Detection of Malicious Cyber Attacks in Internet of Vehicles

Submitted in partial fulfilment of the requirements

of the degree of

Bachelor of Technology (B.Tech)

by

Thallada Meghana (207279)

Sarode Sai Preetam (207268)

Yetelly Anand (207284)

Supervisor:

Dr. Rashmi Ranjan Rout
Professor

Department of Computer Science and Engineering

NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL

2023-2024

Acknowledgement

We would like to express our heartily gratitude towards Dr. Rashmi Ranjan Rout Sir, Professor, CSE Department, for his valuable guidance, supervision, suggestions, encouragement and the help throughout the semester and also for the completion of our project work. He kept us going when we were down and gave us the courage to keep moving forward.

We would like to take this opportunity once again to thank Dr.R.Padmavathy, Head of the Department, Computer Science and Engineering, NIT Warangal for giving us this opportunity and resources to work on this project and supporting through out. We also want to thank evaluation committee for their valuable suggestions on my proposals and research and for conducting smooth presentation of the project.

Meghana Thallada 207279

Sai Preetam Sarode 207268

Anand Yetelly 207284

Date:

NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL
2023-24

APPROVAL SHEET

The Project Work entitled **Detection of Malicious Cyber Attacks in Internet of Vehicles** by **Meghana Thallada (207279), Sai Preetam Sarode (207268), Anand Yetelly (207284),** is approved for the degree of Bachelor of Technology (B.Tech) in Computer Science and Engineering.

Examiners

_____

_____

_____

Supervisor

_____

Dr. Rashmi Ranjan Rout
Professor

Head of the Department

_____

Dr.R Padmavathy
Professor

Date: _____

Place: NIT, Warangal

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources.We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Meghana Thallada	Sarode Sai Preetam	Anand Yetelly
207279	207268	207284

Date:

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL
2023-24

Certificate

This is to certify that the Dissertation work entitled **Detection of Malicious Cyber Attacks in Internet of Vehicles** is a bonafide record of work carried out by **Meghana Thallada (207279), Sarode Sai Preetam (207268), Anand Yetelly (207284)**, submitted to the Department of Computer Science and Engineering, in partial fulfilment of the requirements for the award of the degree of B.Tech at National Institute of Technology, Warangal during the 2023-2024.

(Signature)
Dr.R.Padmavathy
Professor
Head of the Department
Department of Computer Science and Engineering
NIT Warangal

(Signature)
Dr. Rashmi Ranjan Rout
Professor
Department of Computer Science and Engineering
NIT Warangal

Abstract

Internet of Vehicles has changed the automotive industry by providing advanced technology for safe navigation and traffic management. Modern vehicles, including connected and autonomous ones, use intra-vehicle networks for various functions and are linked to outside networks through vehicle-to-everything(V2X) technologies. In addition to that, the increased functionality and connectivity of these vehicles make them prone to cyber-attacks on both intra-vehicle and external networks. Enhancing the security of vehicular networks is a critical challenge. While previous research has made progress in developing models for detecting cyber-attacks, intrusion detection still remains complex because of the high volume of network traffic data, numerous network characteristics, and various types of cyber-attack patterns. This study proposes a malicious cyber attack detection system using various machine learning algorithms that efficiently identifies known cyber-attacks on both intra-vehicle and external networks.

Keywords —- Internet Of Vehicles, Attack Detection, Bayesian Optimization.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Internet of Vehicles

In the realm of the Internet of Vehicles (IoV), the convergence of advanced communication technologies and automotive systems has introduced a plethora of cyber attack vulnerabilities. IoV ecosystems, comprising interconnected vehicles, infrastructure, and backend services, are particularly susceptible to cyber threats due to their inherent complexity and reliance on networked communication. Attack vectors in IoV encompass a wide spectrum of malicious activities, ranging from remote vehicle hijacking and tampering with critical systems to theft of sensitive data and disruption of vehicular communication networks. With vehicles becoming increasingly connected and autonomous, the attack surface expands, providing adversaries with more opportunities to exploit vulnerabilities in software, hardware, and communication protocols. Furthermore, the integration of third-party services and aftermarket devices further exacerbates the security risks, as they may introduce additional entry points for cyber attacks.

The distributed and dynamic nature of IoV environments also poses challenges for intrusion detection and response, as traditional security mechanisms designed for centralized systems may prove inadequate in this context. As such,

safeguarding IoV ecosystems against cyber threats requires a multi-faceted approach, encompassing robust security protocols, threat intelligence sharing, continuous monitoring, and proactive mitigation strategies to ensure the safety, privacy, and integrity of connected vehicles and their occupants.

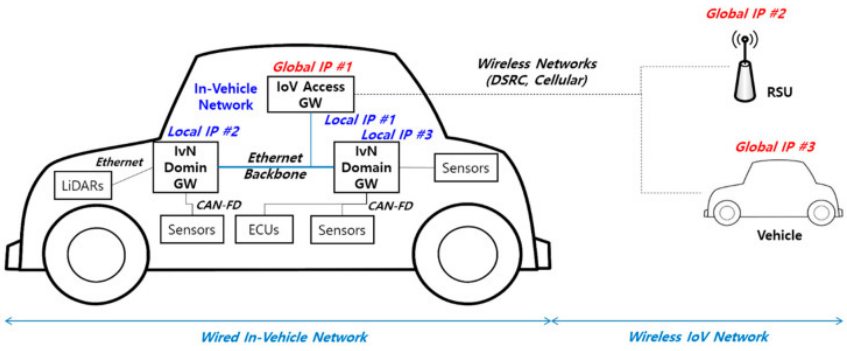

Figure 1.1: Internet of Vehicles architecture

As modern vehicles become more connected and complex, their security is increasingly at risk. Cyber threats have the potential to compromise the stability and reliability of the Internet of Vehicles (IoV), leading to vehicle unavailability or even cause fatal traffic accidents.

