electronics

MDPI

Article

An Effective Ensemble Learning-Based Real-Time Intrusion Detection Scheme for an In-Vehicle Network

Easa Alalwany [1] and Imad Mahgoub [2,*]

[1] College of Computer Science and Engineering, Taibah University, Yanbu 46421, Saudi Arabia; ealwani@taibahu.edu.sa
[2] Electrical Engineering & Computer Science, Florida Atlantic University, 777 Glades Road, Boca Raton, FL 33431, USA
* Correspondence: mahgoubi@fau.edu

Abstract: The emergence of connected and autonomous vehicles has led to complex network architectures for electronic control unit (ECU) communication. The controller area network (CAN) enables the transmission of data inside vehicle networks. However, although it has low latency and enjoys data broadcast capability, it is vulnerable to attacks on security. The lack of effectiveness of conventional security mechanisms in addressing these vulnerabilities poses a danger to vehicle safety. This study presents an intrusion detection system (IDS) that accurately detects and classifies CAN bus attacks in real-time using ensemble techniques and the Kappa Architecture. The Kappa Architecture enables real-time attack detection, while ensemble learning combines multiple machine learning classifiers to enhance the accuracy of attack detection. The scheme utilizes ensemble methods with Kappa Architecture's real-time data analysis to detect common CAN bus attacks. This study entails the development and evaluation of supervised models, which are further enhanced using ensemble techniques. The accuracy, precision, recall, and F1 score are used to measure the scheme's effectiveness. The stacking ensemble technique outperformed individual supervised models and other ensembles with accuracy, precision, recall, and F1 of 0.985, 0.987, and 0.985, respectively.

Keywords: controller area network; machine learning; ensemble learning; in-vehicle network; Kappa Architecture; intrusion detection system

Citation: Alalwany, E.; Mahgoub, I. An Effective Ensemble Learning-Based Real-Time Intrusion Detection Scheme for an In-Vehicle Network. *Electronics* **2024**, *13*, 919. https://doi.org/10.3390/electronics13050919

Academic Editors: Dongkyun Kim and Taeshik Shon

Received: 9 January 2024
Revised: 8 February 2024
Accepted: 26 February 2024
Published: 28 February 2024

1. Introduction

Significant advancements have been made in vehicle technologies over the past few decades, which has led to the development of connected and autonomous vehicles in the automobile industry. This digital transition has resulted in sophisticated vehicles that rely on complicated network architectures to ensure smooth communication among their many components. Among various network topologies, the controller area network (CAN) bus has emerged as a crucial basis for vehicle networks, enabling efficient data transmission between an enormous number of electronic control units (ECUs) [1–3].

The CAN bus has many benefits when used in automobiles, including reduced costs, enhanced real-time communication, and simple installation. The CAN bus is characterized by a number of features, including low latency, broadcast data transfer, and priority arbitration. These same characteristics, however, also increase its vulnerability to attack [4–6]. Conventional security mechanisms, including authentication, firewalls, and encryption are not suitable for CAN due to their structural limitations. These limitations include the absence of support for such security methods [7,8]. Thus, the modern Internet of Vehicles (IoV) is seriously threatened by the susceptibility of automotive networks to malicious threats. An attacker could potentially take control of essential vehicle functions, such as acceleration, steering, and braking [9,10].

As the CAN bus was originally developed without any security mechanism, it is vulnerable to a variety of attacks. Consequently, it is essential to develop and implement

highly accurate detection mechanisms to effectively identify and counter potential CAN bus attacks. The intrusion detection system (IDS) is a notable security mechanism that has received considerable attention among researchers [7,11–13]. To defend against intrusions in environments with limited resources, the authors in [3] recommend implementing IDS for in-vehicle networks (IVNs). Among the security techniques that have garnered interest from researchers is the utilization of intrusion detection systems. Investigating and developing these mechanisms is an effort to strengthen network security in the face of the rising number of attacks. The primary goal of IDSs is to capture information and detect unusual activities in order to prevent unauthorized acts, such as data theft, censorship, or protocol abuse. For the protection of computer resources at the network level, IDSs are implemented. IDSs are highly suitable for detecting anomalies in common networks due to their essential features and capabilities. Furthermore, the integration of IDS is crucial in order to detect and prevent malicious activities automatically, as highlighted in [14]. These systems analyze network traffic and security records, and audit data to find potential security breaches. IDSs have generated considerable interest from both academia and industry due to their critical role in protecting network security, particularly in industrial control networks (ICNs).

The integration of novel technologies can enhance the effectiveness and accuracy of IDSs in in-vehicle networks that are faced with challenging cyberattacks. To enhance the detection capabilities and improve the level of accuracy of results obtained by IDSs, the utilization of ensemble learning and Kappa Architecture real-time prediction techniques can be of essential significance.

Ensemble learning allows IDSs to utilize multiple machine learning (ML) classifiers, thereby enhancing their ability to detect attacks in the CAN bus network. Ensemble methods have been considered to increase intrusion detection accuracy over implementing single classifiers in [15–18]. The Kappa Architecture is used for real-time attack detection. The Apache Spark Streaming Discretized Stream (DStream) is utilized in real-time to process and store CAN bus data records in batches. The real-time stream records will be transmitted to the streaming layer of Kappa Architecture, where Spark Streaming can analyze the real-life data and then utilize the stored ensemble learning techniques to provide real-time detection and classification of attacks. By utilizing Apache Spark's real-time capabilities, the CAN bus network can quickly handle massive amounts of data, allowing for the rapid detection of anomalies or malicious activity. Prediction in real-time Apache Spark has emerged as a prominent strategy for overcoming limitations in numerous fields. It has received considerable attention as an alternative approach to overcoming these limitations. Apache Spark has been extensively implemented in numerous research fields and applications, such as healthcare systems [19], and traffic [20,21]. The integration of ensemble methods and efficient real-time data processing enhances the capability of IDSs to detect and classify common attacks in the CAN bus network including Denial-of-Service (DoS), fuzzing, replay, and spoofing.

The design of the CAN bus lacks robust security mechanisms, which results in vulnerabilities to numerous attacks. Designing IDSs with high accuracy is crucial for detecting and classifying attacks on the CAN bus. No study has proposed real-time CAN bus network threat detection. We developed an IDS that employs ensemble learning and the Kappa Architecture to detect and classify CAN bus attacks in real-time. This study proposes a solution that aims to design a highly effective IDS scheme. The solution uses ensemble methods and Kappa Architecture real-time detection techniques. By integrating ensemble methods into the IDS scheme and employing Kappa Architecture for real-time data analysis, the proposed solution aims to improve the accuracy and efficiency of detecting and classifying common CAN bus attacks.

