AI-Driven Intrusion Detection in IoV Communication: Insights from CICIoV2024 Dataset

Nourah Fahad Janbi
Department of Information Technology, College of Computing and Information Technology at Khulais,
University of Jeddah, Jeddah, Saudi Arabia

***Abstract*—The increasing interconnectivity of vehicular networks through the Internet of Vehicles (IoV) introduces significant security challenges, particularly for the Controller Area Network (CAN), a widely adopted protocol vulnerable to cyberattacks such as spoofing and Denial-of-Service (DoS). To address these challenges, this study explores the potential of Intrusion Detection Systems (IDSs) leveraging artificial intelligence (AI) techniques to detect and mitigate malicious activities in CAN communications. Using the CICIoV2024 dataset, which provides a realistic testbed of vehicular traffic under benign and malicious conditions, we evaluate 25 machine learning (ML) models across multiple metrics, including accuracy, balanced accuracy, F1-score, and computational efficiency. A systematic and repeatable approach was proposed to facilitate testing multiple models and classification scenarios, enabling a comprehensive exploration of the dataset's characteristics and providing insights into various ML algorithms' effectiveness. The findings highlight the strengths and limitations of various algorithms, with ensemble-based and tree-based models demonstrating superior performance in handling imbalanced data and achieving high generalization. This study provides insights into optimizing IDSs for vehicular networks and outlines recommendations for improving the robustness and applicability of security solutions in real-world IoV scenarios.**

***Keywords*—Intrusion Detection System; controller area network; Internet of Vehicles; CICIoV2024; machine learning; Artificial Intelligence; security**

I. Introduction

The Internet of Things (IoT) has revolutionized how devices interact, seamlessly connecting billions of smart devices across homes, industries, and cities [1], [2]. Recent advancements have focused on enhancing real-time data processing, energy efficiency, and scalability. Technologies such as edge computing, 5G/6G networks, and lightweight Machine Learning (ML) models have enabled IoT devices to process data locally, reducing latency, network congestion, and reliance on cloud-based systems [3], [4]. Artificial Intelligence (AI) plays a pivotal role in this transformation by enabling IoT devices to analyze vast amounts of data, derive actionable insights, and adapt to changing environments autonomously [5], [6].

In the domain of IoT security, AI can enhance threat detection, intrusion prevention, and secure authentication. Techniques such as anomaly detection, generative adversarial networks (GANs) for simulating cyber-attacks, and reinforcement learning for adaptive defense strategies enable IoT systems to identify and mitigate potential vulnerabilities proactively [7], [8]. By integrating AI, IoT ecosystems are becoming not only more efficient but also more resilient against evolving cybersecurity threats [9].

Similarly, vehicular networks and the Internet of Vehicles (IoV) leverage IoT to enhance traffic management and enable autonomous driving, but it also faces significant security threats due to its high interconnectivity and dependence on IoT components. Potential attacks exploit both inter-vehicle and intra-vehicle vulnerabilities (see Fig. 1). For instance, GPS spoofing attacks mislead vehicle navigation systems by transmitting false location data, potentially causing accidents [10], [11]. Replay attacks involve retransmitting valid network messages to disrupt real-time vehicle functionality, while Sybil attacks flood the network with fake vehicle nodes to manipulate traffic or force detours [10]. Additionally, Denial-of-Service (DoS) attacks can overwhelm the IoV network, leading to service outages that compromise vehicular operations. Attacks on Electronic Control Units (ECUs) and sensors, such as malware injection, jeopardize vehicle decision-making by tampering with critical system data [11]. One real-world example includes hackers tricking Tesla's Autopilot software into swerving into oncoming traffic lanes, demonstrating the tangible risks of compromised IoV security.

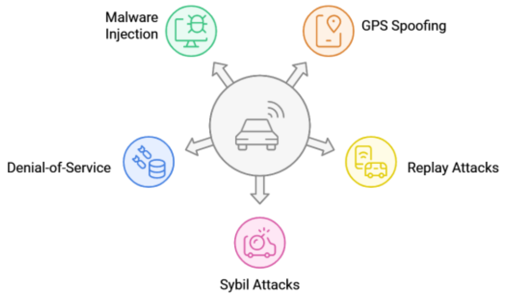


Fig. 1. Potential security threats in IoV.

Addressing these vulnerabilities requires advanced security measures, such as AI-driven intrusion detection systems (IDSs) and robust cryptographic protocols, to safeguard the integrity, availability, and confidentiality of IoV networks.

In this paper, we focus on the security of IoV and, specifically, the security of the Controller Area Network (CAN). The CAN is one of the commonly adopted communication protocols in vehicle and industrial systems for data exchange between ECUs without a central host computer. Despite its widespread use, the CAN protocol suffers from inherent security vulnerabilities, such as the lack of encryption and authentication, making it susceptible to spoofing, DoS, and replay attacks [12].

Traditional IDSs, such as signature-based and rule-based approaches, face significant challenges securing CAN networks. These conventional methods often suffer from high false positive rates, difficulty in adapting to novel attack patterns, and computational inefficiencies that limit their real-time applicability in resource-constrained vehicular environments. Furthermore, their reliance on predefined attack signatures makes them ineffective against zero-day attacks and evolving adversarial techniques [13]. On the other hand, AI-driven IDSs based on advanced ML and Deep Learning (DL) techniques can play a crucial role in enhancing cyberattack detection, prevention, and mitigation [14]. These ML and DL algorithms can effectively identify abnormal IoV traffic and request patterns, contributing to the early detection and mitigation of potential attacks.

As most existing research on CAN has primarily addressed these issues through theoretical solutions or simulated environments, Neto et al. [15] introduced the CICIoV2024 dataset to bridge the gap and provide a realistic testbed for IDSs focusing on CAN security. The dataset includes diverse attack types specific to CAN bus communication, such as DoS and spoofing (steering wheel, RPM, speed, gas). Since this dataset is considered recent, it requires comprehensive investigation.

Researchers in studies [15]–[17] conducted some comparative analyses on multiple ML algorithms using the CICIoV2024 dataset. However, the unrealistically high performance raises concerns about overfitting or dataset-specific optimizations, suggesting the need for comprehensive evaluation and broader testing on diverse datasets. This paper aims to bridge this gap by leveraging the CICIoV2024 dataset and evaluating a diverse range of ML models on this dataset, highlighting their strengths and limitations, and proposing recommendations for enhancing the robustness and applicability of IDSs in real-world IoV scenarios.

The main contributions can be outlined as follows:

- Comprehensively investigate the CICIoV2024 dataset and evaluate a diverse range of ML models on this dataset, highlighting their strengths and limitations.
- Propose a systematic and repeatable approach that facilitates testing multiple models and classification scenarios to comprehensively explore the dataset's characteristics and provide insights into the effectiveness of various ML algorithms on that dataset.
- Perform data cleaning step during preprocessing to ensure a more accurate representation of feature interactions, making the dataset more suitable for reliable ML analysis and reducing the risk of overfitting caused by repeated patterns.
- Provide insights into optimizing IDSs for vehicular networks and outline recommendations for improving security solutions' robustness and applicability in real-world IoV scenarios.

The rest of the paper is organized as follows: Section II reviews the related works. Section III explains our methodology. Section IV discusses the results of ML models and outlines recommendations. Section V concludes the paper.

II. Related Works

This section reviews recent studies that have employed ML and DL techniques for intrusion detection, highlighting their contributions and limitations. Table I provides a summary of related works.

