# Principles of Cybersecurity

## Final Project Report

## On

## "Intrusion Detection System on The Internet of Vehicles"

From Team NTS

**Neeraj Kumar Kondaparthi**

**Tushar Nayan**

**Srimant Kumar Mohanty**

## Table of Contents

## 1. Abstract

Modern automobiles, particularly autonomous and connected vehicles, have incorporated a growing number of features through links and communications with other vehicles, intelligent devices, and infrastructures. As the Internet of Vehicles (IoV) becomes more interconnected, network intrusions have a greater chance of occurring. To protect IoV systems from cyber threats, machine learning (ML) techniques have been used to construct intrusion detection systems (IDSs) that can identify damaging cyber-attacks. In the project we have used a framework called the Leader Class and Confidence Decision Ensemble (LCCDE) to effectively identify various forms of assaults in IoV networks. For each class or type of assault, it is built by selecting the top-performing machine learning model from among five sophisticated ML methods (XGBoost, LightGBM, AdaBoost, GradBoost, and CatBoost). The detection of different kinds of cyberattacks is then accurately decided upon using the class leader models and their prediction confidence ratings. The usefulness of the proposed LCCDE for intrusion detection on both intra-vehicle and external networks is demonstrated by using public IoV security dataset (CICIDS2017).

## 2. Introduction

The rapid development of the Internet of Things (IoT) and the Internet of Vehicles (IoV) technologies has led to the replacement of traditional vehicles with network-controlled automobiles, such as Autonomous Vehicles (AVs) and Connected Vehicles (CVs). Both external networks and intravehicular networks (IVNs) are often used in IoV systems. The Controller Area Network (CAN) bus is the essential communication infrastructure for Electronic Control Units (ECUs) to conduct a range of functions on the IVN. On the other hand, external vehicular networks use Vehicle-To-Everything (V2X) technology to connect intelligent vehicles with other Internet of Things (IoT) things like smart devices, smart units, and roadside infrastructure.

The increased network connectivity of vehicle networks has given rise to a number of security vulnerabilities as a result of the extended network attack surfaces of IoV systems. Additionally, due to the short length of CAN packets, no authentication or encryption mechanisms are used in their processing. Cybercriminals can insert malicious messages into IVNs and carry out a number of attacks, including DoS, fuzzy, BOT Attack and spoofing attacks, in the absence of fundamental security measures. Contrarily, the development of communication between linked automobiles and outside networks has exposed these vehicles to a variety of common cyber-attacks.

Wi-Fi, 5G, Bluetooth, cellular, as well as V2V, V2X, and services are just a few of the network technologies that make up the Internet of Things (IoV). Hardware, software, and services are also included. In the IoV industry, OEMs, technology providers, mobile operators, and component vendors all have plenty of space to expand. The Internet of Vehicles was created as a result of the introduction of the notion of a VANET, or vehicular ad hoc network, which combines vehicle-to-vehicle and vehicle-to-infrastructure communication architectures. The IoV's vehicles are each intelligent objects with a variety

of computer resources, command centers, and sensor platforms that are interconnected with each other through a V2X communication architecture.

