Performance Evaluation of Machine Learning-Based Cyber Attack Detection in Electric Vehicles Charging Stations

Mutaz A.B. Al-Tarawneh, Omar Alirr, Hassan Kanj
College of Engineering and Technology, American University of the Middle East, Egaila 54200, Kuwait

***Abstract*—Electric Vehicles (EV) chargers rely on resource-constrained embedded hardware to execute critical charging operations. However, conventional security solutions may not adequately meet the needs of these devices. Increasingly, machine learning techniques are being leveraged to detect cyber attacks during electric vehicle charging. This study aims to evaluate various base machine learning methods and conduct binary and multi-class classification experiments to enhance security and operational efficiency in EV charging stations. The experiments utilize the CICEVSE2024 dataset, curated by the Canadian Institute for Cybersecurity at the University of New Brunswick, designed specifically for anomaly detection and establishing behavioral patterns in EV charging stations. The analysis highlights nuances in performance across different machine learning classifiers. For instance, Random Forest achieved 95.07% accuracy in binary classification by constructing robust decision trees. Ensemble methods such as CatBoost and LightGBM further improved binary classification to 95.37% and 95.41%, respectively through gradient boosting techniques. In multi-class attack classification, ensemble methods demonstrated superior performance, with the Stacking Ensemble achieving 91.1% accuracy by combining multiple models, and Voting Ensemble achieving 90.7%. Notably, among homogeneous base classifiers, Extra Trees and HistGradient Boosting were particularly effective, achieving 90.2% and 89.8% accuracy respectively in multi-class classification tasks. These findings underscore the efficacy of machine learning in enhancing cybersecurity measures for EV charging infrastructure.**

Keywords*—Machine learning; cyber attack detection; cyber threats; distributed denial of service attack; charging stations

I. Introduction

The proliferation of electric vehicles (EVs) has led to a significant increase in the deployment of electric vehicle charging stations (EVCS) worldwide. However, this expansion has brought attention to cybersecurity vulnerabilities associated with these stations [1]. This section examines the widespread adoption of EVCS, explores their susceptibility to cyber-attacks, discusses the role of machine learning (ML) in bolstering their security, and identifies the common attack patterns targeting EVCS, their implications, and mitigation strategies.

The transition to electric vehicles (EVs) is gaining momentum globally, driven by environmental concerns and advancements in technology. Central to this shift is the development and deployment of electric vehicle charging infrastructure (EVCI), which plays a critical role in supporting the widespread adoption of EVs. This infrastructure has rapidly grown to support the increasing number of electric vehicles on the road [2]. It encompasses a diverse range of charging stations, from residential Level 1 chargers to high-power DC fast chargers installed along highways and in urban centers. Governments, private sector entities, and utilities worldwide are investing in expanding EVCI networks to meet the growing demand for electric mobility [3]. Governments, private companies, and utilities have invested heavily in establishing charging networks to promote sustainable transportation [4]. The deployment spans various types of chargers, including Level 1, Level 2, and DC fast chargers, catering to different charging needs and speeds [5].

Despite significant progress, EVCI deployment faces several challenges as: 1) the uneven distribution of charging stations, with rural and suburban areas often lagging behind urban centers in accessibility [6], 2) the high cost of infrastructure installation and grid capacity upgrades also pose financial challenges for stakeholders [7], 3) interoperability issues between different charging networks and varying charging standards can complicate the user experience and slow down adoption rates [8], and 4) the most severe one is the vulnerability of those station from Cyber attacks [9].

EVCS are vulnerable to cyber-attacks due to their interconnected nature and reliance on communication networks for operation and management [10]. Threats range from unauthorized access to charging data and financial information to potential disruption of service or even physical damage to vehicles through malicious software or hacking attempts [11], [9]. Vulnerabilities can arise from weaknesses in network protocols, inadequate authentication mechanisms, or compromised software updates [12].

Machine learning techniques offer promising solutions to mitigate cybersecurity risks associated with EVCS. ML algorithms can analyze large volumes of data generated by charging stations to detect anomalies indicative of cyber-attacks or unauthorized access attempts [13]. Techniques such as anomaly detection, pattern recognition, and predictive analytics can enhance the ability to identify and respond to potential threats in real-time, thereby fortifying the security posture of EVCS [14], [15].

Recent studies highlight ongoing efforts to integrate ML-based security solutions into EV charging infrastructure [16]. Researchers are exploring adaptive ML models capable of learning from evolving attacks, their threats and improving detection accuracy over time [17]. Furthermore, advancements in cryptographic protocols and secure communication frameworks aim to safeguard data transmission between EVs,

charging stations, and central management systems [18].

Those stations are subjects of attacks of several categories as follows:

1) Man-in-the-Middle (MitM) attacks: occur when an attacker intercepts the communication between the EV and the charging station or the charging station and the backend system. This allows the attacker to eavesdrop, alter, or inject malicious data into the communication stream. MitM attacks can lead to unauthorized charging, data theft, and even manipulation of charging parameters, potentially damaging the vehicle or infrastructure [19].

2) Denial of Service (DoS) attacks: aim to make the charging service unavailable to legitimate users. Attackers can overwhelm the charging station or its network with excessive requests, causing the system to crash or become unresponsive. This type of attack can disrupt the availability of charging services, leading to inconvenience for EV users and potential revenue loss for service providers [20].

3) Malware and Ransomware charging stations: like other networked devices, can be targeted with malware or ransomware. Malware can compromise the station's software, causing it to malfunction or operate incorrectly. Ransomware can encrypt the station's data or control systems, rendering it inoperable until a ransom is paid. Such attacks can lead to service disruptions and financial losses [21].

4) Unauthorized access and physical tampering: Physical access to charging stations can allow attackers to tamper with the hardware or install unauthorized devices. This can lead to direct theft of electricity, physical damage to the station, or insertion of malicious components that facilitate further cyber-attacks. Ensuring physical security is as crucial as securing network communications [22].

5) False data injection attacks: In false data injection attacks, attackers send incorrect data to the charging station or its management system. This can affect billing, load management, and the operational integrity of the station. For example, false readings could lead to incorrect billing or overloads on the power grid if demand is misrepresented [23].

Those attacks can have wide-ranging implications such as financial losses for operators, inconvenience and safety risks for users, and broader impacts on the electrical grid and urban infrastructure. Additionally, compromised EVCS can serve as entry points for attacks on other critical systems, posing significant national security risks [24].

Mitigation actions/processes can be adopted to manage those attacks. Main actions found in literature are:

- Implementing robust encryption protocols and multi-factor authentication. Public Key Infrastructure (PKI) and Transport Layer Security (TLS) are commonly recommended to secure data exchanges [22].
- Keeping software and firmware up-to-date is crucial for addressing vulnerabilities. Regular updates and timely patch management can mitigate the risk of exploits targeting known weaknesses [22].
- Deploying Intrusion Detection and Prevention Systems (IDPS) can help detect and respond to suspicious activities in real-time. Machine learning-based IDPS can analyze patterns and identify anomalies that indicate potential attacks [25].
- Securing the physical infrastructure of charging stations with surveillance, tamper-evident seals, and restricted access can prevent unauthorized physical interactions that could compromise cybersecurity [18].
- Building redundancy into the charging network and ensuring resilience through backup systems and alternative power supplies can help maintain service continuity during and after an attack [18].

The remainder of this paper is organized as follows. Section II presents the applied research methodology, Section III discusses the obtained results and Section IV concludes and summarizes this work.

II. Methodology

The framework and the stages followed in this research including data collection, machine learning implementation for cyber-attack detection, and performance evaluation are described in Fig. 1.

A. Data Collection

1) Dataset Description: This work is based on the dataset named CICEVSE2024, designed to enhance the security of Electric Vehicle Charging Stations (EVCS) through the application of machine learning techniques for cyber-attack detection [26]. The dataset was generated using a comprehensive and realistic setup involving real Electric Vehicle Supply Equipment (EVSE) to capture authentic power consumption data under various operational states. A Raspberry Pi was employed to simulate network traffic and host activities, providing a versatile and cost-effective solution for capturing data in a controlled environment. The data collection framework integrated sensors and monitoring tools to continuously record power usage, network traffic, and host activities. Various cyber-attack scenarios, such as Denial of Service (DoS), spoofing, and malware injection, were simulated to generate labelled instances of attack conditions. Additionally, data under normal operational conditions was collected to establish baseline patterns of power consumption, network traffic, and host activities.

