Deep Learning and Machine Learning for Enhanced Anomaly Detection in EVSE systems

MSc Research Project
MSc in Cloud Computing

Shivangi Pandey
Student ID: 23188006

School of Computing
National College of Ireland

Supervisor: Jorge Mario Cortes Mendoza

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Shivangi Pandey
Student ID:	23188006
Programme:	MSc in Cloud Computing
Year:	2025
Module:	MSc Research Project
Supervisor:	Jorge Mario Cortes Mendoza
Submission Due Date:	15/09/2025
Project Title:	Deep Learning and Machine Learning for Enhanced Anomaly Detection in EVSE systems
Word Count:	7695
Page Count:	21

Signature:	Shivangi Pandey
Date:	15/09/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Deep Learning and Machine Learning for Enhanced Anomaly Detection in EVSE systems

Shivangi Pandey
23188006
MSc in Cloud Computing

Link for Video: `https://www.youtube.com/watch?v=1WD9zO4yOaM`
Link for Colab: `https://drive.google.com/drive/folders/1Czy2G_SAyWrv4IuaZaLqkAyD1uuOocqM?usp=drive_link`

Abstract

The increasing digitization and connectivity of Electric Vehicle Charging Systems (EVCS) have made them vulnerable to a variety of cyberattacks. This research introduces a comprehensive anomaly detection framework that employs Deep Neural Networks (DNNs) and traditional Machine Learning (ML) algorithms to enhance cybersecurity in Electric Vehicle Supply Equipment (EVSE) environments. The research evaluates the performance of four individual models, Random Forest (RF), Support Vector Machine (SVM), Gradient Boosting (GB), and DNN to detect both benign and malicious communication patterns between EVSE and charging management systems. Accuracy, precision, recall, F1-score, Area Under the Curve (AUC), and execution time are used as evaluation metrics. On balanced data, the DNN model achieved the highest performance with an accuracy of 99.86 % and AUC of 0.9986, followed by RF with 87.6 % accuracy and 0.8762 AUC. GB achieved 78.4 % accuracy, while SVM showed the weakest performance with only 53.1% accuracy and 0.5312 AUC. DNN also maintained low execution time (1.53s) and robust generalization. The findings highlight the effectiveness of DNN and classical ML models in detecting anomalies within EVSE systems and provide a foundation for improving security in future EVCS deployments.

Keywords: EVCS, EVSE, Deep Neural Network, Support Vector Machine, Gradient Boosting, Random Forest.

1 Introduction

Electric Vehicle (EV) proliferation continues to take place, so there has been a subsequent increase in EV Supply Equipment (EVSE) infrastructure which has now found a central place in modern transportation networks. As the mediator between vehicle-side equipment and utility-level grid control, EVSEs exchange real-time data-thereby being vulnerable to cyber-attacks. These systems operate in the realm of critical infrastructure, any successful intrusion may trigger catastrophic consequences, such as the disruption of services, energy distribution manipulation, and breach of user data Aljohani and Almutairi (2024). Figure 1 illustrates the conceptual framework of an EVSE system, highlighting the interaction between electric vehicles, the charging station, and the utility grid. It also shows potential points of vulnerability where cyberattacks may occur during communication and energy exchange. The existing modern EV-charging network infrastructure can produce traffic volumes and degree of complexity that exceed the functionality of traditional cybersecurity tools, the rule-based intrusion detection systems, and firewalls Buedi (2024)Buedi et al. (2024).

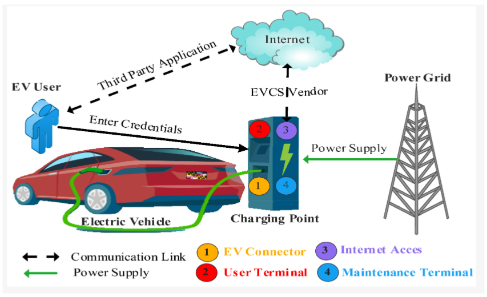


Figure 1: Conceptual framework of an EVSE system (Hamdare et al., 2023)

1.1 Problem Statement

The increasing cyber threat environment posed against electronic vehicle-charging systems has highlighted the weaknesses of traditional and standalone Artificial Intelligence (AI)-based security frameworks. Despite the high predictive power of Deep Learning (DL) techniques, their susceptibility to overfitting and reliance on large annotated datasets pose challenges in real-world deployment Makhmudov et al. (2025). Traditional Machine Learning (ML) methods, though desirable for their speed and interpretability, often struggle with the complexity of high-dimensional data and evolving attack vectors, making it difficult to accurately detect sophisticated and previously unseen anomalies Huang et al. (2025).

EVSE systems have been the target of numerous types of cyberattacks, including Denial-of-Service (DoS) attacks that flood the network with traffic to render services unavailable Rahman et al. (2024), Reconnaissance attacks that exploit protocol vulnerabilities to gather sensitive information Rahal et al. (2025), and Cryptojacking attacks where attackers hijack computing resources of EVSEs for unauthorized cryptocurrency mining Almadhor et al. (2025). Backdoor attacks, which allow persistent unauthorized access, have also been identified as critical threats that compromise the integrity of charging operations Makhmudov et al. (2025). The real-time and critical nature of EVSE operations, often integrated with national power infrastructure, demands anomaly detection solutions that are highly precise, computationally efficient, scalable, and resilient to heterogeneous and evolving threats Graf et al. (2025). As cyberattacks on EV charging infrastructure grow in frequency and complexity, developing robust detection frameworks becomes essential for securing future smart transportation systems.

1.2 Motivation

Building upon the challenges highlighted in the problem statement, this section explains the driving factors behind the need for a more resilient cybersecurity approach in EVSE systems. The consideration of EV charging stations as a component of national energy ecosystems emphasizes the dire need of those architectures, particularly in the context of the increasing number of attacks against infrastructure Martins and Rodrigues (2025). This research is driven by the goal of ensuring that the transition to sustainable transportation remains secure by addressing cybersecurity gaps through the development and evaluation of robust, AI - based anomaly detection mechanisms.

1.3 Research Question

How can a comprehensive anomaly detection framework, integrating Deep Neural Networks and traditional Machine Learning algorithms, effectively enhance cybersecurity by accurately identifying security breaches and malicious activities within the EVSE environment?

1.4 Research Aim

This research aims to enhance cybersecurity in EVSE environments by developing and evaluating a comprehensive anomaly detection framework that combines DNN architectures, traditional ML algorithms, and their integrations to identify potential security breaches or malicious activities. It will assess the effectiveness of DNN models and three ML models— Random Forest, Support Vector Machine and Gradient Boosting individually across various domains Guo et al. (2025).

1.5 Research Objectives

1. To study the latest advancements in the field of malicious communication between EVSE and charging management systems for anomaly detection model training.
2. To use a standard methodology to train and deploy the DNN, RF, SVM and GBoosting models for the detecting of anomalies.
3. To identify the hyperparameters of the models that maximise their efficiency.
4. To compare different models using standard metrics in the literature of ML (Accuracy, Precision, Recall, F1-Score, AUC Score, TPR and FPR).
5. To identify the best performing models that yield the most robust and accurate anomaly detection performance.

1.6 Research Contributions

Traditional security systems like firewalls are not enough to stop new threats. ML and DL offer better ways to detect cyber threats. This research builds a system using ML and DL to protect EVSE from attacks. It compares different models and aims to improve cybersecurity in EV charging networks.

