In-Vehicle Network Anomaly Detection Using Extreme Gradient Boosting Machine

Afia Anjum^1, Paul Agbaje^1, Sena Hounsinou^2, Habeeb Olufowobi^1
1: *University of Texas at Arlington*, Arlington, TX, USA
2: *University of Colorado Colorado Springs*, Colorado Springs, CO, USA
{afia.anjum, pauloluwatowoju.agbaje, habeeb.olufowobi}@uta.edu, shoueto@uccs.edu

***Abstract*—Modern vehicles have significantly increased the number of Internet of Things (IoT) devices called electronic control units (ECUs) connected by in-vehicle networks to provide enhanced features and safety. These devices communicate with the environment and have brought the notion of the Internet of vehicles. The controller area network (CAN) bus facilitates efficient ECU communications and is the standard protocol used by every vehicle, but it is susceptible to remote attacks. Consequently, it is desirable to monitor the CAN bus for malicious activities, such as data injection attacks, that can compromise the vehicular operations. We present an anomaly detection technique that uses an extreme gradient boosting machine (GBM) learning algorithm to categorize unexpected occurrences in the CAN data payload. We further combine GBM with a ten-fold cross-validation method to improve prediction performance. Moreover, we use the early-stopping and grid search approaches to overcome overfitting without affecting model accuracy. We evaluate our detection approach on real CAN bus datasets collected from Hyundai Sonata, a KIA Soul, and Chevrolet Spark with different attack scenarios, such as Denial-of-Service (DoS), fuzzy, spoofing, and malfunction attacks. Using standard metrics, such as accuracy, recall, precision, F1 score, and false-positive rate, the performance analysis of the proposed model achieved an overall accuracy of over 99 percent.**

I. Introduction

The Internet of Vehicles (IoV) is an emerging cyber-physical system (CPS) that fuses the Internet of Things (IoT), communication networks, and the cloud to support vehicular operations as a result of the increasing proliferation of embedded sensors integrated into automobiles and their connectivity to the Internet. These sensors are attached to devices called electronic control units (ECUs) that have become a fundamental part of the vehicle architecture. ECUs increase vehicular functionality in terms of safety, comfort, and automation. This proliferation has also opened up the once closed vehicular systems to cyber and physical attacks [1]. The security of ECUs and their communication across the in-vehicle networks are of paramount importance, as their ubiquity has given rise to many vulnerable threat points for cyber attackers. To date, there is no standardized framework for securing automotive in-vehicle networks, such as the controller area network (CAN) that facilitates ECU communications. Hence, the connected vehicle ecosystem is vulnerable to cyber-attacks. Previous research [2]–[6] has shown that attacks on in-vehicle networks represent severe risks to the automotive sector since they directly affect driver safety, data privacy, and service continuity.

Many studies have been conducted that apply classification-based machine learning techniques on CAN bus traffic to detect anomalous events, which indicate an attack. However, these techniques are computationally intensive and time-consuming during training, they are optimized to detect systems behaviors or require domain expertise. To address the limitations of the prior art, we propose an anomaly detection system using gradient boosted machine (GBM) algorithms [7], which can identify new patterns or features of CAN data frames that have not been previously transmitted. The algorithm uses consecutively fitted models to map the attributes of the presented data payloads to the attack and predicts final results. Gradient boosting is a machine learning approach that can learn problems in noisy, large datasets and their complex dependencies with its prediction speed and accuracy [8]. A GBM algorithm uses an ensemble approach (different formations) by combining relatively large, weak, and simple classifiers to derive a strong ensemble prediction. GBM builds trees in sequence and sums the predictions from each tree for the final prediction [9].

In this paper, we present an anomaly-based detection system using the extreme GBM (XGBoost), an optimized, computationally efficient, and flexible boosting algorithm for the automotive network. XGBoost has an advantage over other gradient boosting algorithms in terms of efficiency and speed, which is essential for the accurate detection of intrusions. We demonstrated the effectiveness of our approach using CAN data signals from real vehicles comprising different data injection attack scenarios. The performance of the model is also evaluated on other open-source datasets using standard metrics, such as recall, precision, false-positive rate (FPR), accuracy, and F1 score.