This dataset offers several key advantages. The use of real EVSE equipment ensures the capture of realistic power consumption patterns, enhancing the reliability of machine learning models trained on this data. The multi-dimensional nature of the dataset, encompassing power consumption, network traffic, and host activities, provides a holistic view of EVCS operations and potential attack vectors. The inclusion of labelled instances of both normal and attack conditions facilitates supervised learning, enabling the development of accurate and effective anomaly detection models. The use of Raspberry Pi for simulating network and host activities allows for flexible and scalable data collection, accommodating various types of cyber-attacks and operational scenarios. The detailed annotations and comprehensive coverage of different aspects of EVCS operations make the dataset suitable for benchmarking and comparing different machine learning algorithms for cyber-attack detection. By leveraging this dataset,

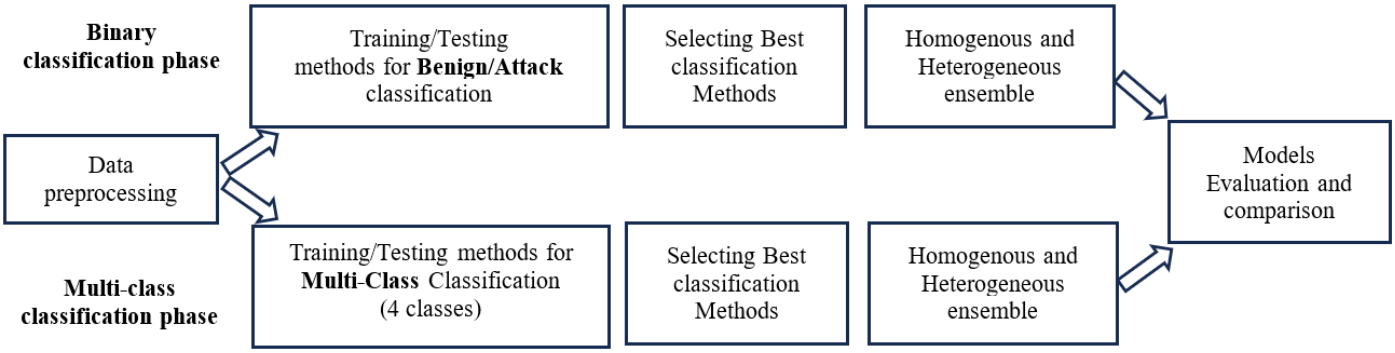

Fig. 1. Research methodology framework.

researchers and practitioners can develop robust machine learning models that enhance the security of EV charging stations, ensuring reliable and safe operation in the face of potential cyber threats. This work focuses primarily on attack detection based on the electric vehicle supply equipment (EVSE) power consumption data under both normal and attack settings. Table I summarizes the power consumption features used in this work. As shown, the dataset contains four numeric features along with a single categorical feature. The numeric features include shunt voltage (mV), Bus voltage, EVSE Current, and EVSE power consumption. On the other hand, the categorical feature indicates whether the EVSE is in the idle or the charging state. Table II presents descriptive statistics of numeric features within a dataset, including shunt voltage, bus voltage, current, and power measurements. On average, the shunt voltage is approximately 619.79 mV, with a standard deviation of 197.19 mV, indicating considerable variability. In contrast, the bus voltage remains relatively stable around 5.19 V, with a minimal standard deviation of 0.01 V. Current readings average around 619.76 mA, displaying a similar level of variability to the shunt voltage. Power consumption averages 3212.78 mW, with a wider range from 2160 mW to 6300 mW. These descriptive statistics offer insights into the distribution and variability of the dataset's numeric features.

On the other hand, Table III delineates descriptive statistics of numeric features categorized by two classes: "attack" and "benign".

In terms of shunt voltage, the "attack" class exhibits a higher mean of approximately 631.17 mV compared to the "benign" class, which averages around 539.83 mV. Both classes display variability, with the "attack" class having a wider standard deviation of 204.85 mV compared to 99.72 mV for "benign" (Fig. 2).

The bus voltage remains relatively consistent across classes, hovering around 5.19 V to 5.20 V, with minimal standard deviations (Fig. 3).

Moving to current values, the "attack" class shows a higher mean of approximately 631.32 mA, indicating potentially more intense activity compared to the "benign" class, which averages about 538.54 mA. Furthermore, the "attack" class displays a wider spread in current readings, with a larger standard deviation of 204.96 mA compared to 98.91 mA for "benign" (Fig. 4).

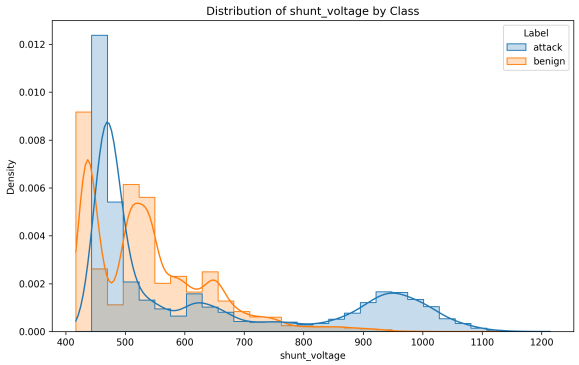

Fig. 2. Shunt voltage histogram per class.

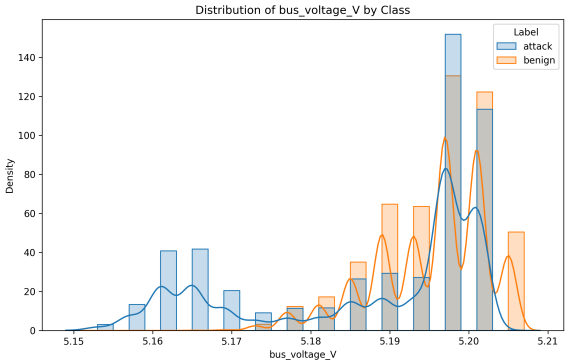

Fig. 3. Bus voltage histogram per class.

Regarding power consumption, the "attack" class exhibits a higher mean of approximately 3271.47 mW, reflecting increased energy usage during potential attacks, while the "benign" class averages around 2800.39 mW. Similarly, the "attack" class demonstrates greater variability in power consumption, with a larger standard deviation of 1050.57 mW compared to 513.59 mW for "benign" (Fig. 5).

In order to delve into deeper details of the dataset and assess the degree of variability exhibited by each numeric

TABLE I. EV POWER CONSUMPTION FEATURES

Feature name	Description	Type	
Shunt_voltage (mV)	Voltage drop that occurs across a shunt resistor of I2C Wattmeter		
Bus_voltage	DC Voltage supply	numeric	
Current_mA	EVSE-B Current consumption	numeric	
Power_mw	EVSE-B Power consumption	numeric	
State	EVCS state (idle, charging)	categorical	

TABLE II. DESCRIPTIVE STATISTICS OF NUMERIC FEATURES

	shunt_voltage	bus_voltage_V	current_mA	power_mW
count	115298	115298	115298	115298
mean	619.79	5.19	619.76	3212.78
std	197.19	0.01	197.31	1011.57
min	417	5.15	417	2160
25%	467	5.18	467	2420
50%	510	5.2	510	2660
75%	746	5.2	747	3860
max	1214	5.21	1220	6300

TABLE III. DESCRIPTIVE STATISTICS OF NUMERIC FEATURES PER BINARY CLASS

	shunt_voltage		bus_voltage_V		current_mA		power_mW	
	attack	benign	attack	benign	attack	benign	attack	benign
count	100935	14363	100935	14363	100935	14363	100935	14363
mean	631.17	539.83	5.19	5.2	631.32	538.54	3271.47	2800.39
std	204.85	99.72	0.01	0.01	204.96	98.91	1050.57	513.59
min	458	417	5.15	5.16	456	417	2360	2160
25%	467	445	5.17	5.19	467	445	2420	2320
50%	506	521	5.2	5.2	506	520	2620	2680
75%	831	593	5.2	5.2	834	591	4300	3040
max	1214	995	5.2	5.21	1220	991	6300	5180

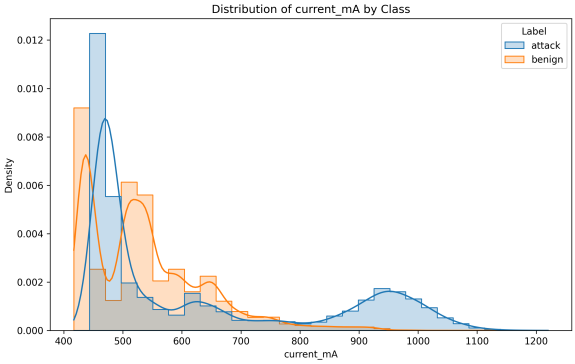

Fig. 4. Current dissipation histogram per class.

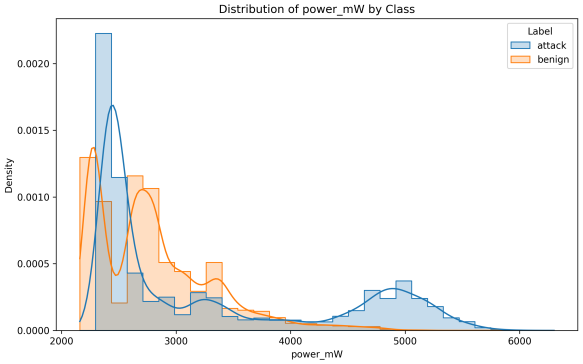

Fig. 5. Power consumption histogram per class.

feature under each attack type, Table IV and Fig. 6 illustrate descriptive statistics for shunt voltage per attack type. The confined statistics reveal distinct differences in feature values among Backdoor, cryptojacking, and syn-flood attacks. Backdoor attacks show a mean shunt voltage of 643.23 mV with a high standard deviation of 130.17, indicating significant variability. The range spans from a minimum of 466 mV to a maximum of 1149 mV, suggesting a broad distribution of shunt voltage values within this attack type. Cryptojacking attacks have a much higher mean shunt voltage of 946.59 mV, but they exhibit lower variability, as indicated by the standard deviation of 53.35. The values range from 752 mV to 1214 mV, showing a more concentrated distribution compared to Backdoor attacks. The lower standard deviation and tighter interquartile range (25% to 75%) indicate that shunt voltage values for cryptojacking attacks are more consistent. Syn-flood attacks, with a mean shunt voltage of 927.73 mV and a standard deviation of 134.21, show variability similar to Backdoor attacks. The range of shunt voltage values for syn-flood attacks spans from 474 mV to 1203 mV, indicating considerable overlap with Backdoor attacks. However, the distribution is slightly more consistent than that of Backdoor attacks but less so than cryptojacking attacks. In summary, cryptojacking attacks stand out with higher and more consistent shunt voltage values, while Backdoor and syn-flood attacks exhibit greater variability and broader ranges, resulting in a higher degree of overlap in their shunt voltage distributions.

