Volume: 13 Issue: IX Month of publication: September 2025

DOI: https://doi.org/10.22214/ijraset.2025.74093

Securing an Electric Vehicle Using a Predictive Model

Akshatha D [1], Dr. Girish L
[1]M.Tech Scholar, Department of CSE, Shridevi Institute of Engineering and Technology Tumkur
[2]Professor,Department of AI & DS, Shridevi Institute of Engineering and Technology Tumkur

Abstract: ***This study presents an efficient approach to enhance the security of electric vehicles (EVs) using interpretable machine learning (IML). Each EV communication activity is a combination of patterns generated by various control systems and data exchanges. Threat detection faces challenges because these patterns evolve with time, and they are described by specific statistical and behavioral features. These features serve as the inputs for machine learning algorithms. Multiple classifiers are proposed for detecting cyberattacks in EV environments, and their performance is improved through advanced tuning techniques. To ensure transparency in decision-making, interpretability methods are employed to explain feature contributions and detection outcomes. The integration of interpretability highlights the most influential features driving classification, thereby improving trust, usability, and applicability of the proposed framework in real-world EV cybersecurity.***

Keywords: Interpretable Machine Learning. Multiple Classifiers.

I. INTRODUCTION

The rapid growth of electric vehicles (EVs) driven by sustainability and digital connectivity has introduced new security challenges that demand advanced solutions. Modern EVs rely heavily on artificial intelligence (AI) and connected systems, making them vulnerable to cyberattacks. Intrusion Detection Systems (IDS) powered by machine learning (ML) and deep learning (DL) have emerged as effective tools for identifying anomalies and unauthorized access within vehicle networks. By analyzing communication patterns and network traffic, ML-based IDS can detect abnormal behaviors and issue proactive alerts to mitigate risks. In this study, a machine learning-based intrusion detection approach is proposed, utilizing feature selection to improve model compatibility and accuracy.

To ensure transparency, SHAP and Partial Dependence Plots are employed to explain feature importance and interpret model decisions. A comparative evaluation of multiple ML algorithms is conducted, and the results demonstrate improved detection performance, offering a robust solution for strengthening EV cybersecurity

A. Problem statement

With the growing adoption of electric vehicles (EVs), traditional security measures often struggle to keep pace with rapidly evolving threats and lack transparency in detection. This project proposes an interpretable machine learning (IML) model that not only identifies cyberattacks with high accuracy but also explains its decisions. By providing clear insights into threat detection, the model enables security teams to respond more effectively and confidently, strengthening the overall cybersecurity of EV environments.

B. Objectives

1) Develop a machine learning-based intrusion detection system to analyze simulated EV network traffic and accurately identify cybersecurity threats.
2) Employ interpretable machine learning (IML) methods such as SHAP and Partial Dependence Plots to explain predictions and enhance transparency.
3) Validate the system's ability to detect diverse intrusion types, including Denial-of-Service, Fuzzers, and Remote Access attacks in EV communication scenarios.
4) Recommend the integration of the interpretable detection framework into future EV security architectures to improve resilience against evolving threats.
5) Advance research in electric vehicle cybersecurity and explainable AI by presenting a hybrid, intelligent, and interpretable intrusion detection approach.

C. Proposed System

This study proposes a proactive, AI-driven intrusion detection framework tailored for electric vehicle (EV) environments, overcoming the limitations of traditional reactive security systems. Using the UNSW-NB15 dataset, the system applies preprocessing, outlier filtering, and SMOTE resampling to ensure balanced learning across attack categories. Machine learning and deep learning models are trained and evaluated using accuracy, precision, recall, and F1-score, achieving reliable detection of both common and rare threats. A key contribution is the integration of interpretable machine learning techniques, including SHAP, LIME, and Partial Dependence Plots (PDPs), which provide transparent insights into model decisions. This enhances trust, enabling EV manufacturers, fleet operators, and cybersecurity teams to not only detect intrusions but also understand the reasoning behind them. The framework offers a scalable, intelligent solution that strengthens EV security and supports future integration with adaptive policies and telemetry data.

II. LITERATURE REVIEW

Cybersecurity attacks on computer networks are a potentially catastrophic adverse outcome and one that can cause substantial disruption to organizations. Anticipating malicious activity before it leads to critical damage allows at-risk systems to be identified. Abnormalities in network behavior, traffic patterns, or protocol usage without appropriate monitoring can lead to intrusions. Intrusion Detection Systems (IDS) have been widely adopted as a useful tool to identify suspicious traffic that requires immediate attention. IDS relies on routinely captured network traffic data, which includes measurements of flow duration, packet length, source and destination addresses, service type, and protocol behavior. IDS is a practical, effective tool that can be applied to safeguard modern digital infrastructures.

Several studies have shown that traditional signature-based IDS approaches cannot detect novel attacks, while anomaly-based systems provide better generalization by identifying unknown threats. Each improvement in feature engineering, model selection, and ensemble methods has been shown to increase detection accuracy and reduce false alarms. Machine learning techniques such as Random Forest, XGBoost, K-Nearest Neighbors, and Deep Learning models like LSTM have been proposed as effective classifiers in intrusion detection.

Over the past years, benchmark datasets such as KDD’99, NSL-KDD, and more recently UNSW-NB15 have been widely used for validating IDS performance. The limitation of many studies, however, is that experiments are often performed on a single dataset, which raises questions about their applicability to real-world, large-scale networks. Furthermore, issues such as class imbalance, dataset bias, and lack of updated attack patterns limit the effectiveness of models. These challenges highlight the need for robust, adaptive, and scalable IDS frameworks.

III. SYSTEM ANALYSIS

A. Proposed System

The proposed system is a machine learning–driven intrusion detection framework designed specifically for electric vehicle (EV) environments to overcome the limitations of conventional reactive security systems. Built on the UNSW-NB15 dataset, the framework begins with extensive preprocessing that includes handling missing values, encoding categorical data, boxplot-based outlier filtering, and correlation-based feature reduction. To address class imbalance, SMOTE is applied, ensuring that both common and rare attack categories are effectively represented for training.