- We design IDS with high accuracy for detecting and classifying attacks on the CAN bus. The scheme uses ensemble methods and Kappa Architecture's efficient real-time data processing technique. We enhance IDS accuracy and efficiency by integrating

ensemble methods and leveraging Kappa Architecture for real-time data analysis in the CAN bus attack detection scheme.

- We develop and test three supervised models that include hyperparameters tuning, balancing of data, and feature selection. The models are random forest (RF), eXtreme Gradient Boosting (XGBoost), and decision tree (DT).
- With three different ensemble methods, we combine the three supervised models to enhance our scheme's ability to detect and classify DoS, fuzzing, replay, and spoofing attacks. These ensemble methods are stacking, voting, and bagging.
- In the evaluation of our scheme, we use well-known evaluation metrics, including accuracy, recall, F1 score, precision, and area under the curve receiver operating characteristic (ROC-AUC). The four attacks are accurately detected with an error rate of 1.5%.

This study's remaining sections are structured as follows: Section 2 addresses the relevant work. Section 3 covers the CAN bus, ensemble methods, Kappa Architecture, and Apache Spark overview. The proposed scheme is illustrated in Section 4. Section 5 contains the results and analysis. Section 6 finally concludes the paper.

2. Related Work

This section contains a discussion of studies relevant to our study on IDSs for CAN bus networks. Moulahi et al. [22] proposed a CAN-bus-based system for detecting malicious intrusions in vehicles. They effectively detected three attacks by utilizing RF, DT, support vector machine (SVM), and multi-layer perceptron (MLP) classifiers to identify between normal and malicious messages. CAN bus intrusion detection may be a potential future enhancement. Reduced inception-ResNet, a deep convolutional neural network (DCNN) technique, was presented in a study [23] for identifying in-vehicle attacks. This technique outperformed DT, NB, SVM, and ANN in terms of accuracy of detection. The study showed that the CAN bus attack performance requires further improvement.

Furthermore, a method for identifying CAN bus intrusion threats was proposed in [24]. Combining a convolutional neural network with an attention-based gated recurrent unit, their hybrid model demonstrates a potential strategy for detecting vehicle threats. A comprehensive study conducted by [25] investigated communication vulnerabilities in in-vehicle networks and proposed machine learning-based security methods. While the survey critically evaluated the effectiveness of these security solutions, it emphasized the need to evaluate their performance in real-time.

Another study demonstrated the current focus on securing vehicular networks but also highlighted the lack of coverage for real-time approaches. The survey indicated that additional studies are needed to investigate real-time approaches and their implementation in the overall framework of security in vehicular networks [16]. In a different field, researchers used Apache Spark for real-time prediction. For instance, to improve prediction accuracy for real-time high-dimensional data challenges, ref. [26] developed a cascade structure on the Spark MLlib Big Data Analytics platform by combining MLP with LSTM. The results of the study showed that this method is effective in enhancing the accuracy of detection.

To identify DDoS attacks, ref. [27] proposed S-DDoS, a streaming-based distributed and real-time DDoS detection system built on the Apache Hadoop framework. Their system identified DDoS attack traffic in real-time using the K-means clustering algorithm, and it achieved a high detection accuracy. In another study [28], the authors used the UNSW-NB15 dataset to investigate efficient Spark-based anomaly identification. The authors proved that their proposed method produced higher accuracy when utilizing Spark for anomaly identification.

To our knowledge, no studies have proposed real-time CAN bus network attack detection. We designed an IDS that uses ensemble learning methods and the Kappa Architecture to achieve real-time highly accurate attack detection and classification in CAN bus networks, with a low error rate of 1.5%.

3. Background

This section introduces fundamental concepts essential to the study. It begins with an overview of the CAN bus, followed by a discussion of ensemble learning for enhancing detection accuracy. The significance of Kappa Architecture in enabling real-time data processing is then demonstrated, and Apache Spark is highlighted as an effective data processing engine.

3.1. CAN Bus

CAN bus is a serial communication protocol widely used for ECU-to-ECU communication because of its high speed and robust communication. In the 1980s, Robert Bosch GmbH developed it for automotive applications [29]. By allowing numerous ECUs to exchange data and control signals over a shared bus architectural concept, the CAN bus provides real-time communication. Its benefits include low cost, easy installation, and fault tolerance. A broadcast technique is used in the CAN bus protocol to ensure that all network nodes receive data and messages. Each bus is capable of performing a range of functions, including engine and braking control [30,31]. Seven fields represent a CAN packet. The start of the frame starts CAN message transmission to all nodes that are connected with one bit. The arbitration field (CAN ID) determines the message's significance. High-priority CAN ID messages have low values. The control field includes a data length code (DLC). The data field transfers data. The cyclic redundancy check (CRC) field validates data packets. The acknowledge (ACK) field confirms that network receiver nodes received CAN packets. The 'end of frame' completes the CAN messages. Attackers can use the ID-based priority mechanism of the CAN protocol to launch DoS attacks by sending high-priority messages through it and gaining access to other services. The vulnerability of the protocol to deceptive data injection into the bus network is attributed to its simplicity and high adaptability. Moreover, injection attacks are possible due to the absence of authentication and source/destination addresses in CAN messages. Such attacks frequently exploit the onboard diagnostics II (OBDII) interface to inject malicious messages and disable functionalities, which may come from external sources such as Wi-Fi, cellular, or Bluetooth, or internal sources such as the OBD [32–35]. The CAN protocol lacks sufficient security support, which limits vehicle communication security. Cyberattacks have targeted the CAN bus because of its vulnerabilities [16,36,37].

3.2. CAN Bus Attacks

Attackers can utilize these vulnerabilities to compromise the communication system's availability, confidentiality, and integrity [38–40]. In [10], hacking was employed to gain unauthorized access and control over the functionalities of a Jeep Cherokee. The most common inter-vehicle attacks are message injection attacks, which include DoS, spoofing, replay, and fuzzing [41]. A DoS attack occurs when the attacker intends to render communication services inaccessible. This is achieved by increasing the number of messages transmitted via the CAN bus. The attacker will send high-frequency messages that either have a higher priority or lower priority than a normal message. A malicious node can intentionally raise the level of bus occupancy, resulting in delays or perhaps preventing other communications. Fuzzing is a technique employed by attackers wherein they inject messages with randomly generated CAN IDs that have the aim of imitating legitimate IDs. Attackers can launch this type of attack against a network at any time by transmitting a large number of compromised IDs and data. This attack is possible even if the victim node is unknown. The attack happens due to weaknesses in the CAN bus protocol's authentication and data integrity. Replay attack: the aim of this attack is to intercept authentic CAN messages during transmission and subsequently re-transmit them into the CAN bus. This attack may cause unauthorized actions or the broadcast of false information throughout the network. A spoofing attack involves the unauthorized injection of messages into the CAN bus by utilizing CAN IDs that are associated with authorized nodes. This is made

possible due to the absence of authentication in the communication of the CAN bus. This attack has the potential to reduce communication efficiency.