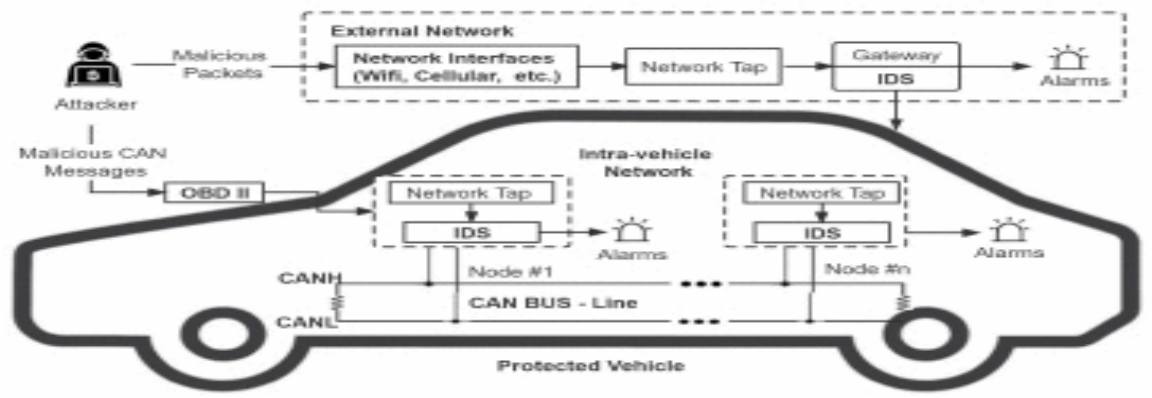


*Figure 1* The IDS-protected vehicle architecture

Figure depicts the IoV attack possibilities, including IVN and external network attacks. Intrusion detection systems, or IDSs, have been developed as potential defenses against cyberattacks on Internet of Vehicles (IoV) systems and smart automobiles. IDS installation in IoV systems is also shown in Fig. 1. To stop malicious CAN communications and protect IVNs, IDSs can be installed on top of the CAN-bus. In order to detect malicious packets arriving from outside networks, the gateways can additionally include IDSs.

ML-driven IDSs in IoV applications have lately piqued the interest of researchers and automakers due to the advancement of ML techniques. Through the analysis of network traffic data, ML techniques are frequently used to create classifier-based IDSs that can distinguish between typical network activity and various cyberattacks.

It is typical to see that the prediction performance of various ML models vary considerably for different types of cyber-attack detection when applying ML models to IDS systems. This work suggests a novel ensemble method known as the Leader Class and Confidence Decision Ensemble (LCCDE)1 to get the best performance on all types of attack detection. Five of the most sophisticated gradient-boosting machine learning approaches are integrated: Extreme Gradient Boosting (XGBoost), Light Gradient Boosting Machine (LightGBM), AdaBoost, GradBoost, and Categorical Boosting (CatBoost). By identifying the top-performing basic ML model with the highest level of prediction confidence for each class, LCCDE seeks to optimize model performance.

The primary contributions made in this report are as follows:

- It suggests a unique ensemble framework called LCCDE for efficient intrusion detection in IoVs employing gradient-boosting ML algorithms, class leader and confidence decision strategies, and decision-boosting decision-making techniques.
- The Car-Hacking and CICIDS2017 datasets, which provide IVN and external network data, respectively, are used to assess the proposed system.

- It contrasts the performance of the suggested model with that of other cutting-edge techniques.

The remaining sections of the essay are structured as follows. The related work on IoV intrusion detection using ML and ensemble models is introduced in Section II. The suggested LCCDE framework is presented in full in Section III. The experimental findings are presented and discussed in Section IV. Section V brings the paper to a close.

## 3. Literature Review

As the number of intelligent cars has recently increased, so has the development of ML models as effective solutions for IoV intrusion detection and security enhancement. Song et al. released a deep convolutional neural network model framework for in-vehicle networks intrusion detection. It does well in tests using the Car-Hacking dataset. Zhao et al. suggested a simple, deep neural network model-based IDS framework for IoT devices. Principal Component Analysis (PCA) is also used to reduce feature dimensionality and computational cost.

The CAN-Intrusion and CICIDS2017 datasets demonstrate the high degree of accuracy of the stacking ensemble model. Three deep learning models—Deep Neural Networks (DNN), Long Short-Term Memory (LSTM), and Deep Belief Networks (DBN)—were proposed by Elmasry et al. as an ensemble model for network intrusion detection. All Predict Wisest Decides (APWD), a revolutionary ensemble IDS architecture developed by Chen et al., allows for the detection of intrusions and decision-making based on the wisest model for each class. On the NSL-KDD dataset, it was only able to achieve an accuracy of 79.7%.