On the other hand, Table V and Fig. 7 show descriptive

TABLE IV. DESCRIPTIVE STATISTICS FOR SHUNT VOLTAGE PER ATTACK TYPE

	Backdoor	cryptojacking	syn-flood
count	21137	11596	13517
mean	643.23	946.59	927.73
std	130.17	53.35	134.21
min	466	752	474
25%	545	911	907
50%	625	944	962
75%	724	981	1008
max	1149	1214	1203

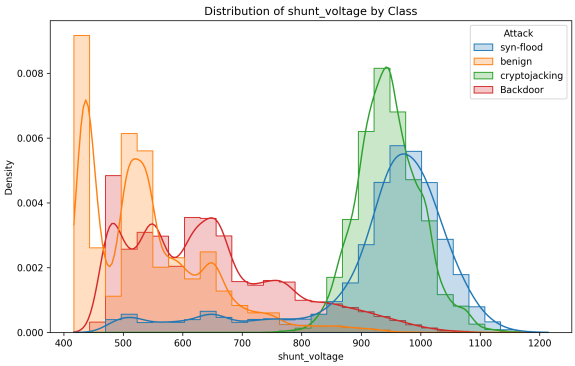

Fig. 6. Shunt voltage histogram per attack type.

statistics for bus voltage across three distinct attack types: Backdoor, cryptojacking, and syn-flood. Each attack type demonstrates a comparable degree of variability in bus voltage, as evidenced by similar standard deviations. Backdoor attacks showcase a mean bus voltage of 5.1869 (V) with a standard deviation of 0.0096, cryptojacking attacks exhibit a mean of 5.1649 with a standard deviation of 0.0037, and syn-flood attacks display a mean of 5.1647 (V) with a standard deviation of 0.0095. Despite this similarity in variability, subtle differences emerge in their respective ranges. Backdoor attacks span from 5.1530 to 5.2050, cryptojacking attacks range from 5.1490 (V) to 5.1770 (V), and syn-flood attacks span from 5.1490 (V) to 5.2010 (V). These ranges suggest overlapping distributions of bus voltage values among the different attack types, despite their comparable degrees of variability.

TABLE V. DESCRIPTIVE STATISTICS FOR BUS VOLTAGE PER ATTACK TYPE

	Backdoor	cryptojacking	syn-flood
count	21137	11596	13517
mean	5.1869	5.1649	5.1647
std	0.0096	0.0037	0.0095
min	5.1530	5.1490	5.1490
25%	5.1810	5.1610	5.1610
50%	5.1890	5.1650	5.1610
75%	5.1930	5.1690	5.1650
max	5.2050	5.1770	5.2010

Moreover, Table VI and Fig. 8 depict current values per attack type, highlighting discernible differences among Backdoor, cryptojacking, and syn-flood attacks. Backdoor attacks exhibit a mean current value of 643.97 mA with a standard deviation of 130.73 mA, indicating notable variability. The range spans from a minimum of 466 mA to a maximum of

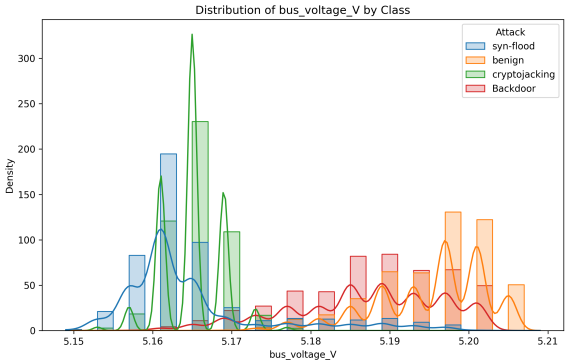

Fig. 7. Bus voltage histogram per attack type.

1101 mA, suggesting a wide distribution of current values within this attack type. Cryptojacking attacks demonstrate a significantly higher mean current value of 946.69 mA, accompanied by a lower standard deviation of 52.79 mA, implying a more consistent distribution. The values range from 753 mA to 1184 mA, showcasing a narrower spread compared to Backdoor attacks. The lower standard deviation and tighter interquartile range (25% to 75%) suggest that current values for cryptojacking attacks are more uniform. Syn-flood attacks, with a mean current value of 927.80 mA and a standard deviation of 134.25 mA, display variability akin to Backdoor attacks. The range of current values for syn-flood attacks extends from 473 mA to 1220 mA, indicating considerable overlap with Backdoor attacks. However, the distribution is slightly more consistent than that of Backdoor attacks but less so than cryptojacking attacks. In summary, cryptojacking attacks stand out with higher and more consistent current values, while Backdoor and syn-flood attacks exhibit greater variability and broader ranges, resulting in a higher degree of overlap in their current value distributions.

TABLE VI. DESCRIPTIVE STATISTICS FOR CURRENT VALUES PER ATTACK TYPE

	Backdoor	cryptojacking	syn-flood
count	21137	11596	13517
mean	643.97	946.69	927.80
std	130.73	52.79	134.25
min	466	753	473
25%	545	912	906
50%	626	945	963
75%	726	981	1007
max	1101	1184	1220

Furthermore, different attack type reveal varying patterns in their power usage characteristics, measured in milliwatts (mW) as shown in Table VII and Fig. 9. Backdoor attacks show a mean power consumption of 3335.85 mW with a standard deviation of 664.86 mW, indicating considerable variability. The range spans from a minimum of 2420 mW to a maximum of 5840 mW, suggesting a broad distribution of power consumption values within this attack type. Cryptojacking attacks exhibit a substantially higher mean power consumption of 4887.07 mW, coupled with a lower standard deviation of 273.09 mW, implying a more consistent power usage pattern. The values range from 3800 mW to 6100 mW, showcasing

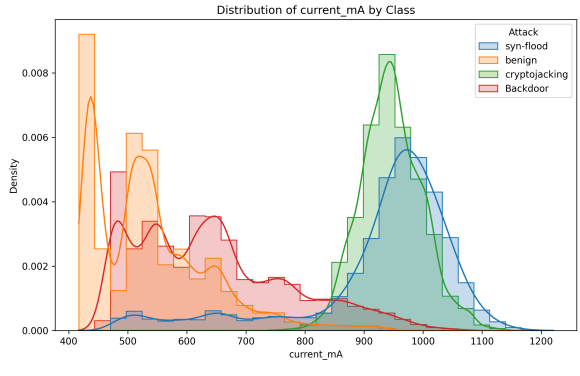

Fig. 8. Current values histogram per attack type.

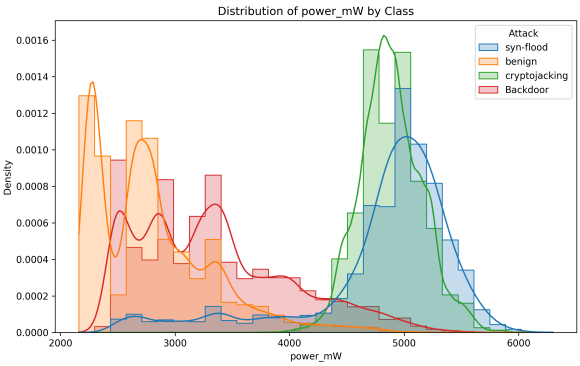

Fig. 9. Power consumption histogram per attack type.

a narrower spread compared to Backdoor attacks. The lower standard deviation and tighter interquartile range (25% to 75%) suggest that power consumption values for cryptojacking attacks are more uniform. Syn-flood attacks, with a mean power consumption of 4796.12 mW and a standard deviation of 680.34 mW, display variability similar to Backdoor attacks. The range of power consumption values for syn-flood attacks extends from 2460 mW to 6300 mW, indicating considerable overlap with Backdoor attacks. However, the distribution is slightly more consistent than that of Backdoor attacks but less so than cryptojacking attacks. In summary, cryptojacking attacks stand out with higher and more consistent power consumption values, while Backdoor and syn-flood attacks exhibit greater variability and broader ranges, resulting in a higher degree of overlap in their power consumption distributions.