3.3. Ensemble Methods

Ensemble learning is a methodology in machine learning that combines several different models to enhance the accuracy of predictions or classifications. Ensemble learning is a method for enhancing performance, robustness, and generalization by combining the strengths of multiple models. Bagging, voting, and stacking are common ensemble learning techniques. Each model contributes its own predictions to the ensemble. Ensemble learning through stacking is a common technique employed for combining the outcomes of several basic classifiers via a meta-model. The objective is to enhance the performance of the ensemble through the fixing of errors. The main concept is to use a different classifier to fix the weaknesses of previous classifiers [42]. Two-stage learners are used in the stacking technique to determine the most effective way of combining the individual models of the base learners with the meta-learner. The process of voting learning involves combining multiple model predictions to produce the final prediction. It is predicated on the idea that integrating the decisions of multiple models can typically enhance performance. The process of voting-based learning applies to both classification and regression tasks [43]. Ensemble learning uses bagging, to increase the machine learning model's stability and accuracy [44]. It utilizes random sampling with replacement to create various subsets of the original training data. Each subset trains a model, and the final prediction is based on all models' predictions.

3.4. Kappa Architecture

The Kappa is an architecture intended to handle real-time data processing with high efficiency, scalability, and fault tolerance. As an alternative to Lambda, it was first proposed by Jay Kreps. By combining batch and real-time processing into a single layer so it consists only of stream and service layers, the Kappa Architecture streamlines the data processing pipeline, as depicted in Figure 1. It utilizes a single stream processing system that manages both historical and incoming data, resulting in low latency and simplified data management. The Kappa Architecture employs a message-passing system to process real-time data. Apache Spark MLlib is used by the streaming layer for developing machine learning models for real-time prediction and analytics. The results are transmitted to the service layer for storage in Hadoop Base, Cassandra, or MySQL, among others. The service layer allows for real-time analysis, decision-making, and user access via visualization, dashboards, and application programming interfaces (APIs). To implement the Kappa, we use Apache Spark and Spark Streaming so we can develop a real-time prediction system that efficiently analyzes data, performs machine learning, stores results, and offers real-time analysis [45].

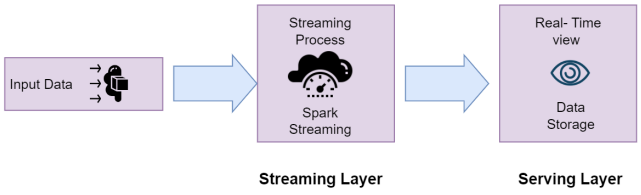

Figure 1. Kappa Architecture.

3.5. Apache Spark

Apache Spark is a scalable and efficient distributed computing framework for processing and analyzing large amounts of data. Spark provides a single computational model for running a variety of applications, including batch processing, real-time streaming, machine learning, and graph processing. It offers high-level APIs in a number of languages, which programmers can use to create sophisticated data processing pipelines. These program languages include Java, Python, R, and Scala. Spark's in-memory computational capabilities

allow for rapid data processing and iterative algorithms, making it suitable for real-time analytics and machine learning tasks. The analysis of streaming data can be performed in real-time using machine learning and statistical approaches. This helps identify and deal with issues that are time-sensitive and crucial. Streaming data lets systems analyze data in real-time and make predictions. Streaming data are usually ingested in real-time but are processed in small amounts, which is sufficient to train machine learning models [46–50].

4. Proposed Method

It is essential for the safety of drivers, passengers, and the vehicle itself that communication within the vehicle be protected. The scheme aims to enhance CAN bus communication security by detecting attacks on the CAN bus network. The objective of the study is to design an IDS by utilizing ensemble learning methods for the purpose of detecting and classifying real-time attacks on the CAN bus. The ensemble approach can effectively and accurately, identify and detect attacks in CAN bus communication by leveraging the strengths of multiple ML models. For analyzing the real-time data collected by the CAN bus and using ensemble learning algorithms for intrusion detection, The Kappa Architecture enables real-time attack detection. The effectiveness of the proposed approach was evaluated using well-known evaluation metrics. Our proposed model's workflow is presented in Figure 2, which includes a learning layer and real-time capabilities. As shown in Figure 3, our IDS scheme can be integrated into the architecture of the in-vehicle network in [51,52]. A network-based IDS is strategically positioned in the vehicle's CAN architecture at the CAN bus. The system actively monitors and inspects the communication system within the vehicle by placing sensors to collect CAN messages and utilizing IDS. The IDS detects any cybersecurity threat by analyzing the communication within the CAN network of the powertrain, head unit, chassis, and body electronics [5].

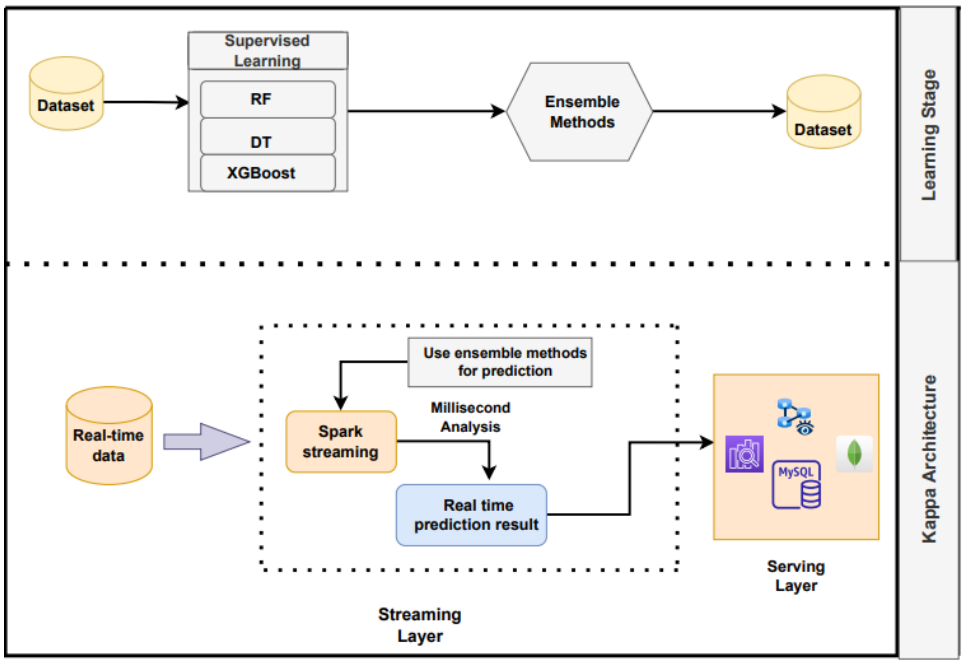

Figure 2. The workflow of our proposed model.