The system employs multiple machine learning and deep learning models, each evaluated on balanced data using accuracy, precision, recall, and F1-score to guarantee reliable performance across diverse attack types. Unlike black-box systems, this approach emphasizes interpretability. SHAP values highlight feature contributions at a global and instance level, LIME provides local explanations for individual predictions, and Partial Dependence Plots reveal complex feature interactions. Together, these explainability tools enable transparent detection, allowing cybersecurity teams, EV manufacturers, and fleet operators to trust and act on alerts with confidence.

Beyond detection, the system facilitates real-time, proactive monitoring of EV communication networks, enabling early intervention against threats such as DoS, fuzzing, or unauthorized access attempts. By uniting robust predictive performance with explainable AI, the proposed system strengthens EV cybersecurity while also creating a scalable foundation for future enhancements like adaptive policy learning, contextual awareness, and integration with vehicle telemetry data.

IV. SYSTEM DESIGN

A. System Architecture

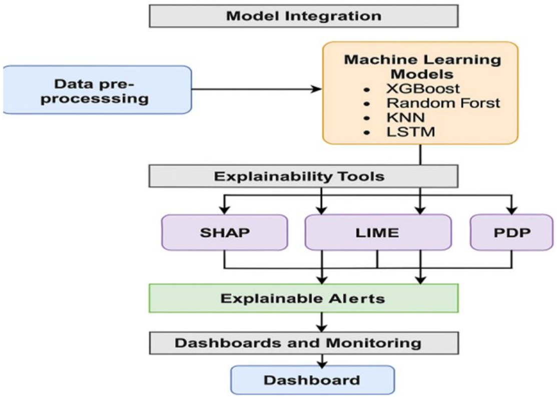


Fig.1: IDS-protected vehicle architecture

As shown in Fig. 1, the proposed IDS uses the UNSW-NB15 dataset with preprocessing steps including missing value handling, feature encoding, scaling, and outlier removal. Class imbalance is addressed using SMOTE, and models (KNN, RF, XGBoost, LSTM) are trained with hyperparameter tuning. Performance is evaluated using accuracy, precision, recall, and F1-score, with XGBoost achieving the best results. To ensure transparency, SHAP, LIME, and PDP are applied for interpretability, enabling real-time and explainable intrusion detection.

B. System Methodology

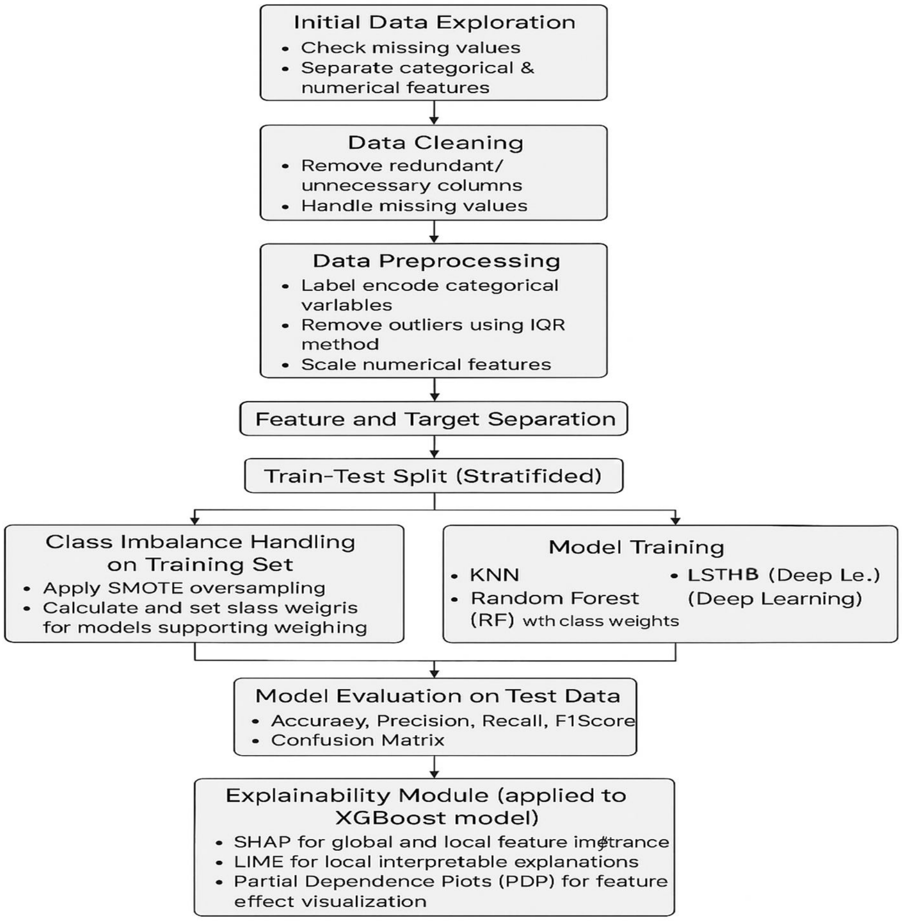
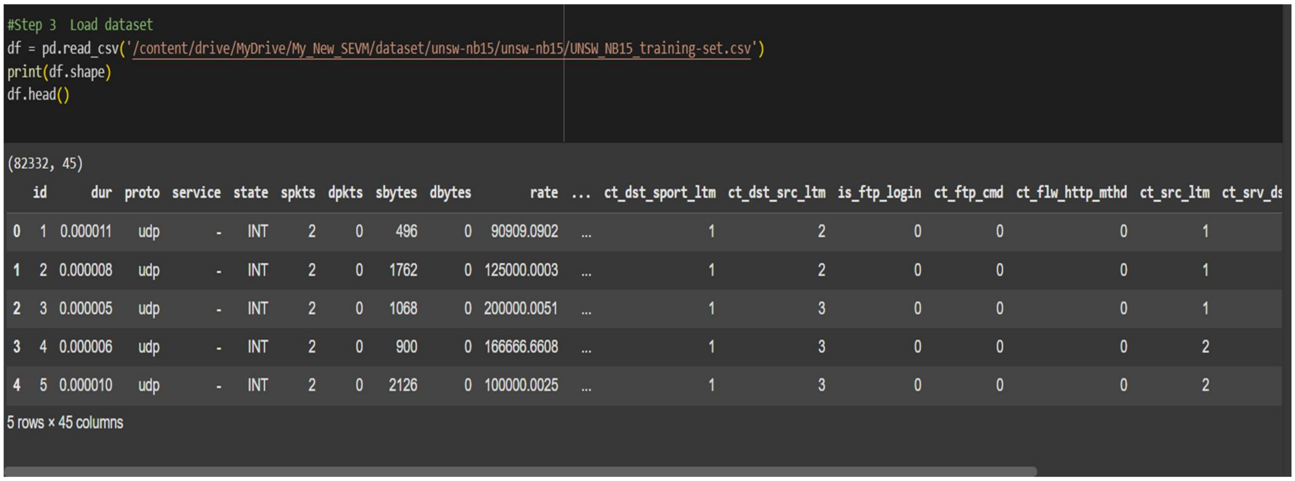

