2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

# Optimized Machine Learning-based Intrusion Detection System for Internet of Vehicles

Elnaz Limouchi
*Department of Electrical and Computer Engineering*
*Royal Military College of Canada*
Kingston, ON, Canada
https://orcid.org/0000-0003-4669-0476

Francois Chan
*Department of Electrical and Computer Engineering*
*Royal Military College of Canada*
Kingston, ON, Canada
https://orcid.org/0000-0002-2719-9430

***Abstract*—Internet of Vehicles (IoV) represents the application of Internet of Things (IoT) within vehicular communication environments. Internet of vehicles refers to a network of interconnected sensors, network layers, and communication systems that enable vehicles to connect with everything (V2X communication). IoV networks face numerous security challenges due to the emergence of modern types of attacks with unusual patterns. Therefore, it is a crucial and demanding task to design intelligent Intrusion Detection Systems (IDSs) for IoV networks. In this paper, we propose an optimized Machine Learning-based IDS to detect attacks in IoV networks. We deploy highly efficient Machine Learning models, Light Gradient Boosting Machine, Extra Trees Classifier, and Extreme Gradient Boosting, to detect attacks in the CICDDoS2019 dataset. We apply the Synthetic Minority Oversampling Technique to resolve the issue of imbalanced data distribution of target class. A Correlation-based Feature Selection is conducted to reduce the computational cost by decreasing the number of input variables. In order to enhance the performance of the attack detection, hyperparameters are optimized using the Bayesian Optimization algorithm. The performance evaluation results show that these ML models perform well. Notably, the Extreme Gradient Boosting classifier outperforms other Machine Learning models, and our proposed solution outperforms existing systems in terms of Accuracy score.**

***Index Terms*—IoV, Intrusion Detection System, Machine Learning, LightGBM, Extra Trees Classification, XGboost, Bayesian Optimization, PyCaret.**

## I. Introduction

Internet of Vehicles (IoV) indicates the incorporation of vehicles, roads, and transportation infrastructure with the communication networks and internet. The implementation of advanced vehicle technologies enables the establishment of a greater variety of communication modes in IoV as compared to Vehicular Adhoc Network (VANET) [1]. IoV is a promising concept, which aims to enable vehicles to communicate with other vehicles (V2V), network infrastructures (V2I), personal devices (V2P), sensor (V2S), and the cloud (V2C). IoV allows vehicles to exchange real-time data and access to various services, which leads to enhanced safety, comfort, and efficiency. The emergence of the Internet of Things (IoT) empowers vehicles to connect with everything (V2X), allowing them to exchange information about their surroundings. IoV networks are vulnerable to various types of cyberattacks, and it is a highly challenging task to identify complex attacks in such networks.

Intrusion Detection System (IDS) in IoV intends to detect and prevent unauthorized access, malicious activities, and cyber threats, which target the networks. IDS monitors network traffic and analyzes the communication patterns to identify suspicious behavior or anomalies that may define an intrusion. Intrusion Detection Systems can employ Machine Learning (ML) techniques to build baseline behavioral models. Then, it compares real-time behavior with the built models/patterns to detect anomalies, which can be potentially security threats.

In this work, we propose a Machine Learning-based Intrusion Detection System for IoV networks. We apply Light Gradient Boosting Machine, Extra Trees (ET), and Extreme Gradient Boosting (XGBoost) classification algorithms to classify traffic of the CICDDoS2019 dataset into normal and attack. In our proposed work, we perform feature selection, resolve the imbalanced target class problem, and optimize Hyperparameters of the applied ML classifiers. Additionally, we use one of the most recent reliable real world traffic datasets (CICDDoS2019) to train the IDS model. More specifically, the key contributions of this work include the following:

- Performing data pre-processing to resolve the issues related to the CICDDoS2019 dataset.
- Deploying Pearson correlation coefficient feature selection to establish a more efficient intrusion detection system by reducing the number of input variables.
- Using the Synthetic Minority Oversampling Technique to balance the distribution of target class.
- Using Bayesian Optimization (BO) algorithm to optimize the Hyperparameters of ML models and enhance the performance of the classification tasks.
- Applying Machine Learning classifiers (LightGBM, ET, Xgboost) on the CICDDoS2019 dataset to recognize normal and attack instances.
- Using 10-fold cross validation to evaluate the performance and accordingly select the best model. Accuracy, Precision, Recall, F1-score, Kappa index, and MCC are used as evaluating metrics.