Our contributions are summarized as follows:

- design and implementation of intrusion detection technique using XGBoost to categorize anomalies in CAN data payload;
- evaluation of the model on a real CAN dataset containing different attack scenarios;
- comparative study of XGBoost-based anomaly detection approach applied on various open-source datasets from real vehicles;
- accuracy demonstration of the proposed model using standard performance metrics such as precision, recall, false-positive rate (FPR), accuracy, and F1 score;

The remainder of the paper is organized as follows. We provide an overview of CAN bus and XGBoost along with a threat model in Section II. In Section III, we present the proposed anomaly detection system using XGBoost and Section IV includes the experimental results. Moreover, we review related work in Section V and Section VI concludes the paper.

II. Background

A. CAN Overview

The CAN bus is the most used internal network communication protocol in modern vehicles to interconnect safety-critical ECUs. ECUs are called nodes on the network and can broadcast signals relating to their functions via a single/dual wire bus. Signals are transmitted in the clear, and the bus does not implement any security mechanisms. Messages sent on the bus are broadcast to all nodes as CAN signals do not contain source nor destination addresses. The CAN bus implements message arbitration based on the identifier (ID) field of the transmitter's message frame, which also indicates the message's priority. Lower IDs have a higher priority, and the protocol detects collisions of signals. Dominant bits of a transmitted signal are decoded as logic 0, and recessive bits are decoded as logic 1. Each frame contains a data payload of up to 8 bytes as dictated by the data length code (DLC) field. Figure 1 shows the details of the CAN data frame.

The CAN bus is susceptible to cyber and physical attacks. An adversary with a window of opportunity can permeate, manipulate, and perform different attacks on the communication protocol that can affect traffic safety and endanger human lives or property. Furthermore, the advent of connected autonomous cars and their interconnection to the Internet and to their environment has made vehicles an attractive target to cyber attackers. An adversary can leverage this interconnectedness to perform attacks that impedes the legitimate vehicular operation, including signal injection, DoS, and spoofing attacks targeting the safety-critical components of the vehicle [10]–[12]. Therefore, a subject of significance to the automotive industry is an appropriate and well-designed security protocol for the connected car ecosystem.

B. Extreme Gradient Boosting Overview

Extreme gradient boosting [13] is an ensemble supervised machine learning algorithm that uses decision trees as base estimators. Gradient boosting algorithms construct strong prediction models by building weak models and combining them [9]. The decision trees in the model are sequentially built to enable subsequent trees to minimize the errors in previous trees.

XGBoost is an efficient implementation of gradient boosting. It is an ensemble of classification and regression trees, and the trees are built using residual class labels. The algorithm is a robust, distributed, and can be used for classification problems such as anomaly detection. Given a dataset $D = (x_i, y_i)$ with n samples, XGBoost uses additive functions f_k to predict the output y_i given in equation 1 [14]:

$$\hat{y_i} = \sum_{k=1}^{m} f_k(x_i), f_k \in F \tag{1}$$

The regularized objective function is given by:

$$L^t = \sum_{i}^{n} l(\hat{y_i}, y_i) + \sum_{k}^{K} \Omega_k(f_k) \tag{2}$$

where $\sum_i^n l(\hat{y_i}, y_i)$ represents the loss function and $\sum_k^K \Omega_k(f_k)$ the regularization parameter. If $y_i^{(t)}$ is the prediction of y_i at the *t-th* iteration, the loss function is minimized by adding a base learner $f_t(x_i)$ to the objective function which is then given by:

$$L^t = \sum_{i}^{n} l(y_i, \hat{y}_i^{t-1}) + f_t(x_i) + \Omega_k(f_k) \tag{3}$$

For a tree with a fixed structure, a derivative loss function for a fixed base learner with K nodes is used to obtain an optimal weight for the leaf nodes. The loss function is given by:

$$\tilde{L}^t = \frac{-1}{2} \sum_{j=1}^{K} \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma K \tag{4}$$

where γ is the pruning parameter and λ is a hyperparameter for tuning the model. The first and second-order derivative solution of the loss during previous iterations is given by g_i and h_i respectively. Since it is impossible to explore every possible tree structure, XGBoost uses a greedy approach to iteratively build a tree [15]. By splitting the tree into left and right nodes, the split that minimizes the loss is selected. If I_L and I_R represents the instances of left and right nodes respectively, and $I = I_L \cup I_R$, then the loss resulting from the split is given by:

$$L_{split} = \frac{1}{2}\left[\frac{(\sum_{i=I_L} g_i)^2}{(\sum_{i=I_L} h_i)+\lambda} + \frac{(\sum_{i=I_R} g_i)^2}{(\sum_{i=I_R} h_i)+\lambda} - \frac{(\sum_{i=I} g_i)^2}{(\sum_{i=I} h_i)+\lambda}\right] - \gamma \tag{5}$$

XGBoost's split finding algorithm makes it efficient for handling data that may contain missing values. Also, the algorithm has inbuilt cross-validation method that reduce overfitting [16].