TABLE I. RELATED WORKS SUMMARY

Ref.	Key Features	Limitations
Subasi et al. [18]	-Focus on interpretable and explainable ML -Use of Decision Trees and Ridge Classifiers -Introduction of cross-explanations	-Limited to feature-based explanations -Challenges with aleatoric uncertainties and feature correlations
Mahdi et al. [19]	-Hybrid approach combining LSTM and Naive Bayes -Three-stage methodology	-High complexity of hybrid model -May require significant computational resources
Aswal et al. [16]	-DL-based intrusion detection model for CAN -Focus on real-time detection	-Lacks comparative evaluation with hybrid methods
Neto et al. [15]	-Introduced the CICIoV2024 dataset -Emphasizes the importance of realistic CAN scenarios	-Dataset limited to specific attack scenarios -Immobile vehicle constraints
Amirudin et al. [17]	-Comparative analysis of ML algorithms like LightGBM, XGBoost, CatBoost	-Unrealistically high performance raises concerns about overfitting or dataset-specific optimizations
Tasci [20]	-Optimized CNN model for IoT security -Focus on lightweight architecture for real-time applications	-Limited exploration of diverse attack types -Scalability to larger datasets requires further validation

Subasi et al. [18] explored interpretable ML approaches for intrusion detection, focusing on enhancing model explainability using Decision Trees and Ridge Classifiers. By incorporating cross-explanation mechanisms and evaluating models with metrics like balanced accuracy and Matthews Correlation Coefficient, the study emphasized the need for interpretable systems in IDSs. However, challenges such as feature correlations and aleatoric uncertainties limit their applicability in more complex scenarios.

Building on this, Mahdi et al. [19] proposed a hybrid ML-DL framework, combining LSTM and Naive Bayes models for intrusion detection in IoT networks. This hybrid approach leverages the strengths of both DL's pattern recognition and ML's efficiency, achieving strong results on the CICIoV2024 dataset. However, its computational intensity highlights a trade-off between accuracy and feasibility for real-time applications, raising the importance of lightweight and scalable solutions.

Focusing on the IoV, Neto et al. [15] introduced the CICIoV2024 dataset that was designed for testing ML-based IDSs in IoVs. They evaluated Logistic Regression, Random

Forest, AdaBoost, and Deep Neural Network (DNN) model's ability to detect and classify malicious activities. Their findings highlight the challenges of addressing cybersecurity in IoV due to imbalanced datasets and similarities between benign and malicious traffic.

Aswal et al. [16] developed a DL-based IDS targeting vulnerabilities in the CAN protocol. Their model demonstrated effective real-time detection of various attacks using the CICIoV2024 dataset. Similarly, Amirudin et al. [17] conducted a comparative analysis of advanced ML algorithms, including LightGBM, XGBoost, and CatBoost, using the CICIoV2024 dataset. However, the unrealistically high performance raises concerns about overfitting or dataset-specific optimizations, suggesting the need for comprehensive evaluation and broader testing on diverse datasets.

Expanding beyond vehicular networks, Tasci [20] introduced an optimized convolutional neural network (CNN) for IoT security, achieving high performance on multiple datasets, including CIC-IoT2023, CIC-MalMem-2022, and CIC-IDS2017. This lightweight model demonstrated suitability for real-time applications, addressing computational limitations observed in hybrid approaches. However, its scalability to larger datasets and handling of diverse attack types requires further investigation.

Despite significant advancements in using ML and DL for intrusion detection in IoT and IoV, several critical research gaps remain. High-performing models often exhibit overfitting, as seen in studies achieving near-perfect accuracy, highlighting the need for comprehensive evaluation and robust evaluation across diverse datasets. Computational complexity is another challenge, with many hybrid and DL models being resource-intensive and unsuitable for real-time applications. Furthermore, since CICIoV2024 is a newly introduced dataset, there is a limited amount of research exploring its usability and potential applications. This creates an opportunity to evaluate its effectiveness in training and testing various ML and DL models for intrusion detection in vehicular networks. In addition, most research prioritizes accuracy and F1-scores over other metrics. In this study, we address these gaps by conducting a comprehensive evaluation of advanced ML and DL models on the CICIoV2024 dataset.

III. Methodology

In this section, we discuss the research methodology we followed in detail. We adopted a systematic and repeatable approach that facilitates testing multiple models and classification scenarios to comprehensively explore the dataset's characteristics and provide insights into the effectiveness of various ML algorithms on that dataset.

The flowchart in Fig. 2 provides a visual representation of the methodology followed in this paper for training and testing ML models on the CICIoV2024 dataset. The process starts with initializing the dataset and models, followed by removing duplicate entries to ensure data quality. The dataset is then split into training and testing subsets, guaranteeing a balanced evaluation of the models.

Each classification type (e.g., labels, categories, and specific classes) is processed iteratively, with labels converted to numeric values to make the dataset compatible with ML algorithms. For every classification type, the models are trained using the training dataset and evaluated on the testing dataset. The results of each model, for all classifications, are collected and stored. This process is repeated for all classifications and models to ensure comprehensive experimentation.

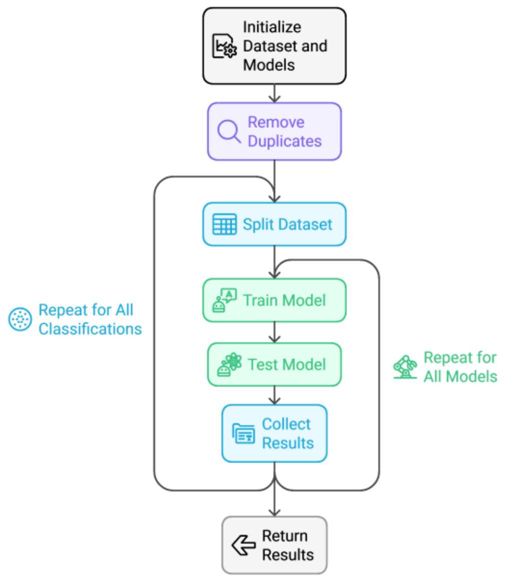


Fig. 2. Methodology flowchart.

The final step involves returning the results for analysis and comparison. This methodology ensures a systematic approach to testing multiple models and classification scenarios, providing insights into the effectiveness of various ML algorithms on the dataset. The combination of data preprocessing, iterative model training, and evaluation ensures a robust experimental setup.

Algorithm 1: Models Training and Testing

```
Input: CICIoV2024 Dataset
Output: Results // lists of results for all models
1  Function: evaluate_models(CICIoV2024)
2    Init: Models ← set of models,
           Classifications ← {label, category, class},
           Results ← empty set for results.
3    For dataset in Dataset
       //Dataset Preprocessing
4      dataset ← dataset.removeDuplicate()
5      For class_type in Classifications
         // Covert labels to numeric values
6        dataset ← dataset.numericValues()
         // Split dataset
7        x_train,y_train,x_test,y_test← dataset.split(test_size=0.3)
8        For model in Models
           // Train model
9          train_result= model.train(x_train, y_train)
           // Test model
10         test_result= model.test(x_test,y_test)
11         Results.add(train_result,test_result)
12       End For
13     End For
14   End For
15   Return Results
```

Algorithm 1 details the step-by-step implementation of the process used in the study. The input to the algorithm is the CICIoV2024 dataset, while the output is a set of results capturing the performance of all models across different classifications. The following subsections will discuss steps in detail.

A detailed breakdown of the features of the CICIoV2024 dataset is provided in Table II. Each instance in the dataset includes an ID field, which denotes the arbitration ID used to determine message priority on the CAN bus, and DATA_0 to DATA_7, which represents the eight-byte payload of CAN messages. Additionally, the dataset includes labels for classifying traffic as benign or malicious, with malicious traffic further categorized into DoS and Spoofing types. Spoofing attacks are further specified into classes such as Speed Spoofing, RPM Spoofing, Gas Spoofing, and Steering Wheel Spoofing. These features provide a granular view of vehicular communication, enabling detailed analysis and the application of ML techniques for intrusion detection.

