

Enhancing IoV Security: Machine Learning-Based Intrusion Detection for CAN Bus Systems

Ozan Duru
Department of Computer Engineering
Biruni University
Istanbul, Turkey
210408005@st.biruniedu.tr

I. INTRODUCTION

The IoV came with the complete transformation of conventional vehicles into an interacting, intelligent entity that would perform on runtime, with decisions and analytics. Extending the framework of the Internet of Things, IoV has helped develop autonomous driving, fleet management, and traffic optimization that have resulted in the emergence of safer and more efficient transportation networks. However, these innovations have also introduced significant cybersecurity challenges, particularly within the Controller Area Network (CAN) Bus, the primary protocol for vehicular communication [1].

The CAN Bus protocol has been instrumental in the organization of communications between different ECUs that govern critical vehicle functions, including everything from the engine to the braking system. While highly effective and in widespread use, CAN Bus does not provide important security features like encryption or authentication. This inadequacy makes the network prone to various cyber threats, such as DoS, which may overload the system with surplus messages, and Spoofing attacks, where an attacker impersonates ECUs to inject unauthorized commands. This can put into question the functionality and safety of the vehicle, making serious threats to passengers and operators.

Moreover, with the increased complexity of IoV systems, sharing data in real time among vehicles and with other external systems opens more attack surfaces. This causes harmful effects on confidentiality, integrity, and availability of critical data, leading to operational disruption and financial loss. Addressing these challenges requires robust Intrusion Detection Systems (IDS) capable of detecting and mitigating sophisticated attack patterns in dynamic vehicular environments [2].

Machine learning has emerged as a transformative tool in the development of IDS, advancing capabilities for network traffic analysis and identifying anomalous behaviors. Techniques such as Random Forest provide high accuracy in classifying threats, while ensemble methods enhance model robustness and scalability. However, the successful implementation of ML-based IDS relies heavily on effective preprocessing, feature selection, and realistic dataset representation to ensure that the models generalize effectively across diverse scenarios.

The KNIME Analytics Platform was used in this research for the preprocessing and analysis of CAN Bus datasets, hence ensuring that the data is optimized both for training and evaluation. It included filtering irrelevant features, imputing or removing missing values, and conversion into binary, decimal, and hexadecimal formats to give detailed insights into the underlying pattern. The IDS was then trained using the Tree Ensemble Learner in KNIME by combining multiple decision trees, which resulted in better threat detection for DoS and Spoofing attacks. The ensemble reduces overfitting and generalizes well, making it applicable in real-world IoV applications [3].

It hence presents a scalable, adaptive framework of IoV cybersecurity with advanced ML techniques, together with meticulous preprocessing workflows. The proposed system fills up some critical gaps in conventional IDS methods, such as their low accuracy and the inapplicability to real time, and sets grounds for further work on IoV networks security against emerging cyber threats [4].

Row ID	S Interfa...	S ID	S DLC	S DATA_0	S DATA_1	S DATA_2	S DATA_3	S DATA_4	S DATA_5	S DATA_6	S DATA_7	S label	S category	S speci...
Row0	slcan0	041	[8]	60	00	00	00	00	00	00	00	BENIGN	BENIGN	BENIGN
Row1	slcan0	42C	[8]	84	0D	A0	00	00	00	00	00	BENIGN	BENIGN	BENIGN
Row2	slcan0	217	[8]	7F	FF	7F	FF	7F	FF	7F	FF	BENIGN	BENIGN	BENIGN
Row3	slcan0	083	[8]	0F	E0	00	00	00	00	00	00	BENIGN	BENIGN	BENIGN
Row4	slcan0	3A8	[8]	01	00	27	10	00	00	00	00	BENIGN	BENIGN	BENIGN
Row5	slcan0	167	[8]	00	80	00	00	00	01	E3	00	BENIGN	BENIGN	BENIGN

Figure 1

Row ID	I ID	I DATA_0	I DATA_1	I DATA_2	I DATA_3	I DATA_4	I DATA_5	I DATA_6	I DATA_7	S label	S category	S speci...
Row0	291	0	0	0	0	0	0	0	0	ATTACK	DoS	DoS
Row1	291	14	11	4	4	3	3	8	12	ATTACK	DoS	DoS
Row2	291	14	11	4	4	3	3	8	12	ATTACK	DoS	DoS
Row3	291	14	11	4	4	3	3	8	12	ATTACK	DoS	DoS
Row4	291	14	11	4	4	3	3	8	12	ATTACK	DoS	DoS
Row5	291	14	11	4	4	3	3	8	12	ATTACK	DoS	DoS
Row6	291	14	11	4	4	3	3	8	12	ATTACK	DoS	DoS

Figure 2

Row ID	I ID3	I ID4	I ID5	I ID6	I ID7	I ID8	I ID9	I ID10	I ID11	I ID12	I ID13	I ID14	I ID15	I ID16	I DATA...	I DATA...	I DATA...
Row0	0	0	0	0	0	1	0	0	1	0	0	0	1	1	0	0	0
Row1	0	0	0	0	0	1	0	0	1	0	0	0	1	1	0	0	0
Row2	0	0	0	0	0	1	0	0	1	0	0	0	1	1	0	0	0
Row3	0	0	0	0	0	1	0	0	1	0	0	0	1	1	0	0	0

Figure 3

II. DATA PREPROCESSING

A. Data Conversion

In Controller Area Network, data is transferred in hexadecimal form “Fig. 1” because of compact and efficient usage. On the other hand, while such data is prepared for a machine learning task, totally depending on the features obtained with hexadecimal forms may result in poor model performances, since inherently complex and variable data cannot be reflected to their corresponding hexadecimal forms. It further enriches the features in order to meet such a challenge by changing the data into a decimal format “Fig. 2”, and it also converts it into a binary format “Fig.3” for training machine learning models.

This is followed by converting the hexadecimal into binary form. This occurs step by step, with each hexadecimal value being converted to its binary form. The binary is further padded to 8 bits for coherence and consistency. When the binary strings have been obtained, each bit extracted shall go ahead and be stored separately as features within the dataset. Through this in-depth depiction, minute pieces of information can be deducted by the machine learning model from the data, hence promoting pattern and anomaly detection.

What follows is the pseudo-code for the steps of conversion and feature extraction; this is just for the clear explanation of the logic.