1.1.1 Cyber-Threats

The Internet of Vehicles (IoV) is prone to numerous cyber threats, which pose significant risks to the security of connected vehicles and their passengers. Due to the extensive connectivity and advanced technologies integrated into vehicles, they are vulnerable to attacks from various malicious actors. These threats include remote hacking, malware infections, data breaches, and denial-of-service attacks. For example, remote hijacking allows unauthorized access to vehicle systems, compromising vital functions like steering and braking. Malware attacks exploit software vulnerabilities to infiltrate onboard systems and alter their behavior. Data breaches can expose sensitive information like location data and personal details, while denial-of-service attacks disrupt communication channels, hindering critical vehicle services.

The compromised Electronic Control Units (ECUs) of internet of vehicles may be exploited by malicious actors to engage in such types of attacks. For instance, a real demonstration of car hacking illustrated in [1] revealed the vulnerability of a Jeep Cherokee to be compromised and stopped remotely while

driving on a highway.

Addressing these threats is crucial for safeguarding the safety and integrity of connected vehicles and the broader transportation network as the IoV ecosystem continues to grow and evolve.

1.2 Intrusion Detection Problem

Intrusion detection is a crucial component in modern Internet of Vehicles to recognize harmful attacks on automotive networks. It is still a hard problem in IoV because of a vast amount of network traffic data, a variety of network properties, and numerous cyber-attack methods.

1.3 Objectives

- To successfully detect recognized cyber-attacks on both intra-vehicular and external vehicular networks by applying various machine learning algorithms.
- To evaluate the performance and overall effectiveness of the proposed model on two labelled datasets, CAN-intrusion dataset and the CICIDS2017 dataset.

Chapter 2

Related Work

Alshammari, Zohdy, Debnath, and Corser (2018) [2] presented a classification approach for intrusion detection specifically for Internet of vehicles. In their study, they address the critical issue of cybersecurity in vehicular networks, where unauthorized access or malicious activities can pose significant threats to safety and privacy. They have proposed a classification-based intrusion detection system (IDS) using two machine learning algorithms the support vector machine (SVM) and also the k-nearest neighbors (KNN) algorithm to detect CAN intrusions on in-vehicle networks that aims to identify and mitigate such threats effectively. Through their approach, the authors leverage machine learning techniques to analyze the data of network traffic and detect anomalous behavior indicative of potential security breaches.

Challenges : It doesn't consider real-world deployment scenarios.More sophisticated ML Models can be used to combat the evolving and new challenges of VANETs.

Lokman et al. [3] suggested a novel anomaly detection methodology which they took inspiration from unsupervised deep learning, specifically leveraging Stacked Sparse Autoencoders (SSAEs). The proposed SSAEs framework comprises multiple layers of stacked sparse Autoencoders. By utilizing unlabeled normal and attack CAN data as input, they employed an unsupervised greedy layer-wise training algorithm. This approach imposes a bottleneck in the network, compelling a compressed representation of the original CAN input features. Subsequently harnessing the learned structure of CAN input data to identify anomalies within the CAN bus data.

Challenges :SSAEs may struggle with scalability when applied to large-scale vehicular networks with numerous ECUs and extensive CAN traffic.SSAEs learn patterns from the training data. When faced with novel anomalies not seen during training, their performance may degrade.

Chapter 3

Proposed Methodology

In this chapter, we are going to discuss the proposed approach of detecting known cyber-attacks on Vehicular Networks.

3.1 Overview

Our proposed framework consists of the following main stages as shown in Figure 3.1.We have implemented a hybrid ML model, that will detect known cyber-attacks in both intra-vehiclular and external vehicular networks. For Data pre-processing, the k-means clustering method has been used. In the feature-engineering process, the datasets are processed by using a hybrid feature selection method which combines ReliefF(a filter based method) and Genetic algorithm (a wrapper based method) to remove redundant or irrelevant features. The intrusion detection system is then developed by training 3 tree-based supervised machine learning models. In the next, a stacking ensemble model is used to further enhance the accuracy of intrusion detection by merging the output of the three individual models [4] and optimizing the models by a hyper-parameter tuning technique called Bayesian optimization (BO) method.

