleration Y		Acceleration Y
24	Acceleration Z	Acceleration Z	
25	Acceleration Noise X	Acceleration Noise X	
26	Acceleration Noise Y	Acceleration Noise Y	
27	Acceleration Noise Z	Acceleration Noise Z	
28	Header X		Header X
29	Header Y		Header Y
30	Header Z	Header Z	
31	Header Noise X	Header Noise X	
32	Header Noise X	Header Noise Y	
33	Header Noise Z	Header Noise Z	
34	Attack?		Attack?

Table 3.3.1: Table of Features

CHAPTER 4

Results

Although deploying and testing the proposed model in a real-world scenario is more pertinent, attempting to do so is hazardous and troublesome. Conducting experiments to test the proposed model's efficiency in the real world brings high safety measures, unreasonable infrastructure costs, and excessive resource requirements. Consequently, we considered running the experiments on a digital scale using simulation tools more reasonable. This chapter is divided into multiple subsections: where section 4.1 reviews the setup discussion of the simulation tools, the parameters used in the VeReMi extension datasets, experimental setup toolkits, parameters used in the classification process, and finally, details of the evaluation metrics for evaluating the performance of each of the classification algorithms. Next, section 4.2 presents a review, analysis, and discussion of the results obtained by our proposed model for binary classification, section 4.3 presents the review and analysis of the results obtained through multiclass classification, and lastly, section 4.4 compares results with the existing approaches.

4.1 Setup Discussion

In this research, we focus on the low attacker and vehicle density of the Speed Offset attack datasets of the VeReMi extension dataset. The Speed Offset attack dataset implemented in this research comprises two types of attack datasets: Random Speed Offset attack and Constant Speed Offset attack. Each dataset consisted of around 820097 entries, and the merged dataset for multiclass

classification contained around 1640000 data entries. After the data pre-processing step, where non-contributing features are eliminated, we considered three scenarios for each attack type, as follows:

- **Scenario 1**: Models are trained with all 14 selected features listed in Table 3.3.1.
- **Scenario 2**: In this case, the models are trained with all features except Receive Time, Send Time, Receiver ID, Sender ID, Sender Pseudo, and Message ID.
- **Scenario 3**: For the last scenario, we trained the models on the dataset except for Send Time, Receiver ID, sender ID, Sender Pseudo, and Message ID. The major difference between scenarios 2 and 3 is the message received time.

To evaluate the performance of the proposed approach, we selected the following machine learning and deep learning classifiers for detecting speed offset attacks:

- **Machine learning classifiers**: K-Nearest Neighbor, Logistic Regression, Adaptive Boost, XGBoost, Decision Tree and Random Forest.
- **Deep learning classifier**: Multi-layer Perceptron classifier.

4.1.1 Evaluation Metrics

The Speed Offset attacks datasets derived from the VeReMi Extension dataset is an imbalanced dataset with an unequal number of legitimate and attacker vehicle data. The two classes of the dataset are not evenly distributed, which results in an imbalanced classification. For such imbalanced datasets, it cannot be solely relied upon accuracy to act as a good performance metric. Hence, we derive each classifier's F1-score, precision, and recall values using a confusion matrix. For our dataset, the positive values of the contribution matrix denote attacker vehicles, and the negative indicates legitimate nodes. A confusion matrix represents a summary of

the prediction results of a classification algorithm. As demonstrated in table 4.1.1, a confusion matrix provides four outcomes for a binary classification problem like ours.

True Positive: A node is correctly identified as an attacker.

False Positive: A node is legitimate but is predicted as an attacker by the classifier.

True Negative: A node is correctly identified as a legitimate.

False Negative: A node is an attacker but incorrectly predicted as a legitimate node by the classifier.

	Predicted Negative	**Predicted Positive**
Actual Negative	True Negative	False Positive
Actual Positive	False Negative	True Positive

Table 4.1.1: Confusion Matrix

Precision

Using precision, we get the proportion of all accurate positive classifications out of all positive ones. Precision is also known as a positive predicted value. Equation (1) breaks down the concept of precision.

$$Precision = \frac{CorrectPositivePredictions}{TotalPositivePredictions} = \frac{TruePositive}{TruePositive + FalsePositive} \quad (1)$$

Recall

Recall accounts for the proportion of true positive classifications out of all positive predictions. The recall is also known as sensitivity, and recall details are provided in equation (2).

$$Recall = \frac{CorrectPositivePredictions}{TotalActualPredictions} = \frac{TruePositive}{TruePositive + FalseNegative} \quad (2)$$

F1-Score

The perfect trade-off between precision and recall is accumulated using F1-Score. It is the harmonic mean of precision and recall, which is expressed in equation (3). A high F1 score means a high precision and recall value.

$$F1 - score = 2 * \frac{Precision * Recall}{Precision * Recall} \qquad (3)$$

4.1.2 Implementation Environment and Toolkit

Experiments, research and implementation for this research were conducted in the following environment configuration:

- Operating system: Windows 10
- Processor: 2.40 GHz [Octa-Core] Core i5
- Memory: 8 GB

Tools and libraries used for the implementation

- Programming language: Python 3.9
- Integrated Development Environment: Jupyter Notebook, Google Colab
- Libraries: Scikit-learn, matplotlib, NumPy, pandas

4.2 Binary Classification

For binary classification of each attack-type dataset, we implemented the six machine learning algorithms (K-Nearest Neighbor, Logistic Regression, AdaBoost, XGBoost, Decision Tree, and Random Forest) and one deep learning algorithm (Multilayer Perceptron) on our proposed framework. For our research, we implemented the classifiers on the low-density Random Speed offset and Constant Speed offset dataset, each containing around 820097 data entries. Tables 4.2.1 -

4.2.3, and 4.2.4 - 4.2.6 show the performance of the classifiers for binary classification for constant and random speed offset attacks respectively. We have also reported the time taken for training and testing the datasets. The training time was the time taken for training the whole dataset for each classifier. We considered testing time to be the time to predict each sample as malicious or legitimate after the model was trained with the dataset. In addition to the evaluation metrics, the training and prediction times are important factors to consider when evaluating the ML/DL models. A detailed discussion of the proposed model's classification results is presented below.

4.2.1 Constant Speed Offset Classification

Scenario 1: Table 4.2.1 portrays that in terms of accuracy, the Decision tree (with a max depth of 29) gave the best result of around 98.5% followed by MLP with an accuracy score of 98.20%. Most classifiers could correctly detect the attacker vehicles for this dataset scenario with high evaluation metric scores. In terms of the time metrics, Logistic Regression performed the best with a per sample testing time of around 0.00054 milliseconds, followed by the Decision tree with a time count of 0.00067 milliseconds. On the contrary, KNN presented the worst accuracy score of around 88% and took the highest to train and test (approximately 0.80 milliseconds) for this scenario. Analyzing the results, we can conclude that for this scenario Decision tree gave the best trade-off of time and accuracy.

Scenario 2: Table 4.2.2 shows an overall improvement in the overall performance of all the classifiers for this scenario. Although Decision Tree (with a max depth of 30) and KNN gave a high performance with an accuracy of around 99%, MLP gave the highest accuracy score of approximately 99.5%.

In terms of time, both MLP (0.0009 milliseconds) and Logistic regression (0.00043) performed well. Still, considering the time and accuracy tradeoff, we can conclude that MLP was the best classifier for this scenario.