1) Pseudo-code for Hexadecimal to Binary Conversion and Bit Extraction

Loading Data: Importing Libraries. Reading hexadecimal values from a CSV file containing the dataset to a DataFrame.

Identify Hexadecimal Columns: There can be columns containing hexadecimal values in a dataset.

Process Each Hexadecimal Column: Use appropriate functions to convert values into a binary string for each column in which values are in hexadecimal. This will ensure that it's 8-bit even though the first character shows up as a number lower than 8 in binary. Use bitwise operations to pull out individual bits from the binary string. In this portion, take bits pulled out and store them in separate columns for each of the 8-bit binary string bits.

Save the Results: Combine the preprocessed binary and bit-level data into the original dataset. Finally, it should be written into a new CSV file that could be used for further processing.

On doing so, the dataset will completely grow by changing the hexadecimal data into binary and decimal format, which may allow the strong establishment of machine learning models. Hence, this preprocessing step is basically essential to enhance the efficiency and reliability of intrusion detection systems in general in CAN networks.

B. Data Selection and Handling in Preprocessing

First, during the preprocessing phase, the dataset was selected based on three main attributes: label, category, and specific_class. These three features were very helpful in preprocessing for efficient machine learning applications, particularly for classifying benign and attack instances and for identifying the type of attack.

This 'Label' attribute at the beginning identified an attack or benign data. It is a binary classification in both datasets: decimal and binary. The Column Filter node in KNIME was used to extract the label

attribute by excluding unnecessary columns (according to purpose of classification) from the dataset. As the pre-processing step was focused on the said attribute, this pre-processing ensured that only relevant data would be included in the initial classification.

The 'Category' attribute was used for the identification of the attack type, such as Dos and spoofing. It allowed going further in sub-classification under the attack class, providing the possibility to distinguish larger attack classes. Another example of this fine-grained classification—for instance, spoofing attacks such as Spoofing GAS, Spoofing SPEED, Spoofing STEERING WHEEL, and Spoofing RPM—used the 'Specific_Class' attribute. Below, the application of the Column Filter node again filters all but the 'Specific_Class' attribute for drilling deeper into the type of attack.

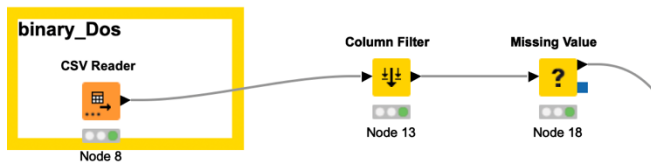


Figure 4

C. Function of the Column Filter Node

Improvements in the preprocessing workflow were significantly enhanced in KNIME with the Column Filter node “Fig.4”, a node that allows users to choose columns or remove them based on relevance to the analysis.

Core functionalities of Column Filter node

include: The process involved either the inclusion of only those columns deemed essential or the exclusion of columns considered extraneous from the dataset.

Data Type Filtering: Enable filtering of columns based on their data type, which may be numerical or categorical, depending on the attribute.

Filtering: It allows filtering based on column name or on specific pattern in column names for flexible and precise selection of data. This improved the preprocessing in the Column Filter node, as it focused on only those essential attributes

required, therefore reducing computation. Both the selection and handling of data attributes have been done with due diligence, providing a very robust grounding for machine learning later.

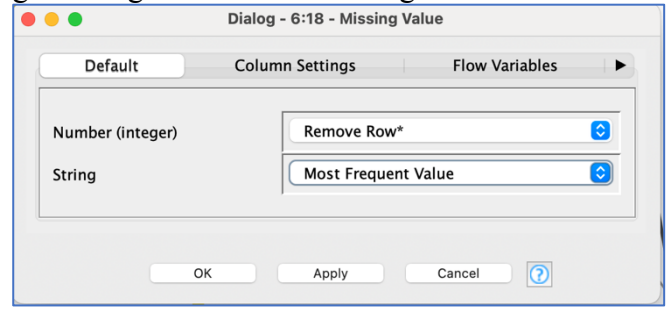


Figure 5

D. Handling Missing Values

The settings of the Missing Value node “Fig.5” in KNIME will be set to handle missing values appropriately. Depending on the data type, different strategies were used:

Numerical values in this case required the Remove Row configuration “Fig.5”. This option removes the complete row containing the missing numerical value, which would also protect the dataset by not including incomplete data that might somehow affect the overall performance of the model.

For categorical variables, the missing value was replaced using the most frequently occurring value of that column. It would make sure that the dataset is being kept consistent and will provide minimal damage to the performance of the running model.

These preprocessing choices were made to reduce the adverse effects of missing data, optimize data quality, and enhance model performance during training.

III. EXPLORATORY DATA ANALYSIS (EDA)

The dataset used in this study contains multiple features that are crucial for analyzing and understanding the behavior of the CAN bus traffic. Each feature has a specific role in defining the characteristics of the data, which ultimately contributes to the detection of benign or malicious activities. Below is a detailed explanation of the key features and their importance:

A. Key Features

ID: arbitration field for CAN bus communication. It determines the priority of the message as the

message with smaller values of ID will have higher priority. Most importantly, it ensures there can never be any delay in transmitting critical messages; this essentially ensures the functionality and safety of the system.

DATA_0 to DATA_7:

These would comprise the very bytes of information transmitted across the CAN bus. Usually, data is from values of sensors or command signals that make up most of the critical on-board operating systems. In detecting anomalies that might insinuate some kind of malicious behavior, it's important to understand these bytes.

B. Designation, Classification, and Particular_Category:

Label: Specifies whether the traffic is benign - benign, or an attack. It provides the ground for a machine learning model to differentiate between normal and anomalous behavior.

Category: The category of the attack, such as DoS and Spoofing, which will allow for more effective targeting of the malicious behavior.

Specific_Class: Very informative on the target of the specific attack, such as RPM, Speed, Steering Wheel, and Gas, which is useful for more detailed threat characterization.

C. Data Representation

This is presented in two different formats to realize the optimal utility, which offers certain benefits in analytics use and machine learning:

Decimal Representation:

The data is in bytes and can be used for the calculation of summary statistics: mean, standard deviation, and percentiles.

This overview depicts higher abstraction data patterns, hence allowing the perception of general trends within the dataset.

Binary Representation:

The data represents it in bits, allowing finer detail for any given characteristic.