1.7 Structure of the Report

This report is structured as follows: Chapter 1 provides an overview of the research, Chapter 2 reviews existing work and highlights the research gap, Chapter 3 describes the experimental framework, Chapter 4 details the architectural design of the proposed system, Chapter 5 presents and interprets the performance results of the individual models across selected metrics, Chapter 6 summarizes key findings, limitations, contributions, and proposes future research directions.

2 Related Work

2.1 DL - Based Approach in detecting anomalies in EVSE

DL techniques have shown significant promise in anomaly detection due to their capacity to model complex temporal and spatial patterns Hussain et al. (2024) proposed a Bi-directional Long-Short Memory (BiLSTM)-based anomaly detection framework that extracts seven statistical features from EVCS parameters such as voltage, current, frequency, and State of Charge (SoC). Evaluated on a custom Hardware-in-the-Loop (HIL) setup, the BiLSTM model achieved an accuracy of 99.42%, outperforming LSTM, Multi-Layer Perceptron (MLP), and SVM models. However, the system's dependence on real-time simulation hardware restricts its scalability in real-world deployments. Mitikiri et al. (2025) addressed replay attacks in EV charging infrastructure using a Temporal Convolutional Network Autoencoder (TCN-AE) framework. Although this model achieved a high detection rate of 99.64%, it relied on synthetic anomalies generated in MATLAB/Simulink, which may limit its generalization to real-world data. While both models achieved high accuracy, they share common limitations: dependency on controlled environments and lack of evaluation on diverse, real-world data sources.

2.2 Machine Learning Approach

ML frameworks offer flexible and interpretable solutions for EVCS anomaly detection. Shirvani and Ghorbani (2024) proposed a five-stage hybrid framework integrating rule-based profiling, supervised learning, and unsupervised detection. Tested on the ACN dataset, the framework's semi-supervised Decision Tree (DT) achieved 96.97% accuracy, though broader validation across diverse EVCS configurations remains unaddressed. Kesavan et al. (2025) introduced the Anomaly Detection with Grid Sentinel (AD-GS) framework, which combines Long Short-Term Memory (LSTM), Random Forest (RF), and Autoencoders for scalable EVCS security. The system achieved 96.8% accuracy and 99.2% operational reliability, demonstrating strong simulation performance, yet facing challenges related to high computational overhead and limited real-time adaptability. Alfardus and Rawat (2024) applied Principal Component Analysis (PCA) with Neural Networks (NN) for anomaly detection in In-Vehicle Networks (IVNs). Their model achieved 95% accuracy on the IVS-Hackathon dataset but experienced several False Positives (FPs) and False Negatives (FNs), indicating a need for enhanced robustness in real-world deployments. These studies highlight the growing role of hybrid models in achieving a balance between performance and interpretability. However, a common limitation lies in the lack of real-time deployment and limited adaptability to emerging, diverse threats. Guo et al. (2025) proposed a federated intrusion detection framework utilizing xDeepCIN optimized with Markov Chain Monte Carlo (MCMC). The system achieved an impressive F1-score of 98.9%, demonstrating strong scalability across 1,000 distributed nodes. However, performance varied under heterogeneous data conditions, highlighting the need for robust federated coordination.

Rahman et al. (2024) employed a hybrid Convolutional Neural Network (CNN)–Recurrent Neural Network (RNN) architecture validated using SHapley Additive exPlanations (SHAP), achieving 97.15% accuracy and an Area Under Curve (AUC) of 97.14%. Despite high interpretability and precision, their study was constrained by the dataset's limited attack diversity. Almadhor et al. (2025) introduced a Transfer Learning approach using pre-trained Deep Neural Networks (DNNs), obtaining 93% accuracy, but its reliance on labeled data and complex model architecture hindered real-time deployment feasibility. Buedi et al. (2024), through a physical testbed setup, applied classical ML models such as RF and SVM, achieving accuracies of 94.13% (SVM) and 93.74% (RF) with strong precision and recall values (91.51%–96.94%), affirming the dataset's practical relevance. However, they did not explore advanced deep or federated learning techniques. Bozömeroğlu and Gürkaş-Aydın (2024) utilized traditional Intrusion Detection System (IDS) techniques such as DT, SVM, and RF with oversampling and normalization, achieving over 90% accuracy (notably with DT), but their limited methodological depth and absence of fusion techniques constrained the scope of findings. Purohit and Govindarasu (2024) introduced Federated Learning for EVCS (FL-EVCS), achieving 96.97% accuracy and F1-score of 97.40%, addressing privacy concerns but falling short in processing latency estimation and client-side heterogeneity. Rahal et al. (2025) developed a Multimodal Federated Model (MFM) based on kernel event logs and network traffic, reaching 98.91% accuracy and an F1-score of 98.90%, though they noted performance degradation in highly diverse local environments.Makhmudov et al. (2025) employed a drift-aware architecture combining Adaptive RF (ARF) with ADaptive WINdowing (ADWIN), yielding 99.13% binary and 98.40% multiclass classification accuracy, with F1-score reaching 99.56%. However, its real-time capability is sensitive to data quality and incurs adaptation overhead, signaling room for optimization. A dataset split column was added in Table 1 to highlight differences in training/testing methodologies across studies. Not all works reported their split, which indicates a limitation in reproducibility.

These approaches advance EVCS security through privacy-preserving, scalable models, though challenges in client variability, latency, and generalizability remain as shown in Table 1.

Table 1: Summary of Recent Research on Anomaly Detection in EVCS/EVSE

Article	Main Goal	Technique	Metrics Used	Dataset Used	DataSet Split	Tools/ Environment	Classification Problem
Shirvani & Ghorbani (2024)	Anomaly detection	Rule-based profiling; Supervised + Unsupervised (semi-supervised DT)	Accuracy: 96.97%	ACN	Not Reported	Not specified	Binary
Kesavan et al. (2025)	Anomaly detection	LSTM; RF; Autoencoders (AD-GS)	Accuracy: 96.8%; Operational reliability: 99.2%	Simulated EVCS	Not Reported	Simulated environment	Binary
Alfardus & Rawat (2024)	IVN anomaly detection	PCA; Neural Networks	Accuracy: 95%	IVS-Hackathon	Not Reported	IVN environment	Binary
Guo et al. (2025)	Federated IDS	xDeepCIN + MCMC	F1-score: 98.9%	Not specified	Not Reported	Distributed (1000 nodes)	MultiClass
Hussain et al. (2024)	EVCS anomaly detection	BiLSTM	Accuracy: 99.42%	Custom HIL-based	70/30	Real-time simulation	Binary
Mitikiri et al. (2025)	Anomaly detection	TCN-AE + Mahalanobis	Accuracy: 95%	Simulink	Not Reported	Synthetic	MultiClass
Rahman et al. (2024)	EVCS anomaly detection	CNN-RNN + SHAP	Accuracy: 97.15%; AUC: 97.14%	CICEVSE2024	80/20	SHAP; Keras/TensorFlow (assumed)	Binary
Almadhor et al. (2025)	Transfer learning for detection	Pretrained DNNs	Accuracy: 93%	CICEVSE2024	80/20	DNN-based platform	Binary
Buedi et al. (2024)	Baseline ML benchmarking	RF; SVM; KNN; PCA	Accuracy: 94.13% (SVM); 93.74% (RF); 92.80%+	CICEVSE2024 (Testbed)	70/30	Real-world testbed	Binary
Bozömeroğlu & Gürkaş-Aydın (2024)	Traditional IDS	SVM; RF; NB; DT + Oversampling	Accuracy: 90% (DT)	Not specified	Not Reported	IDS techniques	Binary
Purohit & Govindarasan (2024)	Federated EVCS detection	FL-EVCS	Accuracy: 96.97%; F1: 97.40%	CICEVSE2024	80/20	Federated learning setup	Binary