Scenario 3: This scenario portrayed similar overall results to scenario 1. Results from table 4.2.3 show that the Decision tree (with a max depth of 31) and

Constant Speed Offset (Scenario 1)						
Algorithms	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Training time {whole sample} (in ms)**	**Testing Time {per sample} (in ms)**
KNN	88.38	88.00	88.00	88.00	2589980.00	0.78
Logistic Regression	90.55	90.00	91.00	90.00	126.00	0.00054
AdaBoost	95.65	96.00	96.00	96.00	5401.00	0.0021
XGBoost	97.64	98.00	98.00	98.00	2058.00	0.0011
Random Forest	94.88	95.00	95.00	95.00	3592.00	0.00152
Decision Tree (Depth=29)	98.50	99.00	99.00	99.00	142.00	0.0005
MLP	98.20	98.00	98.00	98.00	913.00	0.00070

Table 4.2.1: Binary Classification results for Constant Speed Offset (Scenario 1)

Constant Speed Offset (Scenario 2)						
Algorithms	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Training time {whole sample} (in ms)**	**Testing Time {per sample} (in ms)**
KNN	99.13	99.00	99.00	99.00	2377139.00	0.731
Logistic Regression	90.46	90.00	90.00	90.00	114.00	0.00043
AdaBoost	95.51	96.00	96.00	95.00	5048.00	0.00197
XGBoost	97.71	98.00	98.00	98.00	2025.00	0.00108
Random Forest	95.81	96.00	96.00	96.00	3675.00	0.00155
Decision Tree (Depth=30)	99.02	99.00	99.00	99.00	199.00	0.00122
MLP	99.45	99.00	99.00	99.00	1249.00	0.00089

Table 4.2.2: Binary Classification results for Constant Speed Offset (Scenario 2)

MLP performed the best, where the Decision tree had an accuracy of around 98.7%. On the contrary, although Logistic regression had the best test time per sample of approximately 0.0004 milliseconds, it gave the worst accuracy score of around 89%. Following LR, the Decision tree also had quite a low time count of about 0.0007 milliseconds. Therefore, it is evident that the Decision tree was again the best performer as it not only had the most excellent detection rate but also detected the attackers in the least amount of time.

Comparing the overall results of scenarios 2 and 3, it is evident that the removal of message receive time from the dataset improved the overall performance of the classifiers. Furthermore, KNN dramatically performed better for scenario two than for the other two.

Constant Speed Offset (Scenario 3)						
Algorithms	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Training time {whole sample} (in ms)**	**Testing Time {per sample} (in ms)**
KNN	94.28	94.00	94.00	94.00	1824186.00	0.579
Logistic Regression	89.80	90.00	90.00	90.00	99.00	0.00041
AdaBoost	94.59	95.00	95.00	94.00	6025.00	0.00187
XGBoost	97.62	98.00	98.00	98.00	2055.00	0.00106
Random Forest	96.40	97.00	96.00	96.00	3596.00	0.0015
Decision Tree (Depth=31)	98.76	99.00	99.00	99.00	182.00	0.00071
MLP	98.64	99.00	99.00	99.00	204.00	0.0011

Table 4.2.3: Binary Classification results for Constant Speed Offset (Scenario 3)

4.2.2 Random Speed Offset Classification

Scenario 1: From the results presented in table 4.2.4 for scenario 1 (all features), we can observe that in terms of accuracy MLP again dominated the other classifiers with an accuracy score of 98.3%. However, the Decision tree with a maximum depth of 41 followed MLP with a very close score of 97.8% Logistic regression performed the worst in accuracy, with shallow precision scores. Moreover, the Random forest's performance seemed lower than usual for this case. Considering the time metrics, Logistic regression and Decision tree were the fastest classifiers to detect the malicious nodes, and KNN took the longest to train and test the model. We can conclude that in terms of accuracy, solely MLP performed the best, but considering the accuracy and time trade-off, the Decision tree is again the best classifier.

Scenario 2: From table 4.2.5 we can see that MLP and Decision tree (with a maximum depth of 40) performed significantly better than the others. However,

Random Speed Offset (Scenario 1)						
Algorithms	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Training time {whole sample} (in ms)**	**Testing Time {per sample} (in ms)**
KNN	85.41	85.00	85.00	85.00	2028042.00	0.590
Logistic Regression	70.33	49.00	70.00	58.00	**114.00**	0.00050
AdaBoost	85.40	86.00	85.00	84.00	6011.00	0.00200
XGBoost	94.50	95.00	94.00	94.00	2176.00	0.00113
Random Forest	83.94	87.00	84.00	82.00	3311.00	0.00145
Decision Tree (Depth=41)	97.79	98.00	98.00	98.00	**172.00**	0.00050
MLP	**98.30**	98.00	98.00	98.00	1187.00	0.00077

Table 4.2.4: Binary Classification results for Random Speed Offset (Scenario 1)

MLP's score of 99.16% slightly exceeded that of the Decision tree. However, surprisingly, due to the removal of receive time from the dataset, KNN performed significantly well compared to the first scenario, with an accuracy score of around 98.5%. Logistic regression presented the worst accuracy of around 70%. Regarding the time metric, Logistic regression and Decision tree gave the fastest detection rate, and KNN had the slowest one. Analyzing the overall results, in terms of accuracy solely, MLP performed the best, but considering the time and accuracy trade-off, the Decision tree again surpassed the others.

Scenario 3: Table 4.2.6 represents the results for scenario 3 of the Random Speed offset attack dataset. The results were similar to that of the first scenario, where MLP and Decision tree (max depth of 41) gave the best accuracy scores, with the dominance of MLP with a score of around 98.4%. With the addition of receive time to the dataset, KNN's performance deteriorated from that of the second scenario. Logistic regression again presented the worst accuracy score. Moreover, the evaluation metrics scores of XGBoost and Random forest seemed to improve from the last scenarios. Considering the time metrics, again, Logistic regression along with the Decision tree performed the fastest detection with around 0.00063 milliseconds and 0.00051 milliseconds, respectively, and KNN had the slowest

Random Speed Offset (Scenario 2)						
Algorithms	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Training time {whole sample} (in ms)**	**Testing Time {per sample} (in ms)**
KNN	**98.52**	99.00	99.00	99.00	2348642.00	0.661
Logistic Regression	70.33	49.00	70.00	58.00	113.00	**0.00050**
AdaBoost	86.10	86.00	86.00	85.00	4769.00	0.00190
XGBoost	95.13	95.00	95.00	95.00	2075.00	0.0011
Random Forest	88.24	90.00	88.00	87.00	3593.00	0.00154
Decision Tree (Depth=40)	**98.35**	98.00	98.00	98.00	164.00	**0.00050**
MLP	**99.20**	99.00	99.00	99.00	9151.00	0.00068

Table 4.2.5: Binary Classification results for Random Speed Offset (Scenario 2)

detection. Although in terms of accuracy, MLP had the highest score but considering the accuracy and time trade-off, the Decision tree was the constant best performer.

Random Speed Offset (Scenario 3)						
Algorithms	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Training time {whole sample} (in ms)**	**Testing Time {per sample} (in ms)**
KNN	91.82	92.00	92.00	92.00	2352601.00	0.720
Logistic Regression	70.33	49.00	70.00	58.00	**112.00**	0.00063
AdaBoost	86.25	87.00	86.00	85.00	4832.00	0.00180
XGBoost	94.72	95.00	95.00	95.00	2079.00	0.00113
Random Forest	90.30	91.00	90.00	90.00	3551.00	0.00150
Decision Tree (Depth=41)	**97.91**	98.00	98.00	98.00	**161.30**	0.00051
MLP	**98.40**	98.00	98.00	98.00	1265.00	0.00070

Table 4.2.6: Binary Classification results for Random Speed Offset (Scenario 3)

Overall Analysis: From figure 4.2.1, comparing the accuracy performance in the three scenarios indicates that the overall accuracy and other evaluation metrics scores were highest for the second scenario, in which receive time along with a few other features were removed from the feature set in the preprocessing stage. This

signifies the impact of receive time on our dataset. However, the low deviation in the results between the first and third scenarios represented the shallow mark of the sender ID, sender Pseudo, message ID, send time, and receiver ID in the overall performance of the dataset.

Furthermore, comparing the results of the two datasets for the scenarios, we observe that our proposed model could better identify and classify the constant speed offset attacks than the random speed offset attacks.

However, comparing the prediction times from figure 4.2.2, it was noticed that the timing values were comparable for the dataset scenarios.