C. Threat Model

In this paper, we assume an attacker that can eavesdrop, intercept, replay, and transmit anomalous signals on the CAN bus to disrupt or control the vehicle operation. The attack scenarios considered in this paper are consistent with previously demonstrated CAN attacks that include packet injection, spoofing, DoS, and bus-off attacks [2], [4], [17]. The goal of the attacker is to compromise a victim ECU and influence the normal operation mode of the vehicle by attacking the bus. The attacker may gain access through the physical and remote surface of the vehicle, and we assume they have analyzed the vehicle behavior using reverse engineering techniques [18]. The remote attack surfaces connect the vehicle to external networks and the environment. These surfaces are known to be exploitable to compromise vehicular networks and take overall control of vehicular operations [3].

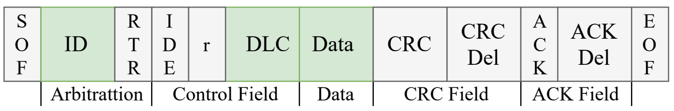

Fig. 1. CAN Data Frame

Packet injection attacks involve the attacker transmitting data frames that mimic a victim ECU frame to alter the internal working of the vehicle. Randomly fabricated data injection or fuzzy attacks cause the modification of the vehicle's functions under the attacker's control. Also, benign messages can be fabricated in the form of spoofing attacks to cause divergence of ECUs from normal operating conditions. Packet injection can lead to DoS attacks when the bus is flooded with high-priority frames, making it unavailable for legitimate frame transmission. The injected frames can have the same or different IDs and are transmitted together to deaden the network. For malfunction attacks, data fields of randomly selected ECUs are manipulated to cause the vehicle to react abnormally.

III. Proposed XGBoost Based Anomaly Detection

Anomaly detection is a process of identifying data patterns with different features from normal behavior or instances. This detection approach has been used in several application domains with significant relevance and can detect anomalies based on density, distance, or isolation. Anomaly-based detectors that learn the patterns of CAN bus messages can signal when an unexpected behavior is observed, which overcomes limitations of signatures but introduces the new problem of false positives. High classifier performance is critical for CAN intrusion prevention systems that automate response mechanisms to alerts [19].

A. Overview of the Proposed Algorithm

The objective of the proposed method is to build a model of the normal CAN bus behavior based on the observed payloads of the data frame transmitted. Our approach to anomaly detection in CAN bus communication relies on a supervised machine learning algorithm to distinguish data frames that do not conform to the typical payload values. The starting point is to measure and understand the normal and abnormal baseline behavior of the CAN bus traffic and then apply a binary classification algorithm to determine if a specific data point is anomalous or expected. This detection method is adaptive as it can adjust to changes in the data payload of the CAN signal.

B. System Architecture

Figure 2 shows the conceptual framework of our approach, composing four layers. The proposed framework starts with a single data frame from the CAN bus signal, from which the desired attributes such as the arbitration ID and data payload are extracted. Since the data bytes in the DLC field represent different signals [20], they are subdivided into eight distinct parameters ($D_0 - D_7$), which are processed separately. In addition, since our solution relies on a supervised learning approach, an additional input is necessary to state whether the input is a normal data point or an outlier. Thus, we use the $Flag$ field to indicate that the corresponding input message is normal or an attack. Together with the message ID, these parameters are transformed using a *Data Filter, Data Converter,* and *Normalizer*. The processed dataset is then fed into a *Data Splitter*, where it is split into training and test datasets ten times using the 10-fold cross-validation method. In parallel, the processed dataset undergoes a parameter tuning process.