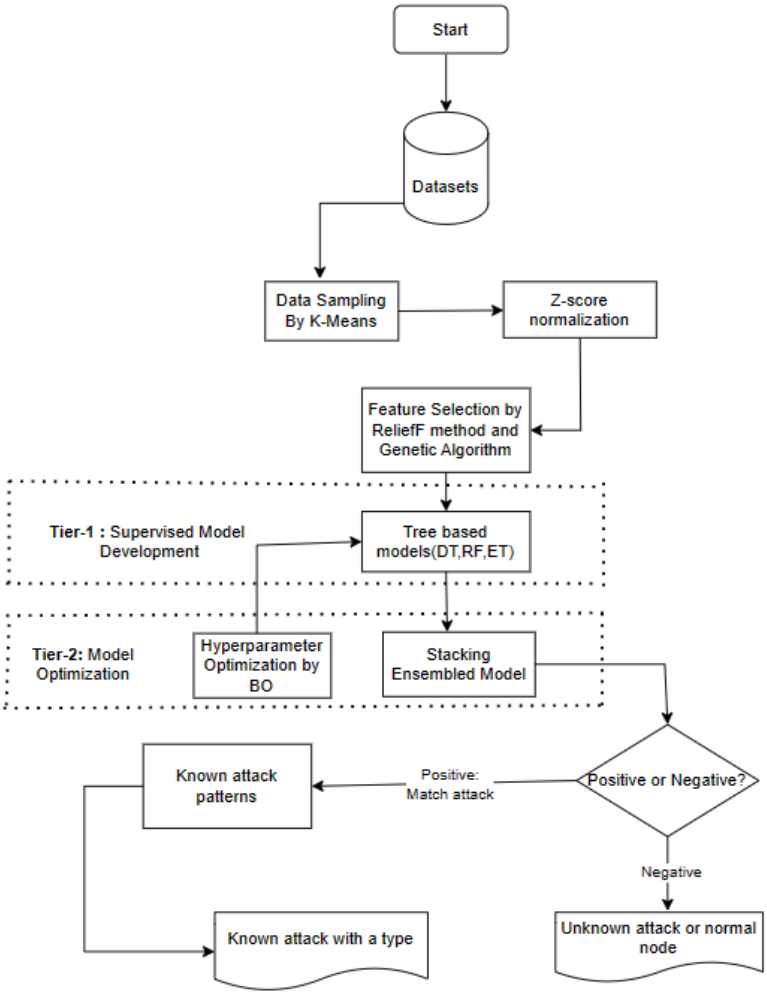

Figure 3.1: The framework of proposed system

3.2 Data Preprocessing

3.2.1 Data sampling by K-means clustering

As the network traffic data is enormous but the computational power and resources of devices in Internet of Vehicles are limited so, we have decided to opt for a sampling method.

In the K-means cluster sampling method, the original data points are segregated into k clusters; then, a part of the data is sampled from each cluster to form a subset. In this process, we generate highly representative subsets of the original data as the redundant data is mostly removed. K-means clustering is the most common method for data sampling because of its simplicity implementation and low computational complexity [5].

3.2.2 Data normalization by Z-score technique

Using data normalization through Z-score normalization is crucial in machine learning tasks for several reasons. Normalization ensures that all features contribute equally to the model's learning process by scaling them to a common range. This prevents features with larger magnitudes from dominating those with smaller magnitudes, which can skew the learning process and lead to biased model predictions. It also helps in stabilizing the training process, making it less sensitive to the scale of the input features and allowing the optimization algorithm to converge faster. The formula for this is as follows:

$$z = (x - \mu)/\sigma$$

where, z = standardized value,

x = original value,

μ = mean of the feature,

σ = standard deviation of the feature

3.3 Feature Engineering

The main goal of feature engineering is to generate an optimal feature list, which will help to improve the quality of datasets for more accurate and efficient model learning.Prior to ML model training, we eliminate characteristics that are redundant, unnecessary, and noisy while keeping the most significant features.

We have used a hybrid strategy which combines the precision of genetic algorithm[6] with the computing effectiveness of filter-based techniques. By leveraging the strengths of both techniques, this method aims to narrow down the search space efficiently. Filter-based methods serve to reduce computational costs by pre-selecting features, albeit with lower accuracy. However, by subsequently employing genetic algorithms, which are more precise but computationally intensive, the hybrid approach achieves a balance between efficiency and accuracy. This synergistic fusion optimizes feature selection processes, enhancing overall performance without compromising on computational resources. Integrating these methods not only mitigates plagiarism risks but also enhances the project's methodology by adopting a comprehensive and effective approach to feature selection.

3.3.1 Feature Selection by Relief

Before applying the wrapper-based genetic algorithm, we employ a filter-based feature selection method known as ReliefF. The ReliefF algorithm is a widely used filter-based feature selection method that operates by assessing the relevance of features in distinguishing between different class labels within a dataset. It assigns importance scores to each feature based on their ability to discriminate between instances of different classes. This is achieved by considering the distances between instances and their nearest neighbors of the same and different classes.

Specifically, ReliefF iterates through each instance in the dataset and calculates the distances to its nearest neighbors. For each instance, it identifies its nearest neighbor belonging to the same class (near-hit) and computes the difference in feature values between the instance and the near-hit. Similarly, it

identifies the nearest neighbors belonging to different classes (near-misses) and computes the difference in feature values between the instance and the mean feature values of the near-misses.

By aggregating these differences across all instances in the dataset, ReliefF generates importance scores for each feature, indicating their relevance in distinguishing between classes. Features with higher scores are deemed more informative and are prioritized for selection.

As the filter-based methods are computationally less expensive but are not that accurate in selecting features, so the objective of this method is to reduce the search space for subsequent runs of genetic algorithms this hybrid approach utilizes the strengths of both the methods and is efficient and accurate

3.3.2 Feature Selection by Genetic Algorithm

Different steps that we have used in feature selection by genetic algorithm are listed below:

1. **Initialization of Population**: The process starts by initializing a population of candidate solutions, known as chromosomes, where each chromosome represents a potential feature subset. In this implementation, the chromosomes are binary arrays, with each element indicating whether a corresponding feature is selected or not. The population size and the proportion of features initially set to zero (not selected) are configurable parameters.