TABLE VII. DESCRIPTIVE STATISTICS FOR POWER CONSUMPTION PER ATTACK TYPE

	Backdoor	cryptojacking	syn-flood
count	21137	11596	13517
mean	3335.85	4887.07	4796.12
std	664.86	273.09	680.34
min	2420	3800	2460
25%	2820	4720	4680
50%	3240	4880	4980
75%	3760	5040	5220
max	5840	6100	6300

These statistics provide nuanced insights into the distinctions in numeric features between the “attack" and “benign" classes within the dataset, suggesting potential patterns related to malicious activity.

2) Dataset filtering: Based on the information provided in the preprint paper, *Enhancing EV Charging Station Security Using A Multi-dimensional Dataset*, the dataset originally contained seven different attack classes: Cryptojacking, Backdoor, None (Benign), TCP-Port-Scan, Service-Version-Detection, OS-Fingerprinting, and Syn-flood. However, in this study, the first step in the preprocessing pipeline, is to filter the dataset to include only four specific classes: "Backdoor", "cryptojacking", "none", and "syn-flood". This selective approach is a well-reasoned decision that serves to enhance the relevance, performance, and interpretability of the machine learning models developed using this dataset. The primary justification lies in the need to tailor the dataset to the specific challenges and threats faced by EV charging infrastructure. Electric vehicles and their supporting charging ecosystem are becoming increasingly prevalent, and ensuring the cybersecurity of these systems is of paramount importance. By focusing the dataset on the most critical attack scenarios, such as backdoor intrusions, cryptojacking, and denial-of-service (syn-flood) attacks, we are aligning the data with the real-world security concerns that need to be addressed. This targeted approach to data selection serves to optimize the performance of the machine learning models trained on the CICEVSE2024 dataset. Including only the most relevant attack classes and the normal (non-attack) condition allows the models to focus on distinguishing between these key scenarios, rather than being distracted by less critical attack types. Furthermore, the decision to filter the dataset to these specific classes also simplifies the analysis of feature importance across the different attack types. When working with a comprehensive dataset that includes a wide range of attack scenarios, the assessment of which features are most significant for each class can become a complex and challenging task.

3) Features and labels encoding: Next, an encoding is applied on the state feature, which represents the charging state of the electric vehicle. So, we have chosen to encode "Idle" as 0 and "Charging" as 1. This binary encoding is a common approach when dealing with categorical variables that have a natural ordering or hierarchy. By converting the state feature to a numerical representation, that can enable the machine learning models to better understand and incorporate this important feature into their decision-making process. In addition, the encoding step is applied on the attack labels, to ensure that the proposed models can properly interpret and learn from the different types of attacks present in the dataset. This include encoding the four selected classes: "Backdoor", "cryptojacking", "none", and "syn-flood".

4) Class balancing: In the context of Power Consumption Data, the class imbalance problem is a significant challenge that needs to be addressed in order to develop effective machine learning models for detecting cyber attacks on electric vehicle charging stations. The dataset contains a disproportionately high number of normal (non-attack) instances compared to the various attack classes, such as Backdoor, cryptojacking, and syn-flood. To mitigate this issue, this work has chosen to employ the Synthetic Minority Over-sampling

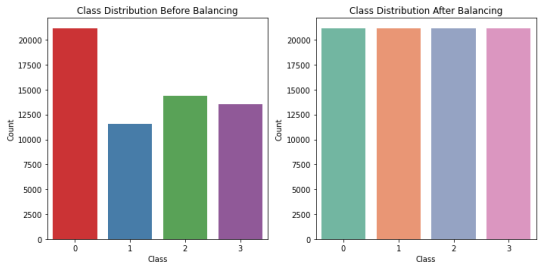

Fig. 10. Dataset class balancing.

Technique (SMOTE) to balance the class distribution. SMOTE is a powerful oversampling method that generates synthetic samples of the minority classes, helping to create a more balanced dataset [27]. The process of applying SMOTE to the Power Consumption Data dataset involves identifying the minority classes, determining the oversampling rate, generating synthetic samples, and combining the original and synthetic samples. For each minority class instance, SMOTE identifies its k nearest neighbors in the feature space and generates synthetic samples by interpolating between the minority class instance and its randomly selected nearest neighbor(s) [27], [28]. This process is repeated until the desired number of synthetic samples is generated for each minority class. By applying SMOTE to the Power Consumption dataset, that leads to effectively increasing the representation of the minority attack classes, which is crucial for training machine learning models to accurately detect these rare and anomalous events, the effect of data balncing is explained in Fig. 10. The benefits of using SMOTE in this context are twofold: it helps to prevent the machine learning models from being biased towards the majority (non-attack) class, and it can improve the models' ability to generalize and detect previously unseen attack instances. However, it's important to note that while SMOTE is a powerful technique, it also has some limitations, such as not working well for datasets with overlapping classes or high-dimensional feature spaces. Additionally, the quality of the synthetic samples generated by SMOTE can vary depending on the choice of hyperparameters, such as the number of nearest neighbors (k) to consider [28].

5) Standardizing the Features: The final step in your preprocessing pipeline is to standardize the features. Standardization is a crucial data preprocessing technique used in machine learning to ensure that all features are on a similar scale. In this work we used the scikit-learn library for this purpose. In this work the StandardScaler is employed to ensures that the features have a mean of zero and a standard deviation of one, which is often a requirement for many machine learning algorithms (e.g. linear regression, logistic regression, SVM, k-means, PCA). The StandardScaler works by subtracting the mean from each feature and then dividing by the standard deviation, as explained in the formula in Eq. 1.

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

where:

- x is the original feature value,
- μ is the mean of the feature,
- σ is the standard deviation of the feature,
- z is the scaled feature value.

This process is performed independently for each feature, ensuring that the resulting features have a mean of 0 and a standard deviation. Standardization is particularly important when working with algorithms that are sensitive to the scale of the input features, such as logistic regression, support vector machines, and neural networks. By standardizing the data, these algorithms can focus on the underlying relationships between the features and the target variable, rather than being influenced by the differences in scale. Another benefit of standardization is that it can improve the numerical stability and convergence speed of optimization algorithms used in machine learning models. This is because the standardized features have a similar range of values, which can help prevent numerical overflow or underflow issues during the optimization process.

B. Classification Techniques

1) Base classifiers: In this study, the intermediate classification strategy is used to assess individual classifiers to be used and decide to include in the ensemble methods, this involves evaluating the performance of individual classifiers using different measures. This approach helps in selecting the best-performing models to include in the final ensemble [29], [30].

a) Decision Trees (DT): are a popular machine learning algorithm used for classification tasks. DTs build a model that resembles a tree-like structure, where each internal node represents a test on a feature, each branch represents an outcome of the test, and each leaf node represents a class label. The algorithm works by recursively partitioning the feature space based on the information gain of each attribute. The attribute with the highest information gain is selected as the root node, and the process continues until a stopping criterion is met, such as reaching a maximum depth or a minimum number of samples in a leaf node. DTs are known for their interpretability and ease of visualization, making them valuable for understanding the decision-making process of the model. They can handle both numerical and categorical features and are robust to outliers and noise in the data. However, DTs can be prone to overfitting, especially when the tree grows too deep or the dataset is small [31], [32].

b) Naive Bayes (NB): is a family of probabilistic algorithms based on the Bayes theorem, which calculates the probability of an event occurring given the probability of another event that has already occurred. In the context of machine learning, Naive Bayes classifiers are used for text classification tasks, such as spam filtering, sentiment analysis, and topic modeling. The algorithm assumes that the features are independent of each other given the class label, which simplifies the computation and allows for efficient training and prediction. Despite this strong assumption, Naive Bayes classifiers often perform well in practice, especially when the features are truly independent or when the dependencies are weak. However, Naive Bayes can be sensitive to the scale of the features and may not perform well when the features

are highly correlated or when the class distributions are not Gaussian. Additionally, the algorithm assumes that the features are independent, which may not always be the case in real-world datasets [33].

c) Support Vector Machines (SVMs): are a powerful tool for both one-class and binary classification tasks, offering a flexible and robust approach to classification that can handle high-dimensional data and non-linear relationships. The SVM algorithm is based on the idea of finding the best hyperplane that separates the data into two classes, with the mathematical formulation involving minimizing the dual expression subject to constraints on the Lagrange multipliers, class labels, and regularization parameter. Key concepts include support vectors, which are the data points closest to the separating hyperplane, and the margin, which is the distance between the hyperplane and the support vectors. The choice of kernel function, such as the linear kernel, polynomial kernel, or radial basis function (RBF) kernel, is crucial in SVMs, as it transforms the data into a higher-dimensional space where the data can be separated by a hyperplane. The regularization parameter, C, controls the trade-off between the margin and the complexity of the decision boundary, with a larger value leading to a more complex decision boundary and a smaller value leading to a simpler decision boundary [34], [35].

d) K-Nearest Neighbors (KNN): is a non-parametric algorithm used for both classification and regression tasks. In the context of classification, KNN assigns a class label to a new instance based on the majority vote of its K nearest neighbors in the feature space. The algorithm works by calculating the distance between the new instance and all the training instances, typically using metrics such as Euclidean distance or Manhattan distance. The K nearest instances are then selected, and the class label with the highest frequency among these neighbors is assigned to the new instance. One of the main advantages of KNN is its simplicity and ease of implementation. KNN is also effective for multi-class classification problems and can be easily adapted to handle imbalanced datasets. However, KNN can be computationally expensive, especially when the training dataset is large or the number of features is high. It can also be sensitive to the choice of K and the distance metric used. Additionally, KNN can be affected by the curse of dimensionality, where the performance of the algorithm deteriorates as the number of features increases [32], [36].