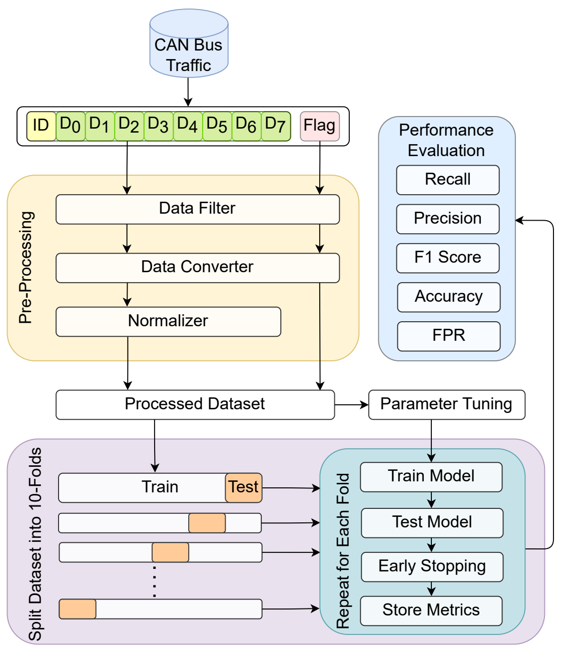

Fig. 2. CAN Bus Intrusion Detection System

Both tuned parameters and split datasets are used for model training and testing, followed by an evaluation process.

We can categorize the levels of the proposed CAN bus intrusion detection system (IDS) as follows:

1) Data Pre-processing: The *data filter* module is used as the first step to transform the parameters obtained from the CAN frame. It removes any row from the dataset which has a missing value. Since the XGBoost model can only accept numerical vectors as input, the module is followed by a *data converter* that converts the attribute to corresponding integer values. These attributes are then normalized using the min-max function. The min-max normalization technique is used to speed up the learning phase before fitting to the model. It also assigns an equal weight to all attributes by performing a linear transformation on the data, resulting in the transformed attributes falling into the range of [0,1]. This transformation helps in preserving the relationships among the extracted attributes of the CAN dataset before feeding it to the model.

2) Hyperparameter Tuning: Hyperparameters are specific values or weights that influence an algorithm's learning process. A large variety of hyperparameters is required to develop an XGBoost model. We optimized frequently used XGBoost hyperparameters, such as alpha, learning rate, and max depth, to enhance prediction accuracy. We used the grid search technique to find the optimal parameter values for each dataset and examine every possible combination of parameter values on the grid. This technique returns the parameters that yield the best prediction for the given dataset [21]. A summary of the values obtained for the hyperparameters is shown in Table I.

3) Model Training and Validating: In this stage, we train our intrusion-detection model using the processed dataset. However, training alone does not ensure that the model will perform well when fed data that has not been seen before. Splitting the dataset in an 80:20 or 70:30 ratio at random may result in a

TABLE I
PARAMETER VALUES OBTAINED USING GRID SEARCH

Parameter	Usage	Car-hacking Dataset				Survival Dataset (Sonata)			Survival Dataset (Soul)			Survival Dataset (Spark)		
		DoS	Fuzzy	RPM	Gear	Flooding	Fuzzy	Malfunction	Flooding	Fuzzy	Malfunction	Flooding	Fuzzy	Malfunction
$alpha$	Improve model speed	5	0	1	0	0	0	0	0	0	0	0	0	0
$colsample\_bytree$	Improve Overfitting	0.8	0.3	0.1	0.3	0.1	0.3	0.3	0.1	0.5	0.2	0.1	0.5	0.3
$learning\_rate$	Optimize the chances to reach the best optimum	0.3	0.1	0.05	0.3	0.01	0.01	0.01	0.01	0.1	0.1	0.01	0.3	0.05
max_depth	Control model performance and complexity	10	6	6	6	1	9	6	1	6	5	1	6	6
$n\_estimators$	Determine the number of boosting rounds	100	100	100	100	100	100	100	100	100	100	100	100	100
min_child_weight	Control the complexity of the trees	3	1	1	1	1	1	3	1	1	1	1	1	3

model with a high bias since we would overlook information about the data that we have not utilized for training [22]. The 10-fold cross-validation approach is used to decrease biases. This approach divides the dataset into ten groups, of which nine are used for training, and the last one is used for validation. The model is trained and tested ten times until all groups are utilized to test the model. This method assures that every observation from the original dataset is included in the model's training and testing. We also implement an early stopping mechanism to avoid overfitting the model: when the model's performance on a validation dataset starts to deteriorate, the early stopping approach terminates it, avoiding overtraining.