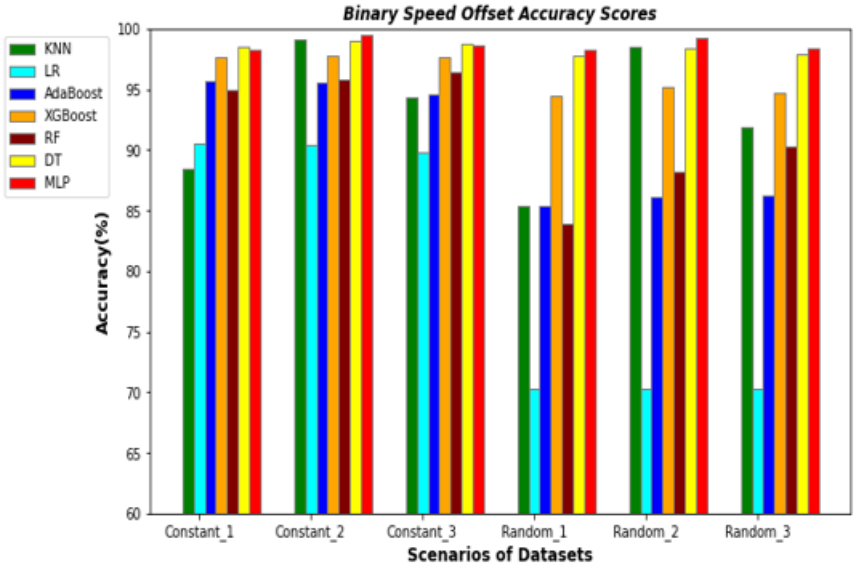

Fig. 4.2.1: Binary Classification Accuracy Comparison

4.2.3 Visualizing the results for Binary Classification

We used a precision-recall curve [86] to visualize the results obtained for both the Random Speed and Constant Speed offset datasets. The precision-recall curve is a valuable medium for visualizing prediction success for an imbalanced dataset like ours. We use the precision-recall curve for our binary classification to study and compare the attack detection performance of each classifier. In a classification model, precision depicts how much the results obtained are relevant to the dataset,

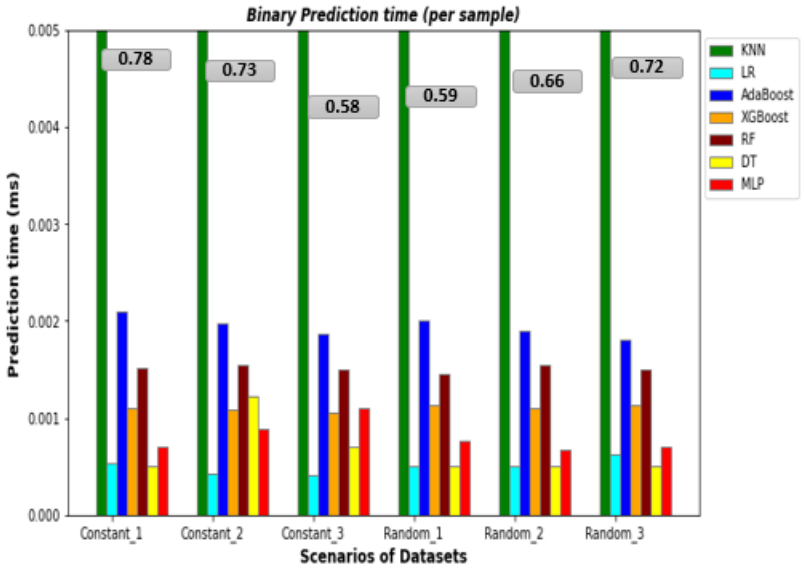

Fig. 4.2.2: Binary Classification Prediction time Comparison

and recall shows the amount of genuinely classified results. Using the precision-recall curve, we get the tradeoff between precision and recall for different parameters. A larger area under the curve represents a classifier's high recall and precision values. The high values of precision and recall denote low false positive and low false negative rates, respectively. Therefore, the higher the area under the curve, the classifier returns a more significant number of accurate results while correctly classifying the dataset with higher positive outcomes. Now, in cases with high recall but low precision, the system generates many results, but most of them are incorrectly classified. On the contrary, for higher precision and lower recall, the number of predictions is lower but primarily correct. In an ideal scenario with high recall and precision values, the model results in many prophecies, of which the majority are correctly classified.

We generate six visualization graphs for the three scenarios of the Constant Speed offset (figure 4.2.3) and Random Speed offset (figure 4.2.4) datasets. Each diagram depicts the combined performance of all the classifiers. The classifiers used for the precision-recall curves are K-Nearest neighbor, Logistic regression, AdaBoost, XG Boost, Random Forest, Decision Tree, and Multilayer perceptron. Moreover, these classifiers' parameters are kept the same for obtaining the binary

classification results. The overall results show that more vehicles were correctly classified for the Constant Speed offset dataset than for the Random Speed Offset attacks. MLP, Decision Tree, and XGBoost show remarkably good results for the constant speed offset dataset with smooth lines signifying high recall and precision scores. In contrast, Logistic regression performed the worst with noisy lines for all three scenarios. Such noisy lines represent low recall scores, hence less number of attacker vehicles were identified. Moreover, the performance of KNN seemed to be the best for scenario two, where we removed the message received from the dataset. The random speed offset dataset generated disappointing results compared to the constant speed offset. Except for MLP, the overall performance of all the classifiers was lower. Random Forest and AdaBoost produced noisy lines signifying low recall rates. Moreover, Logistic regression showed the worst performance with shallow precision values depicting the scarcity of accurately identified attacker vehicles. Furthermore, AdaBoost's performance significantly deteriorated compared to the constant dataset. However, a similarity was found between the performance pattern of KNN, where the removal of receive time greatly enhanced the precision and recall values.

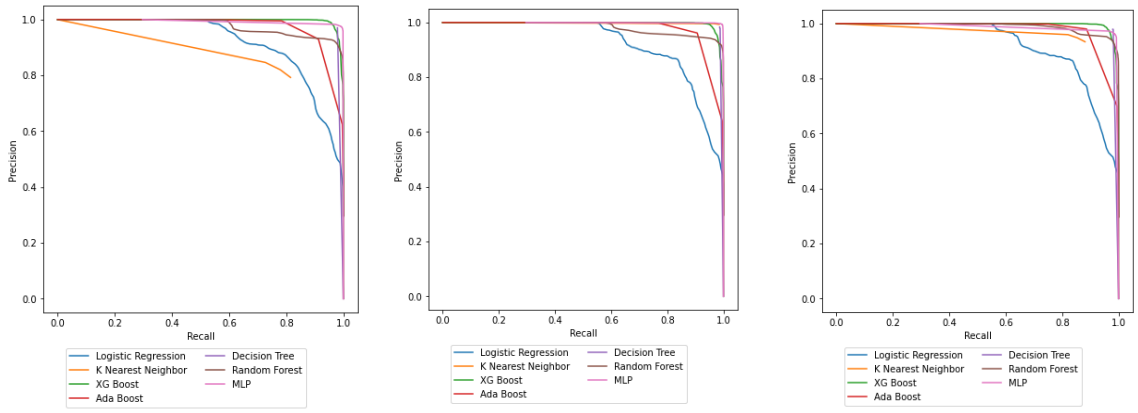

Fig. 4.2.3: Precision-Recall curves for Constant Speed offset for scenarios 1,2 & 3