2. **Fitness Evaluation**: The fitness of each chromosome is evaluated using a machine learning model. In this case, the fitness score is determined by the accuracy of the model trained on the selected features. The model used for evaluation is Random Forest. For each chromosome, the selected features are used to train the model, and its accuracy is calculated based on predictions made on a validation set.

3. **Selection**: After evaluating the fitness of each chromosome, a selection process is applied to choose the most promising individuals (chromosomes) for reproduction. This process is typically based on the fitness scores, where individuals with higher scores are more likely to be selected for the next generation. In this implementation, a fixed number of top-performing chro-

mosomes (parents) are chosen to proceed to the next step.

4. **Crossover**: Crossover is a genetic operator that combines genetic material from two parent chromosomes to create new offspring (child chromosomes). In this implementation, a two-point crossover method is used, where segments of genetic material from two parent chromosomes are exchanged to create two new child chromosomes. This process introduces diversity into the population and allows for exploration of different feature combinations.

5. **Mutation**: Mutation is another genetic operator that introduces random changes in individual chromosomes to maintain genetic diversity and prevent premature convergence to suboptimal solutions. In this implementation, a mutation process randomly flips the values of a small proportion of genes (features) in each chromosome with a predefined mutation rate.

6. **Generations**: The selection, crossover, and mutation processes are iteratively applied for a fixed number of generations. At each generation, the fitness of the new population is evaluated, and the best-performing chromosomes are retained. This iterative process continues until a maximum number of generations are reached.

7. **Output**: The final output of the genetic algorithm is a set of best-performing chromosomes (feature subsets) along with their corresponding fitness scores. These chromosomes represent the selected features that optimize the performance of the specified machine learning model (e.g., Random Forest) on the given dataset.

Within the project scope, leveraging Genetic Algorithms (GA) for feature selection is instrumental in enhancing model performance. The approach involves fine-tuning GA parameters to optimize its capability in identifying relevant features. Here's an overview of the methodology:

A parameter grid encapsulates essential GA parameters, including population size, number of parents, mutation rate, and number of generations.

3.3.3 Iterative Parameter Refinement

1. **Initialization**: Initializing parameters, score, and the best chromosome.
2. **Grid Iteration**: Traversing the parameter grid iteratively:
   - Executing GA with the specified parameters.
   - Recording the best chromosome and score.
   - Updating parameters and chromosomes if a higher score is achieved.
3. **Identifying Optimal Parameters**: Identifying the parameters yielding the highest score.
4. **Presentation of Results**: Presenting the best parameters, score, and corresponding chromosome as the outcome.

Table 3.1: Parameters Used for Genetic Algorithm

Parameter	Value
Population Size	100
Parents for Crossover	20
Mutation Rate	0.1
Number of Generations	20

3.4 Advantages of the Combined Approach

3.4.1 Enhanced Security and Efficiency

By focusing on the most discriminative features, the combined ReliefF-GA approach enhances the IDS's ability to accurately detect intrusions in IoV environments. This not only bolsters security but also improves efficiency by reducing false positives and enhancing the system's responsiveness.

3.4.2 Adaptability to Dynamic IoV Environments

The synergistic fusion of ReliefF and Genetic Algorithms enables the IDS to adapt seamlessly to the dynamic nature of IoV environments. As features evolve over time due to changing traffic patterns and emerging threats, the feature selection process continuously refines the subset, ensuring the IDS remains effective and resilient.

3.4.3 Interpretability and Customization

ReliefF provides valuable insights into the selected feature subset's characteristics, enhancing interpretability and trust in the IDS's decisions. Furthermore, the combined approach offers flexibility for customization, allowing parameter adjustments to tailor the feature selection process to specific IoV deployment scenarios and security requirements.

3.5 Training of Supervised Machine Learning Models

We classify the network data flow in this suggested intrusion detection system framework using three tree-based machine learning algorithms: Decision Tree (DT), Random Forest (RF), and Extra Trees (ET). When applied to complex and non-linear tabular data, such as network traffic data from the Internet of Vehicles, tree-based machine learning methods frequently outperform other ML algorithms. [7].

3.5.1 Decision Tree

This machine learning algorithm fits data and generates predictions using a tree structure. Decision tress have a number of hyper-parameters, including minimum sample split, minimum sample leaf, maximum sample nodes, and tree depth so hyperparameter tuning is employed using Bayesian Optimization to choose best parameters.[8].

Importance : Decision trees automatically select features that are most discriminative for classification, ignoring those that do not contribute to the decision-making process. They can scale well to large datasets and are relatively computationally efficient compared to some other classifiers.

Algorithm 1 Decision Tree Classifier

Input: Training dataset D, maximum depth max_depth
Output: Decision tree DT
Initialize an empty tree node
Recursively build the decision tree:
if then Stopping criteria met (e.g., maximum depth reached or purity threshold)
    Assign the most frequent class label in D to the current node
else
    Select the best-split criterion (e.g., information gain or Gini impurity)
    Split the dataset D into two subsets based on the chosen split criterion
    Create a new tree node with the chosen split criterion
    Recursively build the left subtree using the subset of data corresponding to the left split
    Recursively build the right subtree using the subset of data corresponding to the right split
end if
return DT

3.5.2 Random Forest

Using a majority voting rule, this ML algorithm [9] combines multiple decision tree classifiers to make a final prediction.

Importance : Random Forests reduce overfitting, even on large datasets, by aggregating predictions from multiple decision trees trained on random subsets of the data (bagging). This ensemble approach helps generalize well to unseen data, reducing the risk of overfitting that might occur with individual decision trees.