Table 1 (continued)

Article	Main Goal	Technique	Metrics Used	Dataset Used	DataSet Split	Tools/ Environment	Classification Problem
Rahal et al. (2025)	Multimodal federated detection	MFM (Kernel + Traffic)	Accuracy: 98.91%; F1: 98.90%	CICEVSE2024	Not Reported	Federated, edge-based	MultiClass
Makhmudov et al. (2025)	Concept drift detection	ADWIN + ARF	Accuracy: 99.13%; 98.40%; F1: 99.56%	CICEVSE2024	80/20	Real-time stream processing	Binary and MultiClass
Proposed Research (2025)	EVSE anomaly detection	RF; SVM; GB; DNN	Accuracy: 87.62%; AUC: 0.8762; Time: 0.0646s, Accuracy: 53.12%; AUC: 0.5312; Time: 2.5226s, Accuracy: 78.38%; AUC: 0.7838; Time: 0.0636s, Accuracy: 100%; AUC: 0.9986; Time: 1.5370s	CICEVSE2024	80/20	Python (TensorFlow/Keras, Scikit-learn, Colab)	Binary

2.3 Research Gap and Summary

In summary, while DL, hybrid, and federated approaches show promise in anomaly detection for EVCS, critical gaps remain. Few studies explore ensemble architectures that combine strengths of deep and traditional models. Moreover, evaluation often centers on accuracy, neglecting critical metrics like FP rate (FPR) and AUC, essential for real-world trust and reliability. There is also a lack of comparative studies using identical datasets to benchmark models on equal footing. Federated models, though privacy-friendly, often suffer from performance drops due to data heterogeneity. Evaluated on the standard CICEVSE2024 dataset, the model aims to enhance generalizability, minimize false positives, and support real-time deployment—ultimately improving the cybersecurity posture of EV charging infrastructure.

3 Research Methodology

The Knowledge Discovery in Databases (KDD) methodology model (Fayyad et al., 1996) served as the framework for the data preprocessing pipeline used in this study. The chapter presents the methodology structure of designing and assessing a system for detecting anomalies.

3.1 Research Design

The Research methodology consists of four steps (i) Acquisition and preprocessing of the dataset, (ii) Training of single models, (iii) Implementing individual DL-ML models, and (iv) Performance assessment based on a variety of metrics. Traditional ML algorithms RF, SVM, and GB and DNN were trained first individually. The effectiveness and efficacy of all the algorithms were assessed through various evaluation metrics.
This research aligns with the KDD process: Selection – CICEVSE2024 dataset chosen; Preprocessing – duplicates removed, imbalance handled with SMOTE, categorical features encoded; Transformation – EDA and normalization applied; Modeling – ML (RF, SVM, GB) and DNN models trained with grid search; Evaluation – performance measured using Accuracy, Precision, Recall, F1, and AUC.

3.2 Model Training and Testing Procedure

Training of the models was initiated by training of individual models on the training segment of the preprocessed dataset. These models were traditional ML algorithms RF,SVM, and GB, and DNN with dense layers with dropout and Rectified Linear Unit (ReLU) activation. The standardized metrics were used to test all the models on the test dataset, such as Accuracy, Precision, Recall, F1-Score, AUC Score, Confusion Matrix, and Execution Time.

3.2.1 Evaluation Metrics:

To evaluate the efficacy of all algorithms, this study uses standard metrics in the literature (Makhmudov et al., 2025).

- **Accuracy**: evaluates a model's overall accuracy by calculating the ratio of correctly predicted cases to the total number of predictions.

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

where True Positives (TP) are cases where the model correctly predicted a positive class (e.g., detected anomaly), True Negatives (TN) are cases where the model correctly predicted a negative class (e.g., normal behavior), False Positives (FP) occur when the model incorrectly predicts positive for a negative case, and False Negatives (FN) occur when the model incorrectly predicts negative for a positive case (e.g., missed an anomaly).

- **Precision**: evaluates the proportion of true positives among all positive predictions, indicating how many predicted anomalies were actually correct.

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

- **Recall (or Sensitivity)**: calculates the proportion of actual anomalies that were correctly identified by the model.

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

- **F1-Score**: provides a harmonic mean of Precision and Recall, offering a balanced measure.

$$F1\text{-}Score = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

- **AUC-Score**: assesses a model's capacity to differentiate between classes at various threshold levels. A higher AUC signifies improved discriminative performance (Rahman et al., 2024).

- **Confusion Matrix**: offers a comprehensive assessment of the model's accuracy in categorising (Almadhor et al., 2025).

Table 2: Confusion Matrix

Predicted \Actual	**Positive**	**Negative**
Positive	True Positive (TP)	False Positive (FP)
Negative	False Negative (FN)	True Negative (TN)

- **ROC Curve**:The balance that exists between True Positive Rate and False Positive Rate at different levels is presented clearly.
- **Execution Time**: is the amount of time it takes to train and test out each model. The models demonstrate notable computational efficiency and scalability, essential for real-time applications in electric vehicle supply equipment (Kesavan et al., 2025).

3.3 Machine Learning Models

Each ML algorithm was selected for its relevance in anomaly detection and robustness in classification tasks.

3.3.1 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised learning algorithm that aims to find the optimal hyperplane that separates classes in a dataset with the maximum margin. The support vectors are the data points that lie closest to the hyperplane and are most critical in defining its position. SVM is particularly effective in high-dimensional spaces and performs well even when the number of dimensions exceeds the number of samples Hossen et al. (2025).

The model includes a penalty parameter (C) which controls the trade-off between achieving a low error on the training data and maximizing the margin. A higher value of C encourages the model to classify training examples correctly, while a lower value allows for a larger margin with some misclassifications.

$$\min_{\omega,b,\xi} \quad \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{n}\xi_i \quad \text{subject to} \quad y_i\left(\omega^\top \phi(x_i) + b\right) \geq 1 - \xi_i \tag{5}$$

Here, w represents the weight vector that defines the orientation of the separating hyperplane, while b is the bias term that shifts the hyperplane to better fit the data and C is Penalty Parameter and (x) denotes a kernel transformation.

In the SVM optimization formulation, ω denotes the weight vector that defines the orientation of the decision boundary, while b represents the bias term that shifts the decision boundary. The slack variables ξ_i allow certain training samples to violate the margin constraints, making the model tolerant to misclassification in the soft-margin setting. The regularization parameter C controls the trade-off between maximizing the margin and minimizing the classification error. The function $\phi(x_i)$ is a feature mapping that transforms the input data into a higher-dimensional space, enabling the use of kernel methods for non-linear classification. Finally, $y_i \in \{-1, +1\}$ are the class labels associated with each training instance.

Figure 2 shows a SVM classifier separating two classes of data points, labeled as Class A (red) and Class B (blue). The solid black line represents the hyperplane, which is the optimal decision boundary that separates the two classes. The dashed lines indicate the margins, which define the maximum distance between the hyperplane and the nearest data points of each class. These nearest data points, which lie on the margin, are known as support vectors. The

SVM algorithm maximizes this margin to achieve the best separation between classes, thereby improving the generalization ability of the model.