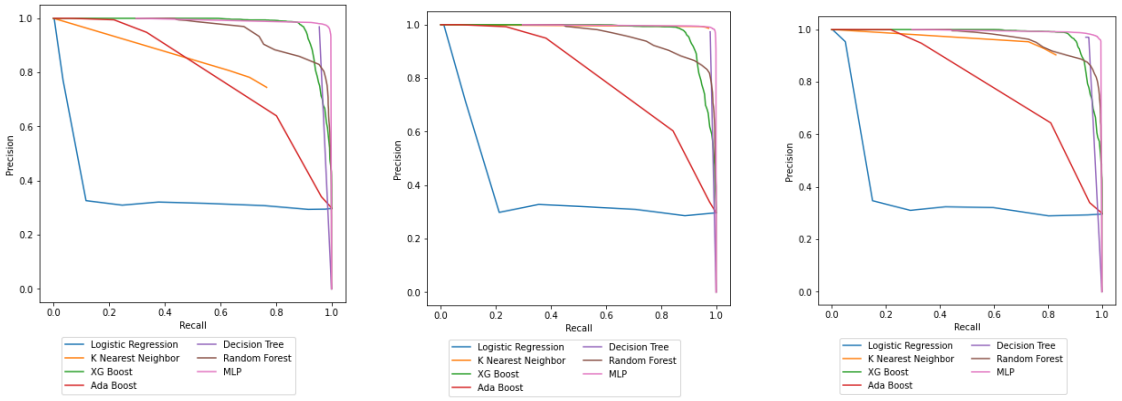

Fig. 4.2.4: Precision-Recall curves for Random Speed offset for scenarios 1,2 & 3

4.3 Multiclass Classification

Using multiclass classification, datasets containing more than two classes/labels are identified. To serve the cause, we created a new dataset by combining the Random Speed Offset and Constant Speed Offset datasets. The merged dataset contained two types of attacker data and one legitimate vehicle data. Both the individual datasets had a similar ratio of the attacker to legitimate nodes data, making the merged dataset fair. However, the proportion of total legitimate to attacker data was still imbalanced. The advantage of multiclass classification being it can train the models for all the classes at once and depict their contribution to the classifier's performance. Based on results obtained from binary classification, we selected and ran the multiclass classification on the top 5 classifiers: K-Nearest Neighbor, XGBoost, Random Forests, Decision Tree, and Multilayer perceptron. Tables 4.3.1, 4.3.2, and 4.3.3 portrays the multiclass classification on the merged dataset for the three scenarios mentioned in section 4.1. We used the same hyperparameter tuning obtained parameters for each classifier as binary classification.

Multiclass classification Scenario 1: Through analysis of the first scenario results from table 4.3.1, we can deduce that in terms of accuracy, XGBoost surpassed other classifiers with an accuracy of around 90.5%. Like binary classification with all features, KNN gave relatively low accuracy scores in this case. Regarding detection time, the Decision tree was the fastest, with a testing time of

around 0.0002 milliseconds, followed by MLP. However, compared to the binary classification, the Decision tree was a poor performer in terms of evaluation metrics in this scenario. Overall, due to the high detection time of XGBoost, MLP served the best trade-off between accuracy and time for this scenario of multiclass classification.

Multiclass Classification (Scenario 1)						
Algorithms	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Training time {whole sample} (in ms)**	**Testing Time {per sample} (in ms)**
XGBoost	**90.52**	91.00	91.00	90.00	12018.00	0.00394
KNN	78.12	80.00	78.00	79.00	68226.00	0.062
Random Forest	81.63	84.00	82.00	77.00	6909.00	0.00142
Decision Tree(D=45)	78.98	85.00	79.00	80.00	282.00	**0.00020**
MLP	**87.15**	88.00	87.00	88.00	5468.00	**0.00080**

Table 4.3.1: Multiclass Classification results Scenario 1

Multiclass classification Scenario 2: Table 4.3.2 represents the results obtained for the second scenario of multiclass classification. Similar to the binary classification, KNN's performance drastically improved from around 78% to around 91% between the first and second scenario due to the removal of message receive time. However, the best accuracy score was presented by MLP of around 94%. Furthermore, for this scenario, the accuracy of the Decision tree (maximum depth of 42) and Random forest improved. MLP provided the best trade-off between time and accuracy considering the time metric.

Multiclass classification Scenario 3: From table 4.3.3, we can deduce that similar to the binary classification, the first and third scenarios performed relatively equally. XGBoost had the highest accuracy score of around 90%, followed by MLP's score of around 88%. However, there was an improvement from the first scenario in the evaluation metrics score for KNN, Random Forest, and Decision tree (maximum depth of 43). Nonetheless, the best trade-off was presented by the MLP classifier for this scenario with a per sample detection time of 0.000724 milliseconds.

Multiclass Classification (Scenario 2)						
Algorithms	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Training time {whole sample} (in ms)**	**Testing Time {per sample} (in ms)**
XGBoost	90.50	91.00	90.00	90.00	11826.00	0.0038
KNN	**91.20**	91.00	91.00	91.00	43259.00	0.0130
Random Forest	83.93	86.00	84.00	81.00	7433.00	0.00135
Decision Tree(D=42)	89.62	90.00	90.00	89.00	354.00	**0.00024**
MLP	**94.10**	94.00	94.00	94.00	3571.00	**0.00070**

Table 4.3.2: Multiclass Classification results Scenario 2

Multiclass Classification (Scenario 3)						
Algorithms	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Training time {whole sample} (in ms)**	**Testing Time {per sample} (in ms)**
XGBoost	**90.27**	91.00	90.00	90.00	11702.00	0.00374
KNN	84.43	85.00	84.00	84.00	48033.00	0.0303
Random Forest	88.00	89.00	88.00	86.00	6973.00	0.00134
Decision Tree(D=43)	85.22	87.00	85.00	85.00	413.00	**0.000324**
MLP	**88.04**	90.00	88.00	89.00	3564.00	**0.00072**

Table 4.3.3: Multiclass Classification results Scenario 3

On graphically comparing the accuracy scores from figure 4.3.1 for each scenario for Multiclass classification, we observe that accuracy scores were slightly better for Scenario 2 than the other two. Moreover, KNN had a noticeable improvement in the score for the second scenario. On the contrary, similar to binary classification, scenarios 1 and 3 had comparable results.

Moreover, on comparing the prediction time from figure 4.3.2, we notice that prediction times were the lowest for scenario 2. This again signifies that this scenario had a better trade-off between time and accuracy than the other two scenarios.

Figure 4.3.3 to figure 4.3.7 present a normalized confusion matrix with attack misclassification rates for each of the 5 algorithms considered in this section. The

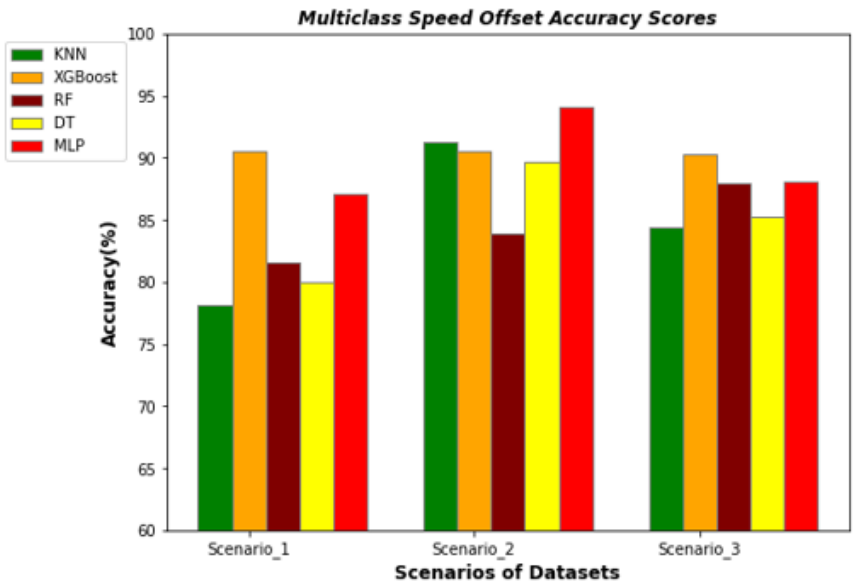

Fig. 4.3.1: Multiclass Classification Accuracy Comparison

predicted labels are on the horizontal axis, and the True labels are on the vertical. The left to right diagonal boxes in each matrix figure represented the correct classifications, whereas the boxes around them denote the misclassifications. Here, Label 0 denotes a legitimate vehicle, Label 1 denotes the random speed offset attack, and Label 2 represents the constant speed offset attack.