## 4. Methodology

### i. System Overview

The purpose of this work is to develop a framework for an ensemble IDS that can effectively identify various types of attacks against both IVN and outside vehicular networks. Figure 2 depicts the general structure of the proposed system, which consists of model training and model prediction. Three potent machine learning (ML) algorithms—XGBoost, LightGBM, and CatBoost—are trained on the IoV traffic dataset to produce the leader models for all assault classes and types. At the model prediction stage, attacks are precisely identified by using the class leader models and their prediction confidences. Information on the algorithm can be found in this section.

Here, you should outline the procedures followed to carry out the project. Put your actions in the past tense and be as specific as you can in your description. Data measurement and analysis, program and algorithm design and implementation, problem survey and reflection, etc. are a few examples of approaches.

## ii. Base Machine Learning Models

A decision tree (DT) is a fundamental machine learning method that applies the divide and conquer strategy to decisions by using a tree structure. In DTs, the leaves denote the result classes while the decision nodes represent the decision tests. An iterative DT approach called Gradient Boosting Decision Tree (GBDT) builds many DTs and aggregates their prediction results. Five sophisticated gradient-boosting algorithms—XGBoost, LightGBM, AdaBoost, GradBoost, and CatBoost—have been created and are frequently utilized in numerous applications to enhance the performance of fundamental GBDTs. The LCCDE ensemble framework is constructed in the proposed system using these five gradient-boosting algorithms.

- A well-known gradient-boosting DT method called **Extreme Gradient Boosting (XGBoost)** was created to increase the speed and effectiveness of GBDTs. To minimize the loss function and lessen over-fitting, XGBoost employs a regularization term and a Second-Order Taylor Approximation for the summing of the squared errors. When K is the number of trees, d is the maximum tree depth, ||x|| is the number of non-missing samples, and n is the size of the data, XGBoost has a low computational complexity of $O(Kd \|x\| \log n)$. For faster model learning, XGBoost supports parallel execution.
- **Light Gradient Boosted Machine (LightGBM)** is a rapid and dependable multi-DT ensemble ML model. One of LightGBM's primary advantages over other ML algorithms is its capability to handle large-scale and high-dimensional data. Exclusive Feature Bundling (EFB) and Gradient-based One Side Sampling (GOSS) are the two primary methods used in LightGBM. GOSS is a down sampling strategy that only preserves data samples with big gradients and randomly tosses out samples with tiny gradients in order to speed up model training and consume less memory. By combining mutually exclusive features into bundles as single features, EFB is a feature engineering technique that decreases feature size and improves the efficiency of model training.
- **Gradient Boosting (GradBoost)** is based on the hypothesis that the overall prediction error is reduced when earlier models are combined with the best upcoming model. The key idea is to specify the expected outcomes for this following model in order to minimize error. Gradient boosting is advantageous for both classification and regression. In the beginning, Gradient Boost creates a single leaf rather than a tree or a stump. This leaf provides a preliminary estimate of the weights of each sample. The average value is the first prediction when attempting to predict a continuous metric, such as MPG, after which Gradient Boosting constructs a tree. This tree dwarfs a stump in size. The size of a tree is still constrained by Gradient Boost, though.
- **Adaptive Boosting (AdaBoost)**, often known as adaptive boosting, is a boosting algorithm. By focusing more on the underfitted training instances produced by the prior model, this algorithm corrects the errors introduced by its predecessor. As a result, the harder instances will always be the focus when there is a new predictor. Similar to Gradient Descent in sound, AdaBoost is a sequential learning technique that gradually improves the ensemble of predictors rather than adjusting the parameters of a single

predictor to minimize the cost function. As a result of the fact that each predictor can only be trained after the previous one has been trained and evaluated, this technique has the significant drawback that the model cannot be parallelized. From the training data, Boost first creates a brief tree called a stump.