TABLE II. CICIoV2024 DATASET FEATURES

Feature Name	Description
ID	ID indicating the message priority and type of data being transmitted
DATA_0 to DATA_7	Data fields (Byte 0 to Byte 7) contain the payload of CAN bus messages
Label	Classification of the traffic as benign or malicious
Category	Category of the traffic (DoS or Spoofing)
Specific Class	Specific malicious traffic class (Speed Spoofing, RPM Spoofing, or Gas Spoofing)

Table III summarizes the dataset composition, labeling traffic data instances into benign and malicious types. Benign traffic, representing normal vehicle operations, forms the bulk of the dataset with 1,223,737 instances. Malicious traffic, with 184,482 instances, is divided into two primary categories: DoS and Spoofing. DoS traffic consists of 74,663 instances while Spoofing traffic includes subcategories (specific classes) like Gas Spoofing (9,991 instances), Steering Wheel Spoofing (19,977 instances), Speed Spoofing (24,951 instances), and RPM Spoofing (54,900 instances). This distribution reflects the predominance of benign operations in real-world scenarios and highlights specific malicious activities.

TABLE III. CICIoV2024 DATASET SUMMARY

Traffic type	Category	Specific Class	Number of Instances	Total
Benign	-	-	1,223,737	1,223,737
Malicious	DoS	-	74,663	184,482
	Spoofing	Gas Spoofing	9,991	
		Steering Wheel	19,977	
		Speed Spoofing	24,951	
		RPM Spoofing	54,900	

A. *Machine Learning Models*

The CICIoV2024 dataset was used to train a diverse set of ML models (25 models) representing a variety of algorithm families and to evaluate its effectiveness comprehensively. Ensemble-based methods included AdaBoost Classifier (AdaBoostClassifier), Bagging Classifier (BaggingClassifier), and Random Forest Classifier (RandomForestClassifier), which combine multiple models to enhance prediction accuracy. Naive Bayes algorithms, such as Bernoulli Naive Bayes (BernoulliNB) and Gaussian Naive Bayes (GaussianNB), were utilized for probabilistic modeling. The Calibrated Classifier Cross-Validation (CalibratedClassifierCV) was employed as a probability calibration method to refine predictive probabilities. Decision tree-based models encompassed Decision Tree Classifier (DecisionTreeClassifier), Extra-tree Classifier (ExtraTreeClassifier), Extra-trees Classifier (ExtraTrees), Light Gradient Boosting Machine Classifier (LGBMClassifier), and Extreme Gradient Boosting Classifier (XGBClassifier), which are widely used for their interpretability and efficiency. Neighbors algorithms, including k-nearest Neighbors (KNeighborsClassifier) and Nearest Centroid Classifier (NearestCentroid), were applied for instance-based learning.

Linear models trained on the dataset included Logistic Regression Classifier (LogisticRegression), Passive Aggressive Classifier (PassiveAggressiveClassifier), Linear Perceptron Classifier (Perceptron), Ridge Classifier (RidgeClassifier), Ridge Classifier with Cross-Validation (RidgeClassifierCV), and Linear classifiers with Stochastic Gradient Descent (SGDClassifier), which are effective for high-dimensional data. Semi-supervised learning techniques, such as Label Propagation Classifier (LabelPropagation) and LabelSpreading Classifier (LabelSpreading), were also employed. Support Vector Machine algorithms, including C-Support Vector Classification (SVC) and Linear Support Vector Classification (LinearSVC), were used for their robustness in handling complex classification problems. In addition, Linear Discriminant Analysis model (LinearDiscriminantAnalysis) and Dummy Classifier (DummyClassifier) were trained. This extensive range of algorithms ensured a thorough exploration of the dataset's predictive potential.

B. *Duplicate Removal*

Data duplication removal from the dataset is one of the essential steps during data cleaning, ensuring that the data is accurate and reliable for further analysis or modeling [21]. In addition, a large volume of duplicated data might reduce data diversity and representativeness, leading to overfitting or biased models.

In our study, we used the drop_duplicates method from the Pandas library to remove duplicates in the CICIoV2024 dataset. Table IV and Fig. 3 show the distribution of data across different classifications after duplicate entries were removed, reducing the dataset size significantly from 1,408,219 instances to 3,588 instances. The distribution is displayed for three levels of classification: Label, Category, and Specific Class. At the Label level, the data is divided into benign and malicious, with a noticeable decrease in benign traffic proportion due to deduplication. At the Category level, malicious traffic is further subdivided into DoS and various spoofing types. Finally, at the Specific Class level, the spoofing category is broken down into detailed subcategories, including gas spoofing, steering wheel spoofing, and RPM spoofing, each with a smaller representation.

TABLE IV. DATASET DISTRIBUTION AFTER DEDUPLICATION

	Benign	**Malicious DoS**	**Malicious Spoofing Gas Spoofing**	**Malicious Spoofing Steering Wheel**	**Malicious Spoofing RPM Spoofing**
Label	3,547	41			
Category	3,547	21	20		
Specific Class	3,547	21	10	5	3

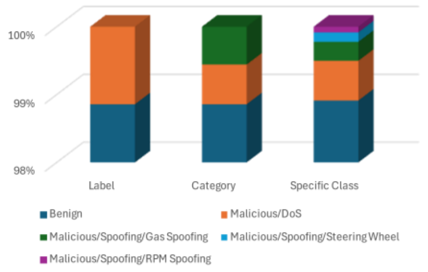


Fig. 3. Data distribution across label, category, and specific class classifications after deduplication.

After that, we compared the feature correlations in the CICIoV2024 dataset before and after duplicate removal, emphasizing the impact of preprocessing on data quality (see Fig. 4 and Fig. 5). Fig. 4, which represents the original dataset with duplicates, shows amplified correlations across several features, as evident from the brighter areas in the heatmap. These inflated relationships are likely caused by repeated data points, which can obscure unique interactions between features and introduce biases in ML models. In contrast, Fig. 5, generated after duplicate removal, exhibits more balanced and refined correlations, with reduced intensity in previously dominant relationships. This indicates a cleaner dataset where the true relationships among features are better preserved. The duplicate removal process not only eliminates redundancy but also ensures a more accurate representation of feature interactions, making the dataset more suitable for reliable ML analysis and reducing the risk of overfitting caused by repeated patterns.

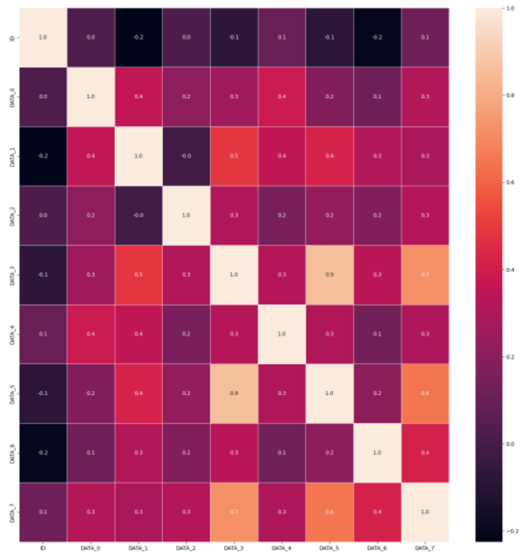


Fig. 4. Dataset heatmap of features correlation (Original dataset).

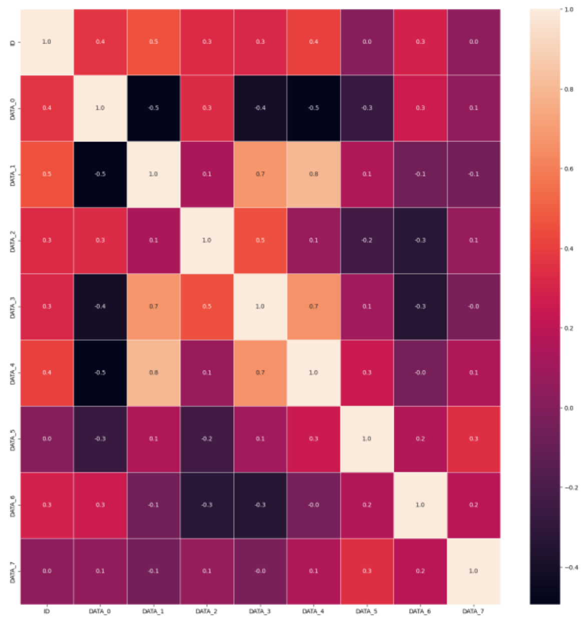


Fig. 5. Dataset heatmap of features correlation (Duplicate removed).