e) Random Forest (RF): is an ensemble learning method that combines multiple decision trees to improve the accuracy and stability of predictions. RF builds a collection of decision trees, each trained on a random subset of the features. The final prediction is made by taking the majority vote of the individual trees. The algorithm works by introducing randomness at two levels: feature selection and sample selection. At each node of a decision tree, a random subset of features is considered for splitting, and the best split is chosen based on the information gain. Additionally, each tree is trained on a random subset of the training instances, obtained through a process called bagging. Random Forest inherits the interpretability and robustness of decision trees while overcoming their tendency to overfit. By combining multiple trees, RF reduces the variance of individual trees and improves the overall performance of the model. RF is known for its ability to handle high-dimensional data, missing values, and outliers. It can also provide feature importance scores, which can be useful for understanding the relative contribution of each feature to the prediction. However, Random Forest can be computationally expensive, especially when the number of trees is large or the dataset is large. It may also not perform well when the features are highly correlated or when the class distributions are imbalanced [37], [38], [39].

f) The Multilayer Perceptron (MLP): is a feedforward artificial neural network that comprises multiple layers of interconnected nodes, each layer linked to the next. In contrast to a single-layer perceptron, the MLP can learn complex non-linear relationships in data. The network architecture typically includes an input layer for data input, hidden layers for processing, and an output layer for generating predictions. During training, the MLP undergoes forward propagation, where input data is processed through the network, and the output is computed at each layer. Subsequently, the error between the predicted and actual output is calculated, initiating the backpropagation process. Backpropagation involves adjusting the weights and biases iteratively to minimize the error, enhancing the network's predictive accuracy. MLPs are known for their ability to handle high-dimensional data, learn intricate patterns, and generalize well to unseen data. Activation functions like sigmoid, tanh, or ReLU introduce non-linearity, enabling the network to model complex relationships within the data. Despite their effectiveness, MLPs can be computationally intensive, especially with large datasets or complex architectures, and may be prone to overfitting if not appropriately regularized [40], [41].

2) Homogeneous ensemble classifiers: This section presents the Homogeneous ensemble methods that utilize some of the previously mentioned methods, Decision Trees (DT), Multilayer Perceptron (MLP), Random Forest (RF), or K-Nearest Neighbors (KNN) as base estimators that form a powerful classification technique. These methods leverage the strengths of individual base estimators to enhance predictive performance and robustness. The next paragraphs outline the main methods used for homogeneous ensemble classification [30], [36].

a) Bagging (Bootstrap aggregating): is an ensemble learning method that combines multiple base models, typically decision trees, to improve the accuracy and stability of predictions. Bagging works by creating multiple subsets of the training data through a process called bootstrapping, where samples are drawn randomly with replacement. Each subset is used to train a separate base model, and the final prediction is made by aggregating the outputs of all the models, either through majority voting (for classification) or averaging (for regression). Bagging helps to reduce overfitting and improve the generalization performance of the base models by introducing randomness and reducing the variance of individual models. It is particularly effective when dealing with high-variance models like decision trees [29].

b) AdaBoost (Adaptive boosting): is an ensemble learning algorithm that combines multiple weak learners, such as decision stumps, to create a strong classifier. AdaBoost works by iteratively training base models on the training data, with each subsequent model focusing more on the instances that were misclassified by the previous models. The final prediction

is made by combining the outputs of all the base models, with each model weighted by its performance on the training data. AdaBoost is known for its ability to improve the performance of weak learners and its robustness to overfitting. However, AdaBoost can be sensitive to noisy data and outliers, and it may not perform well when dealing with imbalanced datasets or complex non-linear relationships [36].

c) Gradient boosting: is an ensemble learning method that combines multiple weak learners, typically decision trees, to create a strong predictive model. Gradient Boosting works by iteratively training base models on the residuals (the difference between the true output and the predicted output) of the previous models. The final prediction is made by summing the outputs of all the base models, each weighted by a learning rate. Gradient Boosting is known for its ability to handle a wide range of data types, including numerical, categorical, and text data. It is also effective in dealing with missing values and can provide feature importance scores [29], [30].

d) XGBoost (Extreme gradient boosting): is a highly efficient and scalable implementation of gradient boosting, which has gained popularity due to its superior performance and computational efficiency. XGBoost incorporates several optimizations, such as parallel processing, sparse data handling, and regularization, to improve the training speed and generalization performance of gradient boosting models. XGBoost has been widely used in various machine learning competitions and has achieved state-of-the-art results in many applications, such as credit card fraud detection, click-through rate prediction, and bioinformatics. Its efficiency and flexibility make it a popular choice for large-scale machine learning problems [42].

e) Extra tree: is an ensemble learning method that combines multiple extremely randomized decision trees to create a strong predictive model. Extra Tree works by introducing randomness at two levels: feature selection and split point selection. At each node of a decision tree, a random subset of features is considered for splitting, and the split point is chosen randomly within the range of the selected feature. Extra Tree is known for its ability to handle high-dimensional data, missing values, and outliers. It is also computationally efficient and can provide feature importance scores [42], [43].

f) CatBoost: is a gradient boosting framework that can handle categorical features without the need for explicit encoding. CatBoost automatically encodes categorical features using a technique called target encoding, which replaces each category with the mean of the target variable for that category. CatBoost also incorporates several other features, such as overfitting prevention, missing value handling, and GPU acceleration.

g) Hist gradient boosting: is a variant of gradient boosting that uses histogram-based decision trees to improve computational efficiency. Instead of storing the individual feature values, Hist Gradient Boosting uses a histogram-based approach to approximate the feature values, which reduces the memory footprint and speeds up the training process. Hist Gradient Boosting is particularly useful for large-scale machine learning problems and has been successfully applied in various domains, such as click-through rate prediction, recommendation systems, and bioinformatics [29], [36].

3) Heterogeneous ensemble classifiers: In order to tackle different attack scenarios, there is a need to develop robust and accurate methods for identifying and categorizing these attacks. One such approach is the use of heterogeneous classifiers, which combine the strengths of multiple classification algorithms to improve overall performance. In this section, stacking and voting are two popular ensemble methods that can be used to combine the predictions of heterogeneous classifiers. In the context of host and network attack detection, heterogeneous classifiers can be used to classify different types of attacks. For example, in the context of host and network attack detection, heterogeneous classifiers can be used to classify different types of attacks. For example, RF can be used to classify attacks based on their characteristics, such as the type of traffic and the source IP address. MLP can be used to classify attacks based on their patterns, such as the sequence of packets and the duration of the attack. KNN can be used to classify attacks based on their proximity to other attacks, such as the similarity in traffic patterns. DT can be used to classify attacks based on their decision tree structure, such as the sequence of decisions made during the attack. Combining diverse models, such as linear models, decision trees, and neural networks, is often more effective than using only one type of model. Voting and stacking are two popular ensemble techniques that can leverage this diversity to achieve superior performance. The choice between voting and stacking depends on the specific problem, the available data, and the characteristics of the base models. In general, voting is a good choice when the base models are already performing well and have different strengths, while stacking is more appropriate when the base models have room for improvement and can benefit from the meta-learner's ability to learn the optimal combination weights.

a) Voting classifiers: Classifiers aim to combine diverse models for robust predictions. Voting classifiers are a powerful ensemble learning technique that combines the predictions of multiple trained models to create a final, more robust classifier. By leveraging the strengths of diverse base models, voting classifiers can achieve superior performance compared to individual models. The key to effective voting is ensuring the underlying classifiers are sufficiently different, which is often accomplished by training them on distinct subsets of features. Soft voting allows assigning weights to each base model, while hard voting relies on majority vote. However, it's important to note that training all ensemble members on the same set of features is generally not recommended, as it can limit the diversity of the models. Instead, using different subsets of features or even different types of models, such as decision trees and random forests, can lead to more effective voting and better predictive performance [35], [44].

b) Stacking classifiers: aim for learning to optimally combine models. Stacking is another ensemble learning technique that combines the predictions of multiple base models to produce a final prediction. Unlike voting, which uses prespecified weights or majority vote, stacking employs a meta-learner to learn the optimal way to combine the base model predictions from data. This meta-learner is a higher-level model that takes the base model outputs as input features and the true labels as the target variable. By allowing the meta-learner to learn the combination weights, stacking can often outperform voting when the base models are diverse and

have different strengths and weaknesses. Stacking can improve overall performance by leveraging the unique capabilities of each base model while mitigating their individual limitations. The key steps in stacking are; first: splitting the data into training and holdout sets. Second, training the base models on the training data. Third, using the trained base models to make predictions on the holdout set. Finally, using the holdout set predictions as input features and the true labels as the target for training the meta-learner [36], [44].