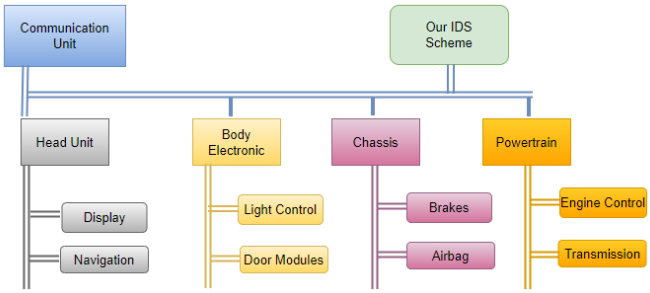

Figure 3. The architecture of the in-vehicle network with our proposed scheme integrated.

4.1. Dataset

For this study, we obtained the CAN intrusion dataset from IEEE DataPort [53]. The dataset was selected since it was collected in 2020. One important component of this dataset is that it has full labels, which are required for supervised models. The dataset's authors gathered millions of CAN bus messages by using data taken from actual vehicles. DoS, spoofing, replay, and fuzzing were the four types of attacks launched against the dataset. Features in the dataset include the timestamp feature which records each CAN bus message's capture time. CAN communications are identified by an arbitration ID. Each CAN message's data length code (DLC) shows its data payload length. CAN data, the network's payload, are represented by the data feature. The dataset's target feature detects CAN message attacks. It labels instances as normal or associated with DoS, spoofing, replay, or fuzzing attacks. The number of benign and attacked samples is shown in Table 1.

Table 1. The number of benign and attacked samples.

Message Type	Count
Normal	3,372,743
Flooding	154,180
Fuzzing	89,879
Replay	47,593
Spoofing	7756
Total	3,672,151

4.2. Learning Stage

In this study, multiple preprocessing methods were used to prepare the dataset for analysis. The preprocessing stage utilized label encoding to transform categorical variables into numerical representations. The process involves assigning distinct numerical codes to individual categories, thereby enabling the efficient processing of data by machine learning models. To address problems such as duplicate and null values, the data were cleaned. Redundancy was eliminated, and data integrity was ensured by removing duplicate instances. The feature selection was employed to determine the most significant features present in the given dataset. The extra tree technique was used to determine the significance of each feature and eliminate those with minimal or no relevance to the analysis. Dimensionality was reduced and modeling efficiency was improved with this approach. Figure 4 presents an overview of our proposed model's workflow, which includes the following phases: the CAN bus input dataset, data preprocessing, supervised model building, ensemble modeling, attack classification, and scheme evaluation. The dataset split consists of 2,495,077 training samples and 817,042 test samples, with training taking 60 min and testing taking 20 min. We balanced the data during model training to eliminate the data imbalance across classes. With the majority class dominating predictions, imbalanced classes may affect model performance. To address this, we used near-miss undersampling. Near-miss balances the dataset by reducing the majority class data to match the minority class, enhancing the classification accuracy. During the supervised model phase, we utilized three models that have demonstrated high performance in our previous research [15]: RF [54], DT [55],

and XGBoost [56]. The previous research indicated that these models show a high level of accuracy, making them appropriate candidates for the present study. We tuned hyperparameters to improve the machine learning model performance. Each model's optimal hyperparameters were found via a random search. This helps find the most effective model performance. In order to enhance the performance of a single classifier, ensemble methods can be used to decrease overfitting, strengthen the classifier, and improve its effectiveness at generalizing. These ensemble methods are stacking voting and bagging. The stacking ensemble approach was used during the ensemble phase, and the RF, DT, and XGBoost models were used as the base models. The XGBoost model is used as the meta-model for this ensemble of methods. The predictive ability of the meta-model is improved by using the predictions of the basic models as inputs, which is achieved by stacking. The bagging classifier has been combined with the XGBoost model in order to reduce overfitting in the bagging process. Aggregating these models' predictions improves the model's stability and generalization. the combination of the RF, DT, and XGBoost models is utilized for the voting ensemble technique. Voting combines the predictions of multiple models to generate an accurate prediction. The predictions of each model are given equal weight, and the majority prediction is chosen as the final output. The objective of our study was to enhance predictive performance by utilizing a combination of supervised models and ensemble techniques, thus leveraging the unique strengths of each model.

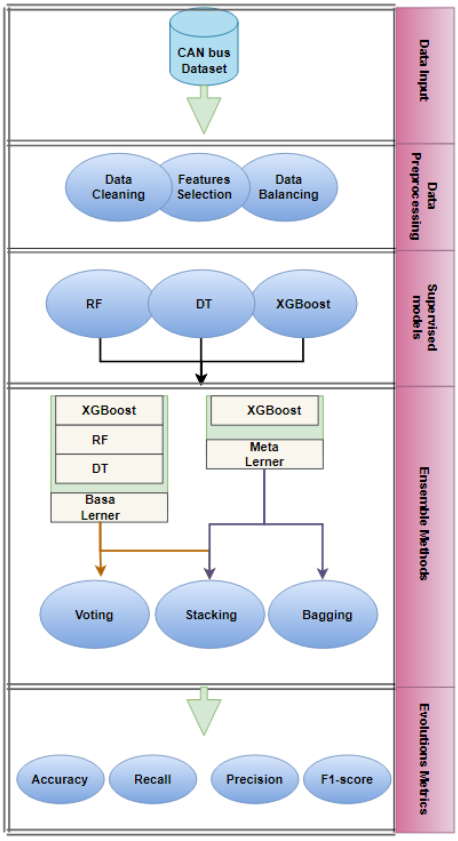

Figure 4. The workflow for ensemble classifiers.