C. Dataset Splitting

Data splitting is a critical step to ensure robust evaluation of ML models. After preprocessing the dataset by removing duplicates and converting labels into numeric values, the dataset is split into training and testing subsets using the StratifiedShuffleSplit method, which combines the characteristics of ShuffleSplit (randomized splitting) and StratifiedKFold (maintaining the proportion of classes in each subset). This ensures that the training and testing sets have similar class distributions, preserving the balance of the data. The dataset is divided into a 70/30 ratio, where 70% is used for training the models to learn patterns and relationships, and 30% is reserved for testing, providing an unbiased evaluation of model performance. The final training set size was 2,511, and the training set size was 1,007 instances.

IV. RESULTS AND DISCUSSION

This section discusses the training and testing results of the ML models trained using the CICIoV2024 dataset with both Decimal (D) and Binary (B) formats. Performance metrics evaluated include accuracy, balanced accuracy, F1-score, and processing time. Balanced accuracy takes into account the accuracy of different classes separately and then calculates the mean. On the other hand, F-score keeps the balance between precision and recall. Both metrics help evaluate the sensitivity and specificity of the models, and they are particularly important when dealing with imbalanced data.

A. Accuracy

Results in Table V, Table VI, and bar charts in Fig. 6 compare the training and testing accuracy of various ML models across different classifications (Category, Label, and Class) in both decimal and binary formats. This comparison highlights their performance consistency and generalizability.

In the training accuracy results, most models, such as DecisionTreeClassifier, ExtraTreesClassifier, XGBClassifier, and LGBMClassifier, achieved near-perfect scores (1.00) across all classification levels (Category, Label, and Class in both Decimal and Binary formats), reflecting their ability to learn from the training data thoroughly. However, models like NearestCentroid and GaussianNB showed slightly lower training accuracies in specific scenarios.

TABLE V. TRAINING ACCURACY OF ALL MODELS

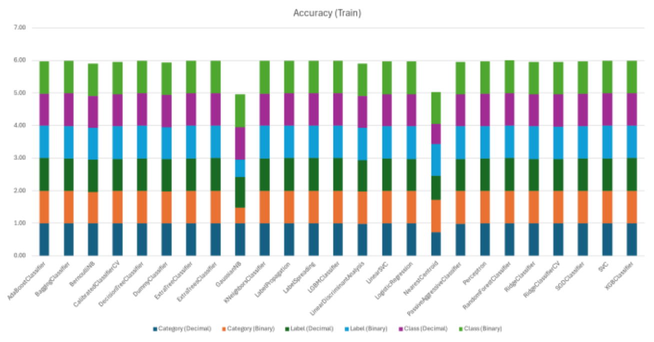
Model	Category (D)	Category (B)	Label (D)	Label (B)	Class (D)	Class (B)
AdaBoost	1.00	0.99	1.00	1.00	0.99	0.99
Bagging	1.00	1.00	1.00	1.00	1.00	1.00
Bernoulli NB	0.99	0.97	0.99	0.97	0.99	0.99
Calibrated CV	0.99	1.00	0.99	1.00	0.99	1.00
Decision Tree	1.00	1.00	1.00	1.00	1.00	1.00
Dummy Classifier	0.99	0.99	0.99	0.99	0.99	0.99
Extra Tree	1.00	1.00	1.00	1.00	1.00	1.00
Extra Trees	1.00	1.00	1.00	1.00	1.00	1.00
Gaussian NB	0.99	1.00	0.95	0.53	1.00	1.00
K Neighbors	1.00	1.00	1.00	1.00	0.99	1.00
Label Propagation	1.00	1.00	1.00	1.00	1.00	1.00
Label Spreading	1.00	1.00	1.00	1.00	1.00	1.00
LGBM Classifier	1.00	1.00	1.00	1.00	1.00	1.00
Linear Discriminant	0.97	1.00	0.97	1.00	0.97	1.00
Linear SVC	0.99	1.00	0.99	1.00	0.99	1.00
Logistic Regression	0.99	1.00	0.99	1.00	0.99	1.00
Nearest Centroid	0.72	0.99	0.74	0.99	0.62	0.97
Passive Aggressive	0.99	1.00	0.99	1.00	0.98	1.00
Perceptron	0.99	1.00	0.99	1.00	0.99	1.00
Random Forest	1.00	1.00	1.00	1.00	1.00	1.00
Ridge Classifier	0.99	1.00	0.99	1.00	0.99	1.00
Ridge CV	0.99	1.00	0.99	1.00	0.99	1.00
SGD Classifier	0.99	1.00	0.99	1.00	0.99	1.00
SVC	1.00	1.00	1.00	1.00	0.99	1.00
XGB Classifier	1.00	1.00	1.00	1.00	1.00	1.00

In contrast, the testing accuracy results showed minor variations, with some models slightly underperforming compared to their training accuracy. For instance, GaussianNB exhibited a noticeable drop in accuracy for Label (Binary) classification, indicating challenges in generalization. Similarly, NearestCentroid demonstrated lower accuracy across most classifications, reflecting its limitations with complex data structures. However, ensemble-based models, including AdaBoostClassifier, BaggingClassifier, and RandomForestClassifier, maintained consistently high accuracy in both training and testing, demonstrating their robustness and ability to generalize effectively.

TABLE VI. TESTING ACCURACY OF ALL MODELS

Model	Category (D)	Category (B)	Label (D)	Label (B)	Class (D)	Class (B)
AdaBoost	0.99	1.00	1.00	1.00	0.99	0.99
Bagging	1.00	0.99	1.00	1.00	1.00	0.99
Bernoulli NB	0.99	0.97	0.99	0.96	0.99	0.99
Calibrated CV	0.99	1.00	0.99	1.00	0.99	1.00
Decision Tree	1.00	0.99	0.99	1.00	0.99	0.99
Dummy Classifier	0.99	0.99	0.99	0.48	0.99	0.99
Extra Tree	1.00	0.99	1.00	1.00	0.99	0.99
Extra Trees	1.00	1.00	1.00	1.00	1.00	1.00
Gaussian NB	0.99	0.49	0.94	0.84	1.00	1.00
K Neighbors	0.99	1.00	1.00	1.00	1.00	0.99
Label Propagation	1.00	1.00	1.00	1.00	1.00	0.99
Label Spreading	1.00	1.00	1.00	1.00	1.00	0.99
LGBM Classifier	1.00	1.00	1.00	0.88	1.00	1.00
Linear Discriminant	0.97	1.00	0.96	0.84	0.96	0.99
Linear SVC	0.99	1.00	0.99	1.00	0.99	0.99
Logistic Regression	0.99	1.00	0.99	1.00	0.99	0.99
Nearest Centroid	0.71	0.99	0.73	0.92	0.62	0.97
Passive Aggressive	0.99	1.00	0.99	1.00	0.98	0.99
Perceptron	0.99	1.00	0.99	1.00	0.99	0.99
Random Forest	1.00	1.00	1.00	1.00	1.00	1.00
Ridge Classifier	0.99	1.00	0.99	1.00	0.99	1.00
Ridge CV	0.99	1.00	0.99	1.00	0.99	1.00
SGDClassifier	0.99	1.00	0.99	1.00	0.99	0.99
SVC	0.99	1.00	1.00	1.00	0.99	0.99
XGB Classifier	1.00	1.00	0.99	1.00	0.99	0.99

Overall, the comparison underscores the reliability of tree-based and ensemble models, which consistently perform well in both training and testing scenarios. It also highlights the importance of balanced model evaluation in identifying overfitting or generalization issues, as seen with certain algorithms like GaussianNB and NearestCentroid.