The remainder of the paper is structured as follows: Section II reviews some related work on intrusion detection for vehicular networks. In Section III, we introduce the proposed work. The performance evaluation of the proposed work is

978-0-7381-4408-5/23/$31.00 ©2023 IEEE

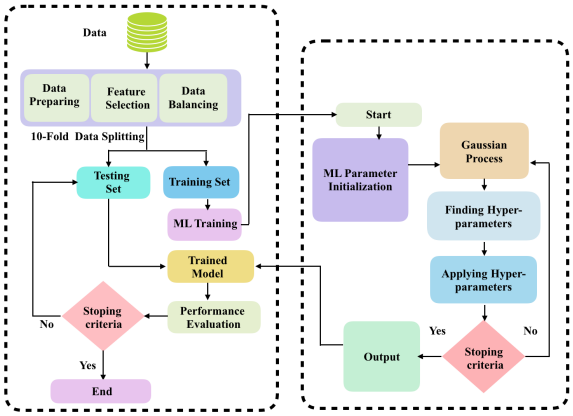


Fig. 1: The Proposed Workflow

discussed in Section IV. Finally, we conclude the paper in Section V.

## II. Related work

Network security applications, including Intrusion Detection Systems (IDSs), are crucial to detect anomaly activities more accurately and robustly [2]. This section highlights some learning-based IDS models proposed for vehicular communication networks. Additionally, some attack detection systems proposed for other types of network environments, which use the CICDDoS2019 dataset to train their models, are introduced in this section.

SHIELDNET is an adaptive botnet detection scheme proposed for vehicular Adhoc networks (VANETs) [3]. It uses the K-Nearest Neighbors algorithm to detect botnets on a dataset extracted from simulations in Veins. The SHIELDNET attack identifier system is capable of detecting 77% of the vehicular bots.

In [4], a misbehaviour detection model is introduced. This model is proposed to detect position falsification attacks and false alarm verification. Various machine learning techniques, like k-Nearest Neighbors (KNN), Decision Tree (DT), Random Forest (RF), and bagging are applied on VeReMi. VeReMi is a dataset generated by LuST and Veins simulations. This dataset consists of three different density levels, five distinct attack types, and three various attacker densities. The bagging classifier shows the best performance compared to the other ML techniques with a Precision score of 0.98 for the position verification system.

In [5], several ML models are examined to perform binary and multi-class classification on the Ton-IoT dataset. Logistic Regression (LR), Naive Bayes (NB), Decision Tree (DT), Random Forest (RF), AdaBoost, k-Nearest Neighbors (KNN), Support Vector Machine (SVM), and XGBoost models are utilized to complete the attack detection task. Comparing the results of the ML models, it is observed that XGBoost classifiers yield the best results with an Accuracy score of 0.986 (binary classification). This work tries to resolve the imbalanced data distribution problem; however, it does not consider Hyperparameters optimization.

In [6], a machine learning-assisted misbehavior detection for VANETs is proposed. To prevent the consequences resulting from false information injection in basic safety messages (BSMs), this system detects position falsification attacks, where KNN, NB, RF, and DT are applied to the simulated dataset, VeReMi. According to the results, the K-Nearest Neighbour and Random Forest classifiers outperform the other ML techniques.

The authors in [7] used Explainable Neural Network (xNN) technique to classify network traffic into normal and attack. Binary classification is conducted by using the UNSWNB15 dataset. The xNN binary classifier provides a good performance with the Accuracy score of 0.997.

In [8], an intelligent intrusion detection system for a 5G-enabled vehicular network is introduced. This IDS model uses DT and RF classifiers to detect Flooding attacks. The dataset used in this work is extracted from ns3. The results show that the Decision Tree classifier yields a good performance, achieving a F1 Score of 1.

In [9], the authors propose a hybrid deep learning-based model to detect DDoS attacks by combining Autoencoder (AE) and Multi-layer Perceptron (MLP) Network, while using the CICDDoS2019 dataset to train the model. This model uses the Autoencoder to conduct the feature selection task and utilizes the Multi-layer Perceptron Network to classify the attacks. The achieved Accuracy score is 0.9834, and the F1 score is 0.9818.

In [10], a Bi-directional LSTM-based multiclass classifier is introduced to detect and classify DDoS attacks. CICDDoS2019 is used to train the model, which yields an Accuracy score of 0.98.

In [11], the CICDDoS dataset is used to train binary and multiclass classifiers to detect DDoS attacks. The authors examine 8 ML classifiers (RF, K-NN, LightGBM, XGBoost, AdaBoost, SVM, Linear Discriminant Analysis) and one deep learning classifier. The binary classification scenario achieves an approximate Accuracy score of 0.99.

In the context of vehicular communications, the models proposed in [3], [4], [6], and [8] use network simulators to generate the datasets while we train our IDS models using a more realistic dataset (CICDDoS2019). The dataset we use in this work is relatively more recent compared to the dataset that is used for binary classification in [7].

## III. Proposed Methodologies

In this work, we propose an effective Intrusion Detection System (IDS) methodology using machine learning techniques to detect attack traffic in Internet of Vehicle networks. Fig. 1 shows the overall workflow of our proposed system. The IDS module is responsible for analyzing the traffic data of the network. Since each data packet has its fixed packet format (according to the IEEE 802.11p standard), each node/sensor can follow the packet data format. The IDS module is designed to analyze both data packets and network traffic.