Initial Data Exploration
Check missing values
Separate categorical & numerical features
Data Cleaning
Remove redundant/ unnecessary columns
Handle missing values
Data Preprocessing
Label encode categorical varlables
Remove outliers using IQR method
Scale numerical features
Feature and Target Separation
Train-Test Split (Stratifided)
Class Imbalance Handling on Training Set
Apply SMOTE oversampling
Calculate and set slass weigris for models supporting weighing
Model Training
KNN
Random Forest (RF) wth class weights
LSTHB (Deep Le.) (Deep Learning)
Model Evaluation on Test Data
Accuraey, Precision, Recall, F1Score
Confusion Matrix
Explainability Module (applied to XGBoost model)
SHAP for global and local feature imǿtrance
LIME for local interpretable explanations
Partial Dependence Piots (PDP) for feature effect visualization

Fig.2: System Methodology

As shown in Fig.2, the system methodology begins with dataset loading, exploration, and cleaning, followed by feature encoding, scaling, and outlier removal. The data is split into training and testing sets, with SMOTE and class weighting used to address imbalance. Models (KNN, RF, XGBoost, LSTM) are trained with grid search and cross-validation for optimal F1-score. Performance is assessed using accuracy, precision, recall, and confusion matrices. Finally, explainability tools (SHAP, LIME, PDP) are applied to the best model, ensuring transparent predictions for real-time deployment.

C. Dataset Loading



```
#Step 3  Load dataset
df = pd.read_csv('/content/drive/MyDrive/My_New_SEVM/dataset/unsw-nb15/unsw-nb15/UNSW_NB15_training-set.csv')
print(df.shape)
df.head()
```

```
(82332, 45)
   id       dur proto service state spkts dpkts sbytes dbytes         rate ... ct_dst_sport_ltm ct_dst_src_ltm is_ftp_login ct_ftp_cmd ct_flw_http_mthd ct_src_ltm ct_srv_ds
0   1  0.000011   udp       -   INT     2     0    496      0   90909.0902 ...                1              2            0          0                0          1
1   2  0.000008   udp       -   INT     2     0   1762      0  125000.0003 ...                1              2            0          0                0          1
2   3  0.000005   udp       -   INT     2     0   1068      0  200000.0051 ...                1              3            0          0                0          1
3   4  0.000006   udp       -   INT     2     0    900      0  166666.6608 ...                1              3            0          0                0          2
4   5  0.000010   udp       -   INT     2     0   2126      0  100000.0025 ...                1              3            0          0                0          2
5 rows × 45 columns
```

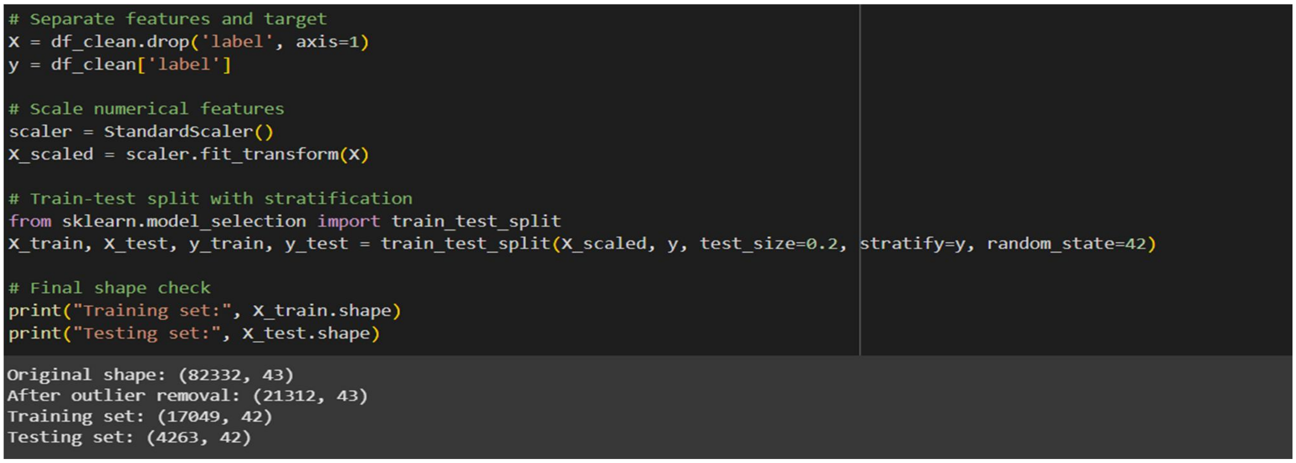
Fig.3: Dataset Loading

As shown in Fig.3, the UNSW-NB15 training dataset is loaded from Google Drive using pandas.read_csv(), resulting in 82,332 records with 45 features. A preview with df.head() confirms labeled network traffic data containing both normal and attack instances, prepared for preprocessing and analysis.

D. Data Cleaning and Transformation

Irrelevant and redundant features were removed, missing values were handled, and categorical variables were label encoded. Numerical features were normalized, and outliers were detected and removed using the IQR method to reduce noise.