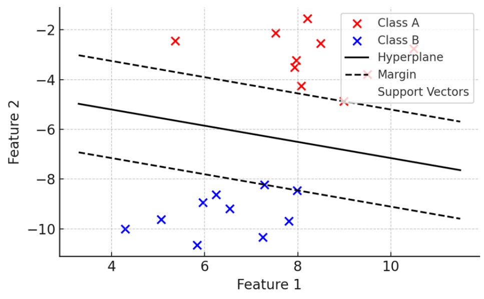


Figure 2: SVM (Original Source)

3.3.2 Random Forest (RF)

RF is an ensemble learning method that builds multiple DTs during training and aggregates their outputs through majority voting for classification tasks. Each DT is trained on a random subset of the data and considers a random subset of features when splitting nodes, which enhances model diversity and reduces correlation between individual trees (Buedi, 2024). This randomness, combined with ensemble averaging, makes RF robust to overfitting and effective for handling high-dimensional and noisy datasets. The final prediction is made by taking the class that appears most frequently among all DT outputs.

$$\text{Final predicted label } \hat{y} = \text{mode}\,(h_1(x), h_2(x), \ldots, h_T(x)) \tag{6}$$

In RF, each base classifier (usually a DT h_t) outputs a class label for the input x. The final predicted label $\hat{y}$ is determined by taking the mode (i.e., the most frequent label) across all predictions $h_1(x), h_2(x), \ldots, h_T(x)$.

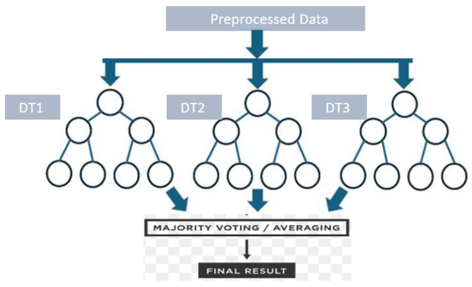


Figure 3: Random Forest (Original Source)

The Figure 3 illustrates the architecture of a RF model, which is an ensemble learning technique based on multiple DTs. The process begins with preprocessed input data, which is randomly sampled (with replacement) and independently fed into multiple decision trees labeled as DT1, DT2, DT3, and so on. Each of these decision trees is trained on a different subset of the data, introducing variability and reducing the risk of overfitting. Once all the individual decision trees make their predictions, the final output is determined through majority voting (for classification tasks) or averaging (for regression tasks). This ensemble approach improves the overall model's robustness, generalization, and accuracy compared to a single DT.

3.3.3 Gradient Boosting (GB)

GB is a powerful sequential ensemble method where each model is trained to correct the residual errors of the previous model. Unlike RF, which builds DTs independently, GB builds models in a stage-wise fashion, adding new weak learners (typically shallow decision trees) that minimize the loss function of the overall model Khaleghian et al. (2025). Each stage aims to improve the previous ensemble by focusing more on the examples that were misclassified. The learning rate determines the contribution of each new model, and the process continues until a predefined number of learners is added or performance stops improving.

Figure 4 illustrates the working principle of a GB model, which builds a sequence of weak learners (typically shallow decision trees) to improve prediction accuracy. The process begins with an initial model (Model 1) trained on the full dataset. This model's errors, known as residuals, are calculated and used as input for the next weak learner (Model 2). This iterative correction continues with each subsequent learner (Model 3, etc.) focusing on minimizing the residuals of the previous one. The final prediction is obtained by summing the outputs of all weak learners. Mathematically, this is expressed as:

$$F_m(x) = F_{m-1}(x) + \nu h_m(x) \tag{7}$$

where Fm(x) is the current model, Fm1(x) is the previous ensemble output, (x) is the new weak learner trained on residuals, and is the learning rate, which controls how much each new model contributes. This staged approach enables GB to capture complex patterns through error correction, making it a powerful tool for both classification and regression tasks.

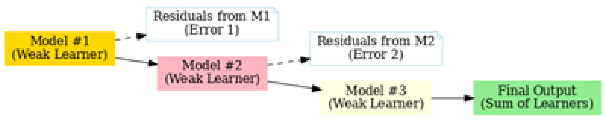

Model #1 (Weak Learner)
Residuals from M1 (Error 1)
Model #2 (Weak Learner)
Residuals from M2 (Error 2)
Model #3 (Weak Learner)
Final Output (Sum of Learners)

Figure 4: Gradient Boosting (Original Source)

3.3.4 Deep Neural Network (DNN)

A DNN consists of multiple interconnected layers of neurons that process data in a hierarchical manner. In this project, the DNN architecture includes several hidden layers, each using the ReLU activation function to introduce non-linearity, and dropout regularization to reduce overfitting by randomly disabling a fraction of neurons during training (Almadhor et al., 2025). The objective is to minimize the loss by iteratively adjusting weights to improve the model's predictive accuracy.
The forward propagation is defined as:

$$a^{(l)} = f\left(W^{(l)} a^{(l-1)} + b^{(N)}\right), \quad a^{(0)} = x \tag{8}$$

The final Output is:

$$\hat{y} = \sigma(z), \quad \text{with } \sigma(z) = \frac{1}{1 - e^{-z}} \tag{9}$$

where $a^{(l)}$ is the activation at layer l. $W^{(l)}$ and $b^{(N)}$ are the weights and biases, respectively. $\hat{y}$ is the predicted output, and $\sigma(z)$ is the sigmoid activation function.
The binary cross-entropy loss is minimized during training:

$$\mathcal{L} = -\left[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\right] \tag{10}$$

$\mathcal{L}$ is the binary cross-entropy loss used to measure the difference between the predicted probability $\hat{y}$ and the true label y Backpropagation and the Adam optimizer are used for weight updates.

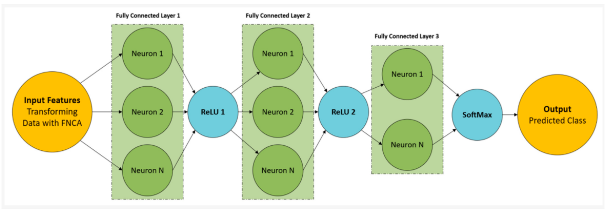


Figure 5: DNN Al-Gburi et al. (2025)

The choice of three hidden layers [128, 64, 32] in this research was based on balancing model complexity and computational efficiency. Too few layers underfit, while more layer's risk overfitting. ReLU was chosen for faster convergence and generalization.

3.3.5 Tools and Libraries

The project was implemented in Python 3.8 using Google Colab as the development environment. It utilized TensorFlow/Keras (2.18.0), Scikit-learn (1.6.1), Pandas (2.2.2), NumPy (2.0.2), Matplotlib (3.10.0), and Seaborn (0.13.2) for model development, data processing, and visualization.

4 Design Specification and its Implementation

4.0.1 Architecture of Project

Figure 6 depicts the high-level architecture of the proposed anomaly detection system for EVCS environments. The system begins with the CICEVSE2024 dataset, which undergoes preprocessing steps such as SMOTE-based oversampling, normalization, and label encoding. The data is then split into training and testing sets. Models implemented: RF, SVM, GB, and DNN. Both model categories undergo hyperparameter tuning using Grid Search Cross-Validation. Finally, all models are evaluated using standard performance metrics, including Accuracy, Precision, Recall, F1-score, AUC-ROC, Confusion Matrix, and Execution Time.