### A. Data Description

In this work, we use one the most recent intrusion datasets, CICDDoS2019 [12], to enable development of our IDS model to detect intrusions in Internet of Vehicle networks. The CICDDoS2019 dataset is a publicly available network attack detection dataset, which is derived from the Canadian Institute for Cybersecurity (CIC) Cybersecurity dataset [13]. The dataset includes a comprehensive collection of real-world network traffic data, containing both normal and attack traffic with labeled data. The dataset has various types of attacks, including: UDP, MSSQL, Syn, NetBIOS, UDPLag, LDAP, DrDoS-DNS, WebDDoS, TFTP, DrDoS-UDP, DrDoS-SNMP, DrDoS-NetBIOS, DrDoS-LDAP, DrDoS-MSSQL, and DrDoS-NTP. We randomly select part of the dataset (431370 samples) and ensure that all the attack categories are included.

### B. Data Pre-processing

Data pre-processing refers to transforming and cleaning the data to verify its quality and consistency for further processing. For the CICDDoS2019 dataset, data cleaning consists of removing white spaces and null/missing values. To decrease the chance of overfitting in training process, we remove socket information. Additionally, we encode the categorical labels of attacks into numerical representations. Since the dataset contains a wide variety of numerical values, we use the Max-normalization method to transform the feature values to $[0, 1]$ scale and simplify the training task.

### C. Feature Selection

Feature selection is one of the most important steps in Machine Learning tasks. The CICDDoS2019 dataset contains 80 features, making it impractical to use all of them. Furthermore, not every feature in a dataset will necessarily have an impact on the output. Selecting the most informative and influential feature can reduce the complexity, improve the performance, decrease the risk of overfitting, and enhance interpretability. In order to select the most relevant features, we apply an univariate method to select features based on their correlation with the target variable, label of attack. We determine the Pearson correlation coefficient to rank the features and select the top ones. The absolute value of 0.5 is set as the threshold to select the features. The pairwise relationships of selected features are plotted in Fig. 2.

### D. Data Balancing

The subclass of dataset we use in this work has an uneven data distribution in target class. Considering binary classes, the $Attack$ class has a very high number of observations (majority class) while the $Normal$ class has a low number of observations (minority class). The main concern with an imbalanced dataset is that the classifier may tend to be biased towards the majority class and fail to recognize the minority class, even if it has a high Accuracy score. The Synthetic Minority Oversampling Technique (SMOTE) is an effective solution to handle the imbalanced data problem. This technique uses the k-nearest neighbors method to pick a random

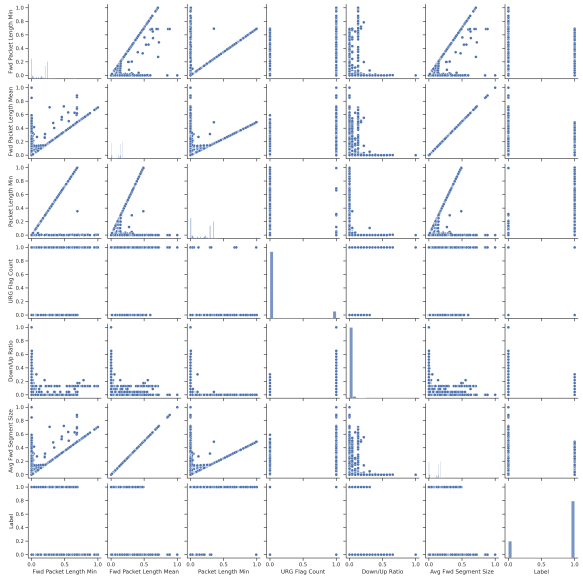


Fig. 2: Pairwise Relationships of Features

neighbor of minority class instances and randomly generate a synthetic instance. By generating new data instances instead of duplicating existing ones, this method effectively reduces the risk of overfitting, providing an advantage over conventional oversampling techniques.

### E. Classification Models

In this work, we use three supervised classification models: Light Gradient Boosting Machine, Extra Trees Classifier, and Extreme Gradient Boosting.

*1) Light Gradient Boosting Machine:* Light Gradient Boosting Machine (LightGBM) is a relatively new machine learning model, which belongs to the gradient boosting framework. As a lightweight and efficient classifier, lightGBM can handle large-scale datasets. This efficiency is attributed to the use of techniques such as leaf-wise tree growth. Algorithm 1 explains details of the Light Gradient Boosting technique.