E. Feature and Target Separation



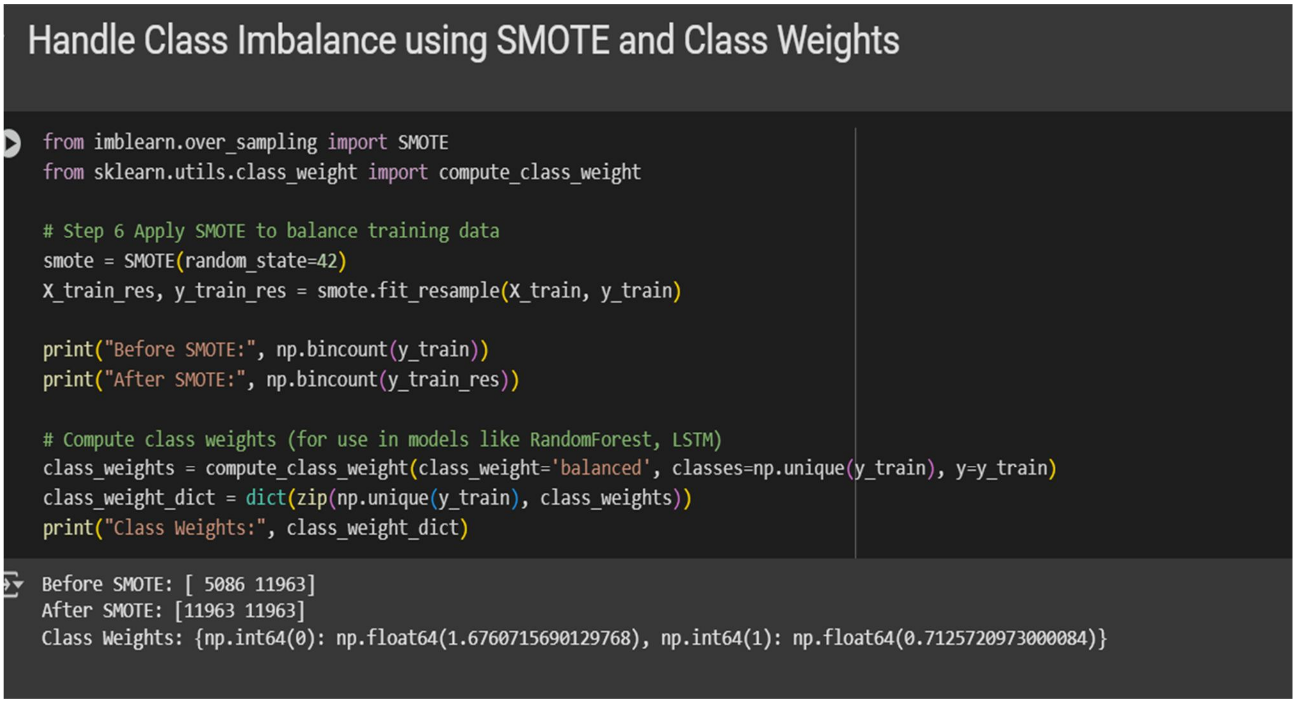
```
# Separate features and target
X = df_clean.drop('label', axis=1)
y = df_clean['label']

# Scale numerical features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train-test split with stratification
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, stratify=y, random_state=42)

# Final shape check
print("Training set:", X_train.shape)
print("Testing set:", X_test.shape)
```

```
Original shape: (82332, 43)
After outlier removal: (21312, 43)
Training set: (17049, 42)
Testing set: (4263, 42)
```

Fig.4: Feature and Target Separation

The dataset was split into input features and target labels, preparing it for model training as shown in the Fig.4 .

F. Class Imbalance Handling



Handle Class Imbalance using SMOTE and Class Weights

```
from imblearn.over_sampling import SMOTE
from sklearn.utils.class_weight import compute_class_weight

# Step 6 Apply SMOTE to balance training data
smote = SMOTE(random_state=42)
X_train_res, y_train_res = smote.fit_resample(X_train, y_train)

print("Before SMOTE:", np.bincount(y_train))
print("After SMOTE:", np.bincount(y_train_res))

# Compute class weights (for use in models like RandomForest, LSTM)
class_weights = compute_class_weight(class_weight='balanced', classes=np.unique(y_train), y=y_train)
class_weight_dict = dict(zip(np.unique(y_train), class_weights))
print("Class Weights:", class_weight_dict)
```

```
Before SMOTE: [ 5086 11963]
After SMOTE: [11963 11963]
Class Weights: {np.int64(0): np.float64(1.6760715690129768), np.int64(1): np.float64(0.7125720973000084)}
```

Fig.5: Class Imbalance Handling using SMOTE and Class Weights

The Synthetic Minority Oversampling Technique (SMOTE) was applied to balance the training set, while class weights were computed for algorithms that support weighted learning, improving minority-class recognition as shown in the Fig.5.

1) Model 1: Intrusion Detection Using XGBoost with Hyperparameter Tuning

The algorithm trains an XGBoost classifier on the UNSW-NB15 dataset with SMOTE-balanced inputs and optimized hyperparameters. A parameter grid (estimators, depth, learning rate, subsample, colsample, and scale_pos_weight) is tuned using GridSearchCV with 5-fold cross-validation, maximizing the F1-score. The best model is selected and evaluated on the test set using accuracy, precision, recall, F1-score, classification report, and confusion matrix visualization.

XGBoost is well-suited for intrusion detection due to its ability to manage imbalanced data, capture nonlinear patterns, handle missing values, prevent overfitting through regularization, and scale efficiently for large datasets.

Model Interpretability Using SHAP

SHAP was applied to the trained XGBoost model for global and local interpretability. A SHAP explainer was built with the resampled training data to measure each feature’s contribution. SHAP values for the test set highlighted positive or negative per-sample impacts.

The bar plot ranked features by their average importance, while the summary plot detailed per-feature impacts with color coding to show direction and magnitude. Together, these visualizations improved transparency, enabling better understanding and trust in the intrusion detection system’s predictions.

Partial Dependence Plots (PDP)

PDPs were generated for key features—sbytes, dbytes, dur, and sttl—to show their marginal effects on attack probability predicted by XGBoost. These plots illustrate how varying a single feature, while holding others constant, influences the model’s output.

The visualizations revealed whether each feature had a positive or negative impact on intrusion likelihood, complementing SHAP results with an intuitive, feature-level understanding of model behavior.