This broad view shows the regularity with which each bit's appearance as a 0 or 1 makes such subtle anomalies easier to spot than in a decimal representation.

D. Importance of Data Presentation

Both the binary and decimal views of the data can show that this is important in making the machine learning models effective. Though it gives a wider view in the decimal format, the binary format delivers minute details on the architecture. This synergy allows:

Simplified threat detection: In the data, patterns start to become more distinctive by which the models identify malicious behavior. **Improved Model Performance:** Having clean and structured features will undoubtedly increase the accuracy and reliability of machine learning algorithms. **Data analysis with propriety:** processing at a micro or macro level analyzes the data and researches it to the core. With such a representation, this dataset will really let the power of machine learning models accurately distinguish between normal and malicious traffic to pave the way for developing robust security methodologies in in-vehicle networks.

E. Data Integration and Classification Process

This part of study will pre-process and integrate data into different categories on which machine learning models are to be applied. The CSV dataset was imported into KNIME via the CSV Reader node, following which several preprocessing steps had to be performed in order to get the data ready for analysis. Once the preprocessing was done, the Concatenate Node "Fig.6" was used to merge the datasets, aligning them for the classification wanted. Then, features were filtered with the Column Filter node to focus on some attributes relevant for the analysis.

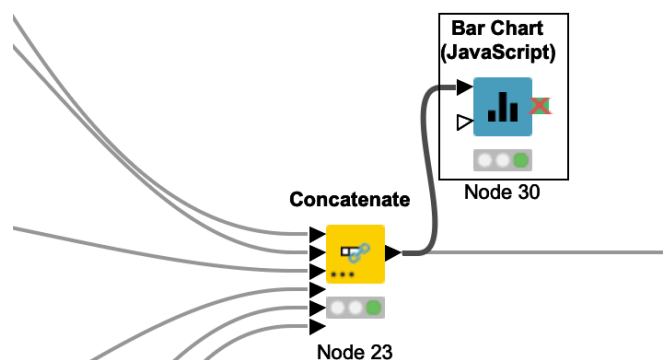


Figure 6

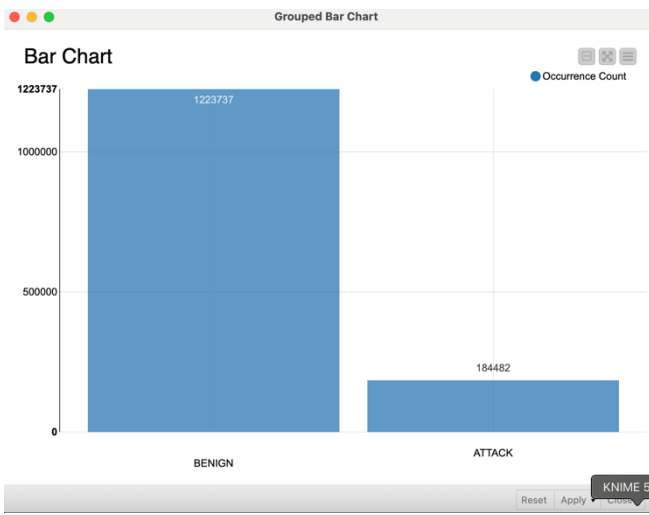


Figure 7

First of all, the dataset was classified into Attack and Benign classes by using the ‘Label’ attribute. Then, a Bar Chart Node “Fig.6” was created to show the numerical distribution “Fig.7” of these two classes. It proved that the number of benign data was much larger than the attack data, which simply justifies the realistic conditions [5] where benign traffic usually dominates. The inclusion of a dataset with more benign samples helped in developing the model closer to the real world. Unlike existing work which was studied on only specific classes [5], which did not consider this differentiation between benign and attack data, the present research does. Therefore, it allows a more realistic basis on which to train the model. While previous approaches were concerned with the classification of specific attack types, this work focuses on the more general categorization.

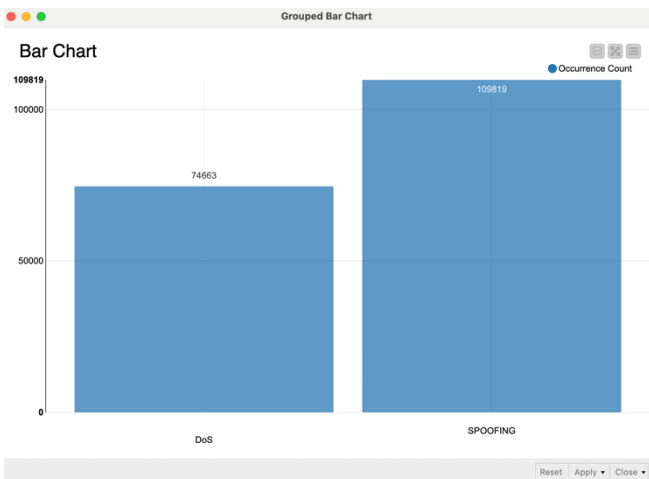


Figure 8

The integration was based on categories using the Concatenate node after the classification between attack and benign. In this step, further

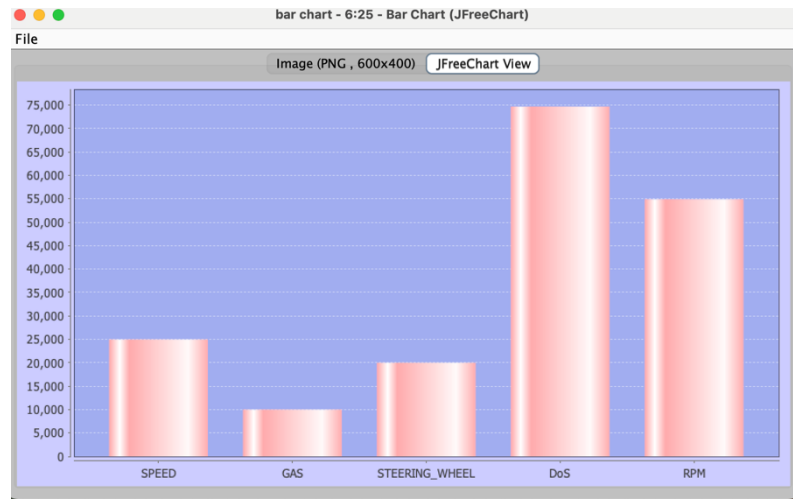


Figure 9

categorization was to be performed between DoS and Spoofing. Numerical distribution “Fig.8” for the two categories of attack shown here gives a better picture of their proportion in this dataset.