*2) Extra Trees Classifier:* The Extra Trees Classifier (ET) is an ensemble machine learning classifier, which performs based on decision trees technique. The Extra Trees Classifier introduces extra levels of randomness in the tree-building process. ET has two main parameters: randomly selected number of attributes, $k$, and minimum sample size for splitting, $n_{min}$. ET is used to generate an ensemble model with $M$ trees. These parameters conduct different effects. The strength of attribute selection process can be determined by $k$, while $n_{min}$ influences the strength of averaging output noise. The parameter $M$ modifies the strength of the variance reduction of the model. A visual explanation of Extra Trees Classifier is shown in Fig. 3.

*3) Extreme Gradient Boosting:* Extreme Gradient Boosting (XGBoost) is an advanced supervised learning algorithm that, like LightGBM, belongs to the gradient boosting framework. As a highly optimized implementation of gradient boosting, XGBoost provides high performance and scalability. XGBoost

**Algorithm 1** Light Gradient Boosting Machine Algorithm

**Input:** Data: $D$=$\{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$, $x_i \in x$, $y_i \in \{0, 1\}$;
**Input:** Loss Function: $L(y, \theta(x))$;
**Input:** Big Gradient Data Sampling Ratio: $a$;
**Input:** Slight Gradient Data Sampling Ratio: $b$;
**Input:** Iterations: $M$;
Set $\theta_0(x) = arg\ min_c \Sigma_i^N L(y_i, c)$;
**for** $m = 0$ to $M$ **do**
  Gradient absolute values calculation:
  $r_i = |\frac{\delta L(y_i, \theta_i(x))}{\delta \theta(x_i)}|_{\theta(x)=\theta_{m-1}(x)},\ i = \{1, ..., N\}$
  Gradient-based one-side sampling process:
  $topN \leftarrow a \times len(D)$;
  $randN \leftarrow b \times len(D)$;
  $sorted \leftarrow GetSortedIndices(abs(r))$;
  $A \leftarrow sorted[1 : topN]$;
  $B \leftarrow RandomPick(sorted[topN : len(D)], randN)$;
  $D\prime = A + B$;
  Information Gains calculation:
  $\gamma_l = \frac{(\Sigma_{x_i \in A_l} r_i + \frac{1-a}{b} \Sigma_{x_i \in B_l} r_i)^2}{n_l^j(d)}$
  $\gamma_r = \frac{(\Sigma_{x_i \in A_r} r_i + \frac{1-a}{b} \Sigma_{x_i \in B_r} r_i)^2}{n_r^j(d)}$
  $G_j(d) = \frac{1}{n}(\gamma_l + \gamma_r)$
  New decision tree $\theta\prime_m(x)$ developing on data $D\prime$
  Updating $\theta_m(x) = \theta_{m-1}(x) + \theta_m(x)$
**end for**
**return** $\hat{\theta}(x) = \theta_M(x)$

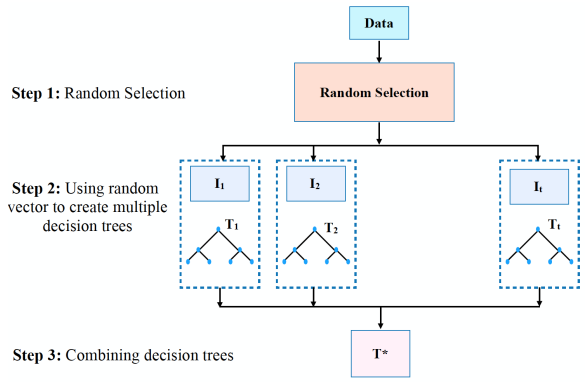


Fig. 3: Extra Trees Classifier

employs an ensemble of decision trees as base learners, while each tree is added sequentially. Subsequent trees join to correct the mistakes made by the previous trees. Algorithm 2 describes the XGBoost model.

### F. Hyperparameter Optimization

Hyperparameter optimization involves finding the right combination of Hyperparameter values, which maximizes the performance of ML models within a reasonable time. Since the default values do not always guarantee the best performance, Hyperparameter optimization potentially enhances the results. Bayesian optimization process involves the following steps:

- It generates a probabilistic Gaussian model of the objective function and updates it based on the previous evaluation results.
- It finds the most promising Hyperparameters with maximum acquisition function.
- It applies the Hyperparameters to the objective function.

**Algorithm 2** Extreme Gradient Boosting Algorithm

**Input:** Data: $D$=$\{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$, $x_i \in x$, $y_i \in \{0, 1\}$;
**Input:** Loss Function: $L(y, \theta(x))$;
**Input:** Iterations: $M$;
Set $\theta_0(x) = arg\ min_c \Sigma_i^N L(y_i, c)$;
**for** $m = 0$ to $M$ **do**
  calculating:
  $g_m = \frac{\delta L(y_i, \theta_i(x))}{\delta \theta(x_i)}$ and $h_m = \frac{\delta^2 L(y_i, \theta_i(x))}{\delta \theta^2(x_i)}$
  Choosing splits with maximum gains:
  $A = \frac{1}{2}[\frac{G_L^2}{H_L} + \frac{G_R^2}{H_R} + \frac{G^2}{H}]$
  Leaf weights calculation: $w^* = -\frac{G}{H}$
  Base learner calculation: $\hat{b}(x) = \Sigma_{j=1}^T wI$
  Updating $\theta_m(x) = \theta_{m-1}(x) + \hat{b}(x)$
**end for**
**return** $\hat{\theta}(x) = \Sigma_{m=0}^M \theta_m(x)$

- It determines the performance with the current parameters and repeat the steps until the maximum performance score is achieved.