LIME for Local Interpretability

LIME was applied to explain individual predictions of the XGBoost model. By perturbing features of a selected test instance and observing output changes, LIME identified the top contributing features influencing that prediction. Unlike global methods, it provides local interpretability, helping analysts validate specific intrusion alerts and improving trust in the system.

2) Model 2: LSTM for Intrusion Detection

The LSTM model was trained on SMOTE-balanced data after reshaping inputs into sequence format. Its architecture included LSTM, dropout, and dense layers, optimized with categorical cross-entropy and Adam optimizer. Over 50 epochs, training and validation curves were monitored to avoid overfitting. Final evaluation on the test set measured accuracy, precision, recall, and F1-score.

LSTM effectively captures temporal dependencies in network traffic, enabling robust detection of evolving attack behaviors.

3) Model 3: Random Forest for Intrusion Detection

Random Forest was trained with hyperparameter tuning using GridSearchCV on SMOTE-balanced data. Best parameters were retrieved, and predictions on the test set were evaluated with accuracy, precision, recall, and F1-score. A confusion matrix was also visualized.

RF's ensemble of decision trees handles high-dimensional, noisy data well and offers strong generalization, making it reliable for intrusion detection with resilience to imbalance and interpretability via feature importance.

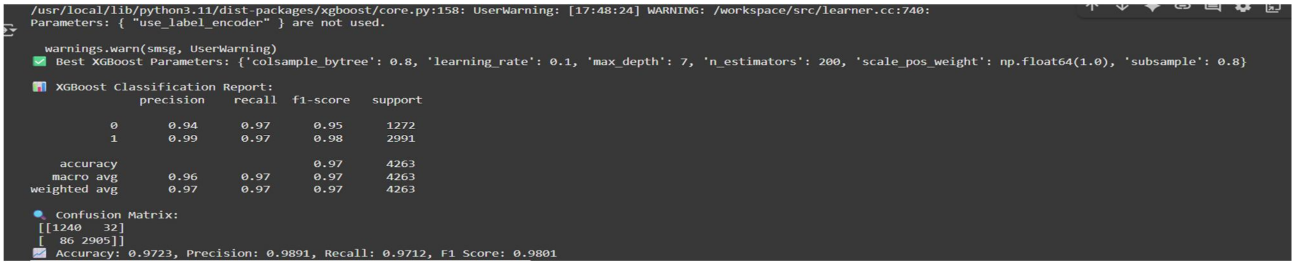
4) Model 4: KNN for Intrusion Detection

KNN was optimized through hyperparameter tuning (neighbors, distance metric, weighting scheme). The best model was evaluated on the test set with standard metrics and confusion matrix visualization.

KNN's non-parametric, instance-based learning allows it to adapt to nonlinear traffic patterns by comparing new samples to known behaviors. Its simplicity and adaptability make it useful for anomaly detection and real-time monitoring.

V. RESULTS

The XGBoost classifier, as shown in the Fig.6 optimized with 200 estimators, max depth 7, learning rate 0.1, subsample 0.8, colsample_bytree 0.8, and scale_pos_weight 1.0, achieved 97.23% accuracy. For normal traffic, it recorded 94% precision, 97% recall, and 95% F1-score, while for attack traffic, it reached 99% precision, 97% recall, and 98% F1-score. The confusion matrix (1240 TN, 2905 TP, 32 FP, 86 FN) confirms its strong balance between sensitivity and specificity for intrusion detection as shown in the Fig.7.



```
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [17:48:24] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
✅ Best XGBoost Parameters: {'colsample_bytree': 0.8, 'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 200, 'scale_pos_weight': np.float64(1.0), 'subsample': 0.8}

📊 XGBoost Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.97      0.95      1272
           1       0.99      0.97      0.98      2991

    accuracy                           0.97      4263
   macro avg       0.96      0.97      0.97      4263
weighted avg       0.97      0.97      0.97      4263

🔍 Confusion Matrix:
 [[1240   32]
 [  86 2905]]
📈 Accuracy: 0.9723, Precision: 0.9891, Recall: 0.9712, F1 Score: 0.9801
```

