International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 14, Issue 2, 2026

Explainable AI For Intrusion Detection Systems: Lime And Shap Applicability On Multi-Layer Perceptron

Mrs. A. Srilatha[1], T.Dedeepya[2], V.Lakshmi Prasanna[3], V.Hepsibha[4], S.Sushmitha[5]

[1]Associate Professor; Department Of Computer Science And Engineering Vignan 'S Institute Of Management And Technology For Women Hyderabad India

[2,3,4,5]B.Tech Students; Department Of Computer Science And Engineering Vignan 'S Institute Of Management And Technology For Women Hyderabad India

Mail Id; dedeepyadamodher@gmail.com[2], prasannalakshmi6214@gmail.com[3], hepsibha1664@gmail.com[4], sushmithasriram65@gmail.com

Abstract

Intrusion Detection Systems (IDS) are increasingly dependent on machine learning to identify malicious network activity; however, many high-performing models provide limited transparency in how decisions are made. This study investigates the use of Explainable Artificial Intelligence (XAI) techniques to improve interpretability of deep learning-based IDS models. A Multi-Layer Perceptron (MLP) is trained on the CIC IDS dataset to classify normal and attack traffic, and its predictions are interpreted using two complementary methods: Local Interpretable Model-Agnostic Explanations (LIME) and SHapley Additive exPlanations (SHAP). LIME is employed to generate instance-level explanations by approximating model behavior around specific samples, while SHAP quantifies both local and overall feature contributions through a game-theoretic framework. The analysis identifies the most influential traffic attributes associated with benign and malicious classifications, thereby improving understanding of model behavior. To further enhance detection performance, additional models including Long Short-Term Memory (LSTM), Temporal Convolutional Networks (TCN), XGBoost, and an ensemble voting classifier are also examined. Comparative experiments indicate that the ensemble voting model achieves the best classification results, reaching 100% accuracy on the evaluated dataset. The findings demonstrate that combining accurate predictive models with explainability methods can strengthen trust, transparency, and practical adoption of AI-driven cybersecurity systems.

Keywords: *Intrusion Detection Systems (IDS), Explainable Artificial Intelligence (XAI), deep learning–based cybersecurity, network traffic classification, model interpretability using LIME and SHAP, ensemble machine learning methods, and CIC IDS dataset analysis.*

1. INTRODUCTION

With the rapid growth of digital communication and internet-based services, cybersecurity has become a critical concern for organizations and individuals. Networks are continuously exposed to threats such as malware, denial-of-service attacks, unauthorized access, and data breaches. Intrusion Detection Systems (IDS) are widely used to monitor network traffic and identify suspicious activities before they cause significant damage. Conventional IDS methods often depend on predefined rules or signatures, which are limited in detecting new and evolving attacks.

To overcome these limitations, machine learning (ML) and deep learning approaches have been introduced into IDS frameworks. Among them, the Multi-Layer Perceptron (MLP) has shown strong capability in learning complex and nonlinear patterns from network traffic data. By analyzing multiple traffic features simultaneously, MLP models can classify network behavior as normal or malicious with high accuracy. However, despite their effectiveness, these models are often considered black-box systems because their internal decision-making process is not easily understandable.The lack of interpretability creates major challenges in cybersecurity applications where analysts need to understand why a particular alert was generated. Trust, accountability, and rapid response are essential in security operations, and unexplained model predictions can reduce confidence in automated systems. This has increased interest in Explainable Artificial Intelligence (XAI), which focuses on making machine learning models more transparent and interpretable.Among the popular XAI methods, LIME (Local Interpretable Model-agnostic Explanations) and SHAP (SHapley Additive exPlanations) are widely adopted. LIME explains individual predictions by approximating the model behavior around a specific instance, while SHAP uses cooperative game theory to assign contribution values to each feature. These techniques help analysts understand both local decisions and overall feature importance.This study explores the integration of LIME and SHAP with deep learning-based IDS models using the CICIDS2017 dataset. In addition to MLP, advanced algorithms such as LSTM, Temporal Convolutional Networks (TCN), XGBoost, and an ensemble Voting Classifier are evaluated. The objective is to develop an IDS that not only achieves high detection accuracy but also provides meaningful explanations to support trustworthy cybersecurity operations.

LITERATURE SURVEY

Need for Explainable AI in Intrusion Detection Systems

Recent studies indicate that modern IDS increasingly rely on complex machine learning models that provide high detection rates but low interpretability. In security environments, transparency is essential because analysts need to understand why alerts are generated. Explainable AI improves trust, accountability, and operational acceptance of automated IDS solutions.