### G. Performance Evaluation Metrics

We use several performance evaluation metrics to evaluate the performance of the proposed intrusion detection framework. A higher value of these metrics indicate a better performance.

*1) Accuracy:* Accuracy score is one the most straightforward metrics to evaluate the performance of classifiers. As described in (1), this metric represents the fraction of correct predictions.

$$Accuracy = \frac{Numberof CorrectPredictions}{TotalNumberofPredictions} \tag{1}$$

*2) AUC:* The curve Receiver Operating Characteristic (ROC) inspects the performance of a classifier by plotting the classifier's true positive rate (TPR) versus its false positive rate (FPR). Calculating the Area Under the Curve results in AUC score. A perfect classifier keeps an AUC score of 1, while a random classifier has a score of 0.5.

*3) Recall:* The metric Recall is the proportion of positive instances that are correctly identified as positive:

$$Recall = \frac{TruePositive(TP)}{TruePositive(TP) + FalseNegative(FN)} \tag{2}$$

*4) Precision:* The ratio of the number of True Positives to the total number of instances, which are predicted as positive is a metric known as Precision. The expression of Precision is given in (3).

$$Precision = \frac{TruePositive(TP)}{TruePositive(TP) + FalsePositive(FT)} \tag{3}$$

*5) F1 Score:* The F1 Score takes the harmonic mean of Precision and Recall to introduce a new metric. Equation (4) indicates the relationship between F1 and the other two scores, Precision and Recall.

$$F1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \tag{4}$$

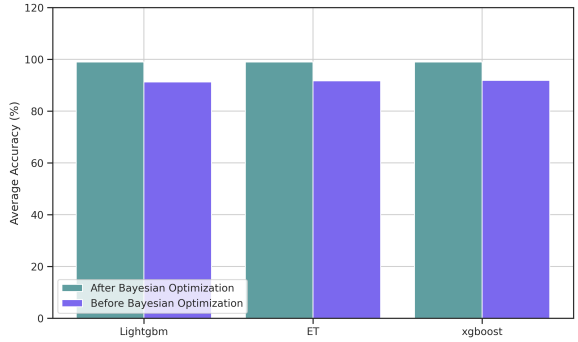


Fig. 4: Impact of Bayesian Optimization on Accuracy Scores

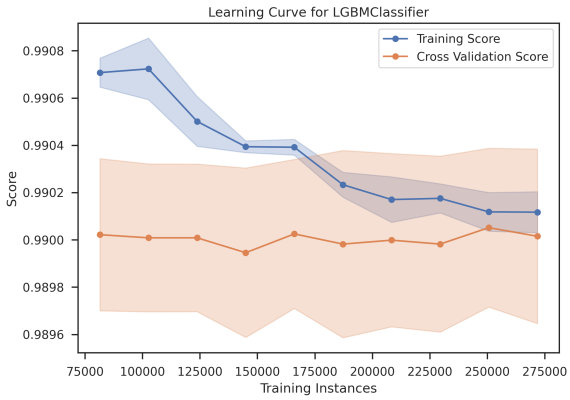


Fig. 5: Learning Curve of LightGBM classification

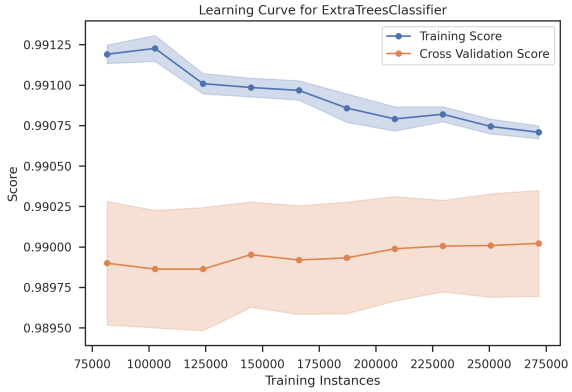


Fig. 6: Learning Curve of Extra Trees classification

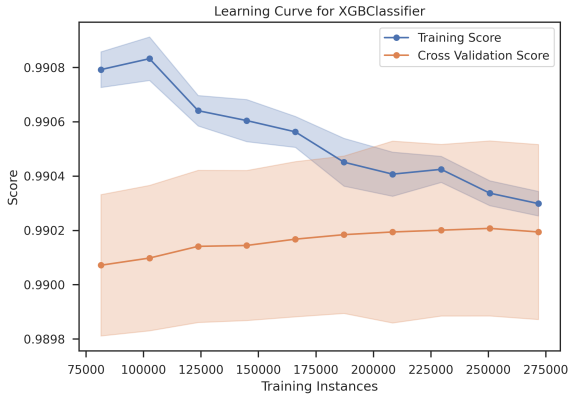


Fig. 7: Learning Curve of XGBoost classification

*6) Kappa Index:* The basic idea of the Kappa index is to compare an observed accuracy with a random guess (expected accuracy). It measures the agreement between two ratings, as shown in (5):