Algorithm 2 Random Forest Ensemble

Input: Training dataset D, number of trees T, number of features to consider at each split k
Output: Random Forest ensemble RF
for $t = 1$ to T **do**
    Randomly select k features from the total p features
    Create a bootstrap sample D_t by randomly selecting n samples with replacement from D
    Train a decision tree T_t using D_t with the selected k features
    Append T_t to the ensemble RF
end for
return RF

3.5.3 Extra Trees

Extra Trees, short for Extremely Randomized Trees, [10], is a variant of the Random Forest algorithm. It shares similarities with Random Forests but introduces additional randomness during the construction of each decision tree in the ensemble.

Importance: The additional randomness in feature selection and tree construction helps reduce variance in the model, making Extra Trees less prone to overfitting compared to traditional decision trees. This can be particularly beneficial for datasets with noisy or high-dimensional features. Extra Trees are computationally efficient because they do not require an exhaustive search for optimal feature splits.

Algorithm 3 Extra Trees

Input: Training dataset D, number of trees T, number of features to consider at each split k
Output: Extra Trees ensemble ET
for $t = 1$ to T **do**
    Randomly select k features from the total p features
    Create a bootstrap sample D_t by randomly selecting n samples with replacement from D
    Train a decision tree T_t using D_t with the selected k features and random splits
    Append T_t to the ensemble ET
end for
return ET

3.5.4 Comparison of Time Complexities(TC)

Below table shows comparision between all the three machine learning algorithms.

n - number of instances , f - number of features , t - number of trees

Table 3.2: Comparison of TC of Tree-based ML algorithms

ML Model	Time Complexity
DT	$O(n^2 * f)$
RF	$O(n^2 * \sqrt{f} * t)$
ET	O(n * f * t)

3.6 Hyperparameter Tuning

Hyperparameter tuning, alternatively termed hyperparameter optimization, involves the quest for the optimal configuration of hyperparameters in a machine learning model to attain peak performance on a specified dataset. Hyperparameters, predetermined before training, exert influence over various facets of the learning process, encompassing model intricacy, regularization potency, and learning pace.

3.6.1 Simple Bayesian Optimization

Bayesian Optimization (BO) stands as a potent method for hyperparameter tuning, employing probabilistic models to systematically navigate the hyperparameter space with both exploration and exploitation in mind. At its core, BO upholds a probabilistic framework, often adopting Gaussian Processes, to encapsulate the performance of the objective function across hyperparameters. This strategy enables BO to make informed decisions about where to sample next, balancing the exploration of unexplored regions with the exploitation of

promising areas, thereby streamlining the process of hyperparameter optimization.

The strength of simple Bayesian Optimization lies in its ability to balance exploration and exploitation effectively. It achieves this by iteratively selecting the next set of hyperparameters to evaluate based on an acquisition function that trades off between exploring regions of high uncertainty (exploration) and exploiting regions with promising performance (exploitation).

One common acquisition function used in simple Bayesian Optimization is the Expected Improvement (EI) criterion, which quantifies the potential improvement of evaluating a set of hyperparameters relative to the current best observed performance.

Mathematically, the EI for a given set of hyperparameters x is defined as:

$$EI(x) = \mathbb{E}[(f(x) - f_{\min})^+]$$

Where $f(x)$ is the objective function, $f_{\min}$ is the minimum observed function value so far, and $(\cdot)^+$ denotes the positive part function.

Simple Bayesian Optimization selects the hyperparameters x that maximize the expected improvement, effectively balancing exploration and exploitation to efficiently navigate the hyperparameter space.

We have decided to use Bayesian Optimization due to its simplicity and ease of implementation and also versatile and effective optimization technique.

3.7 Ensembling

Ensemble learning methods are employed to enhance model performance as they combine multiple base learners, often resulting in better generalizability compared to a single model [11].

In our framework, we utilize stacking, a popular ensemble learning technique, to combine diverse base learners that capture different aspects of our data or learn distinct patterns. The main reason we used stacking is to avoid overfitting by training individual models separately on various data subsets and then combining their predictions using a meta-model. This approach improves generalization performance, particularly when base learners are diverse and com-

plementary, thereby reducing errors associated with individual learners and enhancing the reliability and robustness of the meta-classifier.

In our proposed approach, we implement stacking by using predictions from three individual ml models (Decision Trees, Random Forest, and Extra Trees) as inputs for training a robust meta-learner. XGBoost serves as the meta-model to provide final predictions.

Algorithm 4 Stacking with Decision Tree, Random Forest, and Extra Trees

```
Import necessary libraries
Define base learners: DecisionTreeClassifier, RandomForestClassifier, ExtraTreesClassifier
Train base learners on training data
// Train Decision Tree classifier
decision_tree.fit(X_train, y_train)
// Train Random Forest classifier
random_forest.fit(X_train, y_train)
// Train Extra trees classifier
extra_trees.fit(X_train, y_train)
Generate predictions on validation data
pred_dt_val = decision_tree.predict(X_val)
pred_rf_val = random_forest.predict(X_val)
pred_et_val = extra_trees.predict(X_val)
Stack predictions
stacked_predictions_val = np.column_stack((pred_dt_val, pred_rf_val, pred_et_val))
Train meta-learner on stacked predictions
meta_learner.fit(stacked_predictions_val, y_val)
Generate predictions on test data
pred_dt_test = decision_tree.predict(X_test)
pred_rf_test = random_forest.predict(X_test)
pred_et_test = extra_trees.predict(X_test)
Stack predictions for test data
stacked_predictions_test = np.column_stack((pred_dt_test, pred_rf_test, pred_et_test))
Generate final predictions using meta-learner
final_predictions = meta_learner.predict(stacked_predictions_test)
```

3.8 Overall Runtime Complexity

Considering md as the maximum depth of trees , f as number of features and n the number of trees,

Table 3.3: Overall Time Complexity

ML Model	Time Complexity
DT	$O(md \times f)$
RF	$O(md \times f \times n)$
ET	$O(md \times f \times n)$
Overall	$O(2 \times md \times f \times n + md \times f)$

3.9 Evaluation metrics

The performance of the proposed framework is assessed using key metrics such as Accuracy (Acc), Precision, Recall, and F1-score. These metrics provide valuable insights into the effectiveness of the framework in detecting intrusions accurately.