4) Performance Evaluation: We evaluate our model on different attack scenarios such as DoS, fuzzy, RPM, and spoofing using standard metrics such as accuracy A, precision P, recall R, $F1$ score, and false positive rate FPR. The number of correct predictions divided by the total number of observations yields accuracy, calculated as $A = \frac{TN+TP}{TP+FP+TN+FN}$, where TN, TP, FN, FP are the true negative, true positive, false negative, and false positive values, respectively. The precision $P = \frac{TP}{(TP+FP)}$ is the correctness of positive predictions. The recall $R = \frac{TP}{(TP+FN)}$ is the fraction of correct attack predictions in the attack class. The $F1$ score summarizes a model's prediction effectiveness by averaging accuracy and recall, computed as $F1 = \frac{2\times(R\times P)}{R+P}$. FPR is the number of inaccurate positive findings in all the negative samples available during the test and is estimated by $FPR = \frac{FP}{(FP+TN)}$. In each cross-validation fold, these parameters are calculated, and the final evaluation result is the mean of the values obtained.

IV. EXPERIMENTAL VALIDATION

In this section, we discuss the datasets used to validate our detection model and the outcome of our experimentation.

A. Dataset Analysis

To evaluate the efficiency of our proposed IDS, we used four real-world CAN datasets from the Hacking and Countermeasure Research Lab [23]. The datasets consist of normal vehicle operation and the following attacks: DoS attack, Fuzzy attack, RPM, and gear spoofing attacks. The DoS attack is conducted by injecting a message with ID 0000 every 0.3 milliseconds. Data frames with random IDs and random data values are released to the CAN bus approximately every 0.5 milliseconds for the fuzzy attack, while CAN frames containing IDs of gear and RPM are used to spoof the bus every millisecond.

We also used the survival analysis dataset [24], which contains three separate datasets recorded from three different vehicles: a Hyundai YF Sonata, a KIA Soul, and a Chevrolet Spark. The datasets contain flooding, fuzzy, and malfunction attacks. Consistent with the car-hacking dataset DoS attack, the flooding attack injects messages with ID 0000. The fuzzy attack injects randomized CAN packets with IDs ranging from 0000 to 07FF every 0.0003 seconds. The malfunction attack manipulates data fields of targeted IDs.

For each message, the following parameters are provided: ID, timestamp, data length code, and the frame payload (consisting of 8 data bytes). In addition, a flag value was assigned to indicate whether the message was a genuine message (R) or an attack frame (T). Tables II and III summarize the datasets.

TABLE II
SUMMARY OF CAR-HACKING DATASET

Attack Type	Normal Messages	Injected Messages	Total Messages
DoS	3,078,250	587,521	3,665,771
Fuzzy	3,347,013	491,847	3,838,860
Gear Spoofing	3,845,890	594,252	4,443,142
RPM Spoofing	3,966,805	654,897	4,621,702

B. Experimental Results and Time Complexity Analysis

We pre-processed each dataset and obtained nine features: the CAN ID and eight subfeatures from the data bytes payload. We excluded the DLC and timestamp features of the datasets since the values did not add any advantage for predicting anomalies in our approach. We converted each data byte from hex to its equivalent integer representation. Before feeding the pre-processed dataset into the model, we tune each parameter and split the dataset for training and evaluation using the cross-validation process. We evaluated our model using accuracy, false positive rate, recall, precision, and F1 score. The experimental results obtained using the car-hacking and survival analysis dataset are summarized in Table IV and Table V, respectively.

One of the key purposes of building an anomaly detection system is to classify attack messages from benign signals. For this reason, a model should have low false-positive and false-negative rates. A high precision value refers to a low false-positive rate, and a high recall is associated with a low false-negative rate [25]. Table IV shows the algorithm achieved high recall values of 1 and 0.99 for DoS and fuzzy, respectively while obtaining approximately 0.9 for spoofing attacks. The precision of the model is over 98% for all attacks and a perfect

TABLE III
SUMMARY OF SURVIVAL ANALYSIS DATASET

Attack Type	Total Messages (Sonata)	Total Messages (Soul)	Total Messages (Spark)
Flooding	149,547	181,901	120,570
Fuzzy	135,670	249,990	65,665
Malfunction	132,651	173,436	79,787

precision of 100% for RPM spoofing attacks. Moreover, a lower value of FPR, 0.0029 in DoS and 0 for all other attacks, shows the system's ability to identify attacks efficiently. In addition, Table IV shows an F1 score of over 0.95 in all the attacks scenarios. Such a high value for the F1 score indicates that the model can accurately classify every observation. Furthermore, the model achieved above 0.99 accuracies for each attack, validating the prediction effectiveness of the proposed model. Similarly, Table V shows a high percentage of recall, precision, F1 score, accuracy, and low FPR for each dataset. On average, the proposed XGBoost anomaly detection model achieved over 99% accuracy on different CAN traffic datasets.