- **Categorical Boosting (CatBoost)** is yet another complex gradient-boosting method designed to more accurately assess category features. CatBoost differs from existing gradient-boosting models in three crucial ways: symmetric trees, ordered boosting, and native feature support. All nodes in symmetric trees that have leaves divided according to the same rules as in previous trees receive the pair of feature splits with the lowest loss. By employing symmetric trees, over-fitting can be reduced while model prediction speed is raised. Overfitting on small datasets is prevented with ordered boosting by training a model on a portion of the data and computing residuals on a different portion of the data.

The following are the main justifications for using XGBoost, LightGBM AdaBoost, GradBoost, and CatBoost as base learners:

- ♦ All five of these ML models are strong ensemble models that have performed admirably in numerous data analytics applications.
- ♦ By avoiding the requirement for further feature engineering, these five ML models can automatically create feature relevance scores and choose features throughout their training process, saving time and resources.
- ♦ These five machine learning models are quick and have a manageable level of computational complexity. They all also enable parallel processing and Graphics Processing Unit (GPU) execution, which can accelerate model learning even more.
- ♦ These five ML methods incorporate randomness during the model-building process, allowing people to create a strong ensemble model with a wide range of variables and excellent generalizability.

### iii. LCCDE: Proposed Ensemble Algorithm

Different ML models frequently perform differently on various tasks for attack detection. One ML model might perform best at detecting the first type of attack (such as DoS assaults), while another ML model would perform better at detecting the second type of attack (such as sniffer attacks), when different ML models are applied to the same network traffic dataset. For the purpose of detecting all types of attacks, this work aims to propose an ensemble framework that can achieve optimal model performance. To enhance learning performance and generalizability, ensemble learning is an approach that mixes numerous fundamental ML models.