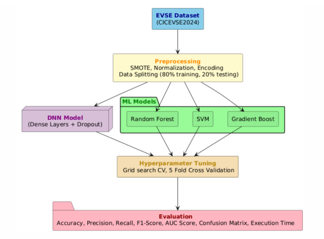

EVSE Dataset
(CICEVSE2024)
Preprocessing
SMOTE, Normalization, Encoding
Data Splitting (80% training, 20% testing)
DNN Model
(Dense Layers + Dropout)
ML Models
Random Forest
SVM
Gradient Boost
Hyperparameter Tuning
Grid search CV, 5 Fold Cross Validation
Evaluation
Accuracy, Precision, Recall, F1-Score, AUC Score, Confusion Matrix, Execution Time

Figure 6: Project Architecture (Original source)

The results from all the models are evaluated, compared, and analyzed in the final step to derive insights and identify the most effective approach for anomaly detection in EVSE environments.

4.0.2 KDD Phase - Selection

This phase involved identifying and obtaining relevant datasets for the research problem. In this case, data related to anomaly detection in EVSE environment is collected. According to the related works, the CICEVSE2024 dataset was selected because it is frequently employed in anomaly detection research studies. The dataset consists of 115,298 samples and 10 features. An initial analysis showed a severe class imbalance, with 100,935 attack samples and only 14,363 benign samples.

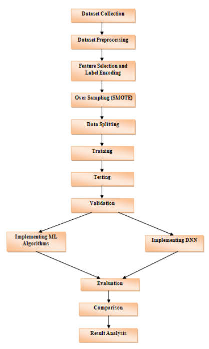
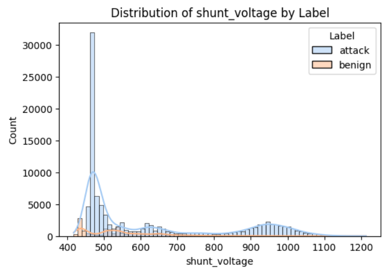


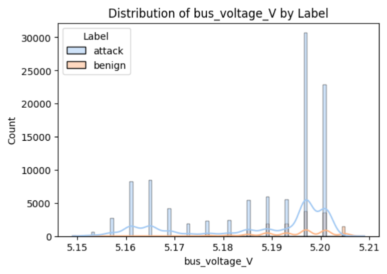
Figure 7: Project Architecture in alignment with KDD Methodologies (Original source)

4.0.3 KDD Phase - Pre-processing / Transformation

CICEVSE2024 dataset offers a realistic, high-dimensional data that includes benign and malicious events of communication between EVSE. The data comprise seven categories, consisting of Normal traffic and six attack types— DoS, Reconnaissance, Cryptojacking, Backdoor, Spoofing, and Man-in-the-Middle (MITM)—which were detected in both idle and active charging modes. It supports binary and multi-class labels and has features such as network traffic packets, power measurements and host-level logs, which makes it a perfect training anomaly detection system in electric vehicle infrastructures (Rahman et al., 2024). Preprocessing was used to clean and organize the data in preparation of the training period. These entailed missing values, duplicates, encoding categorical features and normalizing numerical data (Almadhor et al., 2025). To deal with class imbalance, oversampling was done with the Synthetic Minority Oversampling Technique (SMOTE) Ahmed et al. (2025). Data privacy and consent to use data are guaranteed because the data set utilized is publicly accessible and does not contain any personally identifiable information. Three columns were identified as irrelevant through exploratory data analysis (EDA) and feature correlation checks, which showed that they contained either constant values, identifiers (non-predictive attributes), or had no statistical relationship with the target variable; hence, they were dropped as they did not contribute to the experiment's objectives. The dataset was examined and confirmed to have no null values. A total of 38,807 duplicate entries were identified and dropped. Categorical columns were transformed into numerical format using label encoding. Due to the high-class imbalance, the SMOTE was applied to oversample the minority class (benign). This transformation resulted in a balanced dataset of 128,482 samples (64,241 attack and 64,241 benign), enabling fair and unbiased model training.
During Exploratory Data Analysis (EDA) histograms with Kernel Density Estimation (KDE) were used to visualize the distribution of numerical features by class. EDA was critical to verify dataset quality, detect imbalance, and remove irrelevant features. The KDE plots (Figures 8–10) highlighted differences between benign and attack traffic, confirming feature usefulness.



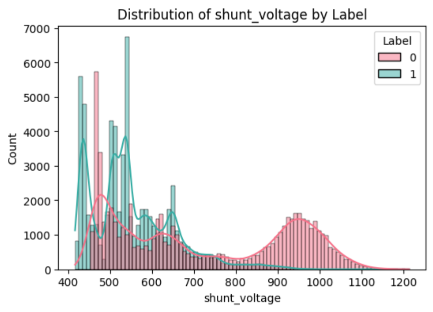
(a) Distribution of `shunt_voltage` by Label



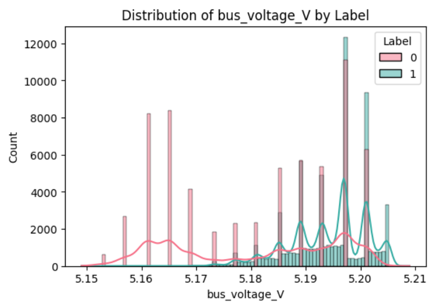
(b) Distribution of `bus_voltage_V` by Label

Figure 8: Histograms with KDE illustrating the distribution of numerical features by class.

The histograms in Figure 8 display the distribution of numerical features by label. Attack samples dominate across all features, while benign samples are significantly fewer, indicating class imbalance. The KDE highlight distinct distribution patterns.

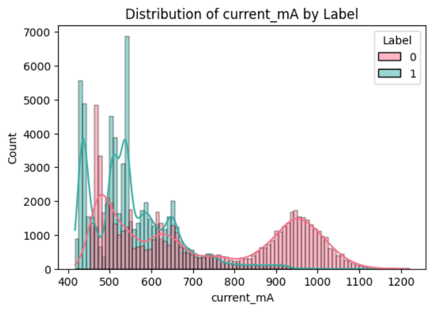


(a) Distribution of `shunt_voltage` by Label

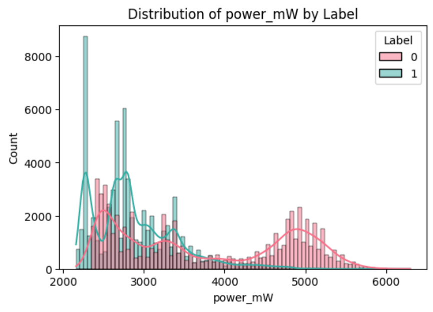


(b) Distribution of `bus_voltage_v` by Label

Figure 9: Histograms with KDE for selected features.



(a) Distribution of `current_mA` by Label



(b) Distribution of `power_mW` by Label

Figure 10: Histograms with KDE illustrating the distribution of selected numerical features grouped by the Label column in the balanced dataset.

The distribution patterns of labels across four numerical features in Figure 9 and Figure 10 are distinct, indicating that these variables could prove useful in distinguishing between normal and assault samples in the balanced dataset. Before training, the balanced dataset was split into training and testing subsets to ensure reliable evaluation.

Table 3: Distribution of splitted data

Data	**Number of Rows**
Training (80%)	102,785
Testing (20%)	25,697

4.0.4 KDD Phase - Implementing ML and DNN models

In this stage, DNN, RF, SVM, GB model are applied: DNN is implemented as a baseline DL model to capture complex patterns. It consists of multiple layers of neurons that learn from features extracted in previous stages. These algorithms are trained to detect anomalies in EVSE systems. Hyperparameter tuning was performed using GridSearchCV with 5 fold cross validation for each model to optimize performance and enhance detection of anomalies in EVSE systems. The model learns patterns from the training set and the unseen test set is used to test the trained model's ability to make accurate predictions.