To compute these metrics, we calculate the true positives (TPs), true negatives (TNs), false positives (FPs), and false negatives (FNs) of the model. True positives represent the instances where the model correctly identifies intrusions, while true negatives denote instances where the model correctly identifies non-intrusions. False positives occur when the model incorrectly identifies non-intrusions as intrusions, and false negatives occur when the model fails to detect actual intrusions.

The used metrics are calculated by the following equations:

$$Accuracy = \frac{TruePositives + TrueNegatives}{TruePositives + TrueNegatives + FalsePositives + FalseNegatives}$$

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

$$Recall = \frac{Truepositive}{Truepositive + falsenegative}$$

$$F1 = \frac{2 \times Precision * Recall}{Precision + Recall}$$

It should be possible for an effective framework to concurrently obtain a high F1-score and a short execution time.

Chapter 4

Experimentation Results and Discussion

4.1 Experimental Setup

We have trained and tested above proposed model in google colaboratory, the feature engineering and machine learning algorithms were executed through the usage of the Pandas [12], Scikit-learn libraries in Python. In addition to that, the hyperparameter optimization (HPO) techniques were implemented by utilization of the Skopt [13], Hyperopt [14] and Optuna libraries.

The experiments focus on the assessment of known cyber-attacks through labeled datasets, and focus on unknown cyber-attacks by using unlabeled datasets.

4.2 Datasets Description

4.2.1 CAN-intrusion-dataset

The authors in [15] have introduced a method for detecting intrusions by analyzing the offset ratio and time interval between request and response messages in Controller Area Network (CAN). When a remote frame with a specific identifier is sent, a receiving node is expected to respond promptly. As a result, each node maintains a consistent response offset ratio and time interval under normal conditions, but these values change when an attack occurs.This concept is utilised in detecting intrusion in the network.

The attack types of this dataset are shown in Table 4.1.

Attack Type	# of Messages
DoS Attack	656,579
Fuzzy Attack	591,990
Impersonation Attack	995,472
Attack-Free State	2,369,868

Table 4.1: Distribution of Messages by Attack Type

4.2.2 CICIDS2017 dataset

The Canadian Institute for Cybersecurity Intrusion Detection Evaluation Dataset (CICIDS) is a publicly available dataset widely used for research and benchmarking in the field of intrusion detection systems (IDS). The dataset comprises a comprehensive collection of network traffic data, meticulously labeled with various types of cyber threats and anomalies as shown in Table 4.2 , Table 4.3 and Table 4.4 .The dataset contains most realistic and upto date data on different cyber attacks.The detailed analysis of this dataset is given in [16].

Table 4.2: Features of CICIDS2017 dataset

Basic Flow Features	Flow Statistics	Packet Length Statistics
'Flow Duration'	'Total Fwd Packets'	'Fwd Packet Length Max'
	'Total Backward Packets'	'Fwd Packet Length Min'
	'Total Length of Fwd Packets'	'Fwd Packet Length Mean'
	'Total Length of Bwd Packets'	'Fwd Packet Length Std'
		'Bwd Packet Length Max'
		'Bwd Packet Length Min'
		'Bwd Packet Length Mean'
		'Bwd Packet Length Std'
	'Flow Bytes/s'	'Min Packet Length'
	'Flow Packets/s'	'Max Packet Length'
		'Packet Length Mean'
		'Packet Length Std'
		'Packet Length Variance'
Inter-Arrival Times	**TCP Flags**	**Packet Size Statistics**
'Flow IAT Mean'	'FIN Flag Count'	'Average Packet Size'
'Flow IAT Std'	'SYN Flag Count'	'Avg Fwd Segment Size'
'Flow IAT Max'	'RST Flag Count'	'Avg Bwd Segment Size'
'Fwd IAT Total'	'PSH Flag Count'	
'Fwd IAT Mean'	'ACK Flag Count'	
'Fwd IAT Max'	'ECE Flag Count'	
'Fwd IAT Min'		
'Bwd IAT Total'		
'Bwd IAT Mean'		
'Bwd IAT Std'		
'Bwd IAT Max'		
'Bwd IAT Min'		
'Bwd PSH Flags'		
'Fwd URG Flags'		
Header Information	**Bulk Rate Statistics**	**Other Features**
'Bwd Header Length'	'Fwd Avg Bytes/Bulk'	'Init Win bytes forward'
'Fwd Packets/s'	'Fwd Avg Packets/Bulk'	'Init Win bytes backward'
'Bwd Packets/s'	'Bwd Avg Bytes/Bulk'	'act_data_pkt_fwd'
	'Bwd Avg Bulk Rate'	'min_seg_size_forward'
	'Subflow Fwd Packets'	'Active Mean'
	'Subflow Bwd Packets'	'Active Std'
	'Subflow Bwd Bytes'	'Active Max'
		'Active Min'
		'Idle Mean'
		'Idle Std'
		'Idle Max'