4.4 Comparison with Existing Approaches

Although considerable work has been done on detecting position falsification attacks, few existing approaches address speed offset attack detection. In this section, we compare the results obtained by the VeReMi Extension dataset and the current works with our proposed model. Based on the performance of the machine learning and deep learning algorithms implemented for binary classification of our model, the decision tree (DT) and Multilayer Perceptron (MLP) based models achieved the best results, and we have used these values in Table 4.4.1 for the proposed approach.

Table 4.4.1 compares the evaluation metric scores for the existing approaches

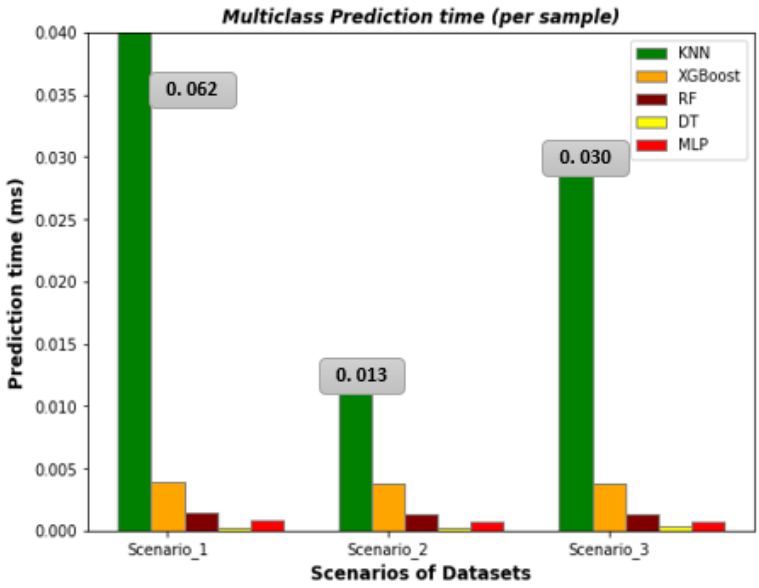

Fig. 4.3.2: Multiclass Classification Prediction time Comparison

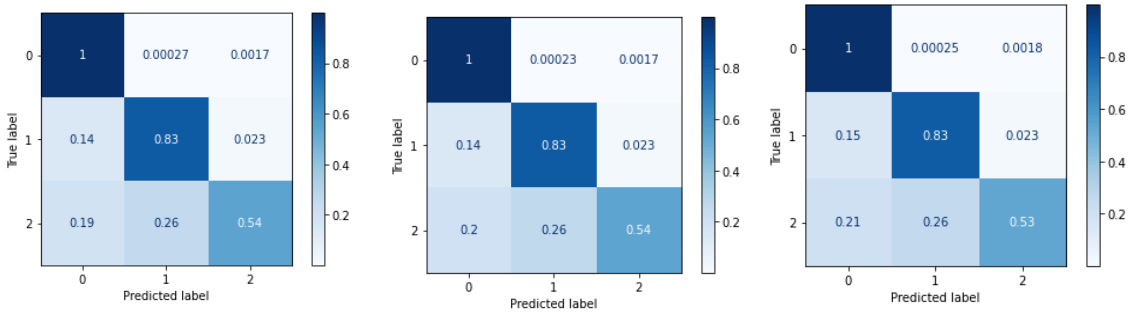

Fig. 4.3.3: Confusion Matrix for XGBoost for Scenarios 1,2 & 3

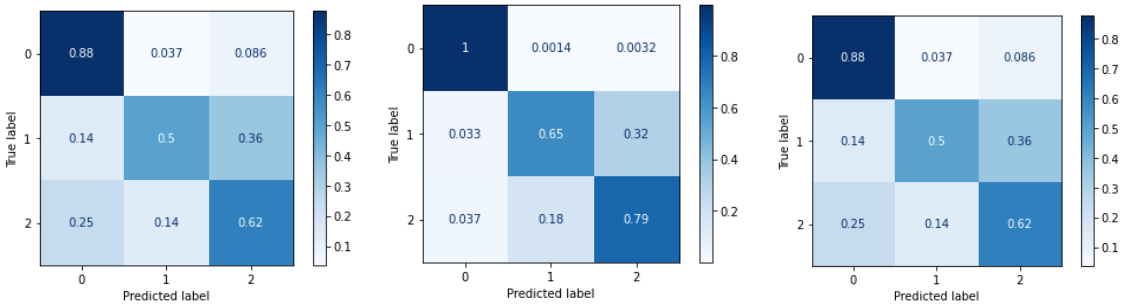

Fig. 4.3.4: Confusion Matrix for KNN for Scenarios 1,2 & 3

with our proposed model. The VeReMi extension paper had good scores for precision; the accuracy scores exceeded 80% but were significantly lower than other approaches and yielded unsatisfactory recall and F1 scores. In terms of the scores

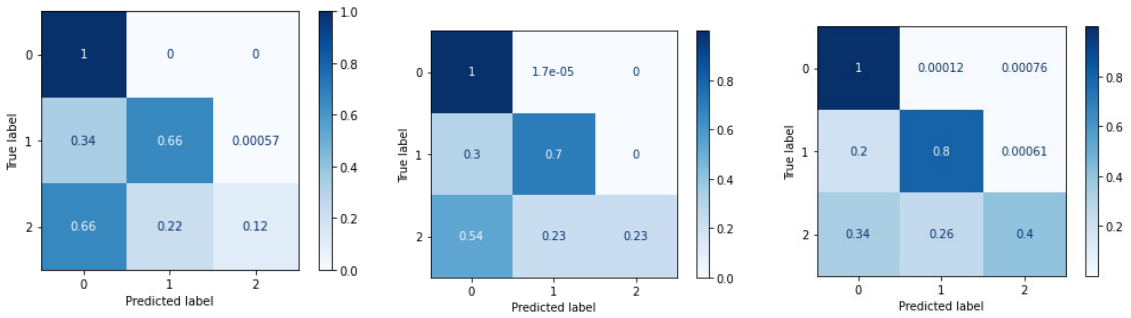

Fig. 4.3.5: Confusion Matrix for Random Forest for Scenarios 1,2 & 3

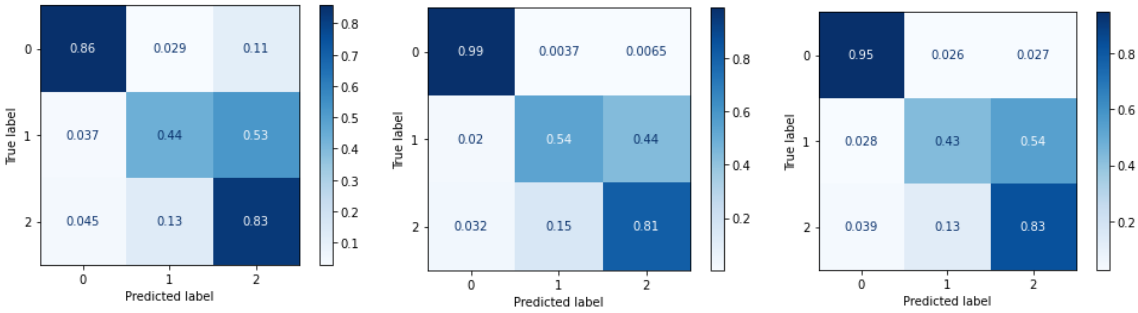

Fig. 4.3.6: Confusion Matrix for Decision Tree for Scenarios 1,2 & 3

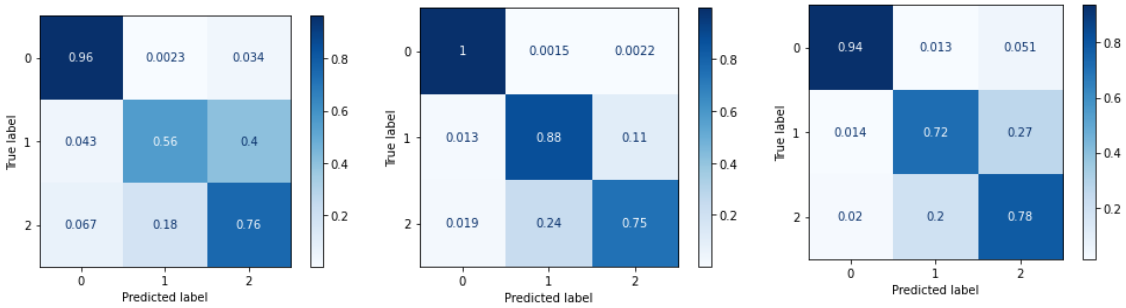

Fig. 4.3.7: Confusion Matrix for MLP for Scenarios 1,2 & and 3

for the Constant Speed offset dataset, Paper 1 [61] outperformed our model's results by a slight margin. However, our proposed model classified Random Speed offset attacks with an accuracy of around 99.20%, which was higher than Paper 1. Based on our results, we expect the proposed DT model will have the best overall performance when both accuracy and time are considered. Although, it had slightly lower scores than the MLP and Paper 1, tables 4.2.2 and 4.2.5 show that it was significantly faster compared to MLP. For random speed offset attacks, the proposed DT model's precision, recall, and F1 scores were similar to Paper 1, and the