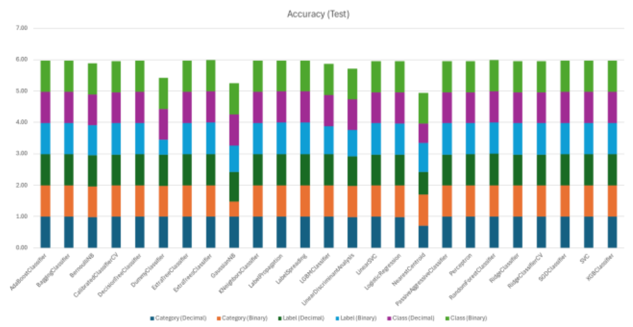


Fig. 6. Comparison of training and testing accuracy of all models.

B. *Balanced Accuracy*

The balanced accuracy results (see Table VII and Table VIII and Fig. 7) reveal how well the models handle imbalanced data during training and testing. Many models, such as DecisionTree, ExtraTreesClassifier, and XGBClassifier, achieved perfect balanced accuracy across all classifications in both formats during training, indicating their ability to learn effectively from imbalanced data.

TABLE VII. TRAINING BALANCED ACCURACY OF ALL MODELS

Model	Category (D)	Category (B)	Label (D)	Label (B)	Class (D)	Class (B)
AdaBoost	0.86	0.85	1.00	1.00	0.30	0.17
Bagging	0.95	1.00	0.98	0.98	0.99	1.00
Bernoulli NB	0.68	0.92	0.77	0.92	0.25	0.67
Calibrated CV	0.33	1.00	0.50	1.00	0.17	0.52
Decision Tree	1.00	1.00	1.00	1.00	1.00	1.00
Dummy Classifier	0.33	0.33	0.50	0.50	0.17	0.17
Extra Tree	1.00	1.00	1.00	1.00	1.00	1.00
Extra Trees	1.00	1.00	1.00	1.00	1.00	1.00
Gaussian NB	0.78	1.00	0.84	0.76	0.95	1.00
K Neighbors	0.81	0.91	0.97	0.97	0.42	0.56
Label Propagation	1.00	1.00	1.00	1.00	1.00	1.00
Label Spreading	1.00	1.00	1.00	1.00	1.00	1.00
LGBM Classifier	1.00	1.00	1.00	1.00	1.00	1.00
Linear Discriminant	0.38	0.95	0.58	0.97	0.50	0.98
Linear SVC	0.36	1.00	0.67	1.00	0.18	1.00
Logistic Regression	0.36	1.00	0.64	0.97	0.21	1.00
Nearest Centroid	0.74	0.97	0.85	0.96	0.84	0.92
Passive Aggressive	0.33	0.95	0.60	0.97	0.50	1.00
Perceptron	0.67	1.00	0.74	1.00	0.20	1.00
Random Forest	1.00	1.00	1.00	1.00	1.00	1.00
Ridge Classifier	0.33	0.93	0.50	0.95	0.17	0.77
Ridge CV	0.33	0.88	0.50	0.90	0.17	0.77
SGDClassifier	0.69	0.95	0.84	1.00	0.37	0.92
SVC	0.77	0.88	0.98	0.93	0.50	0.76
XGB Classifier	1.00	1.00	1.00	1.00	1.00	1.00

TABLE VIII. TESTING BALANCED ACCURACY OF ALL MODELS

Model	Category (D)	Category (B)	Label (D)	Label (B)	Class (D)	Class (B)
AdaBoost	0.72	1.00	0.92	1.00	0.30	0.17
Bagging	0.72	0.91	0.83	1.00	0.69	0.39
Bernoulli NB	0.55	0.95	0.62	0.96	0.25	0.55
Calibrated CV	0.33	0.92	0.54	1.00	0.17	0.39
Decision Tree	0.94	0.91	0.83	1.00	0.53	0.39
Dummy Classifier	0.33	0.50	0.50	0.50	0.17	0.17
Extra Tree	0.72	0.91	0.87	1.00	0.47	0.56
Extra Trees	0.72	0.92	0.92	1.00	0.72	0.69
Gaussian NB	0.67	0.74	0.85	0.84	0.56	0.56
K Neighbors	0.67	0.92	0.87	1.00	0.39	0.42
Label Propagation	0.83	1.00	0.87	1.00	0.72	0.17
Label Spreading	0.78	1.00	0.87	1.00	0.72	0.17
LGBM Classifier	1.00	0.92	0.92	0.88	0.56	0.78
Linear Discriminant	0.33	0.96	0.48	0.83	0.33	0.56
Linear SVC	0.33	1.00	0.58	1.00	0.17	0.56
Logistic Regression	0.33	0.96	0.54	1.00	0.19	0.39
Nearest Centroid	0.63	0.91	0.78	0.92	0.66	0.69
Passive Aggressive	0.33	0.96	0.58	1.00	0.33	0.56
Perceptron	0.67	0.96	0.75	1.00	0.17	0.55
Random Forest	0.94	0.96	0.92	1.00	0.72	0.39
Ridge Classifier	0.33	0.96	0.50	1.00	0.17	0.56
Ridge CV	0.33	0.96	0.50	1.00	0.17	0.56
SGDClassifier	0.61	0.92	0.83	1.00	0.33	0.36
SVC	0.61	0.96	0.83	1.00	0.36	0.22
XGB Classifier	0.89	0.96	0.79	1.00	0.53	0.75

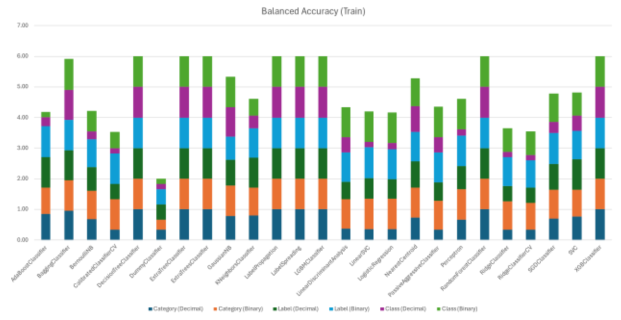
However, testing balanced accuracy showed a decline for some models, such as GaussianNB, NearestCentroid, and SGDClassifier, particularly in challenging classifications like Class (Binary) and Label (Decimal), suggesting overfitting or difficulty in generalizing to unseen data. Ensemble and tree-based methods like RandomForestClassifier and XGBClassifier maintained consistently high performance across both phases, demonstrating their robustness. In contrast, simpler models and linear methods struggled with imbalanced data, especially in more granular classifications. These results highlight the importance of effectively selecting models capable of effectively addressing class imbalance.

C. *F1-Score*

The F1-score results, as shown in the tables (IX and Table X) and charts (Fig. 8), provide a detailed evaluation of the model's F1-score, particularly in balancing precision and recall, which is crucial for imbalanced datasets. During training, most models, such as DecisionTree, ExtraTreesClassifier, XGBClassifier, and LabelPropagation, achieved perfect F1-scores across all classifications in both Decimal and Binary formats, similar to their accuracy and balanced accuracy results. However, models like NearestCentroid and GaussianNB showed lower F1-scores in some scenarios, such as Class (Binary), reflecting their difficulty in managing imbalanced classes effectively.