The specific classification was done in the end by again joining the dataset using the Concatenate Node, this time on the specific_class attribute. This allowed determining specific attack types such as Spoofing GAS, Spoofing SPEED, Spoofing RPM, Spoofing STEERING WHEEL, and DoS. These specific classifications allowed the analysis “Fig.9” to focus on the exact type of attack; therefore, the accuracy of the model in detecting each particular threat would increase.

All the analysis here was done using the binary dataset, but one should realize that all the same operations on the decimal dataset would produce identical numerical results since binary and decimal are direct transforms of each other.

Thus, such a structured approach-classification by label, category, and specific_class-provides a complete framework for making sense of the various types of threats to understand and identify those threats. The incorporation of higher and fine-grained classifications gives robustness to the analysis compared with earlier research that studied individual attack types.

F. Correlation Analysis with Linear Correlation Node

Relationships among numeric features in the dataset were analyzed using the Linear

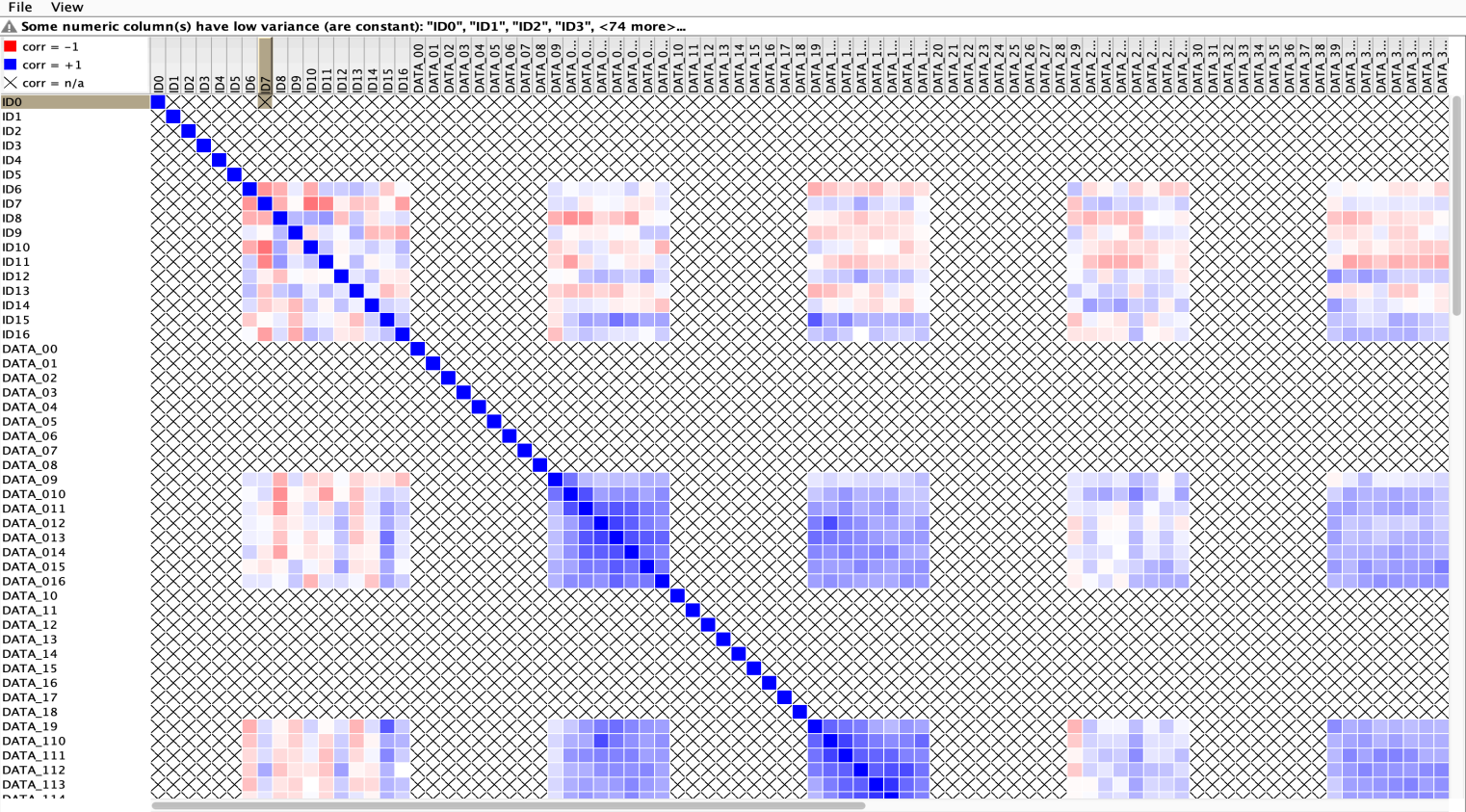


Figure 10

Correlation Node “Fig.10” in KNIME. This step was important for the identification of redundancies and how the features interact. The node calculated the correlation coefficients, highlighting highly correlated features that helped reduce unnecessary attributes and thus improved model efficiency. Besides, it identified those

strongly correlated with the target variable, hence allowing feature selection to be focused and effective. This analysis further improved the preprocessing pipeline, hence providing a better dataset for improved machine learning.