4.0.5 KDD Phase - Interpretation/Evaluation

Model performance is evaluated using accuracy, precision, recall, F1 score, AUC, ROC curve, execution time and confusion matrix. These metrics provided insights into each model's ability to detect anomalies effectively. Comparative analysis is conducted based on their evaluation metrics. This helps determine which approach is most effective for anomaly detection in EVSE data. Key insights are drawn from performance gaps, misclassifications, and model behaviour. These findings are used to inform future improvements and conclusions.

5 Evaluation

In this section, we conduct a thorough assessment of RF, SVM, GB, and DNN, by employing systematic hyperparameter optimisation and analysing results to determine the most effective model.

5.1 RF, SVM and GB

The ML Algorithms along with the corresponding parameters used for model tuning, are listed in the Table 4, 5 and Table 6.

Table 4: Parameters of RF to find the best prediction model using 5CV.

Parameter	**Values**
n_estimators	(50, 100, 150, 200, 250, 300)
max_depth	(2, 3, 4, 5, 6, 7)
min_samples_split	(2, 3, 4, 5)
bootstrap	{True, False}
min_weight_fraction_leaf	[0.25, 0.3, 0.4, 0.5]

Table 5: Parameters of SVM used for model tuning.

Parameter	**Values**
C	[0.1, 0.2, 0.3, 0.4, 0.5]
kernel	{rbf, linear, sigmoid}
degree	[3, 4, 5, 6, 7]
decision_function_shape	{ovo, ovr}
tol	[1e-2, 1e-1]
max_iter	[500, 1000]

Table 6: Parameters of GB used for model tuning.

Parameter	**Values**
min_weight_fraction_leaf	[0.3, 0.35, 0.4, 0.5]
n_estimators	[50, 100, 150, 200, 250]
learning_rate	[0.1, 0.01, 0.02, 0.03, 0.04]
max_depth	[2, 3, 4, 5, 6, 7]
min_samples_split	[2, 3, 4, 5, 6]

The results of the ML models using the hyperparameters that provide the best efficiency of the model are presented in the Table 7.

Table 7: Results of RF, SVM and GB on the training and testing dataset

ML	Best Parameter	Stage	Accuracy	Precision	F1 score	AUC	Time(s)
RF	False-2-2-0.25-100	Train	**0.8767**	0.8262	0.8759	0.8767	0.2172
		Test	**0.8762**	0.8248	0.8754	0.8762	0.0646
SVM	0.5-ovo-3-rbf-1000-0.01	Train	0.5251	0.5229	0.5240	0.5251	17.604
		Test	0.5312	0.5285	0.5301	0.5312	2.5226
GB	0.1-2-2-0.3-250	Train	0.7807	0.7245	0.7772	0.7807	0.2495
		Test	0.7838	0.7270	0.7804	0.7838	0.0636

According to Table 7, the best configuration of parameters for RF, achieving **0.88 accuracy** and AUC, steady precision and F1 scores throughout training and testing. RF has the lowest execution time of 0.2172 during training with respect to other strategies. The SVM model with the configuration, was the worst model with respect to **0.5312 accuracy** and 0.5312 AUC. This suggests that the model has limited predictive power and a lengthier execution time. Performance was robust for the GB model with the configuration, as evidenced by its **0.78 accuracy** and AUC of approximately 0.78. While maintaining a lower execution time, it outperformed SVM but was marginally less effective than Random Forest.

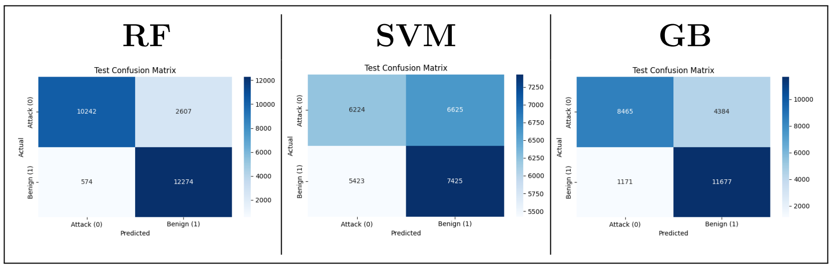


Figure 11: Confusion matrices for RF, SVM and GB on the test dataset.

The confusion matrices in the Figure 11 reveal that the RF model performed best with 12,274 TP, 10,242 TN, 2,607 FP, and 574 FN. Gradient Boosting achieved 11,677 TP, 8,465 TN, 4,384 FP, and 1,171 FN. SVM performed weakest with 7,425 TP, 6,224 TN, 6,625 FP, and 5,423 FN.

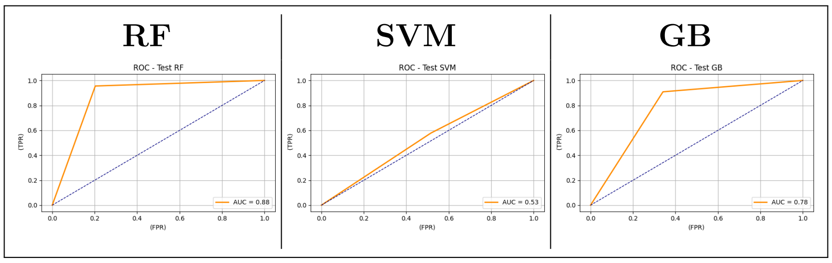


Figure 12: AUC of RF, SVM and GB on the testing dataset.

The ROC curves in the Figure 12 show that the RF model achieved the highest AUC of 0.88, indicating excellent discrimination ability. GB followed with an AUC of 0.78, showing good performance. SVM performed poorly with an AUC of just 0.53.

5.2 Deep Neural Network

The parameters that have been employed in the development of the Deep Neural Network (DNN) model in this research are summarized in the Table 8.

Table 8: Parameters of DNN used for model tuning.

Parameter	**Description / Values**
Hidden Layers	3 Layers → [128, 64, 32]
Output Layer	1 Neuron, Activation='sigmoid'
Dropout	Dropout(0.2) after each layer (Except for selu)
Kernel Initializer	lecun_normal (for selu), glorot_uniform (others)
Batch Size	64
Optimizer	Adam
Loss Function	Binary_crossentropy
Class Weight	Balanced using compute_class_weight
Early Stopping	monitor='val_loss', patience=100, min_delta=0.001
Activation Function	['relu', 'tanh', 'selu', 'elu']
Learning Rate	[0.0001, 0.0002, 0.0003, 0.0004]
Epochs	[100, 150, 200, 300]

In this, Hyperparameter tuning has been carried out in three phases:

1. Hypertuning of Activation function - relu, tanh, selu and elu
2. Hypertuning of Learning rate - 0.0001, 0.0002, 0.0003 and 0.0004
3. Hypertuning of Epochs - 100, 150, 200 and 300

Justification of DNN Structure: The proposed DNN architecture consisted of three hidden layers with 128, 64, and 32 neurons respectively and sigmoid output. This structure was selected to balance learning capacity and computational efficiency. Fewer layers (e.g., one or two) were found to underfit the dataset, whereas deeper networks introduced unnecessary complexity and risk of overfitting. The gradual reduction in neurons across layers follows a funnel-shaped design, which is widely used in anomaly detection studies to encourage hierarchical feature abstraction.