TABLE IX. TRAINING F1-SCORE OF ALL MODELS

Model	Category (D)	Category (B)	Label (D)	Label (B)	Class (D)	Class (B)
AdaBoost	1.00	0.99	1.00	1.00	0.99	0.98
Bagging	1.00	1.00	1.00	1.00	1.00	1.00
Bernoulli NB	0.99	0.98	0.99	0.98	0.99	0.99
Calibrated CV	0.98	1.00	0.98	1.00	0.98	1.00
Decision Tree	1.00	1.00	1.00	1.00	1.00	1.00
Dummy Classifier	0.98	0.98	0.98	0.98	0.98	0.98
Extra Tree	1.00	1.00	1.00	1.00	1.00	1.00
Extra Trees	1.00	1.00	1.00	1.00	1.00	1.00
Gaussian NB	0.99	1.00	0.96	0.68	1.00	1.00
K Neighbors	1.00	1.00	1.00	1.00	0.99	1.00
Label Propagation	1.00	1.00	1.00	1.00	1.00	1.00
Label Spreading	1.00	1.00	1.00	1.00	1.00	1.00
LGBM Classifier	1.00	1.00	1.00	1.00	1.00	1.00
Linear Discriminant	0.98	1.00	0.97	1.00	0.97	1.00
Linear SVC	0.98	1.00	0.99	1.00	0.98	1.00
Logistic Regression	0.98	1.00	0.99	1.00	0.99	1.00
Nearest Centroid	0.83	0.99	0.84	0.99	0.75	0.98
Passive Aggressive	0.98	1.00	0.99	1.00	0.99	1.00
Perceptron	0.99	1.00	0.99	1.00	0.98	1.00
Random Forest	1.00	1.00	1.00	1.00	1.00	1.00
Ridge Classifier	0.98	1.00	0.98	1.00	0.98	1.00
Ridge CV	0.98	1.00	0.98	1.00	0.98	1.00
SGDClassifier	0.99	1.00	0.99	1.00	0.99	1.00
SVC	1.00	1.00	1.00	1.00	0.99	1.00
XGB Classifier	1.00	1.00	1.00	1.00	1.00	1.00



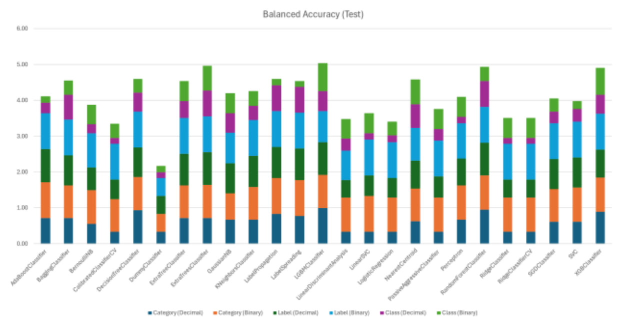


Fig. 7. Comparison of training and testing balanced accuracy of all models.

TABLE X. TESTING F1-SCORE OF ALL MODELS

Model	Category (D)	Category (B)	Label (D)	Label (B)	Class (D)	Class (B)
AdaBoost	0.99	1.00	1.00	1.00	0.99	0.98
Bagging	0.99	0.99	1.00	1.00	1.00	0.99
Bernoulli NB	0.99	0.98	0.99	0.96	0.99	0.99
Calibrated CV	0.98	1.00	0.98	1.00	0.98	0.99
Decision Tree	1.00	0.99	0.99	1.00	0.99	0.99
Dummy Classifier	0.98	0.98	0.98	0.31	0.98	0.98
Extra Tree	0.99	0.99	1.00	1.00	0.99	0.99
Extra Trees	0.99	1.00	1.00	1.00	1.00	1.00
Gaussian NB	0.99	0.64	0.96	0.84	1.00	0.99
K Neighbors	0.99	1.00	1.00	1.00	0.99	0.99
Label Propagation	1.00	1.00	1.00	1.00	1.00	0.98
Label Spreading	1.00	1.00	1.00	1.00	1.00	0.98
LGBM Classifier	1.00	1.00	1.00	0.88	1.00	1.00
Linear Discriminant	0.97	1.00	0.97	0.83	0.97	0.99
Linear SVC	0.98	1.00	0.98	1.00	0.98	0.99
Logistic Regression	0.98	1.00	0.98	1.00	0.99	0.99
Nearest Centroid	0.82	0.99	0.83	0.92	0.76	0.98
Passive Aggressive	0.98	1.00	0.99	1.00	0.98	0.99
Perceptron	0.99	1.00	0.99	1.00	0.98	0.99
Random Forest	1.00	1.00	1.00	1.00	1.00	0.99
Ridge Classifier	0.98	1.00	0.98	1.00	0.98	1.00
Ridge CV	0.98	1.00	0.98	1.00	0.98	1.00
SGDClassifier	0.99	1.00	0.99	1.00	0.99	0.99
SVC	0.99	1.00	1.00	1.00	0.99	0.99
XGB Classifier	1.00	1.00	0.99	1.00	0.99	0.99

In testing, the F1-scores revealed a more nuanced picture compared to accuracy and balanced accuracy. While ensemble models like RandomForestClassifier, ExtraTreesClassifier, and XGBClassifier maintained high F1-scores, models like GaussianNB and NearestCentroid experienced noticeable drops, particularly for imbalanced classes, as seen in Class

(Binary) and Label (Decimal). These drops align with the declines observed in balanced accuracy, reinforcing the importance of metrics like F1-score for evaluating models on imbalanced datasets. Overall, while accuracy may appear high for certain models, the F1-score highlights their limitations in balancing precision and recall, providing a more comprehensive view of model performance in such challenging scenarios.

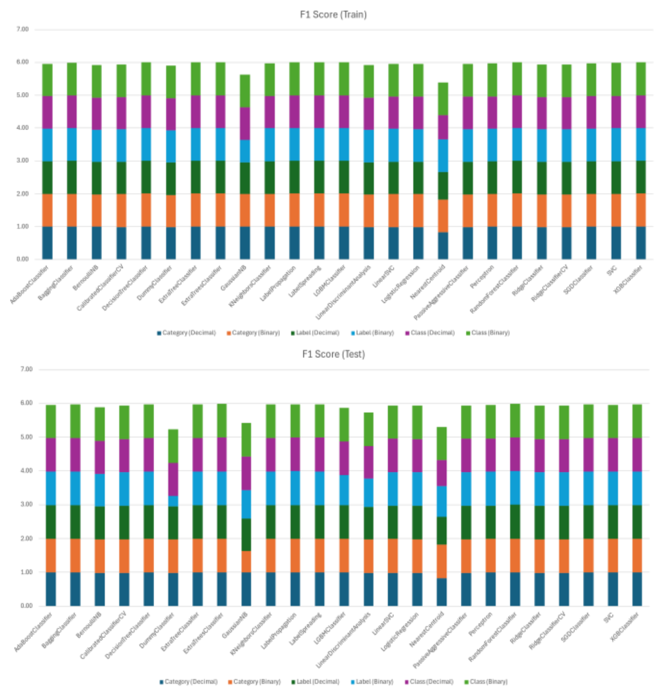


Fig. 8. Comparison of training and testing F1-Score of all models

D. Training and Testing Time

Results in Table XI, Table XII and Fig. 9 compare the training and testing time. The time taken for training and testing ML models highlight the computational efficiency. Lightweight models, such as BernoulliNB, GaussianNB, and LogisticRegression, exhibited minimal training and testing times across all classifications, making them ideal for scenarios with limited computational resources. In contrast, more complex models like CalibratedClassifierCV, LabelPropagation, and LabelSpreading required significantly longer training and testing times, particularly for Class (Binary), due to their iterative or probabilistic nature.

Tree-based ensemble models, such as RandomForestClassifier, ExtraTreesClassifier, and XGBClassifier, balanced efficiency and performance with moderate training and testing times. Notably, CalibratedClassifierCV had the longest training and testing times, especially for Class (Binary), suggesting a high computational cost for its probability calibration. These results underline the importance of considering computational time alongside accuracy and balanced accuracy when selecting models, especially for real-time or resource-constrained applications such as the IoV applications.