Interactive View: Statistics View													
Statistics													
Rows: 154 Columns: 14													
Name	Type	# Missin... ↓	# Unique val...	Minimum	Maximum	25% Quantile	50% Quantile...	75% Quantile	Mean	Mean Absolu...	Standard Dev...	Sum	10 most com...
ID0	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
ID1	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
ID2	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
ID3	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
ID4	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
ID5	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
ID6	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
ID7	Number (intege 0		2	0	1	0	0	0	0.189	0.307	0.392	34,943	0 (149539; 81.0
ID8	Number (intege 0		2	0	1	0	1	1	0.702	0.418	0.457	129,562	1 (129562; 70.2
ID9	Number (intege 0		2	0	1	0	0	1	0.298	0.418	0.457	54,914	0 (129568; 70.2
ID10	Number (intege 0		2	0	1	0	0	1	0.298	0.418	0.457	54,899	0 (129583; 70.2
ID11	Number (intege 0		2	0	1	0	0	1	0.405	0.482	0.491	74,663	0 (109819; 59.5
ID12	Number (intege 0		2	0	1	0	0	1	0.298	0.418	0.457	54,899	0 (129583; 70.2
ID13	Number (intege 0		2	0	1	0	0	1	0.298	0.418	0.457	54,899	0 (129583; 70.2
ID14	Number (intege 0		2	0	1	0	0	0	0.189	0.307	0.392	34,937	0 (149545; 81.0
ID15	Number (intege 0		2	0	1	0	0	1	0.405	0.482	0.491	74,663	0 (109819; 59.5
ID16	Number (intege 0		2	0	1	0	1	1	0.594	0.482	0.491	109,606	1 (109606; 59.4
DATA_00	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
DATA_01	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
DATA_02	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
DATA_03	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
DATA_04	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
DATA_05	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
DATA_06	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
DATA_07	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
DATA_08	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
DATA_09	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
DATA_10	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
DATA_11	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
DATA_12	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
DATA_13	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
DATA_14	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
DATA_15	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
DATA_16	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
DATA_17	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
DATA_18	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
DATA_19	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
DATA_110	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
DATA_111	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
DATA_112	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
DATA_113	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.
DATA_114	Number (intege 0		1	0	0	0	0	0	0	0	0	0	0 (184482; 100.

Figure 11

The correlation coefficient ranges between -1 and +1, with the following interpretations:

- **+1**: Strong positive relationship.
- **-1**: Strong negative relationship.
- **0**: No linear relationship.

The Statistics Node in KNIME “Fig.11” is one of the most useful nodes in summarizing data while doing exploratory analysis. It calculates key metrics: mean, median, standard deviation, and range for numerical columns, and frequency counts for categorical columns. These metrics help in identifying the distribution of data, detection of outliers, validation of integrity of data, and feature selection.

The node provides a rapid view of the dataset, and therefore allows one to identify potential issues early on to make better preprocessing decisions. Its insights improve the performance of models by preparing a clean and well-structured dataset, and it is an essential step in any analysis workflow.

IV. DATA MINING TECHNIQUE SECTION AND APPLICATION

A. Applying Data Mining in Cybersecurity: Random Forest for Classification

In this study, Random Forest was chosen as the data mining technique that performs classification on the dataset. Classification is one of the most fundamental data mining approaches to grouping data points into predefined classes, and Random Forest proves to be an efficient classifier for high-dimensional and intricate datasets like those developed in cybersecurity. Its ensemble-based methodology ensures robust and accurate predictions; hence, it becomes an ideal choice in the detection and classification of cyber threats in Controller Area Network (CAN) Bus systems.

a) Justification of Choice of Random Forest High Accuracy and Robustness

Random Forest has been known to show a high predictive accuracy because it follows the ensemble learning framework; it trains many decision trees and then aggregates their predictions to produce the final output. This

approach will reduce the risk of overfitting, which is very common in the datasets of cybersecurity that have usually noisy or imbalanced data [6] .

Handling High-Dimensional and Complex

Data: The typical dataset in cybersecurity has many features, including hexadecimal, binary, and decimal attributes, which can hardly be handled by the traditional algorithms. Random Forest performs well with high-dimensional data and identifies complex patterns, which makes it quite apt for intrusion detection and threat classification tasks [7] .

Feature Importance Analysis: One of the interesting features of Random Forest is the possibility to get scores for feature importance. This functionality becomes very important in cybersecurity to identify the most important features for an attack, which may be some packet headers or even data payloads. The insight from that not only enhances model interpretability but also helps refine the cybersecurity policies [8] .

b) Relevance to Cybersecurity

The RandomForest algorithm, because of its advantages, has been applied with excellent performance to several important applications in cybersecurity, especially IDS, malware classification, and network anomaly detection. For example:

Intrusion Detection: It has been proved that Random Forest could achieve high accuracy in intrusion detection, such as detecting Denial-of-Service (DoS) and Spoofing attacks by learning patterns from labeled data [9] .

Network Traffic Analysis: It possesses a capability for large-scale processing of network traffic logs, thus it can be applied for identifying threats in real-time environments considering modern IoT and IoV system requirements [10] .

Adaptive Learning: The scalability of the algorithm itself makes it adaptive to develop with emerging threats, which is a critical requirement in cybersecurity.

c) Desired Outcomes

The application of the Random Forest, in this study, shall bring forth the following outcomes:

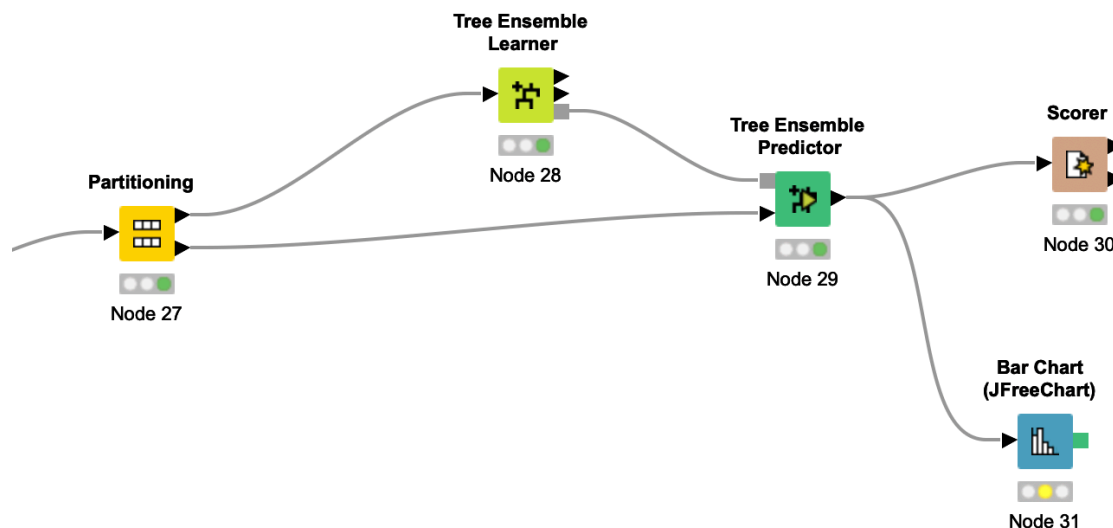


Figure 11

Accurate Classification: It will classify the benign from malicious accurately, which is targeted to detect the exact attack classes such as DoS and Spoofing. Scalable and Adaptive

Performance: Through its ensemble learning method, Random Forest should keep high accuracy and scalability on various data distributions. Better Cybersecurity Insights: Feature importance analysis will reveal top attributes contributing to attacks, leading to improved and more focused cybersecurity measures. By overcoming the obstacles brought about by high-dimensionality, noise, and class imbalance, Random Forest proves to be a strong advocate for the detection and mitigation of cyber threats in both IoV and CAN Bus ecosystems. This research complies with previous studies in light of the importance that these works have placed on ensemble learning for advancing cybersecurity [11] [12] .