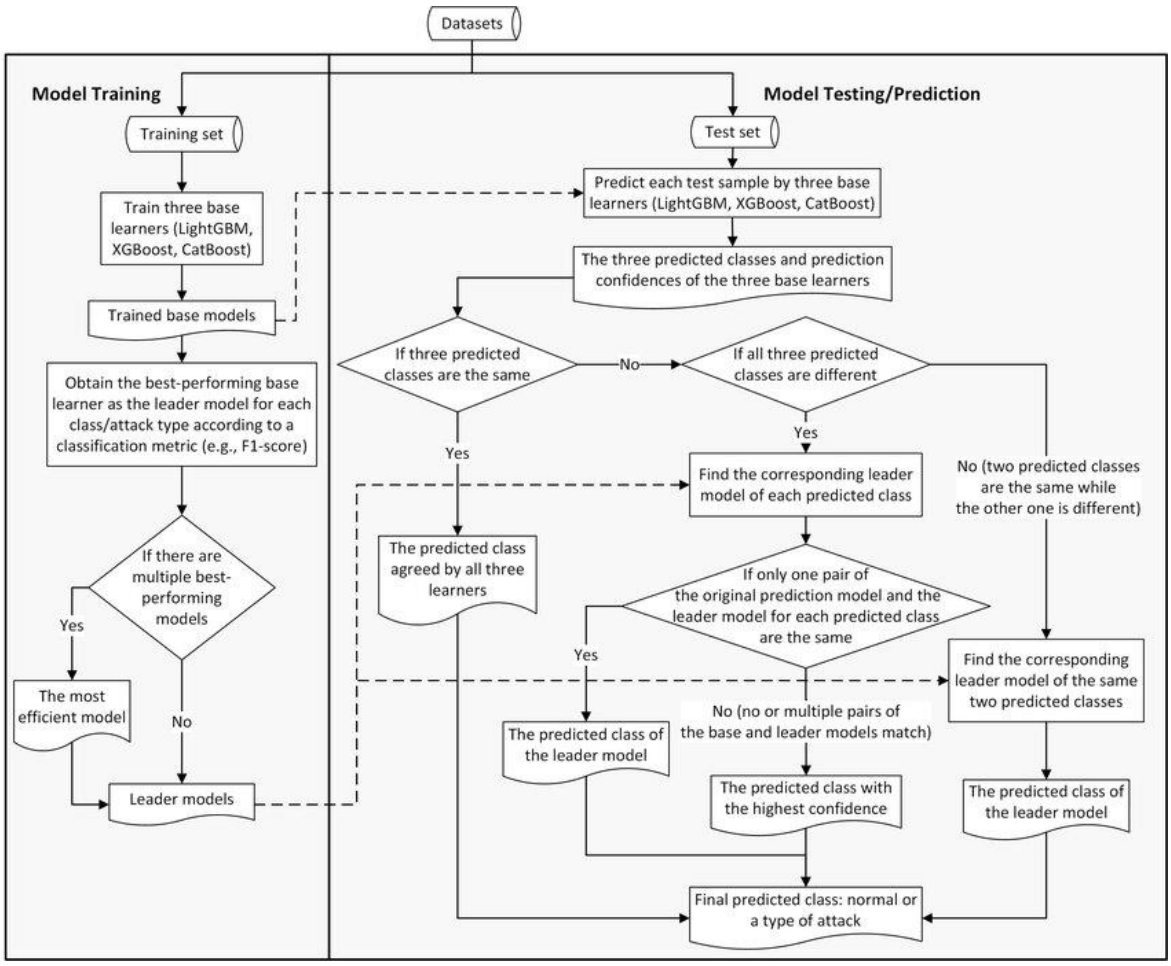


*Figure 2* The Framework of the used LCCDE model

Model training and model prediction are the two stages of the proposed LCCDE framework's procedure, which is shown in Figure 2. The following procedures are used by the LCCDE framework to obtain leader models for all classes throughout the training phase:

- *Develop five fundamental learners.* To create base learners, the five basic ML models (XGBoost, LightGBM, AdaBoost, GradBoost, and CatBoost) are trained on the training data.
- *Assess fundamental students*. Cross-validation and F1-scores are used to assess the performance of the five ML models for each class (normal or an attack type), respectively. Because it is a comprehensive performance metric and performs well with unbalanced datasets, F1- scores are chosen.
- *Choose a leader role model for each class*. The leader model for each class is determined by the top-scoring ML model with the highest F1 score. If numerous top-performing ML models have the same highest F1 scores, the most efficient ML model with the fastest speed is chosen as the final leader model.

- ♦ After the training process, model prediction is done using the trained leader models for all classes. At the model prediction stage, the LCCDE framework predicts each test sample using the following methods:
- ♦ *Form preliminary predictions*. Initial predictions are made using the five trained base ML models that were obtained from the training process. For subsequent studies, their predicted classes and associated prediction confidences are kept. A probability value called confidence is used to express how certain the model is about its forecasts.
- *Verify that the three anticipated classes match*. The predicted class is chosen by all three base learners as the final predicted class is used if they are the same.
- *Determine whether the three expected classes differ*. If they are all different, the matching leader model for each projected class is compared to the base learner that predicted this class. If only one pair of the leader and base models agree, the projected class with the highest prediction confidence is utilized as the final predicted class; otherwise, their predicted class is used.
- *Verify whether two anticipated classes are identical and the third is distinct.* If so, the final prediction is made using the appropriate leader model of the same two predicted classes.

Specifically, LCCDE bases its attack detection on the following three guiding principles:

- It creates first projected classes using the trained ML models.
- It bases its conclusions on the leader models for each class.
- When there are several leader models for various classes, the leader model with the highest forecast confidence is chosen to guide final judgments.