4.3. Kappa Architecture

The Kappa Architecture is a common design pattern utilized for the purpose of real-time data processing and analysis. It is specifically employed in situations where the processing of continuous data streams with low latency is of utmost importance. The Kappa Architecture is used in this scenario for processing real-time traffic records. The design utilizes Apache Spark Streaming for the purpose of ingesting and processing real-time data in the form of a Discretized Stream (DStream), which stands for a sequence of RDDs. Kappa

Architecture's streaming layer uses Spark Streaming to perform a variety of data operations and transformations. Within the framework of the processed data, ensemble methods are employed to provide real-time malicious traffic detection using ensemble methods. These ensemble methods have been trained using historical data. The streaming layer then transmits the prediction results to the serving layer of the Kappa Architecture. The serving layer analyzes and makes the results available to other applications and systems. When applied to traffic data, the Kappa Architecture uses Spark Streaming and ensemble algorithms to process the data in real-time and provide accurate detection. A real-time data stream from the CAN bus is continuously appended to the main dataset in the streaming layer, and the historical database will be updated to accommodate new examples and features, ensuring the system remains current.

5. Results and Discussions

In this study, we utilized accuracy, precision, recall, F1 score, and ROC metrics for evaluating the performance of the scheme. These metrics are defined below [57];

Accuracy is a frequently used metric to evaluate classification performance, calculated as the ratio of correctly classified samples to the total number of samples. Precision is a metric that evaluates the predictive capability of an algorithm. It is calculated by dividing the number of accurately identified positive samples (TPs) by the sum of accurately and incorrectly classified positive samples. Recall measures the reliability of an algorithm in accurately identifying all relevant instances, calculated as the true positives (TPs) divided by the sum of true positives and false negatives (FNs). The F1 score is a single value that combines precision and recall metrics to evaluate a model's performance. ROC graphs illustrate the trade-off between sensitivity (true positive rate, or TPR) and specificity (true negative rate, or TNR), serving as a tool to evaluate the performance of classification models.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{4}$$

$$Specificity = \frac{TN}{FP + TN} \tag{5}$$

$$Sensitivity = Recall \tag{6}$$

In the results of supervised models, we focused on improving the effectiveness of three supervised models: random forest (RF), XGBoost, and decision tree (DT). We were able to enhance the predictive capabilities of the models by employing techniques such as data balancing, feature selection, and model tuning. We utilized a K-fold cross-validation process with K equals 5. As seen in Table 2, the DT model's accuracy, precision, recall, and F1 were 0.981, 0.984, 0.981, and 0.981, respectively. This results in a balanced detection of class labels. The accuracy of the XGBoost model was 0.982, and its precision and recall scores were 0.985 and 0.982, respectively, demonstrating outstanding performance. Its balanced 0.9829 F1 score emphasizes its strong prediction ability. The Rf model performed well, even if its accuracy was 0.973. A 0.977 precision and 0.973 recall score indicate that the model correctly detected positive events. The F1 score of 0.973 indicates a balanced performance in terms of recall and precision. Our analysis demonstrates the efficiency of data balancing, feature selection, and model tuning in enhancing the performance of supervised models.

Table 2. The results of the evaluation of the supervised models.

Models	Accuracy	Recall	Precision	F1
RF	0.973	0.973	0.977	0.973
DT	0.981	0.981	0.984	0.981
XGBoost	0.982	0.982	0.985	0.982

Stacking, bagging, and voting were used to improve the detection and classification performance for CAN bus attacks using ensemble methods. As seen in Table 3, the stacking ensemble obtained an accuracy of 0.985 with high precision, recall, and F1 score, indicating its capacity to accurately detect class labels and recognize positive instances. The bagging ensemble obtained an accuracy of 0.983 and showed balanced precision, recall, and F1 score, demonstrating that it was effective at capturing true positive instances. The ensemble voting system obtained a 0.979 accuracy with the balanced precision, recall, and F1 score, demonstrating its ability to make accurate detection across classes. These ensemble methods were effective in enhancing the performance of the models and providing accurate and robust predictions for detecting CAN bus attacks.

Table 3. Results of the evaluation for the three ensemble classifiers.

Models	Accuracy	Recall	Precision	F1
Stacking	0.985	0.985	0.987	0.985
Bagging	0.983	0.983	0.985	0.983
Voting	0.979	0.979	0.981	0.978

The results of our study showed the efficiency of both supervised models and ensemble methods for improving the performance of CAN bus attack detection. The supervised models performed effectively, and the ensemble approaches, which combine the best features of several models, significantly increased detection accuracy and robustness. The stacking ensemble method outperformed all supervised models and other ensemble methods in detecting CAN bus attacks. The stacking ensemble outperformed RF, XGBoost, DT, voting, and bagging with an accuracy of 0.9856. The stacking ensemble technique enhances the accuracy and robustness of detection by leveraging the strengths of various models. Figure 5 illustrates a comparison to the supervised models and voting and bagging methods, the obtained results for the classification task are improved for the stacking method. In terms of the four metrics. Comparing attack detection improvements between individual models and ensemble learning is crucial as it directly influences the safety of drivers, passengers, and vehicles.

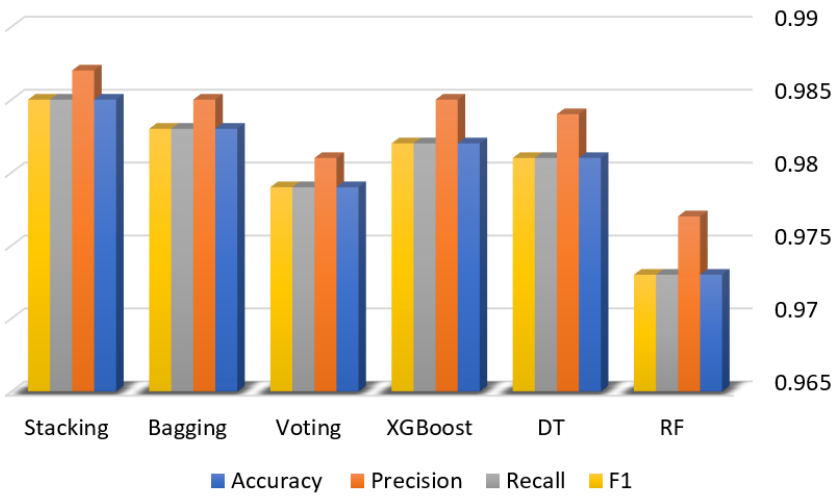

Figure 5. The comparative performance of ensemble classifiers and supervised models.

Figure 6 depicts the AUC-ROC curve for each class in the stacking method. The x-axis represents the false positive rate, while the y-axis represents the true positive rate. A high AUC-ROC score is preferable because it indicates the robustness of the model. A model is considered reliable if its AUC-ROC is close to 1.0. The ROC results reveal outstanding performance in detecting DoS and fuzzing attacks, with an AUC of 1.00, while the detection of replay and spoofing attacks demonstrates a high AUC of 0.99.