Fig.6: XGBoost Classification Report

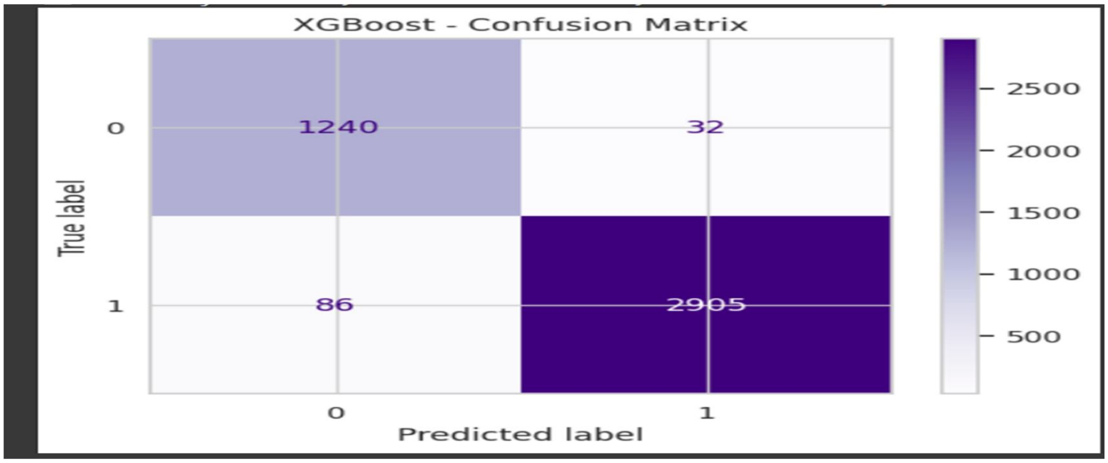


Fig.7: XGBoost Confusion Matrix

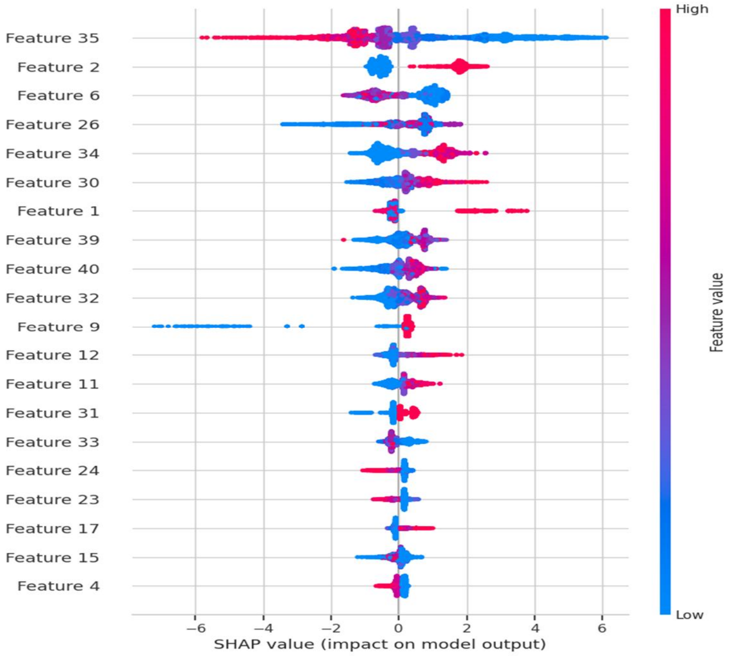


Fig.8: SHAP Model Explainability

SHAP analysis was applied to XGBoost to interpret feature contributions. The summary plot shows that Feature 35, Feature 2, and Feature 6 have the greatest impact on predictions, with red (high) and blue (low) values influencing intrusion likelihood in opposite directions. This visualization highlights the most influential features and clarifies how their values drive the model's decisions, improving transparency and trust as shown in the Fig.8.

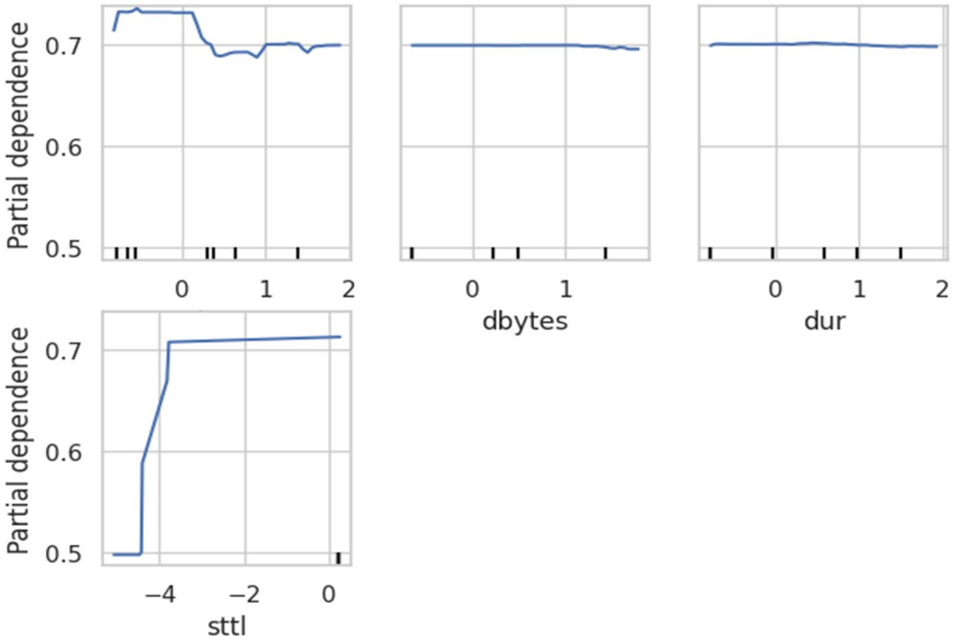


Fig.9: PDP for features dbytes,dur,sttl.

Partial Dependence Plots (PDPs) for features like *dbytes*, *dur*, and *sttl* reveal their impact on attack prediction probability. *sttl* shows a sharp positive influence between values -5 and -2, while *dbytes* and *dur* display relatively flat trends, indicating limited effect. These plots confirm which features meaningfully shape the model's decisions and in what direction as shown in the Fig.9.

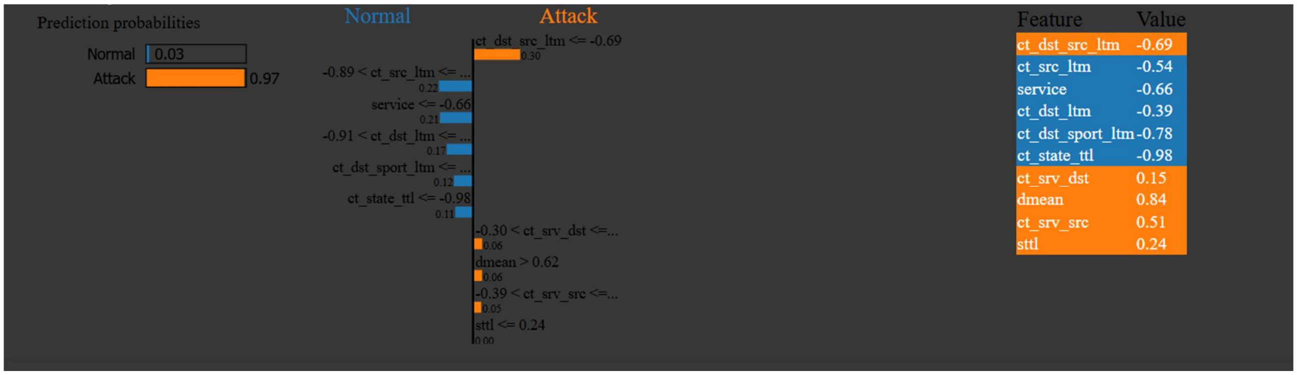


Fig.10: LIME to explain why XGBoost classified a sample as an “Attack” with 97% probability.

LIME was used to explain why XGBoost classified a sample as an “Attack” with 97% probability. Orange bars show features increasing the attack likelihood (*dmean*, *ct_srv_src*, *sttl*), while blue bars show those reducing it (*ct_dst_src_ltm*, *ct_src_ltm*, *service*, *ct_state_ttl*). By perturbing feature values and fitting a local linear model, LIME reveals how each feature influenced the decision as shown in the Fig.10.