Extensive data preprocessing and integration were performed for the current research to classify benign and attack samples efficiently and identify certain attack types. In this respect, the dataset was provided in separate CSV files for each unique class, which needed to combine for proper classification. After the application of different preprocessing methods using nodes like Column Filter and Missing Value, the datasets were combined using the Concatenate Node in KNIME. Accordingly, the Partitioning node was used to divide the dataset into separate training and testing data sets, while the settings are changed depending on the particular classification scenario.

2) Data Integration and Classification Scenarios Benign vs. Attack Classification:

In the initial binary classification, that is, between benign and attack, Label had been considered as the target variable. Since the number of benign instances were quite high in this dataset, splitting for training and testing had been done in the ratio of 70:30 to ensure enough data is available for training with no compromise on computation efficiency.

This reflects real life, since more abundant data does not contain threats; hence, this increases the generalization capability on real-life conditions. DoS vs. Spoofing Classification:

The category attribute has then been used for categorizing these attack types into DoS and Spoofing. The ratio here is 80:20 and was considered appropriate for the training and test samples, as less volume of data is involved as compared to the benign vs. attack classification.

3) Specific Attack Classification:

Bearing in mind that one class attribute refers to the type of attack, namely, Spoofing GAS, Spoofing RPM, Spoofing SPEED, Spoofing STEERING WHEEL, and DoS, the same ratio of 80:20 was used since the total number of instances in the dataset is not very high.

Tree Ensemble Learner and Predictor in KNIME
The Tree Ensemble Learner and Tree Ensemble Predictor nodes “Fig.11” were used for classification. In this case, these nodes represent multipurpose instruments that extend the Random

Forest algorithmic concept to a much more flexible, efficient, and interpretable one. Unlike the classic Random Forest nodes, it provides the possibility to use other ensemble methods such as Gradient Boosted Trees with equally high performance across a wide range of datasets.

Primary advantages of the Tree Ensemble Node: Hyperparameter Optimization: They provide extensive control over parameters, including but not limited to the number of trees, tree depth, and node sizes.

Feature Importance Analysis: Importance of features can be judged by the nodes, which will further enhance interpretability and refinement of the dataset.

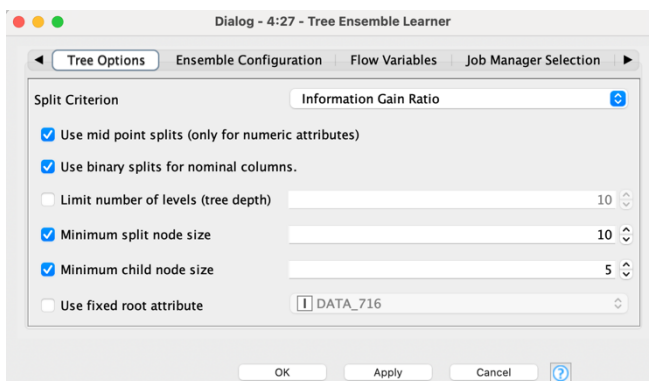
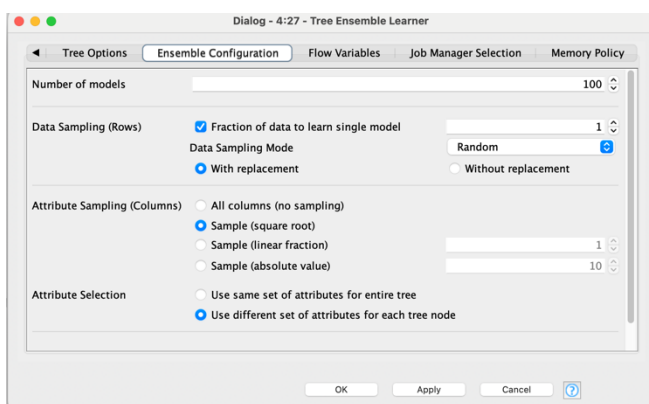


Figure 12

Scalability in handling huge sets of data with high dimensionality that permits their application in cybersecurity.

Configuration for the Benign vs. Attack Classification

Due to the size and complexity of this data, data-specific hyperparameters were tuned for the Tree Ensemble learner in order to optimize both model performance and computational cost. These configurations “Fig.12” include:

Number of Trees:

Set to 100: for the ensemble to capture diverse patterns and to avoid overfitting, this is the best number where the accuracy and computation efficiency balances.

Tree Depth:

Only 10 to keep the individual trees fairly simple, hence keeping the interpretability of the model and not wasting computational overhead.

Minimum Child Node Size:

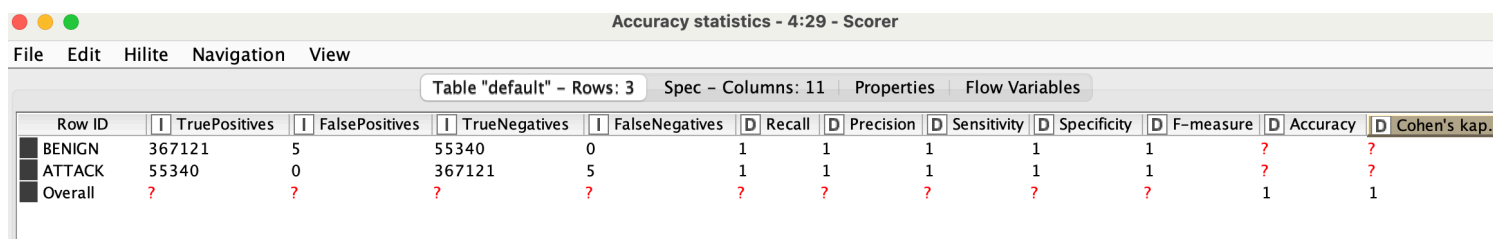
Set to 5, which allows up to at least five samples in each leaf node to avoid over-splittings to improve generalization.