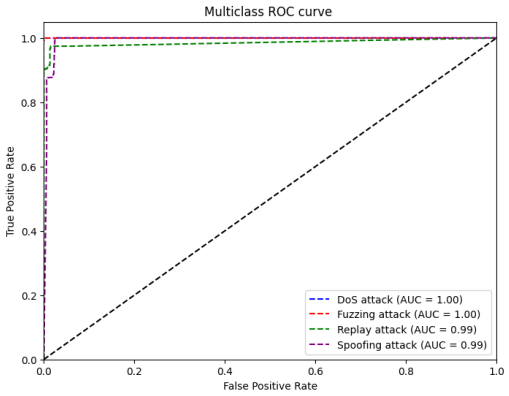

Figure 6. The results of the AUC-ROC curve for each class in the stacking method.

In Figure 7, which illustrates the results of the stacking method, we obtained high performance, comparing actual and predicted values across the four classes. This finding indicates the model demonstrates the ability to consistently and accurately detect each individual class, providing reliable results. The four attacks are accurately detected with an error rate of 1.5%. By utilizing the stacking method that provides high accuracy, our scheme efficiently detects and classifies common attacks on the CAN bus. The shortest time to detect spoofing is 0.26 s, followed by fuzzing, replay, and DoS at 0.29, 0.30, and 0.28 s, respectively. This real-time responsiveness provides robust security along with high accuracy.

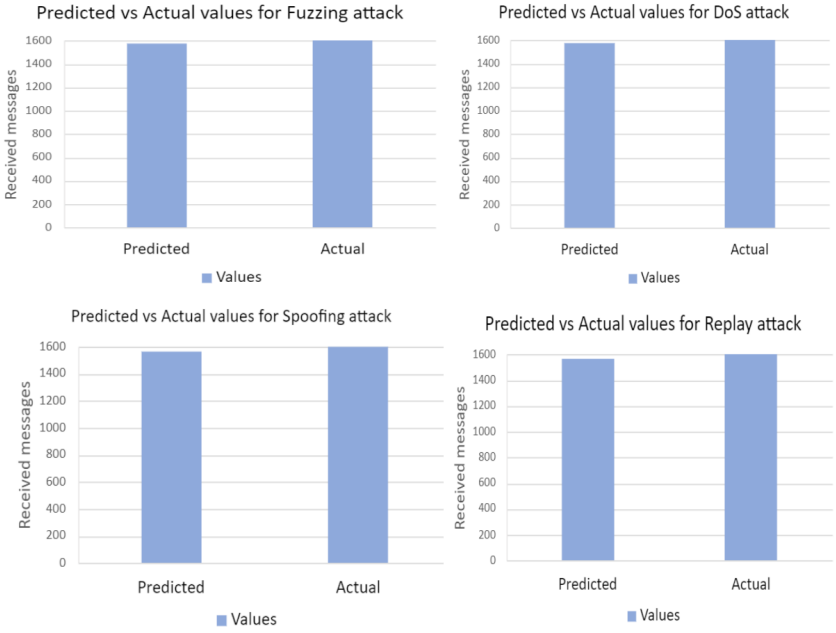

Figure 7. The results of the stacking method in comparing actual and predicted values across the four classes of attacks.

Compare our method to the proposed methods [24,58] in Table 4, demonstrating promising results in evaluation metrics. Results from experiments show the effectiveness of our IDS in detecting and classifying attacks on in-vehicle networks.

Table 4. Comparison of our method to proposed methods.

Citation	Year	Models	Accuracy	Precision	Recall	F1 Score
[24]	2021	CANintelliIDS	-	0.936	0.939	0.937
[58]	2022	SVM	0.979	0.98	0.96	0.97
Our	2024	Stacking	0.985	0.987	0.985	0.985

6. Conclusions

The controller area network (CAN) enables the transmission of data inside vehicle networks. However, although it has low latency and enjoys data broadcast capability, it is vulnerable to attacks on security. Vehicle safety is in danger since conventional security measures are ineffective at addressing these vulnerabilities. To address these challenges, in this study, we proposed an effective IDS scheme designed for the CAN bus network. The scheme's objective is to enhance IDS that accurately detects and classifies CAN bus attacks in real-time using ensemble techniques and the Kappa Architecture. The Kappa Architecture enables real-time attack detection, while ensemble learning combines multiple machine learning classifiers for enhanced attack detection accuracy. We developed and evaluated supervised models and utilized ensemble techniques to enhance the effectiveness of the detection and classification of common CAN bus attacks. The results showed that ensemble methods significantly enhanced the detection accuracy. By combining the strengths of multiple models, the stacking ensemble technique outperformed individual supervised models and other ensembles with accuracy, precision, recall, and F1 of 0.985, 0.987, and 0.985, respectively, and experienced less run-time. Our scheme demonstrated promising results in evaluation metrics, by exhibiting the effectiveness of our IDS in detecting and classifying attacks on in-vehicle networks compared to the proposed methods. For future work, we will investigate the potential of collaborative learning, federated learning, and transfer learning within domain transfer learning-based intrusion detection systems (DTL-IDSs) to enhance the efficiency of security threat prevention and detection [14].

Author Contributions: Conceptualization, E.A. and I.M.; methodology, E.A. and I.M.; software, E.A.; validation, E.A. and I.M.; formal analysis, E.A. and I.M.; investigation, E.A. and I.M.; writing—original draft preparation, review and editing, I.M.; visualization, E.A.; supervision, I.M. and E.A.; writing—project administration, I.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data available upon request.

Acknowledgments: This work is part of the Smart Drive initiative at Tecore Networks Lab at Florida Atlantic University.

Conflicts of Interest: The authors declare that they have no conflicts of interest.

References

1. Kleberger, P.; Olovsson, T.; Jonsson, E. Security aspects of the in-vehicle network in the connected car. In Proceedings of the 2011 IEEE Intelligent Vehicles Symposium (IV), Baden-Baden, Germany, 5–9 June 2011; pp. 528–533.
2. Liu, J.; Zhang, S.; Sun, W.; Shi, Y. In-vehicle network attacks and countermeasures: Challenges and future directions. *IEEE Netw.* **2017**, *31*, 50–58. [CrossRef]
3. Wu, W.; Li, R.; Xie, G.; An, J.; Bai, Y.; Zhou, J.; Li, K. A survey of intrusion detection for in-vehicle networks. *IEEE Trans. Intell. Transp. Syst.* **2019**, *21*, 919–933. [CrossRef]
4. Petit, J.; Shladover, S.E. Potential cyberattacks on automated vehicles. *IEEE Trans. Intell. Transp. Syst.* **2014**, *16*, 546–556. [CrossRef]
5. Lokman, S.F.; Othman, A.T.; Abu-Bakar, M.H. Intrusion detection system for automotive Controller Area Network (CAN) bus system: A review. *EURASIP J. Wirel. Commun. Netw.* **2019**, *2019*, 184. [CrossRef]
6. Young, C.; Zambreno, J.; Olufowobi, H.; Bloom, G. Survey of automotive controller area network intrusion detection systems. *IEEE Des. Test* **2019**, *36*, 48–55. [CrossRef]