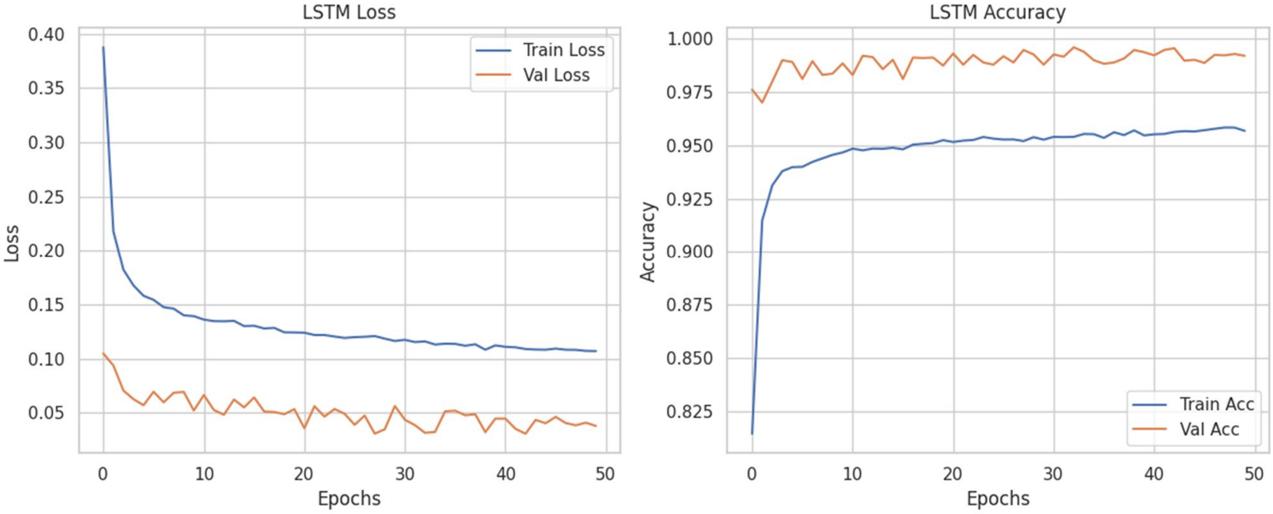
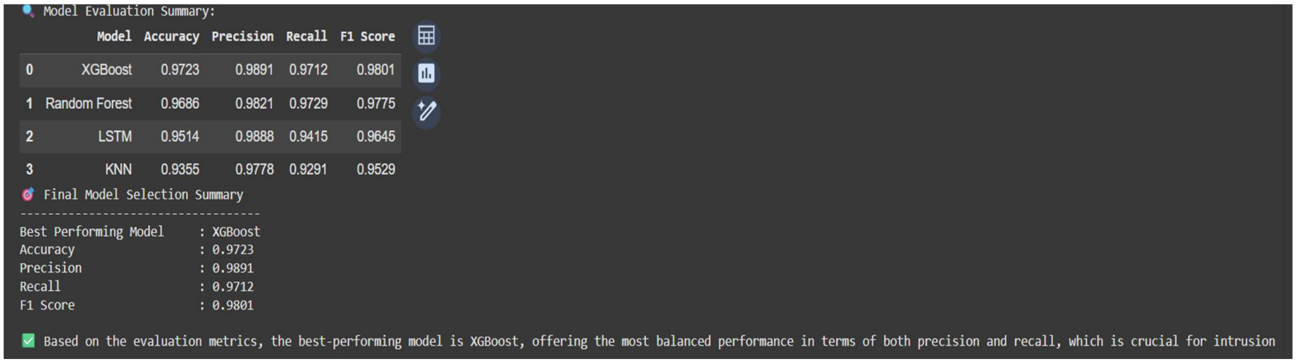


Fig.11: LSTM Model Loss and LSTM Model Accuracy.

The LSTM model effectively captured temporal patterns, achieving 95.14% accuracy. For normal traffic, it scored 88% precision, 97% recall, and 92% F1-score; for intrusion traffic, 99% precision, 94% recall, and 96% F1-score. Training accuracy rose from 76% to over 95%, with validation accuracy peaking at 99.6% and no signs of overfitting, confirming strong generalization for time-dependent intrusion detection as shown in the Fig.11.



🔍 Model Evaluation Summary:

	Model	Accuracy	Precision	Recall	F1 Score
0	XGBoost	0.9723	0.9891	0.9712	0.9801
1	Random Forest	0.9686	0.9821	0.9729	0.9775
2	LSTM	0.9514	0.9888	0.9415	0.9645
3	KNN	0.9355	0.9778	0.9291	0.9529

```
🎯 Final Model Selection Summary
----------------------------------
Best Performing Model     : XGBoost
Accuracy                  : 0.9723
Precision                 : 0.9891
Recall                    : 0.9712
F1 Score                  : 0.9801

✅ Based on the evaluation metrics, the best-performing model is XGBoost, offering the most balanced performance in terms of both precision and recall, which is crucial for intrusion
```

Fig.12: XGBoost, Random Forest, LSTM and KNN Model evaluation Summary

Four models—XGBoost, Random Forest, LSTM, and K-NN—were evaluated on the UNSW-NB15 dataset using accuracy, precision, recall, and F1-score as shown in the Fig.12. XGBoost achieved the best results (97.23% accuracy, 98.91% precision, 97.12% recall, 98.01% F1), showing strong detection capability with minimal errors. Random Forest followed closely (96.86% accuracy, 97.75% F1), highlighting its robustness on noisy traffic. LSTM achieved 95.14% accuracy and 96.45% F1, excelling in precision but with slightly lower recall. K-NN, though simpler, performed reliably (93.55% accuracy, 95.29% F1).