Minimum Split Node Size:

Set to 10, to ensure that nodes had enough samples before splitting further. In this way, splitting based on noise or outliers was minimized.

These are important configurations, crucial indeed, to handle such large datasets for the benign vs attack classification. Without these use of memory and finally resulted in training failure for an example, Java Heap Space error.

Overcoming Computational Challenges Being a large dataset, the classification of benign versus attack instances has been particularly challenging in terms of memory usage in the learning process. It is at this point that optimization of hyperparameters and reduction of dimensions of the dataset-for example, reducing benign samples by half-finally yield high accuracy with computationally feasible outcomes. Remarkably, even with reduced benign samples, the model yielded consistent high results, truly speaking to the efficiency of the Tree Ensemble Learner configuration. Therefore, several classification tasks, such as identifying certain types of attacks, needed less processing since smaller dataset sizes would make the default settings enough. Scoring Using Score Node “Fig.11” The results of all classifications were analyzed using the Score Node, which also provided accuracy, precision, and recall metrics. This assured the model performance assessment comprehensively in all contexts. Importance of Configurations Therefore, these configurations were optimum, achieving a



Row ID	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Sensitivity	Specificity	F-measure	Accuracy	Cohen's kappa
BENIGN	367121	5	55340	0	1	1	1	1	1	?	?
ATTACK	55340	0	367121	5	1	1	1	1	1	?	?
Overall	?	?	?	?	?	?	?	?	?	1	1

Figure 14

perfect balance between accuracy and computation cost. With the parameters optimized concerning the nature of the data and the goal of classification, the research achieved highly performing models that could differentiate between normal instances and attack samples and also identify various kinds of attack categories. The above configurations outline the importance of parameter optimization in ensemble learning algorithms, especially on wide and complex datasets related to cybersecurity.

B. Classification Performance Analysis: Binary Dataset

The evaluation of the classification models was conducted using a series of performance metrics, including accuracy, precision, recall, and F1-score. These metrics provide a comprehensive assessment of how well the models perform under different scenarios, such as distinguishing between benign and attack samples, classifying attack types (DoS vs. Spoofing), and identifying specific attack categories (e.g., Spoofing GAS, Spoofing SPEED). Below are the results and corresponding interpretations for each classification task.

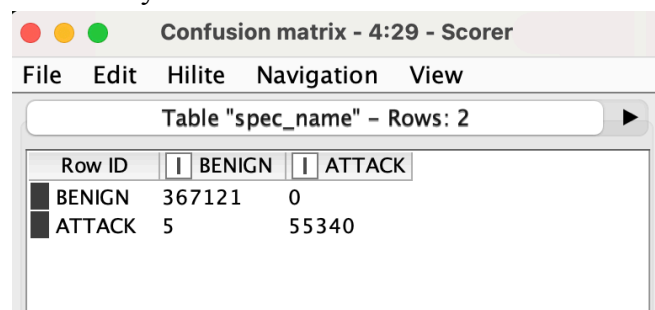
1) Benign vs. Attack Classification (Binary Classification)

Confusion Matrix“Fig.13” :

True Positives (TP): 55,340 attacks correctly classified.

True Negatives (TN): 367,121 benign instances correctly classified.

False Positives (FP): 5 benign instances incorrectly classified as attacks.



Row ID	BENIGN	ATTACK
BENIGN	367121	0
ATTACK	5	55340

Figure 13

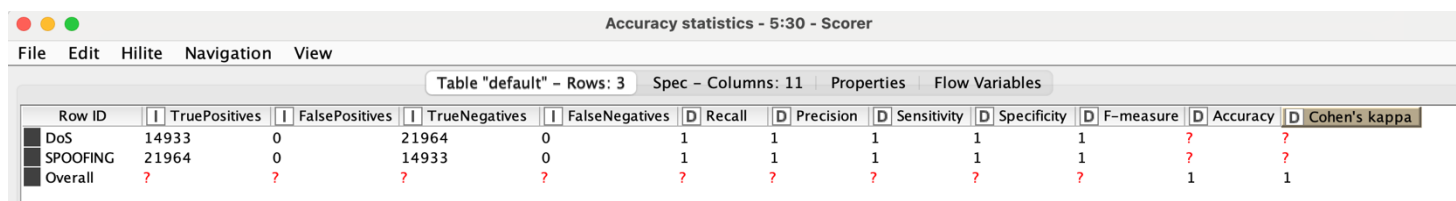
False Negatives (FN): 0 attacks misclassified as benign.

Accuracy Statistics“Fig.14” :

The model achieved **100% recall**, **100% precision**, and **100% F1-score**, indicating exceptional performance. However, the small number of false positives (FP = 5) suggests an imbalance, where benign samples dominate the dataset.

Issue with "?":

The overall accuracy is 100%, but metrics such as **Cohen's Kappa** and some derived statistics appear as question marks ("?"). This is likely due to the overwhelming imbalance between benign and attack classes, causing the evaluation algorithm to



Row ID	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Sensitivity	Specificity	F-measure	Accuracy	Cohen's kappa
DoS	14933	0	21964	0	1	1	1	1	1	?	?
SPOOFING	21964	0	14933	0	1	1	1	1	1	?	?
Overall	?	?	?	?	?	?	?	?	?	1	1

Figure 16

Row ID	DoS	SPOOF...
DoS	14933	0
SPOOFING	0	21964

Figure 15

skip certain calculations (e.g., lack of sufficient variability in the dataset).

Actionable Suggestion:

Consider resampling the dataset to balance benign and attack samples for better generalization.

2) DoS vs. Spoofing Classification

Confusion Matrix“Fig.15” :

True Positives: 14,933 DoS attacks and 21,964 Spoofing attacks correctly classified.

False Positives and Negatives: Both are zero, indicating perfect classification.

Accuracy Statistics“Fig.16” :

Recall, Precision, and F1-score: All metrics are perfect at **1.0** for both classes.

However, overall accuracy statistics like **Cohen's Kappa** are missing ("?"), indicating potential redundancy or lack of variability in the data.

Interpretation:

The perfect scores may reflect that the dataset is well-separated for this classification task. However, real-world scenarios may not provide such clear separations.

Actionable Suggestion:

Validate results with additional noise or variability in the dataset to confirm model robustness.