$$k = \frac{P_o - P_e}{1 - P_e} = 1 - \frac{1 - P_o}{1 - P_e} \tag{5}$$

where $P_o$ and $P_e$ are the observed and expected agreements, respectively.

*7) MCC:* The Matthew's correlation coefficient (MCC) is a correlation coefficient between the predicted values and true values. MCC returns a value between -1 and 1, while perfect predictions result in MCC value of +1.

## IV. Results

In this section, we present the experimental results of the proposed Intrusion Detection System for Internet of Vehicles network. We use various software packages of Python to conduct a better analysis. Pandas is mostly used to complete the data pre-processing tasks [14], while the majority of data analyzing procedures are done by PyCaret and Sklaern [15], [16]. Matplotlib and Seaborn are employed for data visualization [17], [18].

### A. Hyperparameter Optimization

The impact of Hyperparameter tuning based on the Bayesian Optimization algorithm for each classifier is shown in Fig. 4. The figure represents an approximately 8% improvement in Accuracy for all three classifiers after optimizing their parameters.

### B. Classification Performance

Table I specifies the performance evaluation results of the classifiers. According to the information provided in Table I, XGBoost slightly outperforms the other two classifiers in terms of accuracy (0.9902), AUC (0.9993), and recall (0.9755) scores. Figs. 5, 6, and 7 show the learning curves of LightGBM, ET, and XGBoost classifiers, respectively. The learning curves confirm the effective learning process of these three classifiers, while further increasing the size of the training sets may not significantly enhance performance.

### C. K-Fold Cross Validation

The 10-Fold Cross-Validation Accuracy results are given in Table II. In the 10-Fold Cross-Validation process, the first fold is used for testing, and the remaining folds are used for training and this is repeated ten times to test the entire dataset in a fold-by-fold basis. From Table II, the mean Accuracy values of LightGBM, ET, and XGBoost classifiers are 0.9900, 0.9900 and 0.9902, respectively. The Accuracy standard deviation of LightGBM classifier is 0.0004, while ET and XGBoost models have the same value of 0.0003. These results demonstrate the robust performance of the classifiers.

TABLE I: Classification Performance

| Model | Accuracy | AUC | Recall | Prec. | F1 | Kappa | MCC |
|---|---|---|---|---|---|---|---|
| Light Gradient Boosting Machine | 0.9900 | 0.9991 | 0.9753 | 0.9995 | 0.9872 | 0.9469 | 0.9481 |
| Extra Trees Classifier | 0.9900 | 0.9992 | 0.9754 | 0.9995 | 0.9873 | 0.9471 | 0.9483 |
| Extreme Gradient Boosting | 0.9902 | 0.9993 | 0.9755 | 0.9995 | 0.9873 | 0.9470 | 0.9482 |

TABLE II: 10-Fold Cross-Validation Accuracy Scores

| | LightGBM | ET | XGBoost |
|---|---|---|---|
| **Fold-1** | 0.9902 | 0.9902 | 0.9905 |
| **Fold-2** | 0.9905 | 0.9902 | 0.9907 |
| **Fold-3** | 0.9907 | 0.9907 | 0.9907 |
| **Fold-4** | 0.9903 | 0.9903 | 0.9904 |
| **Fold-5** | 0.9898 | 0.9898 | 0.9899 |
| **Fold-6** | 0.9899 | 0.9901 | 0.9902 |
| **Fold-7** | 0.9897 | 0.9895 | 0.9898 |
| **Fold-8** | 0.9897 | 0.9898 | 0.9899 |
| **Fold-9** | 0.9895 | 0.9898 | 0.9899 |
| **Fold-10** | 0.9898 | 0.9898 | 0.9899 |
| **Mean** | 0.9900 | 0.9900 | 0.9902 |
| **Std** | 0.0004 | 0.0003 | 0.0003 |

### *D. Impact of Feature Selection*

Feature selection reduces the complexity of classifications. The time complexity of LightGBM classifier is on the order of $O(T * M * P * D)$, where $M$ denotes the number of features, $T$ is the number of trees, $D$ is the depth of the tree, and $P$ represents the number of buckets.

The complexity of the Extra Trees algorithm can be described as $O(T * K * N * log(N))$, where $N$ is the sample size of the training data, and $K$ is the main parameter of the tree-based algorithm.