proposed MLP model had better scores than Paper 1. Although Paper 2 [81] presented an excellent recall value of 100% for both types of attacks, in their limited work, they did provide results for any other evaluation metrics. Therefore, making a meaningful overall comparison with this approach was impossible.

Papers	**Constant Speed Offset**				**Random Speed Offset**			
	Accuracy	**Precision**	**Recall**	**F1 Score**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
VeReMi Extension [11]	81.57	99.57	39.69	56.76	89.28	99.72	65.0	78.70
Paper 1 [61]	99.85	99.82	99.82	99.82	98.67	98.14	98.73	98.44
Paper 2 [81]	-	-	100	-	-	-	100	-
Proposed Model (DT)	99.02	99.00	99.00	99.00	98.35	98.00	98.00	98.00
Proposed Model (MLP)	99.50	99.00	99.00	99.00	99.20	99.00	99.00	99.00

Table 4.4.1: Comparison of proposed model with existing approaches

CHAPTER 5

Conclusion and Future Work

5.1 Conclusion

This thesis proposes a novel Machine-learning and Deep-learning based approach for detecting Speed Offset attacks in VANET. We implemented the proposed misbehavior detection model on the low-density Constant Speed offset and Random Speed offset attack datasets from the VeReMi extension dataset. Based on our domain knowledge, each dataset consisted of the 14 most essential independent features of each BSM received by the network. After the required preprocessing, the datasets were fed into different classifiers to evaluate the classifier's performance. We performed binary classification for each dataset and multiclass classification by merging the Random speed offset, and Constant speed offset attack datasets. Moreover, we completed the implementation in three different scenarios to determine the impact of certain features on the classifiers' detection performance. In the first scenario, we considered all the selected features; in the second scenario, we excluded the message received and sent time, receiver and sender ID, sender Pseudo, and message ID. Lastly, for the third scenario, we returned the message's receive time into the dataset from the features selected for the second scenario. Our research implemented the machine learning algorithms: K-Nearest neighbor, XGBoost, Logistic regression, AdaBoost, Decision tree, Random forest; and the Deep learning classifier: Multi-layer perceptron. We compared the overall performance of all the classifiers mentioned above based on accuracy and the overall detection efficiency. Analyzing the results obtained from our implementation for

both datasets, we depicted that in the case of binary classification, the Decision tree and MLP mostly yielded the best results in terms of accuracy. Regarding the time and accuracy trade-off, the Decision Tree outperformed the others. Overall we observed that the implementations for the second scenario yielded the best results for binary classification compared to the other two scenarios. However, in terms of Multiclass classification, XGBoost, KNN, and MLP outperformed the other classifiers in terms of accuracy.

The overall performance of our binary classification was compared with existing approaches discussed in the Literature Survey that used the VeReMi extension dataset for identifying Speed offset attacks. The results showed that our proposed misbehavior detection model could improve existing approaches in classifying the Random Speed Offset attacks and is the first to report multiclass classification results. Finally, our research evaluated each classifier's prediction time, which was missing in the compared works.

5.2 Future Work

Our research only focused on the low-density Speed offset attack dataset. As prospective future work, we also want to run our proposed model on high-density datasets and make a comparative analysis of the model's performance. Moreover, we want to explore the Constant Speed and Random Speed attack datasets with our proposed architecture. Some models used in this research did not go through hyperparameter tuning; we would like to overcome this shortcoming in finding the optimal parameters for classification. In this part, we mainly concentrated on the Machine learning models. In the future, we plan to expand our scope by incorporating more deep learning models like Convolutional Neural Networks (CNN) and Artificial Neural Networks models like Long short-term memory (LSTM). Furthermore, we would like to create new features by combining the existing ones and compare the classification performance by incorporating them into the dataset.

REFERENCES

[1] Sheng-hai An, Byung-Hyug Lee, and Dong-Ryeol Shin. A survey of intelligent transportation systems. In *2011 Third International Conference on Computational Intelligence, Communication Systems and Networks*, pages 332–337. IEEE, 2011.

[2] World Health Organization et al. Global status report on road safety 2018. 17 june 2018. *Saatavissa (viitattu 25.8. 2020): www. who. int/publications/i/item/globalstatus-report-on-road-safety-2018*.

[3] Maxim Raya and Jean-Pierre Hubaux. The security of vehicular ad hoc networks. In *Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks*, pages 11–21, 2005.

[4] Rashmi Mishra, Akhilesh Singh, and Rakesh Kumar. Vanet security: Issues, challenges and solutions. In *2016 international conference on electrical, electronics, and optimization techniques (ICEEOT)*, pages 1050–1055. IEEE, 2016.

[5] Sherali Zeadally, Ray Hunt, Yuh-Shyan Chen, Angela Irwin, and Aamir Hassan. Vehicular ad hoc networks (vanets): status, results, and challenges. *Telecommunication Systems*, 50(4):217–241, 2012.

[6] Ghassan MT Abdalla, Mosa Ali Abu-Rgheff, and Sidi Mohammed Senouci. Current trends in vehicular ad hoc networks. *Ubiquitous Computing and Communication Journal*, pages 1–9, 2007.

[7] Federico Poli. *Vehicular communications: from DSRC to Cellular V2X*. PhD thesis, Politecnico di Torino, 2018.

[8] Muhammad Rizwan Ghori, Kamal Z Zamli, Nik Quosthoni, Muhammad Hisyam, and Mohamed Montaser. Vehicular ad-hoc network (vanet). In *2018 IEEE international conference on innovative research and development (ICIRD)*, pages 1–6. IEEE, 2018.

[9] Ahmed Shoeb Al Hasan, Md Shohrab Hossain, and Mohammed Atiquzzaman. Security threats in vehicular ad hoc networks. In *2016 international conference on advances in computing, communications and informatics (ICACCI)*, pages 404–411. IEEE, 2016.

[10] Neeraj Kumar, Rahat Iqbal, Sudip Misra, and Joel JPC Rodrigues. An intelligent approach for building a secure decentralized public key infrastructure in vanet. *Journal of Computer and System Sciences*, 81(6):1042–1058, 2015.

[11] Joseph Kamel, Michael Wolf, Rens W van der Hei, Arnaud Kaiser, Pascal Urien, and Frank Kargl. Veremi extension: A dataset for comparable evaluation of misbehavior detection in vanets. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2020.

[12] AR Deshmukh and SS Dorle. Simulation of urban mobility (sumo) for evaluating qos parameters for vehicular adhoc network. *IOSR J. Electron. Commun. Eng. Ver. I*, 11(1):2278–2834, 2016.

[13] Peter G Gipps. A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological*, 15(2):105–111, 1981.

[14] Peter G Gipps. A model for the structure of lane-changing decisions. *Transportation Research Part B: Methodological*, 20(5):403–414, 1986.

[15] Afsana Ahamed and Hamid Vakilzadian. Issues and challenges in vanet routing protocols. In *2018 IEEE international conference on electro/information technology (EIT)*, pages 0723–0728. IEEE, 2018.