3) Specific Attack Classification (GAS, RPM, SPEED, STEERING, DoS)

Confusion Matri“Fig.17” :

The model successfully classified:

1,998 GAS attacks.

10,979 RPM attacks.

4,990 SPEED attacks.

3,995 STEERING attacks.

14,934 DoS attacks.

Minimal misclassifications are observed, with one false negative for GAS and one false positive for RPM.

Accuracy Statistics“Fig.18” :

Precision, Recall, Sensitivity, Specificity, and F1-scores are all **1.0**, except for GAS, which has a precision of **0.999** due to one misclassification.

The question marks ("??") in overall metrics arise because the model performed perfectly across most categories, causing statistical calculations (like Cohen's Kappa) to fail due to zero variability in predictions.

Interpretation:

This result demonstrates the high specificity of the model for detecting specific attack types. However, near-perfect results might indicate a dataset that is too clean or lacks noise, which could limit the model's generalizability.

Row ID	GAS	RPM	SPEED	STEER...	DoS
GAS	1998	0	0	0	0
RPM	1	10979	0	0	0
SPEED	0	0	4990	0	0
STEERING_...	0	0	0	3995	0
DoS	0	0	0	0	14934

Figure 17

Actionable Suggestion:

Add more variability or introduce a test set with adversarial noise to ensure robustness.

Accuracy statistics - 6:30 - Scorer

FileEditHiliteNavigationView

Table "default" - Rows: 6Spec - Columns: 11PropertiesFlow Variables

Row ID	I TruePositives	I FalsePositives	I TrueNegatives	I FalseNegatives	D Recall	D Precision	D Sensitivity	D Specificity	D F-measure	D Accuracy	D Cohen's kappa
GAS	1998	1	34898	0	1	0.999	1	1	1	?	?
RPM	10979	0	25917	1	1	1	1	1	1	?	?
SPEED	4990	0	31907	0	1	1	1	1	1	?	?
STEERING_...	3995	0	32902	0	1	1	1	1	1	?	?
DoS	14934	0	21963	0	1	1	1	1	1	?	?
Overall	?	?	?	?	?	?	?	?	?	1	1

Figure 18

4) Comparison of Binary and Decimal Dataset Results

Comparison of decimal and binary datasets results End

To compare the performance between binary and decimal datasets, the same classification task is carried out using the decimal dataset, but now only in terms of attack type. This model is trained and tested on the decimal dataset and the results are compared to those obtained with the binary dataset.

The results show that metrics of performance, such as accuracy, precision, recall, and F1-score, had almost the same value on both datasets. Proven, this demonstrates that either binary or decimal representation data is equally effective for the classification of attack type.

Similarly, in these other experiments—the specific attack classifications and benign versus attack classification—the training and testing results on both the decimal and binary datasets were almost identical to each other. Such consistency would thus imply that this change of representation of the data—from binary to decimal or vice versa—does not really seem to affect the model in the detection and classification task.

V. CONCLUSION

This work gives a general framework for the detection and classification of cyber threats in CAN Bus systems of the IoV environment. The proposed IDS, by using Random Forest, shows an outstanding performance on different classification scenarios such as benign vs. attack, DoS vs. Spoofing, and detailed attack types. This study will make use of the KNIME Analytics Platform for data pre-processing and analysis, thereby showing the importance of feature representation in optimizing machine learning models in either binary or decimal formats.

These results demonstrate that the binary and decimal datasets do indeed yield comparable results, further emphasizing the robustness of the preprocessing and classification pipeline. Furthermore, the detailed assessment of hyperparameter tuning provided assurance that the proposed IDS would also be scalable and efficient with large imbalanced datasets. This research not only tackles current IDS methodology limitations—namely, low accuracy and ineffectiveness in real time—but also provides actionable insight toward improving cybersecurity in IoV systems.

More effort can be conducted in this regard by integrating additional adversarial noise to test model robustness and investigating other ensemble learning methods to further increase the performance of classification. This paper paves the way toward developing advanced, scalable, and real-time IDS solutions that contribute to the security and reliability of IoV networks as a whole.

- [1] Fadhil, J. A., & Sarhan, Q. I. (2020, November). Internet of Vehicles (IoV): A survey of challenges and solutions. In *2020 21st International Arab Conference on Information Technology (ACIT)* (pp. 1-10). IEEE.
- [2] Alfardus, A., & Rawat, D. B. (2023, March). Evaluation of CAN Bus Security Vulnerabilities and Potential Solutions. In *2023 Sixth International Conference of Women in Data Science at Prince Sultan University (WiDS PSU)* (pp. 90-97). IEEE.
- [3] Danba, S., Bao, J., Han, G., Guleng, S., & Wu, C. (2022). Toward Collaborative Intelligence in IoV Systems: Recent Advances and OpenIssues. *Sensors*, 22(18),6995. <https://doi.org/10.3390/s22186995>
- [4] Bozdal, M., Samie, M., Aslam, S., & Jennions, I. (2020). Evaluation of CAN Bus Security Challenges. *Sensors*, 20(8), 2364. <https://doi.org/10.3390/s20082364>
- [5] Neto, E. C. P., Taslimasa, H., Dadkhah, S., Iqbal, S., Xiong, P., Rahman, T., & Ghorbani, A. A. (2024). CICIoV2024: Advancing realistic IDS approaches against DoS and spoofing attack in IoV CAN bus. *Internet of Things*, 26, 101209.
- [6] Breiman, L., "Random Forests," *Machine Learning*, vol. 45, pp. 5–32, 2001.
- [7] Neto, E.C.P. et al., "Intrusion Detection System for CAN-Bus in IoV using Random Forest," *Internet of Things*, vol. 26, pp. 101209, 2024.
- [8] Han, J., Kamber, M., "Data Mining: Concepts and Techniques," *Morgan Kaufmann*, 3rd ed., 2011.
- [9] Sommer, R., Paxson, V., "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," *IEEE Symposium on Security and Privacy*, pp. 305–316, 2010.
- [10] Wang, W. et al., "Deep learning-based intrusion detection for IoT systems," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7401–7412, 2020.
- [11] Zhang, X. et al., "Random Forest-Based Detection of IoT Botnet Attacks," *IEEE Access*, vol. 8, pp. 3273–3285, 2020.
- [12] Liaw, A., Wiener, M., "Classification and Regression by randomForest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.