C. Performance Measures

In assessing the performance of the previously outlined classification machine learning (ML) methods, it is crucial to evaluate their accuracy, recall, precision, and F1 score [45]. These metrics provide valuable insights into the model's ability to correctly classify instances, detect relevant instances, and balance between precision and recall [46].

- Accuracy is a measure of how well a model is able to correctly classify instances. It is calculated as the proportion of correctly classified instances out of the total number of instances.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

- Precision is a measure of how well a model is able to avoid false positives. It is calculated as the proportion of true positives out of the total number of instances classified as positive.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3}$$

- Recall measures the proportion of actual positive instances that are correctly identified by the model. It is calculated by dividing the number of true positives by the sum of true positives and false negatives.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4}$$

- F1 score is a harmonic mean of precision and recall, providing a balanced measure of a model's performance.

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5}$$

where:

  - TP is the number of true positives (correctly classified samples).
  - TN is the number of true negatives (correctly rejected samples).
  - FP is the number of false positives (incorrectly classified samples).
  - FN is the number of false negatives (incorrectly rejected samples).

III. Results and Discussion

A. Binary Classification Results

1) Base classifiers results: Table VIII presents the performance metrics of various classifiers for the binary classification task in which each power consumption instance is classified as either benign or attack. The Random Forest classifier outperforms others with an accuracy of 95.074%, precision of 94.890%, recall of 95.074%, and F1-score of 94.914%. The Multilayer Perceptron also shows strong performance with an accuracy of 94.436%, precision of 94.234%, recall of 94.436%, and F1-score of 94.054%. K-Nearest Neighbors and the Decision Tree exhibit solid performance metrics, while the Logistic Regression and Support Vector Machine have lower scores comparatively. The Naive Bayes classifier performs the poorest with significantly lower metrics across all categories, especially with an accuracy of 43.040% and an F1-score of 51.095%.

TABLE VIII. Performance Metrics of Various Base Classifiers for Binary Classification

Classifier	Accuracy	Precision	Recall	F1-score
Decision Tree	93.608%	93.538%	93.608%	93.571%
K-Nearest Neighbors	94.350%	94.118%	94.350%	94.173%
Logistic Regression	87.606%	84.885%	87.606%	84.983%
Multilayer Perceptron	94.436%	94.234%	94.436%	94.054%
Naive Bayes	43.040%	76.572%	43.040%	51.095%
Random Forest	95.074%	94.890%	95.074%	94.914%
Support Vector Machine	91.210%	90.615%	91.210%	89.879%

2) Ensemble methods results: Table IX presents the performance of various ensemble methods for the binary classification task, categorizing instances as either attack or benign. In terms of accuracy, CatBoost and LightGBM lead with 95.37% and 95.41%, respectively, followed closely by HistGradient Boosting at 95.29%. XGBoost and Bagging (Random Forest) also perform well, with accuracies of 95.26% and 95.19%.

TABLE IX. Performance Metrics of Various Ensemble Methods for Binary Classification

Classifier	Accuracy	Precision	Recall	F1-score
Bagging (Decision Tree)	94.63%	94.41%	94.63%	94.44%
Bagging (KNN)	94.36%	94.13%	94.36%	94.18%
Bagging (MLP)	94.21%	93.94%	94.21%	93.89%
Bagging (Random Forest)	95.19%	95.01%	95.19%	95.01%
AdaBoost (Decision Tree)	94.05%	93.80%	94.05%	93.87%
Gradient Boosting	91.11%	91.94%	91.11%	89.02%
XGBoost	95.26%	95.08%	95.26%	95.09%
Extra Trees	94.50%	94.26%	94.50%	94.25%
HistGradient Boosting	95.29%	95.12%	95.29%	95.13%
CatBoost	95.37%	95.21%	95.37%	95.21%
LightGBM	95.41%	95.25%	95.41%	95.25%
Voting Classifier	94.75%	94.53%	94.75%	94.52%
Stacking Classifier	94.39%	94.16%	94.39%	94.22%

Precision is highest for LightGBM at 95.25%, followed by CatBoost at 95.21%, and HistGradient Boosting at 95.12%. XGBoost and Bagging (Random Forest) also maintain high precision at 95.08% and 95.01%.

Recall metrics reveal that LightGBM and CatBoost excel with 95.41% and 95.37%, respectively, with HistGradient Boosting at 95.29%. Bagging (Random Forest) and XGBoost maintain high recall at 95.19% and 95.26%.

F1-scores, which balance precision and recall, are highest for LightGBM (95.25%), CatBoost (95.21%), and HistGradient Boosting (95.13%). Bagging (Random Forest) and XGBoost show strong F1-scores at 95.01% and 95.09%.

Notably, the Voting Classifier and Stacking Classifier are heterogeneous ensembles, achieving accuracies of 94.75%

and 94.39%, respectively, with the Voting Classifier showing slightly higher performance metrics. Other methods, such as Bagging and Boosting techniques, are homogeneous ensembles, demonstrating a range of high to low performance based on the classifier used. The heterogeneous ensembles, despite not having the highest individual metrics, still show competitive performance, illustrating the strength of combining diverse models.

B. Multi-Class Classification Results

1) Base Classifiers Results: Table X presents the performance metrics of selected base classification methods for a multi-class classification task, where each instance is categorized as either benign or one of three possible attack types. Among the classifiers, Random Forest shows the highest performance with an accuracy of 90.857%, precision of 90.819%, recall of 90.857%, and an F1-score of 90.815%. The K-Nearest Neighbors (KNN) classifier follows, achieving an accuracy of 88.788%, precision of 88.750%, recall of 88.788%, and an F1-score of 88.677%. The Decision Tree classifier also performs robustly with an accuracy of 87.711%, precision of 87.670%, recall of 87.711%, and an F1-score of 87.636%. The Multi-Layer Perceptron (MLP) classifier, while slightly lower in performance compared to the others, still maintains a reasonable accuracy of 81.928%, precision of 82.268%, recall of 81.928%, and an F1-score of 81.969%. Overall, Random Forest demonstrates the strongest performance across all metrics for this multi-class classification task.

TABLE X. PERFORMANCE METRICS OF BASE CLASSIFIERS FOR MULTI-CLASS CLASSIFICATION

Classifier	Accuracy	Precision	Recall	F1 Score
Random Forest	90.857%	90.819%	90.857%	90.815%
KNN	88.788%	88.750%	88.788%	88.677%
Decision Tree	87.711%	87.670%	87.711%	87.636%
MLP	81.928%	82.268%	81.928%	81.969%

2) Ensemble methods results: Table XI demonstrates the performance of various ensebmle method for multi-class classification. The ensemble methods employed in the classification task displayed varying levels of performance.

TABLE XI. PERFORMANCE METRICS OF ENSEMBLE METHODS FOR MULTI-CLASS CLASSIFICATION

Ensemble Method	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Bagging (Decision Tree)	89.716%	89.674%	89.716%	89.688%
AdaBoost (Decision Tree)	89.657%	89.605%	89.657%	89.621%
Gradient Boosting	81.490%	81.937%	81.490%	81.627%
XGBoost	86.535%	86.711%	86.535%	86.565%
Extra Trees	90.189%	90.177%	90.189%	90.177%
HistGradient Boosting	89.805%	89.860%	89.805%	89.800%
CatBoost	86.316%	86.498%	86.316%	86.347%
Stacking Classifier	91.076%	91.030%	91.076%	91.040%
Voting Classifier	90.721%	90.694%	90.721%	90.702%

Bagging, utilizing decision trees, achieved an accuracy of 89.716%, closely followed by AdaBoost, which attained 89.657%. While accuracy provides an overall measure of correctness, other metrics offer deeper insights. For instance, Gradient Boosting exhibited a lower accuracy of 81.490%, indicating comparatively weaker performance among the methods. However, its precision, recall, and F1 score values, around 81.937%, 81.490%, and 81.627%, respectively, reveal its ability to maintain a balance between true positives, true negatives, false positives, and false negatives. XGBoost demonstrated a moderate accuracy of 86.535%, with precision, recall, and F1 score values approximately 86.711%, 86.535%, and 86.565%, respectively, showcasing its effectiveness in correctly identifying both positive and negative instances. Extra Trees emerged as the top performer, achieving the highest accuracy of 90.189%. Its precision, recall, and F1 score closely matched the high accuracy, indicating robust and consistent performance across different evaluation metrics. HistGradient Boosting and CatBoost displayed similar accuracies of 89.805% and 86.316% respectively, with corresponding precision, recall, and F1 score values reflecting their performance in handling large datasets and categorical features, respectively. Among the ensemble techniques, the Stacking Classifier outperformed others, reaching an accuracy of 91.076%. Its precision, recall, and F1 score values closely mirrored the high accuracy, indicating robust performance across various evaluation metrics. Similarly, the Voting Classifier demonstrated strong performance with an accuracy of 90.721%. These results underscore the importance of considering multiple evaluation metrics when selecting appropriate ensemble methods for classification tasks, with the Stacking Classifier showcasing the highest overall performance in terms of accuracy and other key metrics.