## 5. Results

We got some promising result here. We could get an accuracy of almost 99%. We trained all five models with the CICIDS2017 dataset. The dataset is huge, so we took almost 27k datapoints out of that and we split the dataset for the test and training. We split the data into 80% and 20% for training and testing respectively.

| | Flow Duration | Total Fwd Packets | Total Backward Packets | Total Length of Fwd Packets | Total Length of Bwd Packets | Fwd Packet Length Max | Fwd Packet Length Min | Fwd Packet Length Mean | Fwd Packet Length Std | Bwd Packet Length Max | ... | min_seg_size_forward | Active Mean | Active Std | Active Max | Active Min | Idle Mean | Idle Std | Idle Max | Idle Min | Label |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 5.416666e-07 | 0.0 | 0.000003 | 4.651163e-07 | 9.153974e-09 | 0.000242 | 0.002581 | 0.00101 | 0.0 | 0.000307 | ... | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 |
| 1 | 5.416666e-07 | 0.0 | 0.000003 | 4.651163e-07 | 9.153974e-09 | 0.000242 | 0.002581 | 0.00101 | 0.0 | 0.000307 | ... | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 |
| 2 | 4.416666e-07 | 0.0 | 0.000003 | 4.651163e-07 | 9.153974e-09 | 0.000242 | 0.002581 | 0.00101 | 0.0 | 0.000307 | ... | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 |
| 3 | 7.499999e-07 | 0.0 | 0.000003 | 4.651163e-07 | 9.153974e-09 | 0.000242 | 0.002581 | 0.00101 | 0.0 | 0.000307 | ... | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 |
| 4 | 7.249999e-07 | 0.0 | 0.000003 | 4.651163e-07 | 9.153974e-09 | 0.000242 | 0.002581 | 0.00101 | 0.0 | 0.000307 | ... | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 |

5 rows × 78 columns

*Figure 3* Overview of Dataset

In the dataset the data distribution was not uniform, so we have to do data sampling using SMOTE technique. Which would help model trained in a regularized way without getting under-fitting.

Below are the images of model accuracies of all five models.

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      3656
           1       0.99      0.99      0.99       387
           2       1.00      1.00      1.00        14
           3       1.00      1.00      1.00       612
           4       1.00      0.75      0.86         8
           5       0.99      1.00      0.99       231
           6       1.00      1.00      1.00       452

    accuracy                           1.00      5360
   macro avg       1.00      0.96      0.98      5360
weighted avg       1.00      1.00      1.00      5360

Accuracy of LightGBM: 0.9973880597014926
Precision of LightGBM: 0.9973964590872451
Recall of LightGBM: 0.9973880597014926
Average F1 of LightGBM: 0.9973629910588964
F1 of LightGBM for each type of attack: [0.99822283 0.99224806 1.         0.99836334 0.85714286 0.99354839
 0.99778271]
```

*Figure 4* LightGBM Model F-1 Score

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      3656
           1       1.00      0.99      0.99       387
           2       1.00      1.00      1.00        14
           3       1.00      1.00      1.00       612
           4       1.00      0.75      0.86         8
           5       0.99      1.00      0.99       231
           6       1.00      1.00      1.00       452

    accuracy                           1.00      5360
   macro avg       1.00      0.96      0.98      5360
weighted avg       1.00      1.00      1.00      5360

Accuracy of XGBoost: 0.9979477611940298
Precision of XGBoost: 0.9979549065880012
Recall of XGBoost: 0.9979477611940298
Average F1 of XGBoost: 0.997921195725638
F1 of XGBoost for each type of attack: [0.99863313 0.99351492 1.         1.         0.85714286 0.9935483
 0.99778271]
```

*Figure 5* XGBoost Model F-1 Score

```
              precision    recall  f1-score   support

           0       0.68      1.00      0.81      3656
           1       0.00      0.00      0.00       387
           2       0.00      0.00      0.00        14
           3       0.00      0.00      0.00       612
           4       0.00      0.00      0.00         8
           5       0.00      0.00      0.00       231
           6       0.00      0.00      0.00       452

    accuracy                           0.68      5360
   macro avg       0.10      0.14      0.12      5360
weighted avg       0.47      0.68      0.55      5360

Accuracy of AdaBoost: 0.682089552238806
Precision of AdaBoost: 0.46524615727333485
Recall of AdaBoost: 0.682089552238806
Average F1 of AdaBoost: 0.5531764425432729
F1 of AdaBoost for each type of attack: [0.81100266 0.         0.         0.         0.         0.
 0.        ]
```