For our algorithm, preprocessing the data incurs a time complexity of $O(n)$. If n is the number of samples, t is the number of trees, h is the height of the trees, and S is the number of entries with non-missing values, training with XGBoost incurs $O(thSlogn)$ [26]. Since we utilize grid search to tune the hyperparameters, the time complexity of the model training increases as Grid Search goes through each possible combination of hyperparameters. To make the process faster, we parallelize the process with the n_jobs argument, which uses each core of a multicore machine and splits the time complexity by n [27]. However, since the data preprocessing and training are completed offline, these stages do not incur a cost during the actual intrusion detection stage. During detection, the XGBoost algorithm takes $O(th)$ for each new CAN payload, while preprocessing the new payload takes $O(1)$ constant time.

TABLE IV
SUMMARY OF XGBOOST MODEL EVALUATION ON CAR-HACKING DATASET

Attack \ Metrics	Recall	Precision	F1	Accuracy	FPR
DoS	1.0000	0.9852	0.9926	0.9976	0.0029
Fuzzy	0.9926	0.9999	0.9963	0.9990	0.0000
RPM Spoofing	0.9000	1.0000	0.9474	0.9858	0.0000
Gear Spoofing	0.9006	0.9982	0.9469	0.9864	0.0000

V. RELATED WORK

Several studies have proposed techniques for anomaly detection in CAN bus from a machine learning (ML) point of view [28], [29]. Some of the approaches in the literature include statistics-based adopting local outlier factor [29], cumulative sum change point [30], support vector machines [31], and neural networks [32].

Islam et al. [33] proposed a detection algorithm using graph properties with statistical tests. The algorithm transforms the CAN messages into a graph structure followed by a statistical analysis, such as computing threshold and chi-square values, to detect the abnormalities within the graphs. Olufowobi et al. [30] explored the CAN message frequencies and detected abrupt changes in the frequencies using a cumulative sum change-point algorithm. Taylor et al. [34] propose an IDS for CAN bus using long short-term memory (LSTM) classifier to predict the next message and significant deviation from the predicted word. The approach also indicates whether an actual message is considered an attack. Other researchers have leveraged this approach to also detect anomalies in CAN bus [28], [35], [36].

Seo et al. [37] introduced a generative adversarial net-based intrusion detection system, a deep-learning model, trained using solely attack-free data to boost the likelihood of detecting any unknown attack not seen during the training phase. D'Andrada et al. [38] presented a real-time isolation forest-based detection approach implemented using hardware descriptive language, based on a binary decision tree suitable for hardware, while Paul and Islam [39] used the binary classification of the artificial neural network to detect legitimate and compromised messages. Islam et al. [40] proposed a Gaussian naive Bayes method to identify a wide variety of CAN bus attacks in a short period by combining common graph characteristics with PageRank-related features. Furthermore, Li et al. [41] proposed an improved support vector domain description (SVDD) based detection system. The authors addressed the limitation of traditional SVDD by incorporating Markov chain to adapt related vehicle features and Gaussian kernel function to improve detection accuracy and reduce model redundancy. However, these detection methods are computationally intensive to efficiently detect anomalies and are often not practical for safety-critical system mainly due to speed (latency and throughput). Because of its learning algorithm-based and interpretable techniques for regression and classification [7], the XGBoost competitive and highly robust trees approach has been employed to best predict anomalous events in the CAN bus. XGBoost offers several parameters used to regulate model speed and complexity to get a better prediction in a shorter amount of time. Moreover, we used additional features such as grid search, early stopping, and cross-validation to enhance the performance of our model.

VI. CONCLUSION

This paper presents an anomaly detection method for the CAN bus using the XGBoost learning algorithm. The proposed method incorporates the CAN IDs and the data payload of the CAN frame into the XGBoost algorithm to predict anomalies in the bus operation. The optimum performance of the model was obtained by performing individual parameter tuning and cross-validation, followed by early stopping rounds. Furthermore, we evaluated the proposed model on open-source real dataset collected from three different vehicles. The evaluation results demonstrate the high detection performance of the proposed model. In future work, we will fine-tune the model to include more features of the CAN data frame for better attack coverage. Moreover, a comparative study with other machine learning-based detection approaches will be conducted.