C. Discussion

1) Binary classification results: The binary classification task aimed to differentiate between benign and malicious instances of power consumption. The evaluation of various base classifiers revealed intriguing nuances in their performance. Random Forest emerged as the standout performer, boasting an impressive accuracy of 95.074%. Its ability to construct numerous decision trees and aggregate their predictions led to robust classification, particularly effective in handling the complexity of distinguishing between benign and attack instances. Conversely, the Naive Bayes classifier exhibited starkly lower accuracy metrics, shedding light on its inherent limitations in capturing the intricacies of power consumption patterns. Transitioning to ensemble methods, we witnessed a paradigm shift in performance dynamics. CatBoost and LightGBM showcased remarkable accuracies of 95.37% and 95.41%, respectively, surpassing even Random Forest. Their gradient boosting mechanisms facilitated iterative refinement, effectively capturing subtle patterns indicative of attack behaviors. Precision, recall, and F1-score analyses further emphasized the superiority of these ensemble methods, reaffirming their efficacy in correctly classifying instances across various evaluation metrics. However, it's essential to acknowledge the interpretability trade-off inherent in these advanced ensemble methods. While they excel in predictive accuracy, the opacity of their internal mechanisms may limit interpretability, posing challenges in explaining model decisions—a crucial consideration in security-critical applications.

2) Multi-class classification results: In the context of the reference preprint of this study, in which the different classifiers are applied on CICEVSE2024 Dataset, where the focus is on detecting and classifying various types of attacks such as syn-flood, cryptojacking, and backdoor attacks this analysis evaluates the performance of several classifiers. These

classifiers include homogeneous models like Bagging (Decision Tree), AdaBoost (Decision Tree), Gradient Boosting, XGBoost, Extra Trees, HistGradient Boosting, and CatBoost. Additionally, ensemble methods such as stacking and voting ensembles are assessed. As shown in Table XI, the performance metrics considered for evaluation are Accuracy, Precision, Recall, and F1 Score.

Starting by Bagging, which is an ensemble method aimed at improving the stability and accuracy of machine learning algorithms. The Bagging classifier with Decision Trees achieved an accuracy of 0.897. The high values of Precision, Recall, and F1 Score indicate a well-balanced performance, suggesting that the model is not only accurate but also consistent in identifying both attacks and normal activities without significant bias towards any specific class. This performance demonstrates Bagging's effectiveness in creating robust models by reducing variance through aggregation.

AdaBoost combines multiple weak classifiers to form a strong classifier. The performance metrics for AdaBoost are slightly lower than Bagging, with an accuracy of 0.897. However, the difference is minimal, showing that AdaBoost is almost as effective as Bagging in this context. The similarity in performance metrics across Accuracy, Precision, Recall, and F1 Score reflects a balanced classifier. AdaBoost's iterative process of focusing on misclassified instances helps improve model accuracy, though it might not significantly outperform Bagging in this dataset.

Gradient Boosting builds models sequentially to correct the errors of its predecessors, achieved an accuracy of 0.815. Despite its lower accuracy, the Precision of 0.819 is slightly higher, suggesting that while it may miss some attacks (hence lower Recall), it is precise in the predictions it makes. The relatively lower performance could be due to the complexity and potential overfitting of Gradient Boosting to specific instances.

XGBoost demonstrated better performance than standard Gradient Boosting with an accuracy of 0.865. XGBoost's enhanced algorithm and regularization techniques often result in better performance and faster training times, which is reflected in its higher Precision and Recall compared to Gradient Boosting. The improvement highlights XGBoost's efficiency in handling the dataset's intricacies through its advanced optimization and handling of missing data. The Extra Trees classifier performed the best among all homogeneous classifiers with an accuracy of 0.902. The high Precision, Recall, and F1 Score indicate that Extra Trees is highly effective in classifying different types of attacks and normal activities. Its randomness in splitting points and selection of features might have contributed to its superior performance by reducing overfitting. This classifier's ability to generate diverse trees by randomizing splits results in a robust and accurate model.

HistGradient Boosting, which bins the data into discrete intervals to speed up computation, achieved an accuracy of 0.898. This method is particularly efficient with large datasets. Its performance metrics are very close to Bagging and Extra Trees, indicating that it is also a strong contender for classifying attacks in this dataset. The binning process helps reduce computational complexity, thereby enhancing performance without sacrificing accuracy. CatBoost designed to handle categorical features, achieved an accuracy of 0.8632. Although its performance metrics are slightly lower than XGBoost and Extra Trees, CatBoost's ability to efficiently handle categorical data might make it a preferred choice in datasets with significant categorical features. Its balanced Precision and Recall further indicate a reliable classification performance. The specialized handling of categorical variables by CatBoost results in a model that is robust and less prone to overfitting.

The Stacking Ensemble, which combines multiple models to improve performance, achieved the highest accuracy of 0.911. By leveraging the strengths of different models, stacking can often outperform individual models. The high Precision, Recall, and F1 Score indicate that this ensemble method is very effective in classifying the different types of attacks. Stacking's ability to combine different models' predictions into a meta-model enhances its accuracy and robustness. The Voting Ensemble method, which aggregates the predictions of several models, also showed strong performance with an accuracy of 0.907. The high Precision, Recall, and F1 Score suggest that this method is effective in making robust predictions. Voting, especially when using a combination of different types of classifiers, helps balance the weaknesses of individual models, leading to a reliable overall performance.

In comparing these classifiers, the ensemble methods, particularly the Stacking Ensemble, demonstrated superior performance with the highest accuracy, precision, recall, and F1 scores. Among the homogeneous classifiers, Extra Trees and HistGradient Boosting showed the best performance, indicating their effectiveness in handling the dataset's complexity. Bagging and AdaBoost showed comparable and slightly lower performance, suggesting that while boosting and aggregating can enhance performance, they might not always outperform more complex methods like Extra Trees. Overall, this analysis of various homogeneous and ensemble classifiers on the CICEVSE2024 dataset reveals that ensemble methods, particularly the Stacking Ensemble, deliver the best performance in classifying different types of attacks. These methods leverage the strengths of multiple models to achieve high accuracy, precision, recall, and F1 scores.

In summary, the binary and multi-class classification results underscored the multifaceted nature of power consumption analysis in cybersecurity. While individual classifiers showcased distinct strengths and weaknesses, ensemble methods emerged as indispensable tools for navigating the intricacies of classification tasks. By harnessing the collective intelligence of diverse models, ensemble methods transcended the limitations of individual classifiers, offering unparalleled accuracy and robustness—a testament to their pivotal role in advancing cybersecurity analytics.

IV. Conclusion

The application of machine learning techniques to cyber attack detection in electric vehicle charging stations has demonstrated significant potential. The analysis of various base classifiers and ensemble methods has provided valuable insights into the nuances of model performance in this domain.

The standout performance of the Random Forest classifier highlights the advantages of ensemble learning through the

construction of multiple decision trees. Its ability to robustly capture the complex patterns in power consumption data, distinguishing between benign and malicious instances, underscores the value of this approach. Conversely, the limitations of the Naive Bayes classifier in this context shed light on the importance of selecting appropriate models that can effectively handle the intricacies of the problem at hand. The superior performance of ensemble methods, such as CatBoost and LightGBM, further reinforces the benefits of leveraging multiple models to enhance predictive accuracy. These gradient boosting-based techniques achieved high accuracy surpassing even the strong performance of Random Forest. Their ability to iteratively refine predictions, capturing subtle indicators of attack behaviors, highlights the potential of ensemble learning in security-critical applications.

The multi-class classification results on the CICEVSE2024 dataset corroborate these findings, with the Stacking Ensemble and Voting Ensemble demonstrating the highest accuracies . These ensemble methods effectively combined the strengths of various homogeneous classifiers, including the well-performing Extra Trees and HistGradient Boosting models, to achieve robust and reliable attack detection. However, As the adoption of electric vehicles continues to grow, the need for robust and reliable cyber attack detection in charging infrastructure becomes increasingly paramount. The findings of this study underscore the significant potential of machine learning, particularly ensemble methods, in enhancing the security and resilience of these critical energy systems.

REFERENCES

[1] S. Hamdare, O. Kaiwartya, M. Aljaidi, M. Jugran, Y. Cao, S. Kumar, M. Mahmud, D. Brown, and J. Lloret, "Cybersecurity risk analysis of electric vehicles charging stations," *Sensors*, vol. 23, no. 15, p. 6716, 2023.

[2] T. Chen, X.-P. Zhang, J. Wang, J. Li, C. Wu, M. Hu, and H. Bian, "A review on electric vehicle charging infrastructure development in the uk," *Journal of Modern Power Systems and Clean Energy*, vol. 8, no. 2, pp. 193–205, 2020.

[3] Q. Zhang, H. Li, L. Zhu, P. E. Campana, H. Lu, F. Wallin, and Q. Sun, "Factors influencing the economics of public charging infrastructures for ev–a review," *Renewable and Sustainable Energy Reviews*, vol. 94, pp. 500–509, 2018.

[4] R. Kene, T. Olwal, and B. J. van Wyk, "Sustainable electric vehicle transportation," *Sustainability*, vol. 13, no. 22, p. 12379, 2021.