7. Bozdal, M.; Samie, M.; Aslam, S.; Jennions, I. Evaluation of can bus security challenges. *Sensors* **2020**, *20*, 2364. [CrossRef]
8. Wang, Q.; Qian, Y.; Lu, Z.; Shoukry, Y.; Qu, G. A delay based plug-in-monitor for intrusion detection in controller area network. In Proceedings of the 2018 Asian Hardware Oriented Security and Trust Symposium (AsianHOST), Hong Kong, China, 17–18 December 2018; pp. 86–91.
9. Koscher, K.; Czeskis, A.; Roesner, F.; Patel, S.; Kohno, T.; Checkoway, S.; McCoy, D.; Kantor, B.; Anderson, D.; Shacham, H.; et al. Experimental security analysis of a modern automobile. In Proceedings of the 2010 IEEE Symposium on Security and Privacy, Oakland, CA, USA, 16–19 May 2010; pp. 447–462.
10. Miller, C.; Valasek, C. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA* **2015**, *2015*, 1–91.
11. Karopoulos, G.; Kambourakis, G.; Chatzoglou, E.; Hernández-Ramos, J.L.; Kouliaridis, V. Demystifying in-vehicle intrusion detection systems: A survey of surveys and a meta-taxonomy. *Electronics* **2022**, *11*, 1072. [CrossRef]
12. Khraisat, A.; Alazab, A. A critical review of intrusion detection systems in the internet of things: Techniques, deployment strategy, validation strategy, attacks, public datasets and challenges. *Cybersecurity* **2021**, *4*, 1–27. [CrossRef]
13. Shichun, Y.; Zheng, Z.; Bin, M.; Yifan, Z.; Sida, Z.; Mingyan, L.; Yu, L.; Qiangwei, L.; Xinan, Z.; Mengyue, Z.; et al. Essential Technics of Cybersecurity for Intelligent Connected Vehicles: Comprehensive Review and Perspective. *IEEE Internet Things J.* **2023**, *10*, 21787–21810. [CrossRef]
14. Kheddar, H.; Himeur, Y.; Awad, A.I. Deep transfer learning for intrusion detection in industrial control networks: A comprehensive review. *J. Netw. Comput. Appl.* **2023**, *220*, 103760. [CrossRef]
15. Alalwany, E.; Mahgoub, I. Classification of Normal and Malicious Traffic Based on an Ensemble of Machine Learning for a Vehicle CAN-Network. *Sensors* **2022**, *22*, 9195. [CrossRef]
16. Aliwa, E.; Rana, O.; Perera, C.; Burnap, P. Cyberattacks and countermeasures for in-vehicle networks. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 21. [CrossRef]
17. Alhowaide, A.; Alsmadi, I.; Tang, J. Ensemble detection model for IoT IDS. *Internet Things* **2021**, *16*, 100435. [CrossRef]
18. Pham, N.T.; Foo, E.; Suriadi, S.; Jeffrey, H.; Lahza, H.F.M. Improving performance of intrusion detection system using ensemble methods and feature selection. In Proceedings of the Australasian Computer Science Week Multiconference, Brisbane, QLD, Australia, 29 January–2 February 2018; pp. 1–6.
19. Ed-Daoudy, A.; Maalmi, K. Real-time machine learning for early detection of heart disease using big data approach. In Proceedings of the 2019 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS), Fez, Morocco, 3–4 April 2019; pp. 1–5.
20. Ameer, S.; Shah, M.A.; Khan, A.; Song, H.; Maple, C.; Islam, S.U.; Asghar, M.N. Comparative analysis of machine learning techniques for predicting air quality in smart cities. *IEEE Access* **2019**, *7*, 128325–128338. [CrossRef]
21. Saraswathi, A.; Mummoorthy, A.; GR, A.R.; Porkodi, K. Real-time traffic monitoring system using spark. In Proceedings of the 2019 International Conference on Emerging Trends in Science and Engineering (ICESE), Hyderabad, India, 18–19 September 2019; Volume 1, pp. 1–6.
22. Moulahi, T.; Zidi, S.; Alabdulatif, A.; Atiquzzaman, M. Comparative performance evaluation of intrusion detection based on machine learning in in-vehicle controller area network bus. *IEEE Access* **2021**, *9*, 99595–99605. [CrossRef]
23. Song, H.M.; Woo, J.; Kim, H.K. In-vehicle network intrusion detection using deep convolutional neural network. *Veh. Commun.* **2020**, *21*, 100198. [CrossRef]
24. Javed, A.R.; Ur Rehman, S.; Khan, M.U.; Alazab, M.; Reddy, T. CANintelliIDS: Detecting in-vehicle intrusion attacks on a controller area network using CNN and attention-based GRU. *IEEE Trans. Netw. Sci. Eng.* **2021**, *8*, 1456–1466. [CrossRef]
25. Rathore, R.S.; Hewage, C.; Kaiwartya, O.; Lloret, J. In-vehicle communication cyber security: Challenges and solutions. *Sensors* **2022**, *22*, 6679. [CrossRef] [PubMed]
26. Khan, M.A.; Karim, M.R.; Kim, Y. A two-stage big data analytics framework with real world applications using spark machine learning and long short-term memory network. *Symmetry* **2018**, *10*, 485. [CrossRef]
27. Patil, N.V.; Rama Krishna, C.; Kumar, K. S-DDoS: Apache spark based real-time DDoS detection system. *J. Intell. Fuzzy Syst.* **2020**, *38*, 6527–6535. [CrossRef]
28. Othman, D.M.S.; Hicham, R.; Zoulikha, M.M. An efficient spark-based network anomaly detection. *Int. J. Comput. Digit. Syst.* **2020**, *9*, 1175–1185. [CrossRef]
29. Bosch, C. *Specification Version 2.0*; Robert Bosch GmbH: Gerlingen, Germany, 1991; Volume 1.
30. Johansson, K.H.; Törngren, M.; Nielsen, L. Vehicle applications of controller area network. In *Handbook of Networked and Embedded Control Systems*; CRC Press: Boca Raton, FL, USA, 2005; pp. 741–765.
31. Takefuji, Y. Connected vehicle security vulnerabilities [commentary]. *IEEE Technol. Soc. Mag.* **2018**, *37*, 15–18. [CrossRef]
32. Bozdal, M.; Samie, M.; Jennions, I. A survey on can bus protocol: Attacks, challenges, and potential solutions. In Proceedings of the 2018 International Conference on Computing, Electronics & Communications Engineering (iCCECE), Southend, UK, 16–17 August 2018; pp. 201–205.
33. Song, H.M.; Kim, H.R.; Kim, H.K. Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network. In Proceedings of the 2016 International Conference on Information Networking (ICOIN), Kota Kinabalu, Malaysia, 13–15 January 2016; pp. 63–68.
34. Lee, H.; Jeong, S.H.; Kim, H.K. OTIDS: A novel intrusion detection system for in-vehicle network by using remote frame. In Proceedings of the 2017 15th Annual Conference on Privacy, Security and Trust (PST), Calgary, AB, Canada, 28–30 August 2017.