Justification of Grid Search Values: The hyperparameter ranges for GridSearchCV—such as DNN layers and neurons, RF and GB tree depth and number of estimators, and SVM kernel and regularization parameter—were carefully selected based on prior research and preliminary experiments. These thoughtfully chosen values balance exploration and efficiency, enabling each model to learn effectively, generalize well, and robustly detect anomalies in EVSE systems, while minimizing overfitting and ensuring practical computational cost.In the initial phase, four different activation functions were optimised to determine the one that produces the most accurate predictive results. After the optimal activation function has been determined, it is fixed, and four distinct learning rates are adjusted in the subsequent phase to increase model convergence. In the third phase, the optimal training without overfitting is achieved by tuning the number of epochs to four distinct values in conjunction with the optimal learning rate and activation function. The controlled optimization that is facilitated by tuning hyperparameters one at a time reduces complexity and prevents interactions that may obscure results.

In the initial phase, four different activation functions were optimised to determine the one that produces the most accurate predictive results. After the optimal activation function has been determined, it is fixed, and four distinct learning rates are adjusted in the subsequent phase to increase model convergence. In the third phase, the optimal training without overfitting is achieved by tuning the number of epochs to four distinct values in conjunction with the optimal learning rate and activation function. The controlled optimization that is facilitated by tuning hyperparameters one at a time reduces complexity and prevents interactions that may obscure results.

The evaluation results of the DNN while testing different activation functions are presented in Table 9.

Table 9: Results of DNN while testing different activation functions

Learning Rate and Epoch	Activation Function	Stage	Accuracy	Precision	F1 Score	AUC	Time (s)
0.0001 – 100	Relu	Train	0.9990	0.9981	0.9991	1.0000	10.2975
		Test	0.9986	0.9973	0.9986	1.0000	1.5370
0.0001 – 100	Tanh	Train	1.0000	1.0000	1.0000	1.0000	5.8271
		Test	1.0000	1.0000	1.0000	1.0000	1.9262
0.0001 – 100	Selu	Train	0.6948	0.9492	0.6682	0.9029	10.3064
		Test	0.6396	0.9451	0.6670	0.9007	2.6544
0.0001 – 100	Elu	Train	1.0000	1.0000	1.0000	1.0000	6.8343
		Test	1.0000	1.0000	1.0000	1.0000	2.6752

Although tanh and elu attained 100% scores, these may suggest overfitting as a result of the perfect match between the train and test data. Strong generalisation and dependable performance led to the selection of ReLU, which achieved a 99.86% test accuracy and a little lower training accuracy.

The evaluation results of the DNN while testing different learning rates are presented in Table 10.

Table 10: Results of DNN while testing different learning rates

Activation Function and Epoch	Learning Rate	Stage	Accuracy	Precision	F1 Score	AUC	Time (s)
Relu – 100	0.0001	Train	0.9999	0.9998	0.9999	1.0000	10.3062
		Test	0.9998	0.9997	0.9998	1.0000	2.6125
	0.0002	Train	0.9353	0.8873	0.9350	0.9904	7.0536
		Test	0.9342	0.8849	0.9339	0.9900	2.6199
	0.0003	Train	0.8566	0.8022	0.8555	0.9180	10.3172
		Test	0.8566	0.8014	0.8553	0.9182	1.6098
	0.0004	Train	0.9235	0.8698	0.9231	0.9882	10.3220
		Test	0.9227	0.8681	0.9223	0.9882	1.7346

Despite the fact that all learning rates generated scores that were reasonably high, the rate of 0.0001 obtained near perfect accuracy (99.98%) on the test set with continuous training results.

Therefore, the most effective learning rate for further tuning has been selected as 0.0001. Epoch 100 accomplished flawless performance with minimal execution time. The accuracy of the model experienced a minor decrease and the training time increased as the number of epochs increased. The evaluation results of the DNN while testing different Epochs are presented in Table 11.

Table 11: Results of DNN while testing different Epochs

Activation Function and Learning Rate	**Epoch**	**Stage**	**Accuracy**	**Precision**	**F1 Score**	**AUC**	**Time (s)**
Relu – 0.0001	100	Train	1.0000	1.0000	1.0000	1.0000	6.7863
		Test	1.0000	1.0000	1.0000	1.0000	2.6164
	150	Train	0.9996	0.9991	0.9996	1.0000	5.8431
		Test	0.9994	0.9988	0.9994	1.0000	1.5181
	200	Train	0.9998	0.9996	0.9998	1.0000	6.0627
		Test	0.9997	0.9995	0.9997	1.0000	2.6248
	300	Train	0.9690	0.9416	0.9690	1.0000	10.3613
		Test	0.9666	0.9373	0.9665	1.0000	2.6656

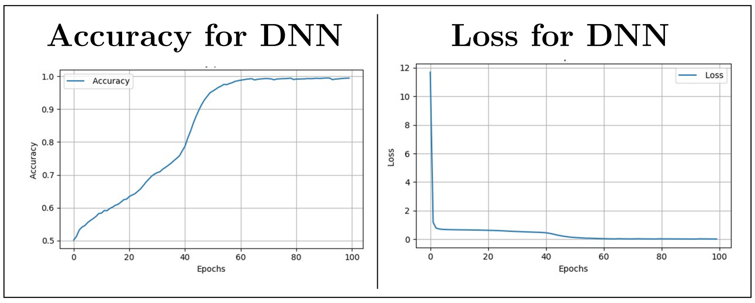


Figure 13: Accuracy and Loss plot of the Training of the best DNN model with the configuration Relu, learning rate = 0.0001, 100 Epoch and adam optimiser.

The loss and accuracy plots of the DNN model with the configuration relu-0.0001-100-adam which scored 100% accuracy and AUC in both training and testing is shown in the above Figure 13. Overall, in this section, the DNN model configured with the Adam optimizer, a learning rate of 0.0001, ReLU activation function, and 100 epochs is selected as the best-performing model with high testing accuracy for the balanced dataset.

5.3 Results and Discussion

RF, SVM, GB, and DNN models were assessed for anomaly detection in this experiment, which was designed to improve cybersecurity in EVSE environments. The RF model was the most reliable choice, as it exhibited strong predictive performance with a testing accuracy of 0.88 and the lowest execution time (0.2172 s). The GB model also performed well, achieving an accuracy of 0.78 while maintaining low computational overhead. In contrast, the SVM model, despite its structured configuration, demonstrated a significant underperformance, with an accuracy and AUC of only 0.5312, indicating limited effectiveness.

Table 12: Comparative Analysis of RF, SVM, GB and DNN Models

Model	Accuracy	Precision	Recall	F1-Score	AUC	Execution Time (s)
RF	0.8762	0.88	0.87	0.87	0.8762	0.0646
SVM	0.5312	0.54	0.53	0.52	0.5312	2.5226
GB	0.7838	0.78	0.78	0.78	0.7838	0.0636
DNN	1.0000	1.00	1.00	1.00	0.9986	1.5370

The ReLU activation function, Adam optimiser, learning rate of 0.0001, and 100 epochs were employed by the Deep Neural Network (DNN) to achieve 100% accuracy, surpassing all other models. As the most effective model, it was affirmed by its high-test performance and robust generalisation, which were achieved without overfitting. Overall, the DNN model is recognised as the most effective approach for the detection of anomalies in EVSE cybersecurity systems.. Although test accuracy was 100%, steps such as dropout, class weighting, and early stopping were applied to reduce overfitting risk.