REFERENCES

[1] H. Olufowobi and G. Bloom, "Chapter 16 - Connected Cars: Automotive Cybersecurity and Privacy for Smart Cities," in *Smart Cities Cybersecurity and Privacy*, D. B. Rawat and K. Z. Ghafoor,

TABLE V
SUMMARY OF XGBOOST MODEL EVALUATION ON SURVIVAL ANALYSIS DATASET

Metrics / Attack	Sonata					Soul					Spark				
	Recall	Precision	F1	Accuracy	FPR	Recall	Precision	F1	Accuracy	FPR	Recall	Precision	F1	Accuracy	FPR
Flooding	0.9000	1.0000	0.9474	0.9783	0.0000	0.9000	1.0000	0.9474	0.9818	0.0000	0.9000	1.0000	0.9474	0.9813	0.0000
Fuzzy	0.9908	0.9998	0.9953	0.9998	0.0000	0.9958	0.9999	0.9978	0.9993	0.0000	0.92	0.9734	0.9460	0.9907	0.0024
Malfunction	1.0000	0.9992	0.9996	0.9999	0.0001	0.9322	1.0000	0.9649	0.9971	0.0000	1.0000	0.9980	0.9990	0.9998	0.0002

Eds. Elsevier, Jan. 2019, pp. 227–240. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780128150320000160

[2] S. Checkoway, D. Mccoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," in *USENIX SECURITY*. USENIX, 2011.

[3] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle." BlackHat USA, 2015.

[4] K.-T. Cho and K. G. Shin, "Error handling of in-vehicle networks makes them vulnerable," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 1044–1055.

[5] S. Hounsinou, M. Stidd, U. Ezeobi, H. Olufowobi, M. Nasri, and G. Bloom, "Vulnerability of controller area network to schedule-based attacks," in *2021 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2021, pp. 495–507.

[6] G. Bloom, "Weepingcan: A stealthy can bus-off attack," in *Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*, 2021, p. 25.

[7] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[8] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," *Frontiers in neurorobotics*, vol. 7, p. 21, 2013.

[9] B. A. Tama and K.-H. Rhee, "An in-depth experimental study of anomaly detection using gradient boosted machine," *Neural Computing and Applications*, vol. 31, no. 4, pp. 955–965, 2019.

[10] H. Olufowobi, C. Young, J. Zambreno, and G. Bloom, "Saiducant: Specification-based automotive intrusion detection using controller area network (can) timing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 1484–1494, 2020.

[11] C. Young, J. Zambreno, H. Olufowobi, and G. Bloom, "Survey of automotive controller area network intrusion detection systems," *IEEE Design & Test*, vol. 36, no. 6, pp. 48–55, 2019.

[12] J. Liu, S. Zhang, W. Sun, and Y. Shi, "In-vehicle network attacks and countermeasures: Challenges and future directions," *IEEE Network*, vol. 31, no. 5, pp. 50–58, 2017.

[13] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen *et al.*, "Xgboost: extreme gradient boosting," *R package version 0.4-2*, vol. 1, no. 4, pp. 1–4, 2015.

[14] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[15] "Unveiling mathematics behind xgboost," *KD Nuggets*. [Online]. Available: https://www.kdnuggets.com/2018/08/unveiling-mathematics-behind-xgboost.html

[16] "Xgboost," *Geeks for Geeks*. [Online]. Available: https://www.geeksforgeeks.org/xgboost/

[17] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental security analysis of a modern automobile," in *2010 IEEE Symposium on Security and Privacy*, May 2010, pp. 447–462.

[18] U. Ezeobi, H. Olufowobi, C. Young, J. Zambreno, and G. Bloom, "Reverse engineering controller area network messages using unsupervised machine learning," *IEEE Consumer Electronics Magazine*, 2020.

[19] H. Olufowobi, S. Hounsinou, and G. Bloom, "Controller area network intrusion prevention system leveraging fault recovery," in *Proceedings of the ACM Workshop on Cyber-Physical Systems Security amp; Privacy*, ser. CPS-SPC'19. New York, NY, USA: Association for Computing Machinery, 2019, p. 63–73. [Online]. Available: https://doi.org/10.1145/3338499.3357360

[20] M. E. Verma, R. A. Bridges, J. J. Sosnowski, S. C. Hollifield, and M. D. Iannacone, "Can-d: A modular four-step pipeline for comprehensively decoding controller area network data," *arXiv preprint arXiv:2006.05993*, 2020.