35. Groza, B.; Murvay, P.S. Efficient intrusion detection with bloom filtering in controller area networks. *IEEE Trans. Inf. Forensics Secur.* **2018**, *14*, 1037–1051. [CrossRef]
36. Avatefipour, O.; Malik, H. State-of-the-art survey on in-vehicle network communication (CAN-Bus) security and vulnerabilities. *arXiv* **2018**, arXiv:1802.01725.
37. Pan, L.; Zheng, X.; Chen, H.; Luan, T.; Bootwala, H.; Batten, L. Cyber security attacks to modern vehicular systems. *J. Inf. Secur. Appl.* **2017**, *36*, 90–100. [CrossRef]
38. Nowdehi, N.; Lautenbach, A.; Olovsson, T. In-vehicle CAN message authentication: An evaluation based on industrial criteria. In Proceedings of the 2017 IEEE 86th Vehicular Technology Conference (VTC-Fall), Toronto, ON, Canada, 24–27 September 2017; pp. 1–7.
39. Zhang, H.; Meng, X.; Zhang, X.; Liu, Z. CANsec: A practical in-vehicle controller area network security evaluation tool. *Sensors* **2020**, *20*, 4900. [CrossRef]
40. Alalwany, E.; Mahgoub, I. Security and Trust Management in the Internet of Vehicles (IoV): Challenges and Machine Learning Solutions. *Sensors* **2024**, *24*, 368. [CrossRef]
41. Seo, E.; Song, H.M.; Kim, H.K. GIDS: GAN based intrusion detection system for in-vehicle network. In Proceedings of the 2018 16th Annual Conference on Privacy, Security and Trust (PST), Belfast, UK, 28–30 August 2018; pp. 1–6.
42. Wolpert, D.H. Stacked generalization. *Neural Netw.* **1992**, *5*, 241–259. [CrossRef]
43. Dietterich, T.G. Ensemble learning. In *The Handbook of Brain Theory and Neural Networks*; MIT Press: Cambridge, MA, USA, 2002; Volume 2, pp. 110–125.
44. Quinlan, J.R. Bagging, boosting, and C4.5. In Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96), Portland, OR, USA, 4–8 August 1996; Volume 1, pp. 725–730.
45. Lin, J. The lambda and the kappa. *IEEE Internet Comput.* **2017**, *21*, 60–66. [CrossRef]
46. Choudhary, P.; Garg, K. Comparative analysis of spark and hadoop through imputation of data on big datasets. In Proceedings of the 2021 IEEE Bombay Section Signature Conference (IBSSC), Gwalior, India, 18–20 November 2021; pp. 1–6.
47. Kumar, K.; Sharma, N.A.; Ali, A.S. Machine Learning Solutions for Investigating Streams Data using Distributed Frameworks: Literature Review. In Proceedings of the 2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE), Brisbane, Australia, 8–10 December 2021; pp. 1–6.
48. Tun, M.T.; Nyaung, D.E.; Phyu, M.P. Performance evaluation of intrusion detection streaming transactions using apache kafka and spark streaming. In Proceedings of the 2019 International Conference on Advanced Information Technologies (ICAIT), Dehradun, India, 5–7 June 2019; pp. 25–30.
49. Karau, H.; Konwinski, A.; Wendell, P.; Zaharia, M. *Learning Spark: Lightning-Fast Big Data Analysis*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2015.
50. Meng, X.; Bradley, J.; Yavuz, B.; Sparks, E.; Venkataraman, S.; Liu, D.; Freeman, J.; Tsai, D.; Amde, M.; Owen, S.; et al. Mllib: Machine learning in apache spark. *J. Mach. Learn. Res.* **2016**, *17*, 1235–1241.
51. Apvrille, L.; El Khayari, R.; Henniger, O.; Roudier, Y.; Schweppe, H.; Seudié, H.; Weyl, B.; Wolf, M. Secure automotive on-board electronics network architecture. In Proceedings of the FISITA 2010 World Automotive Congress, Budapest, Hungary, 30 May–4 June 2010; Volume 8.
52. Studnia, I.; Alata, E.; Nicomette, V.; Kaâniche, M.; Laarouchi, Y. A language-based intrusion detection approach for automotive embedded networks. *Int. J. Embed. Syst.* **2018**, *10*, 1–12. [CrossRef]
53. Kang, H.; Kwak, B.; Lee, Y.H.; Lee, H.; Lee, H.; Kim, H.K. Car hacking: Attack and defense challenge 2020 dataset. *IEEE Dataport* **2021**. [CrossRef]
54. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
55. Song, Y.Y.; Ying, L. Decision tree methods: Applications for classification and prediction. *Shanghai Arch. Psychiatry* **2015**, *27*, 130. [PubMed]
56. Chen, T.; He, T.; Benesty, M.; Khotilovich, V.; Tang, Y.; Cho, H.; Chen, K.; Mitchell, R.; Cano, I.; Zhou, T.; et al. Xgboost: Extreme Gradient Boosting. R Package Version 0.4-2. 2015; Volume 1, pp. 1–4. Available online: https://cran.r-project.org/web/packages/xgboost/vignettes/xgboost.pdf (accessed on 25 February 2024).
57. Sokolova, M.; Japkowicz, N.; Szpakowicz, S. Beyond accuracy, F-score and ROC: A family of discriminant measures for performance evaluation. In Proceedings of the 19th Australasian Joint Conference on Artificial Intelligence, Hobart, Australia, 4–8 December 2006; pp. 1015–1021.
58. Refat, R.U.D.; Elkhail, A.A.; Hafeez, A.; Malik, H. Detecting can bus intrusion by applying machine learning method to graph based features. In Proceedings of the Intelligent Systems and Applications: Proceedings of the 2021 Intelligent Systems Conference (IntelliSys), Amsterdam, The Netherlands, 1–2 September 2022; Volume 3, pp. 730–748.