*Figure 6* AdaBoosting Model F-1 Score

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      3656
           1       0.99      0.99      0.99       387
           2       1.00      1.00      1.00        14
           3       1.00      1.00      1.00       612
           4       1.00      0.75      0.86         8
           5       0.99      1.00      0.99       231
           6       1.00      0.99      0.99       452

    accuracy                           1.00      5360
   macro avg       1.00      0.96      0.98      5360
weighted avg       1.00      1.00      1.00      5360

Accuracy of CatBoost: 0.9966417910447761
Precision of CatBoost: 0.9966457193764396
Recall of CatBoost: 0.9966417910447761
Average F1 of CatBoost: 0.99661603515491
F1 of CatBoost for each type of attack: [0.99781241 0.99353169 1.         0.99673203 0.85714286 0.99137931
 0.9944629 ]
```

*Figure 7* CatBoost Model F-1 Score

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      3656
           1       1.00      0.98      0.99       387
           2       1.00      1.00      1.00        14
           3       1.00      1.00      1.00       612
           4       1.00      0.62      0.77         8
           5       0.99      1.00      0.99       231
           6       1.00      1.00      1.00       452

    accuracy                           1.00      5360
   macro avg       1.00      0.94      0.96      5360
weighted avg       1.00      1.00      1.00      5360

Accuracy of GradBoost: 0.9968283582089552
Precision of GradBoost: 0.9968408780693236
Recall of GradBoost: 0.9968283582089552
Average F1 of GradBoost: 0.996762445603187
F1 of GradBoost for each type of attack: [0.99767791 0.9921875  1.         0.99672668 0.76923077 0.99354839
 0.99889258]
```

*Figure 8* GradBoost Model F-1 Score

Below image shows the LCCDE model classification of leading performing model for different attacks. Which is a better approach comparing to using a single model.

**Leading Model for Each Type of Attack:**

- 0 BENIGN: XGBClassifier
- 1 Bot: CBClassifier
- 2 BruteForce: LGBMClassifier
- 3 DoS: XGBClassifier
- 4 Infiltration: LGBMClassifier
- 5 PortScan: LGBMClassifier
- 6 WebAttack: GBClassifier

*Figure 9* LCCDE model prediction of specified model attacks

```
Accuracy of LCCDE: 0.9981343283582089
Precision of LCCDE: 0.9981409642565935
Recall of LCCDE: 0.9981343283582089
Average F1 of LCCDE: 0.9981079075490825
F1 of LCCDE for each type of attack: [0.99876965 0.99351492 1.         1.         0.85714286 0.99354839
 0.99889258]
```

*Figure 10* LCCDE Model F-1 Score

## 6. Conclusion

To improve and protect IoV systems, which is day by day becoming autonomous, network heavy and sophisticated, it needs another eye to look after the vulnerability it can be a victim of. Machine Learning application into this code heavy software environment is a good approach to solve the problem. So, we have used LCCDE model which have used five advanced high performing Gradient Boosting Decision Tree algorithms to predict the intrusion detection into the network system of IoVs. Often different ML model perform differently with variable attacks. The ML models used to develop LCCDE are CatBoost, LightGBM, XGBoost, AdaBoost, GradBoost. With LCCDE ensemble approach we are selecting the leading performing model for each attack at times. We used CICIDS2017 dataset and able to get 99.81% accuracy. Which shows the advantages of the future leader class-based approach.

## References

[1] LCCDE: “A Decision-Based Ensemble Framework for Intrusion Detection in The Internet of Vehicles” GLOBECOM 2022 - 2022 IEEE Global Communications Conference.

[2] L. Yang and A. Shami, “A Transfer Learning and Optimized CNN Based Intrusion Detection System for Internet of Vehicles,” in 2022 IEEE Int. Conf. Commun. (ICC), 2022, pp. 1–6.