5.3.1 Comparison of results with previous research works

A comparison with prior studies that employed the same dataset is provided in Table 13.
The DNN exhibited superior performance across all evaluation metrics, surpassing the efficacy of previous approaches in the detection of anomalies.
Moreover, this research achieved 100% testing accuracy because the test data was strictly separated and never used during training. This high accuracy is not due to overfitting because proper data preprocessing was applied to remove noise and irrelevant patterns. Suitable parameters were chosen using hyper parameter tuning to avoid overfitting. The model was evaluated using unseen data, ensuring fair testing. The dataset features were highly informative and well-structured, making patterns easier to learn. These steps ensured the models learned general patterns, not just memorized the data.

Table 13: Comparison of results with previous research works

Author	Model	Accuracy	Classification
Rahman et al. (2024)	CNN-RNN+SHAP	97.15%	Binary
Almadhor et al. (2025)	DNN	93%	Binary
Makhmudov et al. (2025)	RF+ADWIN	99.13%	Binary and Multi
Rahal et al. (2025)	MFM	98.91%	Multiclass
Buedi et al. (2024)	RF, SVM	93.74%, 94.13%	Binary
Bozömeroğlu & Gürkaş-Aydın (2024)	DT	> 90%	Binary
Purohit & Govindarasu (2024)	FL-EVCS	96.97%	Binary
Proposed Research	RF, SVM, GB, DNN	**87.62%, 53.12%, 78.38%, 100% testing accuracy**	Binary

6 Conclusion and Future Work

In order to enhance cybersecurity in EVSE environments, this research proposed and assessed a variety of machine learning models for anomaly detection. The DNN was the most effective model among the ones considered, attaining 100% testing accuracy through systematic hyper-parameter tuning and a robust model architecture that utilised ReLU activation, the Adam optimiser, and a learning rate of 0.0001 across 100 epochs. The efficacy of the model was superior to that of conventional models such as RF (0.8762 accuracy), GB (0.7838), and SVM (0.5312). A comparative analysis of the model with the existing literature further substantiated its superiority. At the same time, this investigation is subject to certain constraints. External validation with distinct or evolving datasets was not implemented, and the dataset may not provide a comprehensive representation of all real-world conditions, despite its high accuracy. In addition, the model's efficacy in live deployment environments or against adversarial attacks has not been evaluated. The framework could be extending to real-time anomaly detection using streaming EVSE traffic in future work. Furthermore, the integration of explainable AI (XAI) methodologies would enhance the transparency and confidence in decision-making. Future research may also investigate federated or edge learning models to facilitate distributed, safeguarding privacy security measures in smart charging infrastructure, which have significant commercial potential in the expanding EV ecosystem.

References

Ahmed, M. A. O., AbdelSatar, Y., Alotaibi, R. and Reyad, O. (2025). Enhancing internet of things security using performance gradient boosting for network intrusion detection systems, *Alexandria Engineering Journal* **116**: 472–482.

Al-Gburi, S., Al-Sammak, K., Ion, M., Oprea, C., Drăgulinescu, A.-M., Suciu, G., Ali Alheeti, K. M., Alduais, N. and Al-Sammak, N. (2025). Introducing a novel fast neighbourhood component analysis–deep neural network model for enhanced driver drowsiness detection, *Big Data and Cognitive Computing* **9**: 126.

Alfardus, A. and Rawat, D. B. (2024). Machine learning-based anomaly detection for securing in-vehicle networks, *Electronics* **13**: 1962.

Aljohani, T. and Almutairi, A. (2024). A comprehensive survey of cyberattacks on evs: Research domains, attacks, defensive mechanisms, and verification methods, *Defence Technology* **42**: 31–58.

Almadhor, A., Alsubai, S., Bouazzi, I., Karovic, V., Davidekova, M., Hejaili, A. and Sampedro, G. (2025). Transfer learning for securing electric vehicle charging infrastructure from cyber-physical attacks, *Scientific Reports* **15**: 9331.

Bozömeroğlu, H. and Gürkaş-Aydın, Z. (2024). Cybersecurity of electric vehicle charging stations: Performance analysis of ids systems with cic-evse 2024 dataset, *2024 17th International Conference on Information Security and Cryptology (ISCTürkiye) IEEE*, pp. 1–6.

Buedi, E. D. (2024). Enhancing ev charging station security: A multi-stage approach.

Buedi, E., Ghorbani, A., Dadkhah, S. and Ferreira, R. (2024). *Enhancing EV Charging Station Security Using a Multi-dimensional Dataset: CICEVSE2024*, pp. 171–190.

Graf, J., Moser, C., Fuxen, P. and Hackenberg, R. (2025). Intrusion detection using peer-to-peer distributed context-information for electric vehicle supply equipment, *CLOUD COMPUTING 2025*, p. 57.

Guo, J., Wang, Z., Guo, Y. and Jiang, H. (2025). Anomaly detection of controllable electric vehicles through node equation against aggregation attack, *Computers, Materials & Continua* **82**(1).

Hossen, M., Sarker, M., Al Qwaid, M., Ramasamy, G. and Eng Eng, N. (2025). Ai-driven framework for secure and efficient load management in multi-station ev charging networks, *World Electric Vehicle Journal* **16**(7): 370.

Huang, H., Wang, P., Pei, J., Wang, J., Alexanian, S. and Niyato, D. (2025). Deep learning advancements in anomaly detection: A comprehensive survey, *arXiv preprint arXiv:2503.13195* .

Hussain, A., Yadav, A. and Ravikumar, G. (2024). Anomaly detection using bi-directional long short-term memory networks for cyber-physical electric vehicle charging stations, *IEEE Transactions on Industrial Cyber-Physical Systems* .

Kesavan, V., Hossen, M., Gopi, R. and Joseph, E. (2025). Anomaly detection with grid sentinel framework for electric vehicle charging stations in a smart grid environment, *Scientific Reports* **15**(1): 1–30.

Khaleghian, S., Doan, T., Knox, J., Harris, A. and Sartipi, M. (2025). Data-driven insights into ev charging patterns: Machine learning models reveal key predictors of station utilization in tennessee, *IEEE Access* .

Makhmudov, F., Kilichev, D., Giyosov, U. and Akhmedov, F. (2025). Online machine learning for intrusion detection in electric vehicle charging systems, *Mathematics* **13**(5): 712.

Martins, J. and Rodrigues, J. (2025). Intelligent monitoring systems for electric vehicle charging, *Applied Sciences* **15**(5): 2741.

Mitikiri, S., Srinivas, V. and Pal, M. (2025). Methodology for detecting energy anomalies due to multi-replay attacks on electric vehicle charging infrastructure, *arXiv preprint arXiv:2504.00319* .

Purohit, S. and Govindarasu, M. (2024). Fl-evcs: Federated learning based anomaly detection for ev charging ecosystem, *2024 33rd International Conference on Computer Communications and Networks (ICCCN)*, IEEE, pp. 1–9.

Rahal, R., Korba, A. and Ghamri-Doudane, Y. (2025). Fuse and federate: Enhancing ev charging station security with multimodal fusion and federated learning, *arXiv preprint arXiv:2506.06730* .

Rahman, M., Chayan, M., Mehrin, K., Sultana, A. and Hamed, M. (2024). Explainable deep learning for cyber attack detection in electric vehicle charging stations, *Proceedings of the 11th International Conference on Networking, Systems, and Security*, pp. 1–7.

Shirvani, S. and Ghorbani, A. (2024). A study of ev-evse ecosystem integrity: Machine learning based security monitoring of charging sessions, Available at SSRN:4711137.