[21] "Hyperparameter tuning. grid search and random search," *YOUR DATA TEACHER*. [Online]. Available: https://www.yourdatateacher.com/2021/05/19/hyperparameter-tuning-grid-search-and-random-search/

[22] "Why and how to cross validate a model?" *Towards Data Science*. [Online]. Available: https://towardsdatascience.com/why-and-how-to-cross-validate-a-model-d6424b45261f#

[23] H. Lee, S. H. Jeong, and H. K. Kim, "OTIDS: A Novel Intrusion Detection System for In-vehicle Network by Using Remote Frame," in *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, vol. 00, Aug. 2017, pp. 57–5709.

[24] M. L. Han, B. I. Kwak, and H. K. Kim, "Anomaly intrusion detection method for vehicular networks based on survival analysis," *Vehicular communications*, vol. 14, pp. 52–63, 2018.

[25] "Precision-recall," *Scikit Learn*. [Online]. Available: https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html#

[26] "Time complexity for different machine learning algorithms," *Marco Virgolin*. [Online]. Available: https://marcovirgolin.github.io/extras/details_time_complexity_machine_learning_algorithms/

[27] "Gridsearchcv and time complexity," *Data Science*. [Online]. Available: https://datascience.stackexchange.com/questions/97013/gridsearchcv-and-time-complexity

[28] M. D. Hossain, H. Inoue, H. Ochiai, D. Fall, and Y. Kadobayashi, "Lstm-based intrusion detection system for in-vehicle can bus communications," *IEEE Access*, vol. 8, pp. 185 489–185 502, 2020.

[29] J. Ning, J. Wang, J. Liu, and N. Kato, "Attacker identification and intrusion detection for in-vehicle networks," *IEEE Communications Letters*, vol. 23, no. 11, pp. 1927–1930, 2019.

[30] H. Olufowobi, U. Ezeobi, E. Muhati, G. Robinson, C. Young, J. Zambreno, and G. Bloom, "Anomaly detection approach using adaptive cumulative sum algorithm for controller area network," in *Proceedings of the ACM Workshop on Automotive Cybersecurity*, 2019, pp. 25–30.

[31] O. Avatefipour, A. S. Al-Sumaiti, A. M. El-Sherbeeny, E. M. Awwad, M. A. Elmeligy, M. A. Mohamed, and H. Malik, "An intelligent secured framework for cyberattack detection in electric vehicles' can bus using machine learning," *IEEE Access*, vol. 7, pp. 127 580–127 592, 2019.

[32] M. Kang and J. Kang, "A novel intrusion detection method using deep neural network for in-vehicle network security," in *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*, 2016, pp. 1–5.

[33] R. Islam, R. U. D. Refat, S. M. Yerram, and H. Malik, "Graph-based intrusion detection system for controller area networks," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[34] A. Taylor, S. Leblanc, and N. Japkowicz, "Anomaly detection in automobile control network data with long short-term memory networks," in *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2016, pp. 130–139.

[35] S. Longari, D. H. N. Valcarcel, M. Zago, M. Carminati, and S. Zanero, "Cannolo: An anomaly detection system based on lstm autoencoders for controller area network," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1913–1924, 2020.

[36] S. Tariq, S. Lee, H. K. Kim, and S. S. Woo, "Can-adf: The controller area network attack detection framework," *Computers & Security*, vol. 94, p. 101857, 2020.

[37] 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)*. IEEE, 2018, pp. 1–6.

[38] L. F. P. D'Andrada, P. F. de Araujo-Filho, and D. R. Campelo, "A real-time anomaly-based intrusion detection system for automotive controller area networks," in *Anais do XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. SBC, 2020, pp. 658–671.

[39] A. Paul and M. R. Islam, "An artificial neural network based anomaly detection method in can bus messages in vehicles," in *2021 International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI)*. IEEE, 2021, pp. 1–5.

[40] R. Islam, M. K. Devnath, M. D. Samad, and S. M. J. Al Kadry, "Ggnb: Graph-based gaussian naive bayes intrusion detection system for can bus," *Vehicular Communications*, vol. 33, p. 100442, 2022.

[41] X. Li, H. Zhang, Y. Miao, S. Ma, J. Ma, X. Liu, and K.-K. R. Choo, "Can bus messages abnormal detection using improved svdd in internet of vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 5, 2022.