TABLE XI. TRAINING TIME OF ALL MODELS

Model	Category (D)	Category (B)	Label (D)	Label (B)	Class (D)	Class (B)
AdaBoost	0.62	0.69	0.21	0.86	0.38	0.34
Bagging	0.17	0.41	0.05	0.62	0.10	0.11
Bernoulli NB	0.08	0.33	0.02	0.49	0.03	0.06
Calibrated CV	0.44	1.02	0.07	1.06	0.40	14.61
Decision Tree	0.06	0.27	0.04	0.43	0.04	0.06
Dummy Classifier	0.02	0.26	0.02	0.43	0.03	0.05
Extra Tree	0.03	0.26	0.03	0.43	0.03	0.05
Extra Trees	0.46	0.59	0.18	0.62	0.34	0.36
Gaussian NB	0.08	0.43	0.02	0.25	0.03	0.05
K Neighbors	0.43	0.63	0.17	0.35	0.29	0.25
Label Propagation	0.68	1.46	0.42	0.93	0.44	0.60
Label Spreading	0.76	1.74	0.52	1.41	0.52	0.72
LGBM Classifier	0.28	0.65	0.13	0.42	0.55	0.62
Linear Discriminant	0.04	0.94	0.14	0.43	0.09	0.13
Linear SVC	0.05	0.40	0.07	0.33	0.05	5.06
Logistic Regression	0.06	0.45	0.05	0.39	0.05	0.11
Nearest Centroid	0.02	0.28	0.03	0.28	0.02	0.05
Passive Aggressive	0.02	0.33	0.03	0.27	0.03	0.14
Perceptron	0.03	0.35	0.03	0.31	0.03	0.11
Random Forest	0.28	0.57	0.30	0.55	0.27	0.31
Ridge Classifier	0.02	0.32	0.03	0.42	0.02	0.18
Ridge CV	0.04	0.49	0.12	0.46	0.03	0.15
SGD Classifier	0.09	0.33	0.04	0.30	0.11	0.12
SVC	0.05	0.40	0.06	0.37	0.06	0.50
XGB Classifier	0.11	0.61	0.10	0.43	0.16	0.43

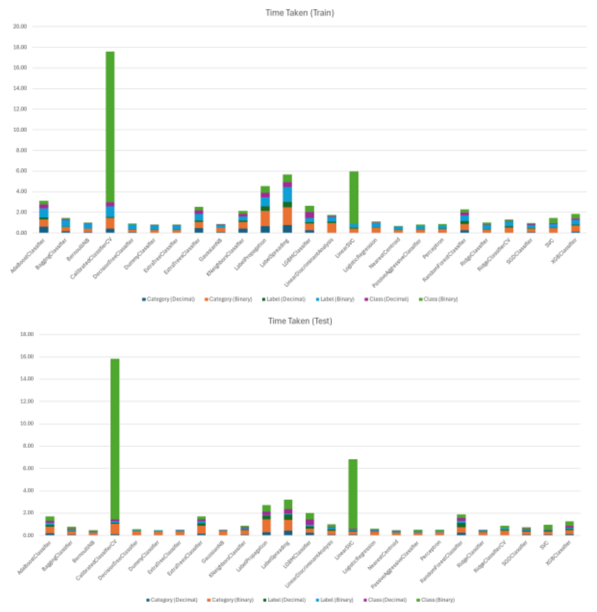


Fig. 9. Comparison of training and testing time of all models.

TABLE XII. TESTING TIME OF ALL MODELS

Model	Category (D)	Category (B)	Label (D)	Label (B)	Class (D)	Class (B)
AdaBoost	0.22	0.57	0.18	0.18	0.18	0.38
Bagging	0.06	0.30	0.06	0.09	0.07	0.18
Bernoulli NB	0.02	0.25	0.02	0.06	0.02	0.10
Calibrated CV	0.11	0.94	0.08	0.12	0.19	14.37
Decision Tree	0.02	0.35	0.03	0.06	0.02	0.05
Dummy Classifier	0.02	0.32	0.03	0.06	0.02	0.04
Extra Tree	0.02	0.31	0.03	0.07	0.03	0.03
Extra Trees	0.17	0.72	0.19	0.18	0.17	0.26
Gaussian NB	0.02	0.35	0.02	0.06	0.02	0.04
K Neighbors	0.08	0.41	0.08	0.07	0.08	0.14
Label Propagation	0.30	1.15	0.33	0.08	0.30	0.54
Label Spreading	0.45	0.97	0.45	0.10	0.40	0.86
LGBM Classifier	0.25	0.36	0.23	0.13	0.47	0.58
Linear Discriminant	0.05	0.39	0.10	0.19	0.05	0.20
Linear SVC	0.06	0.28	0.07	0.10	0.08	6.25
Logistic Regression	0.04	0.29	0.05	0.10	0.04	0.09
Nearest Centroid	0.02	0.24	0.05	0.10	0.02	0.05
Passive Aggressive	0.03	0.22	0.06	0.06	0.03	0.12
Perceptron	0.02	0.26	0.03	0.06	0.03	0.10
Random Forest	0.27	0.47	0.37	0.21	0.27	0.30
Ridge Classifier	0.02	0.28	0.06	0.07	0.02	0.07
Ridge CV	0.04	0.40	0.09	0.10	0.04	0.18
SGDClassifier	0.09	0.25	0.07	0.09	0.10	0.14
SVC	0.06	0.30	0.10	0.06	0.05	0.38
XGB Classifier	0.11	0.35	0.15	0.09	0.15	0.41

E. General Discussion and Recommendation

The analysis of all results, including accuracy, balanced accuracy, F1-scores, and computational time, reveals a comprehensive comparison of model performance on the dataset. Tree-based ensemble models, such as Decision Tree, RandomForestClassifier, ExtraTreesClassifier, and XGBClassifier, consistently achieved near-perfect scores across all metrics, including accuracy, balanced accuracy, and F1-scores, while maintaining moderate computational times, making them reliable and efficient choices for most tasks. Lightweight models, such as LogisticRegression, BernoulliNB, and GaussianNB, demonstrated low computational times with competitive performance in accuracy and F1-scores, but they struggled with balanced accuracy in scenarios with significant class imbalance. On the other hand, models like CalibratedCV, LabelPropagation, and LabelSpreading achieved excellent accuracy and F1-scores but at the expense of significantly higher training and testing times, particularly for more complex classifications like Class (Binary).

While accuracy and F1-scores highlight overall model performance, balanced accuracy provided more profound insights into handling class imbalances, exposing limitations in models like NearestCentroid and GaussianNB. The computational time results underscored the trade-offs between predictive performance and resource efficiency, with certain models offering high accuracy at the cost of increased processing time. In summary, ensemble methods emerged as the dataset's most robust and practical choice, balancing performance and efficiency. At the same time, lightweight models offered a computationally inexpensive alternative with slightly reduced robustness. These findings emphasize the importance of selecting models based on the application's specific requirements, whether prioritizing accuracy, computational efficiency, or the ability to handle imbalanced datasets.

Recommendations: Based on the discussed results, several recommendations can be made to enhance intrusion detection in vehicular networks. Ensemble models, such as RandomForest ExtraTreesClassifier, ExtraTreesClassifier, and XGBClassifier, should be prioritized due to their superior performance in accuracy, balanced accuracy, and F1-score, particularly for handling imbalanced datasets. Lightweight models like LogisticRegression and BernoulliNB can be optimized with techniques such as oversampling, feature scaling, or class-weight adjustments to enhance their performance in imbalanced scenarios. Computationally intensive models like LabelPropagation and CalibratedCV should be optimized for real-time use through hybrid approaches or parallel processing techniques. Additionally, expanding the dataset to include more diverse attack scenarios and vehicular communication protocols will improve model generalizability. Balanced accuracy and F1-scores should be emphasized as key evaluation metrics, particularly in imbalanced datasets, to ensure fair assessments. Finally, integrating high-performing models into real-time systems with optimized preprocessing pipelines, including duplicate removal and stratified splitting, will enhance their practical applicability in real-world vehicular network scenarios.