The time complexity for building each tree in the XGBoost algorithm is approximately $O(M * N * log(N))$.

Using the correlation coefficient feature selection technique and removing the port-dependent features, we change the shape of balanced training set from $(596366, 88)$ to $(596366, 7)$, which means 6 input features and 1 output feature are used to train the classifiers. The decrease in the number of features and data points notably reduces the complexity for all three classifiers.

### *E. Feature Importance*

Feature importance is a score, which shows how useful or valuable each feature was to establish the boosted decision trees within the model. Considering XGBoost as the most effective model in attack detection, "Down/Up Ratio" and "URG Flag Count" receive the highest importance scores, while "Avg Fwd Segment Size" and "Fwd Packet Length Min" have the least importance among the dataset's variables.

We repeat the XGBoost classification task considering two different scenarios. In the first scenario, only the 4 most important features are applied to train the classifier. The second scenario is defined to use only the 4 least important features. As shown in Table III, reducing the number of features from 6 to 4 and using only the 4 most important ones results in an improvement in Accuracy score from 0.9902 to 0.9914. However, using the 4 least important features does not improve the performance, instead, the Accuracy drops to a value of 0.9886.

### *F. Impact of Data Balancing*

We use the Synthetic Minority Oversampling Technique (SMOTE) to balance the target variable. This leads to an increased number of samples (from 43137 to 596366). As previously mentioned in Section III-D, an imbalanced dataset can potentially have a high Accuracy score, where it is biased towards the majority class. Therefore, instead of Accuracy score, other evaluation metrics are more reliable to evaluate the performance. The impact of data balancing on the performance of all the three classifiers in terms of the Precision metric is shown in Fig. 8. Data balancing based on SMOTE technique improves the Precision score for all the classifiers. Due to the imbalanced data, the Precision scores for LightGBM, ET, and XGBoost classifiers were 0.9967, 0.9893, and 0.9897, respectively. However, by using the SMOTE technique, all three classifiers are able to achieve a higher Precision score of 0.9995.

### *G. Comparison with Other Studies*

We compare the performance of this work in terms of Accuracy to the other systems in [5], [7], and [8]. Table IV summarizes this comparison. We choose the results of [5], since similar to our work they present the XGBoost classifier as the most effective ML model to recognize attack instances. Among all the binary classifiers discussed in Section II, the proposed system in [7] has the highest accuracy score. As one of the most recent proposed systems in the context of Intrusion Detection for vehicular communications, the results of [8] are considered. In terms of the Accuracy scores, our proposed system outperforms the work in [5] and [8]. The attack detection model in [7] achieves a higher Accuracy score (0.997) compared to our model's Accuracy score (0.9902). However, we use a more recent dataset to train our attack detection system.

We also examined the performance of all these methods on the CICDDoS2019 dataset. According to Table IV, although all the methods perform accurately, our proposed work achieves the highest Accuracy score. The combination of Bayesian Optimization technique to tune the Hyperparameters and SMOTE to balance the data distribution leads to an enhanced classification accuracy.

## V. Conclusion

We proposed an optimized Machine Learning-based Intrusion Detection System for Internet of Vehicles networks. Light Gradient Boosting Machine, Extra Trees classifier, and Extreme Gradient Boosting models were the supervised ML classifiers employed to classify normal and attack instances. The Hyperparameters of ML models were optimized according to the Bayesian Optimization algorithm. We used one of the

TABLE III: Performance of XGBoost using only the 4 Most/Least Important Features

| | Accuracy | AUC | Recall | Prec. | F1 | Kappa | MCC |
|---|---|---|---|---|---|---|---|
| 4 Most Important Features | 0.9914 | 0.9998 | 0.9986 | 1.0000 | 0.9993 | 0.8860 | 0.8913 |
| 4 Least Important Features | 0.9886 | 0.9797 | 0.9985 | 0.9799 | 0.9792 | 0.8758 | 0.8816 |

TABLE IV: Comparison of the average Accuracy with the other studies

| Reference | Best ML Technique | Data Balancing Technique | Hyperparameter Optimization | Original Dataset | Average Accuracy (Original Dataset) | Average Accuracy (CICDDoS2019) |
|---|---|---|---|---|---|---|
| [5] | Extreme Gradient Boosting | SMOTE | - | Ton-IoT | 0.986 | 0.983 |
| [7] | Explainable Neural Network | - | - | UNSWNB15 | 0.997 | 0.986 |
| [8] | Decision Tree | - | - | Extracted from ns3 | 0.970 | 0.982 |
| Our work | Extreme Gradient Boosting | SMOTE | Bayesian Optimization | CICDDoS2019 | 0.9902 | 0.9902 |