[16] Mustafa Maad Hamdi, Lukman Audah, Sami Abduljabbar Rashid, Alaa Hamid Mohammed, Sameer Alani, and Ahmed Shamil Mustafa. A review of applications, characteristics and challenges in vehicular ad hoc networks (vanets). In *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pages 1–7. IEEE, 2020.

[17] Prinkle Sharma, Jonathan Petit, and Hong Liu. Pearson correlation analysis to detect misbehavior in vanet. In *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pages 1–5. IEEE, 2018.

[18] Ram Shringar Raw, Manish Kumar, and Nanhay Singh. Security challenges, issues and their solutions for vanet. *International journal of network security & its applications*, 5(5):95, 2013.

[19] Gagan Deep Singh, Ravi Tomar, Hanumat G Sastry, and Manish Prateek. A review on vanet routing protocols and wireless standards. *Smart computing and informatics*, pages 329–340, 2018.

[20] Felipe Cunha, Leandro Villas, Azzedine Boukerche, Guilherme Maia, Aline Viana, Raquel AF Mini, and Antonio AF Loureiro. Data communication in vanets: Protocols, applications and challenges. *Ad Hoc Networks*, 44:90–103, 2016.

[21] Saleh Yousefi, Mahmoud Siadat Mousavi, and Mahmood Fathy. Vehicular ad hoc networks (vanets): challenges and perspectives. In *2006 6th international conference on ITS telecommunications*, pages 761–766. IEEE, 2006.

[22] Hassnaa Moustafa and Yan Zhang. *Vehicular networks: techniques, standards, and applications*. Auerbach publications, 2009.

[23] Er Gurpreet Singh. Overview of challenges in vanet. *International Journal of Innovative Research in Science and Engineering*, 2(08):389–394, 2016.

[24] Mohammed Ali Hezam Al Junaid, AA Syed, Mohd Nazri Mohd Warip, Ku Nurul Fazira Ku Azir, and Nurul Hidayah Romli. Classification of security

attacks in vanet: A review of requirements and perspectives. In *MATEC web of conferences*, volume 150, page 06038. EDP Sciences, 2018.

[25] Deepak Kushwaha, Piyush Kumar Shukla, and Raju Baraskar. A survey on sybil attack in vehicular ad-hoc network. *International Journal of Computer Applications*, 98(15), 2014.

[26] Mahmood A Al-shareeda, Mohammed Anbar, Iznan H Hasbullah, Selvakumar Manickam, Nibras Abdullah, and Mustafa Maad Hamdi. Review of prevention schemes for replay attack in vehicular ad hoc networks (vanets). In *2020 IEEE 3rd International Conference on Information Communication and Signal Processing (ICICSP)*, pages 394–398. IEEE, 2020.

[27] Muhammad Sameer Sheikh and Jun Liang. A comprehensive survey on vanet security services in traffic management system. *Wireless Communications and Mobile Computing*, 2019, 2019.

[28] Richard Gilles Engoulou, Martine Bellaïche, Samuel Pierre, and Alejandro Quintero. Vanet security surveys. *Computer Communications*, 44:1–13, 2014.

[29] Irshad Ahmed Sumra, Halabi Bin Hasbullah, and Jamalul-lail Bin AbManan. Attacks on security goals (confidentiality, integrity, availability) in vanet: a survey. In *Vehicular Ad-Hoc Networks for Smart Cities*, pages 51–61. Springer, 2015.

[30] Ajay N Upadhyaya and JS Shah. Attacks on vanet security. *Int J Comp Eng Tech*, 9(1):8–19, 2018.

[31] Sushil Kumar and Kulwinder Singh Mann. Prevention of dos attacks by detection of multiple malicious nodes in vanets. In *2019 International Conference on Automation, Computational and Technology Management (ICACTM)*, pages 89–94. IEEE, 2019.

[32] S Balasubramani, SK Rani, and K Suja Rajeswari. Review on security attacks

and mechanism in vanet and manet. In *Artificial Intelligence and Evolutionary Computations in Engineering Systems*, pages 655–666. Springer, 2016.

[33] Mohssen Mohammed, Muhammad Badruddin Khan, and Eihab Bashier Mohammed Bashier. *Machine learning: algorithms and applications*. Crc Press, 2016.

[34] Vineet Chaoji, Rajeev Rastogi, and Gourav Roy. Machine learning in the real world. *Proceedings of the VLDB Endowment*, 9(13):1597–1600, 2016.

[35] Taiwo Oladipupo Ayodele. Types of machine learning algorithms. *New advances in machine learning*, 3:19–48, 2010.

[36] Payam Refaeilzadeh, Lei Tang, and Huan Liu. Cross-validation. *Encyclopedia of database systems*, 5:532–538, 2009.

[37] Aized Amin Soofi and Arshad Awan. Classification techniques in machine learning: applications and issues. *Journal of Basic & Applied Sciences*, 13:459–465, 2017.

[38] Gopalan Kesavaraj and Sreekumar Sukumaran. A study on classification techniques in data mining. In *2013 fourth international conference on computing, communications and networking technologies (ICCCNT)*, pages 1–7. IEEE, 2013.

[39] Pratap Chandra Sen, Mahimarnab Hajra, and Mitadru Ghosh. Supervised classification algorithms in machine learning: A survey and review. In *Emerging technology in modelling and graphics*, pages 99–111. Springer, 2020.

[40] Ravi Tomar, Manish Prateek, and GH Sastry. Vehicular adhoc network (vanet)-an introduction. *International Journal of Control Theory and Applications*, 9(18):8883–8888, 2016.

[41] Ahmad Riza'ain Yusof, Nur Izura Udzir, and Ali Selamat. An evaluation on knn-svm algorithm for detection and prediction of ddos attack. In *International*

conference on industrial, engineering and other applications of applied intelligent systems, pages 95–102. Springer, 2016.

[42] Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. Knn model-based approach in classification. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 986–996. Springer, 2003.

[43] J Brownlee. Logistic regression for machine learning.[online] machine learning mastery, 2021.

[44] Tae-Ki An and Moon-Hyun Kim. A new diverse adaboost classifier. In *2010 International conference on artificial intelligence and computational intelligence*, volume 1, pages 359–363. IEEE, 2010.

[45] Amanpreet Singh, Narina Thakur, and Aakanksha Sharma. A review of supervised machine learning algorithms. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 1310–1315. Ieee, 2016.

[46] Rens W Heijden, Thomas Lukaseder, and Frank Kargl. Veremi: A dataset for comparable evaluation of misbehavior detection in vanets. In *International conference on security and privacy in communication systems*, pages 318–337. Springer, 2018.

[47] Yan-Yan Song and LU Ying. Decision tree methods: applications for classification and prediction. *Shanghai archives of psychiatry*, 27(2):130, 2015.

[48] Vladimir Vovk. The fundamental nature of the log loss function. In *Fields of Logic and Computation II*, pages 307–318. Springer, 2015.

[49] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[50] Dennis W Ruck, Steven K Rogers, and Matthew Kabrisky. Feature selection using a multilayer perceptron. *Journal of Neural Network Computing*, 2(2):40–48, 1990.

[51] Terry Windeatt. Ensemble mlp classifier design. In *Computational Intelligence Paradigms*, pages 133–147. Springer, 2008.

[52] Christoph Sommer, Reinhard German, and Falko Dressler. Bidirectionally coupled network and road traffic simulation for improved ivc analysis. *IEEE Transactions on mobile computing*, 10(1):3–15, 2010.

[53] Varga András. The omnet++ discrete event simulation system. In *Proceedings of the European simulation multiconference (ESM'2001)*, 2001.

[54] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent development and applications of sumo-simulation of urban mobility. *International journal on advances in systems and measurements*, 5(3&4), 2012.

[55] Lara Codeca, Raphaël Frank, and Thomas Engel. Luxembourg sumo traffic (lust) scenario: 24 hours of mobility for vehicular networking research. In *2015 IEEE Vehicular Networking Conference (VNC)*, pages 1–8. IEEE, 2015.