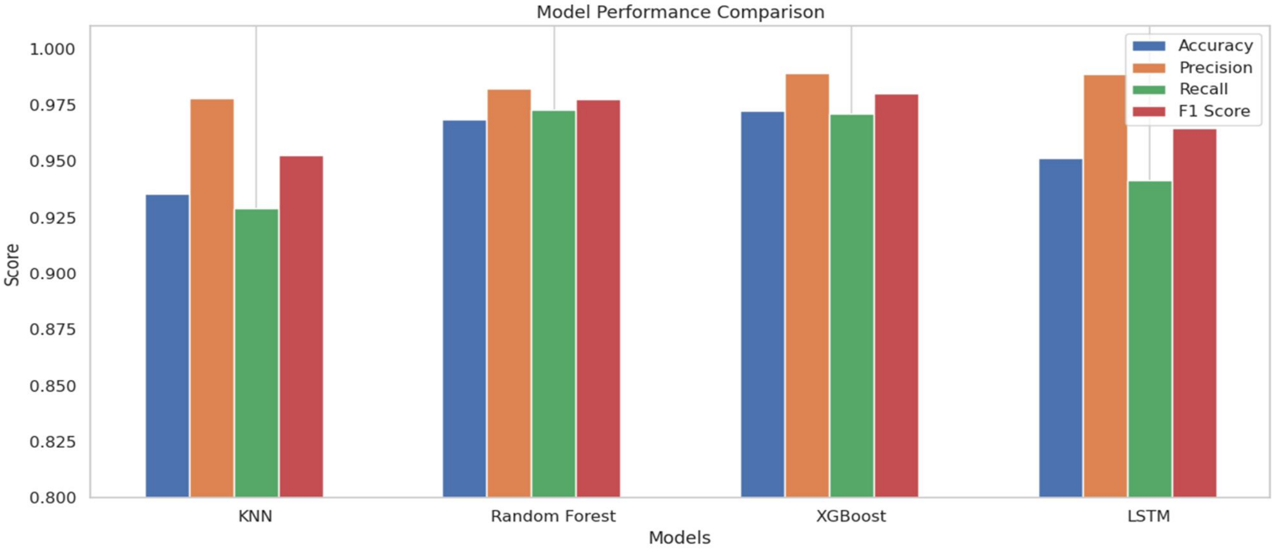


Fig.13: XGBoost, Random Forest, LSTM and KNN Model Performance Comparision.

For interpretability, SHAP identified key features (e.g., dttl, sttl, swin, dwin) influencing predictions and as shown in the Fig.13, performance of all four models are compared. Overall, XGBoost proved most effective, with Random Forest and LSTM as strong alternatives, while K-NN served as a practical baseline.

VI. CONCLUSION

This study proposes a machine and deep learning-based intrusion detection system (IDS) for electric vehicles, emphasizing efficiency, robustness, and interpretability. The methodology includes comprehensive preprocessing steps—normalization, outlier removal, missing value handling, and class imbalance correction—to optimize training data. Various classifiers, including KNN, Random Forest, LSTM, and XGBoost, were evaluated, with XGBoost achieving the best performance (97.23% accuracy, 98.01% F1-score). To ensure transparency, explainable AI methods such as SHAP, LIME, and Partial Dependence Plots were applied, providing insights into feature importance and model behavior. The resulting IDS framework not only delivers high detection accuracy but also offers clear explanations, making it well-suited for real-world EV cybersecurity applications.

REFERENCES

[1] Umer, M. A., Junejo, K. N., Jilani, M. T., & Mathur, A. P. (2022). Machine learning for intrusion detection in industrial control systems: Applications, challenges, and recommendations. International Journal of Critical Infrastructure Protection, 100516. Useful for understanding ML-based IDS approaches.

[2] Han, M., Cheng, P., & Ma, S. (2021). PPM-InVIDS: Privacy protection model for in- vehicle intrusion detection system based on complex-valued neural networks. Vehicular Communications, 31, 100374. Proposes deep learning for vehicle IDS.

[3] Aloqaily, M., Otoum, S., Al Ridhawi, I., & Jararweh, Y. (2019). An intrusion detection system for connected vehicles in smart cities. Ad Hoc Networks, 90, 101842. Focuses on IDS for smart city-integrated vehicles.

[4] Kang, M. J., & Kang, J. W. (2016). Intrusion detection system using deep neural network for in-vehicle network security. PloS One, 11(6), e0155781. Uses DNN for in-vehicle intrusion detection.

[5] Wang, Q., Lu, Z., & Qu, G. (2018). An entropy analysis-based intrusion detection system for controller area network in vehicles. In 2018 IEEE SOCC, pp. 90–95. IEEE. Applies entropy-based ML for CAN bus intrusion detection.

[6] Lombardi, M., Pascale, F., & Santaniello, D. (2022). Two-step algorithm to detect cyber- attack over the CAN-Bus: A case study in connected vehicles. ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, 8(3). Proposes a layered intrusion detection framework.

[7] Lokman, S. F., Othman, A. T., & Abu-Bakar, M. H. (2019). Intrusion detection system for automotive CAN bus: A review. EURASIP Journal on Wireless Communications and Networking, 2019(1), 1–17. Review of existing IDS solutions for CAN systems.

[8] Basnet, M., & Ali, M. H. (2020). Deep learning-based intrusion detection system for electric vehicle charging station. In 2020 SPIES, pp. 408–413. IEEE. Very relevant – IDS for EV charging infrastructure.

[9] Cheng, P., Han, M., Li, A., & Zhang, F. (2022). STC-IDS: Spatial–temporal correlation feature analyzing based IDS for intelligent connected vehicles. International Journal of Intelligent Systems, 37(11), 9532–9561. ML-based IDS leveraging spatiotemporal data in connected vehicles.

[10] Barletta, M., et al. A distance-based IDS for CAN intrusion detection using an XY-fused Kohonen network with k-means algorithm (XYF-K). Focuses on CAN-bus attack detection using unsupervised learning with high accuracy, suitable for EV security contexts.

[11] Song, H., et al, Reduced Inception-ResNet for intra-vehicle attacks using CAN intrusion dataset. A deep CNN architecture tailored for detecting sophisticated CAN-bus attacks.

[12] Ashraf, S., et al. DL-based IDS for IoV using LSTM Autoencoder evaluated on CAN and UNSW-NB15 datasets. LSTM-based anomaly detection for both internal (CAN) and external network threats— matches well with your use of sequential data and time-based threats.