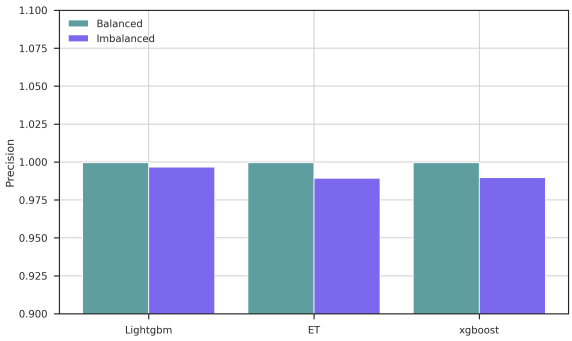


Fig. 8: Impact of Data Balancing on Precision Score

most recent datasets, CICDDoS2019, to train and test the ML models. We performed data pre-processing and feature selection on the dataset before applying ML classification models. Data balancing was conducted by applying Synthetic Minority Oversampling Technique. The performance of the ML models were assessed by using several evaluation metrics: Accuracy, Precision, Recall, F1-score, Kappa index, and MCC. According to the performance outcomes, while all the ML classifiers had very good performance the XGBoost classifier slightly outperformed the others. Finally, we compared the performance of our proposed attack detection system with the results of some of the most recent and high-performing studies using the same CICDDoS dataset. The findings revealed that our scheme outperforms all these other systems, highlighting the effectiveness of our data balancing technique and Bayesian Hyperparameter Optimization.

## REFERENCES


[1] R. Gasmi and M. Aliouat, "Vehicular ad hoc networks versus internet of vehicles - a comparative view," in *2019 International Conference on Networking and Advanced Systems (ICNAS)*, 2019, pp. 1–6.

[2] R. A. Khamis and A. Matrawy, "Evaluation of adversarial training on different types of neural networks in deep learning-based idss," in *2020 International Symposium on Networks, Computers and Communications (ISNCC)*, 2020, pp. 1–6.

[3] M. T. Garip, J. Lin, P. Reiher, and M. Gerla, "Shieldnet: An adaptive detection mechanism against vehicular botnets in vanets," in *2019 IEEE Vehicular Networking Conference (VNC)*, 2019, pp. 1–7.

[4] S. Gyawali and Y. Qian, "Misbehavior detection using machine learning in vehicular communication networks," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[5] A. R. Gad, A. A. Nashat, and T. M. Barkat, "Intrusion detection system using machine learning for vehicular ad hoc networks based on ton-iot dataset," *IEEE Access*, vol. 9, pp. 142 206–142 217, 2021.

[6] A. Sharma and A. Jaekel, "Machine learning based misbehaviour detection in vanet using consecutive bsm approach," *IEEE Open Journal of Vehicular Technology*, vol. 3, pp. 1–14, 2022.

[7] S. Aziz, M. T. Faiz, A. M. Adeniyi, K.-H. Loo, K. N. Hasan, L. Xu, and M. Irshad, "Anomaly detection in the internet of vehicular networks using explainable neural networks (xnn)," *Mathematics*, vol. 10, no. 8, 2022. [Online]. Available: https://www.mdpi.com/2227-7390/10/8/1267

[8] B. Sousa, N. Magaia, and S. Silva, "An intelligent intrusion detection system for 5g-enabled internet of vehicles," *Electronics*, vol. 12, no. 8, 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/8/1757

[9] Y. Wei, J. Jang-Jaccard, F. Sabrina, A. Singh, W. Xu, and S. Camtepe, "Ae-mlp: A hybrid deep learning approach for ddos detection and classification," *IEEE Access*, vol. 9, pp. 146 810–146 821, 2021.

[10] C.-S. Shieh, W.-W. Lin, T.-T. Nguyen, C.-H. Chen, M.-F. Horng, and D. Miu, "Detection of unknown ddos attacks with deep learning and gaussian mixture model," *Applied Sciences*, vol. 11, no. 11, 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/11/5213

[11] J. Halladay, D. Cullen, N. Briner, J. Warren, K. Fye, R. Basnet, J. Bergen, and T. Doleck, "Detection and characterization of ddos attacks using time-based features," *IEEE Access*, vol. 10, pp. 49 794–49 807, 2022.

[12] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (ddos) attack dataset and taxonomy," in *2019 International Carnahan Conference on Security Technology (ICCST)*, 2019, pp. 1–8.

[13] "Ddos evaluation dataset (cic-ddos2019)," updated in 2021, accessed on July 25, 2023. [Online]. Available: https://www.unb.ca/cic/datasets/ddos-2019.html

[14] T. pandas development team, "pandas-dev/pandas: Pandas," Feb. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3509134

[15] M. Ali, *PyCaret: An open source, low-code machine learning library in Python*, April 2020, pyCaret version 1.0.0. [Online]. Available: https://www.pycaret.org

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[17] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[18] M. L. Waskom, "seaborn: statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021. [Online]. Available: https://doi.org/10.21105/joss.03021