[56] Norbert Bißmeyer, Christian Stresing, and Kpatcha M Bayarou. Intrusion detection in vanets through verification of vehicle movement data. In *2010 IEEE Vehicular Networking Conference*, pages 166–173. IEEE, 2010.

[57] James A DiLellio, Edward Carolipio, Jya-Syin Wu Chien, and Kamran Ghassaei. Global positioning system accuracy enhancement, June 28 2011. US Patent 7,969,352.

[58] Kazuyuki Kobayashi, Ka C Cheok, and Kajiro Watanabe. Estimation of absolute vehicle speed using fuzzy logic rule-based kalman filter. In *Proceedings of 1995 American Control Conference-ACC'95*, volume 5, pages 3086–3090. IEEE, 1995.

[59] Joseph Kamel, Arnaud Kaiser, Ines ben Jemaa, Pierpaolo Cincilla, and Pascal Urien. Catch: a confidence range tolerant misbehavior detection approach. In *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–8. IEEE, 2019.

[60] Joseph Kamel, Farah Haidar, Ines Ben Jemaa, Arnaud Kaiser, Brigitte Lonc, and Pascal Urien. A misbehavior authority system for sybil attack detection in c-its. In *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pages 1117–1123. IEEE, 2019.

[61] Hsiao-Yuan Hsu, Nai-Hsin Cheng, and Chun-Wei Tsai. A deep learning-based integrated algorithm for misbehavior detection system in vanets. In *Proceedings of the 2021 ACM International Conference on Intelligent Computing and its Emerging Applications*, pages 53–58, 2021.

[62] Muhammad Arif, Guojun Wang, Md Zakirul Alam Bhuiyan, Tian Wang, and Jianer Chen. A survey on security attacks in vanets: Communication, applications and challenges. *Vehicular Communications*, 19:100179, 2019.

[63] Bryan Parno and Adrian Perrig. Challenges in securing vehicular networks. In *Workshop on hot topics in networks (HotNets-IV)*, pages 1–6. Maryland, USA, 2005.

[64] Sumair Ur Rahman and Hossein Falaki. Security & privacy for dsrc-based automotive collision reporting. *www. cs. ucla. edu/falaki/courses/security project. pdf*, 2009.

[65] Rongxing Lu, Xiaodong Lin, Xiaohui Liang, and Xuemin Shen. Flip: An efficient privacy-preserving protocol for finding like-minded vehicles on the road. In *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, pages 1–5. IEEE, 2010.

[66] Mihail L Sichitiu and Maria Kihl. Inter-vehicle communication systems: a survey. *IEEE Communications Surveys & Tutorials*, 10(2):88–105, 2008.

[67] Aekta Sharma and Arunita Jaekel. Machine learning based misbehaviour detection in vanet using consecutive bsm approach. *IEEE Open Journal of Vehicular Technology*, 3:1–14, 2021.

[68] Fatih Sakiz and Sevil Sen. A survey of attacks and detection mechanisms on intelligent transportation systems: Vanets and iov. *Ad Hoc Networks*, 61:33–50, 2017.

[69] Marwane Ayaida, Nadhir Messai, Sameh Najeh, and Kouamé Boris Ndjore. A macroscopic traffic model-based approach for sybil attack detection in vanets. *Ad Hoc Networks*, 90:101845, 2019.

[70] Maxim Raya and Jean-Pierre Hubaux. Securing vehicular ad hoc networks. *Journal of computer security*, 15(1):39–68, 2007.

[71] Jyoti Grover, Vijay Laxmi, and Manoj Singh Gaur. Misbehavior detection based on ensemble learning in vanet. In *International Conference on Advanced Computing, Networking and Security*, pages 602–611. Springer, 2011.

[72] Steven So, Prinkle Sharma, and Jonathan Petit. Integrating plausibility checks and machine learning for misbehavior detection in vanet. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 564–571. IEEE, 2018.

[73] Secil Ercan, Marwane Ayaida, and Nadhir Messai. New features for position falsification detection in vanets using machine learning. In *ICC 2021-IEEE International Conference on Communications*, pages 1–6. IEEE, 2021.

[74] Steven So, Jonathan Petit, and David Starobinski. Physical layer plausibility checks for misbehavior detection in v2x networks. In *Proceedings of the 12th conference on security and privacy in wireless and mobile networks*, pages 84–93, 2019.

[75] Prinkle Sharma and Hong Liu. A machine-learning-based data-centric

misbehavior detection model for internet of vehicles. *IEEE Internet of Things Journal*, 8(6):4991–4999, 2020.

[76] Benedikt Brecht and Thorsten Hehn. A security credential management system for v2x communications. In *Connected Vehicles*, pages 83–115. Springer, 2019.

[77] Rens Wouter van der Heijden, Stefan Dietzel, Tim Leinmüller, and Frank Kargl. Survey on misbehavior detection in cooperative intelligent transportation systems. *IEEE Communications Surveys & Tutorials*, 21(1):779–811, 2018.

[78] Pranav Kumar Singh, Rahul Raj Gupta, Sunit Kumar Nandi, and Sukumar Nandi. Machine learning based approach to detect wormhole attack in vanets. In *Workshops of the international conference on advanced information networking and applications*, pages 651–661. Springer, 2019.

[79] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. Sumo–simulation of urban mobility: an overview. In *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind, 2011.

[80] Thomas R Henderson, Mathieu Lacage, George F Riley, Craig Dowell, and Joseph Kopena. Network simulations with the ns-3 simulator. *SIGCOMM demonstration*, 14(14):527, 2008.

[81] Tejasvi Alladi, Ayush Agrawal, Bhavya Gera, Vinay Chamola, Biplab Sikdar, and Mohsen Guizani. Deep neural networks for securing iot enabled vehicular ad-hoc networks. In *ICC 2021-IEEE International Conference on Communications*, pages 1–6. IEEE, 2021.

[82] MD Ridwan Al Iqbal, Saiedur Rahman, Syed Irfan Nabil, and Ijaz Ul Amin Chowdhury. Knowledge based decision tree construction with feature importance domain knowledge. In *2012 7th international conference on electrical and computer engineering*, pages 659–662. IEEE, 2012.

[83] GSK Ranjan, Amar Kumar Verma, and Sudha Radhika. K-nearest neighbors and grid search cv based real time fault monitoring system for industries. In *2019 IEEE 5th international conference for convergence in technology (I2CT)*, pages 1–5. IEEE, 2019.

[84] Roshan Kumari and Saurabh Kr Srivastava. Machine learning: A review on binary classification. *International Journal of Computer Applications*, 160(7), 2017.

[85] Ryan Rifkin. Multiclass classification. *Lecture Notes, Spring08. MIT, USA*, page 59, 2008.

[86] Kendrick Boyd, Kevin H Eng, and C David Page. Area under the precision-recall curve: point estimates and confidence intervals. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 451–466. Springer, 2013.

[87] Parul Tyagi and Deepak Dembla. A taxonomy of security attacks and issues in vehicular ad-hoc networks (vanets). *International Journal of Computer Applications*, 91(7), 2014.

[88] SAE V2X Core Technical Committee et al. Dedicated short range communications (dsrc) message set dictionary: A march 2016 update. *J2735_201603, https://saemobilus. sae. org/content/j2735_201603, accessed: October*, 2019.

[89] Jyoti Grover, Manoj Singh Gaur, and Vijay Laxmi. A novel defense mechanism against sybil attacks in vanet. In *Proceedings of the 3rd international conference on Security of information and networks*, pages 249–255, 2010.

[90] Jaydip Kamani and Dhaval Parikh. A review on sybil attack detection techniques. *J Res*, 1(01), 2015.

VITA AUCTORIS

NAME:	Bhuiyan Mustafa Tawheed
PLACE OF BIRTH:	Sylhet, Bangladesh
YEAR OF BIRTH:	1996
EDUCATION:	BRAC University, B.Sc in Computer Science and Engineering, Dhaka, Bangladesh, 2018
	University of Windsor, M.Sc in Computer Science, Windsor, Ontario, 2022.