V. CONCLUSION

This study comprehensively explored the CICIoV2024 dataset to evaluate the effectiveness of various advanced ML algorithms in intrusion detection for vehicular networks, focusing on CAN security. The research highlights the significance of data preprocessing, including duplicate removal and stratified splitting, in ensuring robust model evaluation. A wide range of ML models were assessed across metrics such as accuracy, balanced accuracy, F1-score, and computational efficiency.

The findings underscore the superior performance of ensemble-based and tree-based models, such as RandomForestClassifier, ExtraTreesClassifier, and XGBoostClassifier, consistently demonstrating high generalization and resilience to imbalanced data. Simpler models, such as LogisticRegression and GaussianNB, offered computational efficiency but struggled with complex, imbalanced scenarios. Models like LabelPropagation and CalibratedClassifiers achieved excellent accuracy but incurred higher computational costs, limiting their applicability for real-time environments.

Despite achieving high accuracy, the study identified concerns regarding potential overfitting in some models, emphasizing the need for broader evaluation across diverse datasets. The CICIoV2024 dataset, with its realistic representation of spoofing and DoS attacks, proved to be a valuable resource but requires further exploration to harness its potential fully.

Future work will focus on integrating additional attack scenarios, enhancing the dataset's diversity, evaluating the scalability of ML models across varying vehicular communication protocols, and improving the generalizability of models to diverse communication protocols and real-world conditions. Moreover, we could explore more advanced ML techniques such as reinforcement learning-based IDS, federated learning, or lightweight transformer models for IoV security.

REFERENCES

[1] M. E. E. Alahi et al., "Integration of IoT-Enabled Technologies and Artificial Intelligence (AI) for Smart City Scenario: Recent Advancements and Future Trends," Sensors 2023, Vol. 23, Page 5206, vol. 23, no. 11, p. 5206, May 2023, doi: 10.3390/S23115206.

[2] D. Serpanos and M. Wolf, "The IoT Landscape," in Internet-of-Things (IoT) Systems, Cham: Springer International Publishing, 2018, pp. 1–6. doi: 10.1007/978-3-319-69715-4_1.

[3] N. Janbi, I. Katib, A. Albeshri, and R. Mehmood, "Distributed artificial intelligence-as-a-service (DAIaaS) for smarter IoE and 6G environments," Sensors (Switzerland), vol. 20, no. 20, pp. 1–28, Oct. 2020, doi: 10.3390/s20205796.

[4] N. Janbi, R. Mehmood, I. Katib, A. Albeshri, J. M. Corchado, and T. Yigitcanlar, "Imtidad: A Reference Architecture and a Case Study on Developing Distributed AI Services for Skin Disease Diagnosis over Cloud, Fog and Edge," Sensors 2022, Vol. 22, Page 1854, vol. 22, no. 5, p. 1854, Feb. 2022, doi: 10.3390/S22051854.

[5] M. Merenda, C. Porcaro, and D. Iero, "Edge Machine Learning for AI-Enabled IoT Devices: A Review," Sensors 2020, Vol. 20, Page 2533, vol. 20, no. 9, p. 2533, Apr. 2020, doi: 10.3390/S20092533.

[6] N. F. Janbi, M. A. Ghaseb, and A. A. Almazroi, "ESTS-GCN: An Ensemble Spatial–Temporal Skeleton-Based Graph Convolutional Networks for Violence Detection," Int. J. Intell. Syst., vol. 2024, no. 1, p. 2323337, Jan. 2024, doi: 10.1155/2024/2323337.

[7] N. Srinivasan, "Artificial Intelligence in IoT Security: Review of Advancements, Challenges, and Future Directions," Int. J. Innov. Technol. Explor. Eng., vol. 13, no. 7, pp. 14–20, 2024, doi: 10.35940/ijitee.g9911.13070624.

[8] N. Janbi, I. Katib, and R. Mehmood, "Distributed artificial intelligence: Taxonomy, review, framework, and reference architecture," Intell. Syst. with Appl., vol. 18, p. 200231, May 2023, doi: 10.1016/j.iswa.2023.200231.

[9] S. A. Abdulkareem, C. H. Foh, M. Shojafar, F. Carrez, and K. Moessner, "Network Intrusion Detection: An IoT and Non IoT-Related Survey," IEEE Access, 2024, doi: 10.1109/ACCESS.2024.3473289.

[10] S. M. Karim, A. Habbal, S. A. Chaudhry, and A. Irshad, "Architecture, Protocols, and Security in IoV: Taxonomy, Analysis, Challenges, and Solutions," Secur. Commun. Networks, vol. 2022, no. 1, p. 1131479, Jan. 2022, doi: 10.1155/2022/1131479.

[11] H. Taslimasa, S. Dadkhah, E. C. P. Neto, P. Xiong, S. Ray, and A. A. Ghorbani, "Security issues in Internet of Vehicles (IoV): A comprehensive survey," Internet of Things, vol. 22, p. 100809, Jul. 2023, doi: 10.1016/J.IOT.2023.100809.

[12] M. Hanselmann, T. Strauss, K. Dormann, and H. Ulmer, "CANet: An Unsupervised Intrusion Detection System for High Dimensional CAN Bus Data," IEEE Access, vol. 8, pp. 58194–58205, 2020, doi: 10.1109/ACCESS.2020.2982544.

[13] A. Salehi Shahraki, L. Diana, P. Dini, and D. Paolini, "Overview on Intrusion Detection Systems for Computers Networking Security," Comput. 2025, Vol. 14, Page 87, vol. 14, no. 3, p. 87, Mar. 2025, doi: 10.3390/COMPUTERS14030087.

[14] A. Sivanathan, H. Habibi Gharakheili, and V. Sivaraman, "Managing IoT Cyber-Security Using Programmable Telemetry and Machine Learning," IEEE Trans. Netw. Serv. Manag., vol. 17, no. 1, pp. 60–74, Mar. 2020, doi: 10.1109/TNSM.2020.2971213.

[15] E. C. P. Neto et al., "CICIoV2024: Advancing realistic IDS approaches against DoS and spoofing attack in IoV CAN bus," Internet of Things, vol. 26, p. 101209, Jul. 2024, doi: 10.1016/J.IOT.2024.101209.

[16] K. Aswal and H. Pathak, "Advancing Vehicle Security: Deep Learning based Solution for Defending CAN Networks in the Internet of Vehicles," EAI Endorsed Trans. Internet Things, vol. 10, pp. 1–14, Oct. 2024, doi: 10.4108/EETIOT.6523.

[17] N.' Aliah Amirudin and S. J. Abdulkadir, "Comparative Study of Machine Learning Algorithms using the CICIOV2024 Dataset," Platf. A J. Sci. Technol., vol. 7, no. 1, p. 1, 2024, doi: 10.61762/pjstvol7iss1art27052.

[18] O. Subasi, J. Cree, J. Manzano, and E. Peterson, "A Critical Assessment of Interpretable and Explainable Machine Learning for Intrusion Detection," Jul. 2024, Accessed: Dec. 05, 2024. [Online]. Available: https://arxiv.org/abs/2407.04009v1

[19] Z. S. Mahdi, R. M. Zaki, and L. Alzubaidi, "Advanced Hybrid Techniques for Cyberattack Detection and Defense in IoT Networks," Secur. Priv., p. e471, Oct. 2024, doi: 10.1002/SPY2.471.

[20] B. Taşcı, "Deep-Learning-Based Approach for IoT Attack and Malware Detection," Appl. Sci. 2024, Vol. 14, Page 8505, vol. 14, no. 18, p. 8505, Sep. 2024, doi: 10.3390/APP14188505.

[21] P. Dhawas, A. Dhore, D. Bhagat, R. D. Pawar, A. Kukade, and K. Kalbande, Big Data Preprocessing, Techniques, Integration, Transformation, Normalisation, Cleaning, Discretization, and Binning. IGI Global, 2024. doi: 10.4018/979-8-3693-0413-6.ch006.