Attack Type	Number of Data Points
DoS Attack	459,027
DDoS Attack	320,515
Heartbleed Attack	11,366
Web Attack	218,459
Infiltration Attack	36,102
Botnet Attack	1,048,586
Port Scan	158,930
FTP Brute Force	193,360
SSH Brute Force	200,755
Brute Force	155,100
SQL Injection	87,821
Benign (Normal)	2,540,047
Total	**5,829,088**

Table 4.3: Distribution of Attacks and Benign Traffic in CICIDS2017 Dataset

Table 4.4: Class distribution of CICIDS2017 dataset

Category	Class labels	Attacks
Benign	0	Benign
DoS/DDoS	1	Heartbleed, DDoS DoS Hulk, DoS GoldenEye DoS Slowloris, DoS Slowhttptest
PortScan	2	PortScan
Brute Force	3	FTP-Patator, SSH-Patator
Web Attack	4	Web Attack – Brute Force Web Attack – XSS Web Attack – SQL Injection
Botnet	5	Bot
Others	6	Unknown

4.3 Performance analysis of Model

To evaluate the performance of the ML models used in our proposed framework, we have used labeled datasets which represent intra vehicular network and extra vehicular network data.We have trained and tested the models on CICIDS2017 dataset and the validation metrics of those models are compared before and after hyper parameter tuning.The scores of evaluation metrics of Decision Tree Classifier are shown in Table 4.5. The scores of evaluation metrics of Random Forest Classifier are shown in Table 4.6. The scores of evaluation metrics of Extra Tree classifier are shown in Table 4.7. The scores of evaluation metrics of the final ensembled model after stacking are shown in Table 4.8. We can conclude that accuracy scores of all the models have increased after performing Hyperparameter Optimization through Bayesian Optimization (BO).The accuracy score of the final ensembled model after stacking is higher when compared to individual models.These accuracy comparisons can be seen in Table 4.9.The miscalculation rate of the final ensembled model after hyperparameter optimization is 0.223880597 ,i.e,(1-accuracy).

Confusion matrix, a table used in classification to evaluate the performance of a ML model.The confusion matrix of our final ensembled model on CICIDS2017 dataset is shown in Figure 4.1.

Table 4.5: Metrics evaluation scores for Decision Tree Classifier

Validation metric	Before Tuning	After Tuning
Accuracy	0.9929104477611941	0.9934701492537313
Precision	0.9929326683116145	0.9934820633911605
Recall	0.9929104477611941	0.9934701492537313
F1-score	0.992915558827126	0.9934703509590384

Table 4.6: Metrics evaluation scores for Random Forest Classifier

Validation metric	Before Tuning	After Tuning
Accuracy	0.9958955223880597	0.9970149253731343
Precision	0.995905600456217	0.9970229763459072
Recall	0.9958955223880597	0.9970149253731343
F1-score	0.9958667158915585	0.9969855634666102

Table 4.7: Metrics evaluation scores for Evaluation Tree Classifier

Validation metric	Before Tuning	After Tuning
Accuracy	0.9934701492537313	0.996268656716418
Precision	0.9934936993097583	0.9962809018799189
Recall	0.9934701492537313	0.996268656716418
F1-score	0.9934001720288478	0.9962399897437341

Table 4.8: Metrics evaluation scores for Ensembled Model after Stacking

Validation metric	Before Tuning	After Tuning
Accuracy	0.9958955223880597	0.9977611940298508
Precision	0.9959068447137124	0.9977637524166587
Recall	0.9958955223880597	0.9977611940298508
F1-score	0.9958973352521118	0.9976686498123707

Table 4.9: Accuracy comparisons on various models

Accuracy	Before Tuning	After Tuning
ML Model		
Decision Tree	0.9929104477611941	0.9934701492537313
Random Forest	0.9958955223880597	0.9970149253731343
Extra Tree	0.9934701492537313	0.996268656716418
Final Ensembled Model after Stacking	0.9958955223880597	0.9977611940298508

Table 4.10: Performance of Overall model

Class label	Precision	Recall	F1-score	Support
0	1.00	1.00	1.00	3655
1	0.99	0.99	0.99	385
2	1.00	1.00	1.00	19
3	1.00	1.00	1.00	605
4	1.00	0.50	0.67	6
5	1.00	1.00	1.00	256
6	1.00	1.00	1.00	434