[5] M. Muratori, M. Alexander, D. Arent, M. Bazilian, P. Cazzola, E. M. Dede, J. Farrell, C. Gearhart, D. Greene, A. Jenn *et al.*, "The rise of electric vehicles—2020 status and future expectations," *Progress in Energy*, vol. 3, no. 2, p. 022002, 2021.

[6] S. Sachan and P. P. Singh, "Charging infrastructure planning for electric vehicle in india: Present status and future challenges," *Regional Sustainability*, vol. 3, no. 4, pp. 335–345, 2022.

[7] R. S. Levinson and T. H. West, "Impact of public electric vehicle charging infrastructure," *Transportation Research Part D: Transport and Environment*, vol. 64, pp. 158–177, 2018.

[8] K. Dimitriadou, N. Rigogiannis, S. Fountoukidis, F. Kotarela, A. Kyritsis, and N. Papanikolaou, "Current trends in electric vehicle charging infrastructure; opportunities and challenges in wireless charging integration," *Energies*, vol. 16, no. 4, p. 2057, 2023.

[9] Z. Pourmirza and S. Walker, "Electric vehicle charging station: Cyber security challenges and perspective," in *2021 IEEE 9th International Conference on Smart Energy Grid Engineering (SEGE)*. IEEE, 2021, pp. 111–116.

[10] T. Aljohani and A. Almutairi, "A comprehensive survey of cyberattacks on evs: Research domains, attacks, defensive mechanisms, and verification methods," *Defence Technology*, 2024.

[11] R. Gottumukkala, R. Merchant, A. Tauzin, K. Leon, A. Roche, and P. Darby, "Cyber-physical system security of vehicle charging stations," in *2019 IEEE Green Technologies Conference (GreenTech)*. IEEE, 2019, pp. 1–5.

[12] J. Johnson, B. Anderson, B. Wright, J. Quiroz, T. Berg, R. Graves, J. Daley, K. Phan, M. Kunz, R. Pratt *et al.*, "Cybersecurity for electric vehicle charging infrastructure," Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), Tech. Rep., 2022.

[13] M. Basnet and M. H. Ali, "Deep reinforcement learning-driven mitigation of adverse effects of cyber-attacks on electric vehicle charging station," *Energies*, vol. 16, no. 21, p. 7296, 2023.

[14] Y. Li, L. Zhang, Z. Lv, and W. Wang, "Detecting anomalies in intelligent vehicle charging and station power supply systems with multi-head attention models," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 1, pp. 555–564, 2020.

[15] G. ALMahadin, M. O. Hiari, A. H. Hussein, N. M. M. Turab, A. Alkhresheh, and M. A. B. Al-Tarawneh, "Performance evaluation of an intelligent and optimized machine learning framework for attack detection," *International Journal of Communication Networks and Information Security (IJCNIS)*, vol. 14, no. 3, p. 358–371, Dec. 2022.

[16] M. ElKashlan, M. S. Elsayed, A. D. Jurcut, and M. Azer, "A machine learning-based intrusion detection system for iot electric vehicle charging stations (evcss)," *Electronics*, vol. 12, no. 4, p. 1044, 2023.

[17] M. Basnet, "Deep learning-powered computational intelligence for cyber-attacks detection and mitigation in 5g-enabled electric vehicle charging station," Ph.D. dissertation, The University of Memphis, 2022.

[18] R. Metere, M. Neaimeh, C. Morisset, C. Maple, X. Bellekens, and R. M. Czekster, "Securing the electric vehicle charging infrastructure," *arXiv preprint arXiv:2105.02905*, 2021.

[19] J. E. Rubio, C. Alcaraz, and J. Lopez, "Addressing security in ocpp: Protection against man-in-the-middle attacks," in *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 2018, pp. 1–5.

[20] S. Roy, "Denial of service attack on protocols for smart grid communications," in *Security solutions and applied cryptography in smart grid communications*. IGI Global, 2017, pp. 50–67.

[21] M. Basnet, S. Poudyal, M. H. Ali, and D. Dasgupta, "Ransomware detection using deep learning in the scada system of electric vehicle charging station," in *2021 IEEE PES Innovative Smart Grid Technologies Conference-Latin America (ISGT Latin America)*. IEEE, 2021, pp. 1–5.

[22] L. Xuefeng and Z. Wei, "Risks of cyber threats and developing robust security protocols within electric vehicle charging infrastructure," *Journal of Sustainable Urban Futures*, vol. 12, no. 12, pp. 16–31, 2022.

[23] Y. Liu, O. Ardakanian, I. Nikolaidis, and H. Liang, "False data injection attacks on smart grid voltage regulation with stochastic communication model," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 5, pp. 7122–7132, 2022.

[24] S. Sripad, S. Kulandaivel, V. Pande, V. Sekar, and V. Viswanathan, "Vulnerabilities of electric vehicle battery packs to cyberattacks," *arXiv preprint arXiv:1711.04822*, 2017.

[25] M. ElKashlan, H. Aslan, M. Said Elsayed, A. D. Jurcut, and M. A. Azer, "Intrusion detection for electric vehicle charging systems (evcs)," *Algorithms*, vol. 16, no. 2, p. 75, 2023.

[26] E. D. Buedi, A. A. Ghorbani, S. Dadkhah, and R. L. Ferreira, "Enhancing ev charging station security using a multi-dimensional dataset: Cicevse2024," 2024.

[27] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[28] B. Larsen, "Synthetic minority over-sampling technique (smote)," *GitHub (https://github. com/dkbsl/matlab_smote/releases/tag/1.0)*, 2022.

[29] S. R. Lenka, S. K. Bisoy, R. Priyadarshini, and M. Sain, "Empirical analysis of ensemble learning for imbalanced credit scoring datasets: a systematic review," *Wireless Communications and Mobile Computing*, vol. 2022, no. 1, p. 6584352, 2022.

[30] A. Mellit and S. Kalogirou, "Assessment of machine learning and ensemble methods for fault diagnosis of photovoltaic systems," *Renewable Energy*, vol. 184, pp. 1074–1090, 2022.

[31] R. Timofeev, "Classification and regression trees (cart) theory and applications," *Humboldt University, Berlin*, vol. 54, 2004.

[32] L. I. Kuncheva, *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2014.

[33] B. Scholkopf, "Support vector machines: a practical consequence of learning theory," *IEEE Intelligent systems*, vol. 13, 1998.

[34] T. K. Nguyen and T. P. T. Pham, "Predicting bankruptcy using machine learning algorithms," *Tap chí Khoa hoc và Công nghe-Đai hoc Đà Nang*, pp. 6–9, 2018.

[35] T. J. Lucas, I. S. De Figueiredo, C. A. C. Tojeiro, A. M. G. De Almeida, R. Scherer, J. R. F. Brega, J. P. Papa, and K. A. P. Da Costa, "A comprehensive survey on ensemble learning-based intrusion detection approaches in computer networks," *IEEE Access*, 2023.

[36] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, pp. 3–42, 2006.

[37] A. Criminisi *et al.*, "Regression forests for efficient anatomy detection and localization in ct studies, sep. 20, 2010, medical computer visions. recognition techniques and applications in medical imaging."

[38] E. Mushtaq, A. Zameer, and A. Khan, "A two-stage stacked ensemble intrusion detection system using five base classifiers and mlp with optimal feature selection," *Microprocessors and Microsystems*, vol. 94, p. 104660, 2022.

[39] T.-E. Tai, S.-C. Haw, K.-W. Ng, P. Naveen, and M. Al-Tarawneh, "Performance evaluation on resolution time prediction using decision tree, random forest and extreme gradient boosting," in *2023 International Conference on Computer Applications Technology (CCAT)*, 2023, pp. 74–79.

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

[41] M. A. B. Al-Tarawneh, S. A. Al-Tarawneh, and K. S. Al-Maaitah, "Predicting processor performance using machine learning techniques: A study on spec cpu2017 benchmark suite," *International Journal of Engineering Trends and Technology*, vol. 69, no. 10, pp. 108–117, 2021.

[42] H. Aljamaan and A. Alazba, "Software defect prediction using tree-based ensembles," in *Proceedings of the 16th ACM international conference on predictive models and data analytics in software engineering*, 2020, pp. 1–10.

[43] M. A. Mim, N. Majadi, and P. Mazumder, "A soft voting ensemble learning approach for credit card fraud detection," *Heliyon*, vol. 10, no. 3, 2024.

[44] T. J. Lucas, I. S. De Figueiredo, C. A. C. Tojeiro, A. M. G. De Almeida, R. Scherer, J. R. F. Brega, J. P. Papa, and K. A. P. Da Costa, "A comprehensive survey on ensemble learning-based intrusion detection approaches in computer networks," *IEEE Access*, 2023.

[45] M. Al-Tarawneh, M. Muheilan, and Z. A. Tarawneh, "Hand movement-based diabetes detection using machine learning techniques," *International Journal on Engineering Applications (IREA)*, vol. 9, no. 4, 2021.

[46] M. A. B. Al-Tarawneh, O. Al-irr, K. S. Al-Maaitah, H. Kanj, and W. H. F. Aly, "Enhancing fake news detection with word embedding: A machine learning and deep learning approach," *Computers*, vol. 13, no. 9, 2024.