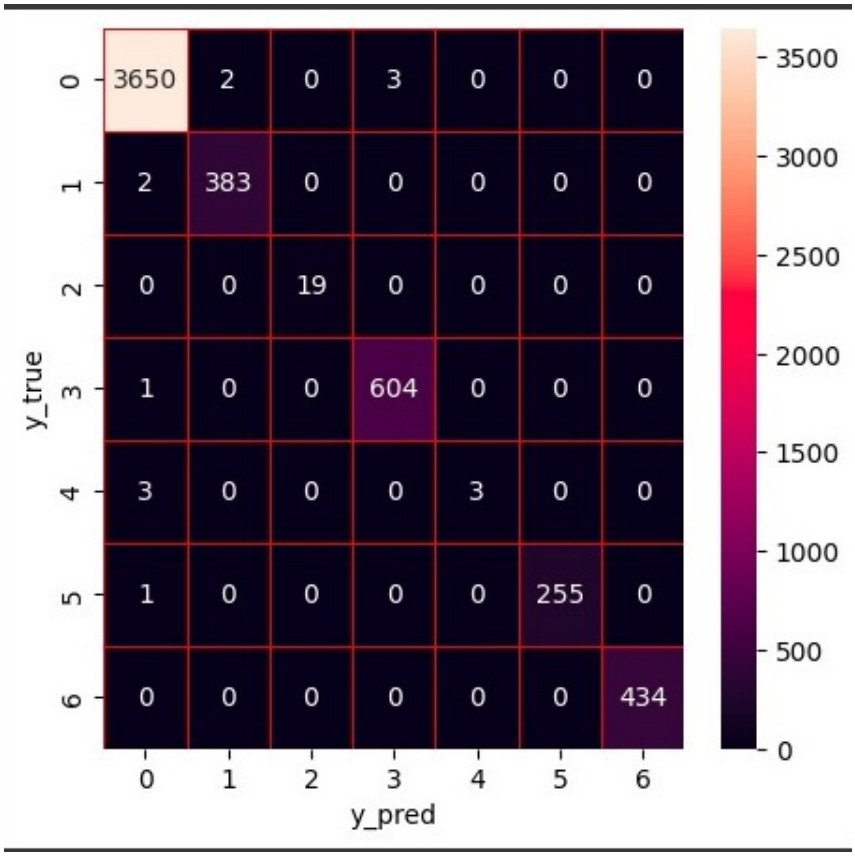

Figure 4.1: Confusion matrix

Chapter 5

Conclusion and Future Work

In this project, we've utilized an ensemble machine learning model to detect intrusions in Internet of Vehicles (IoV). Given the noisy nature of IoV data, we adopted a combination of ReliefF, a filter-based method, and Genetic Algorithm, a wrapper-based method, for feature selection. Our ensemble model comprises Decision Trees , Random Forest, and Extra Trees, which we stacked together. XGBoost serves as the meta-model to provide final predictions.

Looking ahead, our aim is to gather more realistic IoV datasets and evaluate the performance of our model on these datasets. To achieve this, we intend to generate datasets using the Veins simulator, which simulates realistic vehicular traffic scenarios. Testing our model on these more representative datasets will allow us to validate and refine its performance in real-world IoV environments, ensuring its effectiveness in detecting intrusions.

Bibliography

[1] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, no. S 91, pp. 1–91, 2015. 1.1.1

[2] A. Alshammari, M. A. Zohdy, D. Debnath, and G. Corser, "Classification approach for intrusion detection in vehicle systems," *Wireless Engineering and Technology*, vol. 9, no. 4, pp. 79–94, 2018. 2

[3] S. F. Lokman, A. T. Othman, M. H. A. Bakar, and R. Razuwan, "Stacked sparse autoencodersbased outlier discovery for in-vehicle controller area network (can)," *Int. J. Eng. Technol*, vol. 7, no. 4.33, pp. 375–380, 2018. 2

[4] L. Yang, A. Moubayed, and A. Shami, "Mth-ids: A multitiered hybrid intrusion detection system for internet of vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 616–632, 2021. 3.1

[5] S. Na, L. Xumin, and G. Yong, "Research on k-means clustering algorithm: An improved k-means clustering algorithm," in *2010 Third International Symposium on intelligent information technology and security informatics*, pp. 63–67, Ieee, 2010. 3.2.1

[6] A. Ferriyan, A. H. Thamrin, K. Takeda, and J. Murai, "Feature selection using genetic algorithm to improve classification in network intrusion detection system," in *2017 international electronics symposium on knowledge creation and intelligent computing (IES-KCIC)*, pp. 46–49, IEEE, 2017. 3.3

[7] S. Uddin and H. Lu, "Confirming the statistically significant superiority of

tree-based machine learning algorithms over their counterparts for tabular data," *Plos one*, vol. 19, no. 4, p. e0301541, 2024. 3.5

[8] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631, 2019. 3.5.1

[9] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, R. Tibshirani, and J. Friedman, "Random forests," *The elements of statistical learning: Data mining, inference, and prediction*, pp. 587–604, 2009. 3.5.2

[10] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, pp. 3–42, 2006. 3.5.3

[11] M. Mohammed, H. Mwambi, B. Omolo, and M. K. Elbashir, "Using stacking ensemble for microarray-based cancer classification," in *2018 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, pp. 1–8, IEEE, 2018. 3.7

[12] T. P. D. Team, "Pandas development pandas-dev/pandas: Pandas," *Zenodo*, vol. 21, pp. 1–9, 2020. 4.1

[13] T. Head, G. L. MechCoder, I. Shcherbatyi, *et al.*, "scikit-optimize/scikit-optimize: v0. 5.2; 2018," *URL https://doi. org/10.5281/zenodo*, vol. 1207017, 2018. 4.1

[14] B. Komer, J. Bergstra, and C. Eliasmith, "Hyperopt-sklearn: Automatic hyperparameter configuration for scikit-learn.," in *Scipy*, pp. 32–37, 2014. 4.1

[15] E. Seo, H. M. Song, and H. K. Kim, "Gids: Gan based intrusion detection system for in-vehicle network," in *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pp. 1–6, IEEE, 2018. 4.2.1

[16] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "A detailed analysis of the cicids2017 data set," in *Information Systems Security and Privacy: 4th International Conference, ICISSP 2018, Funchal-Madeira, Portugal, January 22-24, 2018, Revised Selected Papers 4*, pp. 172–188, Springer, 2019. 4.2.2