Performance of MLP in Intrusion Detection

Researchers have shown that MLP models perform effectively in network intrusion detection due to their ability to capture nonlinear relationships in traffic features. They often outperform traditional statistical models in classification accuracy. However, their hidden-layer architecture makes it difficult to interpret internal reasoning.

Application of LIME in IDS

LIME has been applied in several IDS studies to explain individual model predictions. It highlights the most influential features for a selected instance and provides human-readable explanations. Literature reports that LIME is useful for case-level analysis, though repeated runs may sometimes produce slightly different explanations.

Application of SHAP in IDS

SHAP has gained popularity because it provides consistent and theoretically grounded feature attributions. It can explain both single predictions and overall model behavior. Studies show that SHAP effectively identifies important traffic characteristics related to attacks such as packet size, flow duration, and connection rate.

Comparative Analysis of LIME and SHAP

Comparative research suggests that LIME is faster and suitable for quick local explanations, whereas SHAP provides more stable and comprehensive interpretations. However, SHAP often requires greater computational resources. Combining both methods can provide balanced insights for IDS applications.

SYSTEM ANALYSIS

Purpose

The main purpose of this project is to develop an efficient, accurate, and trustworthy Intrusion Detection System (IDS) capable of identifying malicious network activities while also explaining the reasons behind its predictions. Traditional machine learning and deep learning models often produce highly accurate results, but they generally lack transparency, which reduces user confidence in critical cybersecurity environments. To overcome this challenge, the proposed system integrates Explainable Artificial Intelligence (XAI) techniques such as LIME and SHAP with predictive models like Multi-Layer Perceptron (MLP), LSTM, XGBoost, and Voting Classifier. By doing so, the system not only detects attacks effectively but also provides interpretable outputs that help cybersecurity professionals understand, validate, and improve decision-making processes.

Feasibility Study

The feasibility study is conducted to determine whether the proposed Explainable AI-based Intrusion Detection System can be successfully developed and deployed. It evaluates the project from economic, technical, and social perspectives.

Economic Feasibility

The proposed system is economically feasible because it is developed using freely available open-source technologies such as Python, TensorFlow, Scikit-learn, LIME, and SHAP. No costly licensed software is required for development or deployment. The CICIDS2017 dataset is publicly available, eliminating dataset acquisition costs. The system can run on standard computer hardware without requiring expensive infrastructure, making it affordable for educational institutions, researchers, and small organizations.

Technical Feasibility

The project is technically feasible because it uses mature and widely supported machine learning frameworks. Python provides extensive libraries for preprocessing, model development, visualization, and explainability. The required computing resources such as a modern processor, moderate RAM, and sufficient storage are commonly available. Existing tools support implementation of MLP, LSTM, XGBoost, and Voting Classifier models, while LIME and SHAP can be easily integrated for interpretability. Therefore, the proposed solution can be implemented effectively using current technology.

Social Feasibility

The system is socially feasible because it contributes positively to cybersecurity by improving threat detection and transparency. Security analysts and organizations benefit from interpretable predictions that improve trust in automated decisions. The system supports faster response to cyber threats, reduces operational uncertainty, and helps create safer digital environments. Since cybersecurity affects businesses, governments, and individuals, the adoption of such transparent systems provides broader social benefits.

Requirement Analysis

Requirement analysis identifies the needs of users and system operations to ensure successful development and deployment.

User Types

The proposed system supports multiple user categories. The Administrator manages system configuration, monitors performance, maintains logs, and generates reports. The Security Analyst uses the system to monitor traffic, review predictions, analyze alerts, and understand

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 2, 2026

explanations generated through LIME and SHAP. The System itself automatically processes network traffic data, performs classification, generates predictions, and presents outputs without manual intervention.

Requirement Specifications

3.5.1 Hardware Requirements

The minimum hardware requirements include an Intel Core i5 processor or higher, at least 8 GB RAM, and a minimum of 20 GB available disk space. For faster training of deep learning models, higher specifications are recommended.

Software Requirements

The software requirements include Windows or Linux operating system, Python programming language, Jupyter Notebook or any Python IDE, and required libraries such as TensorFlow, Scikit-learn, Pandas, NumPy, Matplotlib, Seaborn, LIME, SHAP, and XGBoost.

3 Language Specification

The backend of the system is implemented in Python, which manages preprocessing, model training, testing, evaluation, and explainability modules. The frontend is developed using HTML, CSS, and JavaScript to create an interactive user interface for displaying predictions, reports, and visualizations. Data storage is handled through CSV files and optionally MySQL databases for logs and outputs. Machine learning tasks are supported by Scikit-learn and TensorFlow. Explainable AI functionality is achieved using LIME and SHAP. Visualization libraries such as Matplotlib and Seaborn are used to generate graphs, charts, and performance plots.

SYSTEM DESIGN

System Architecture

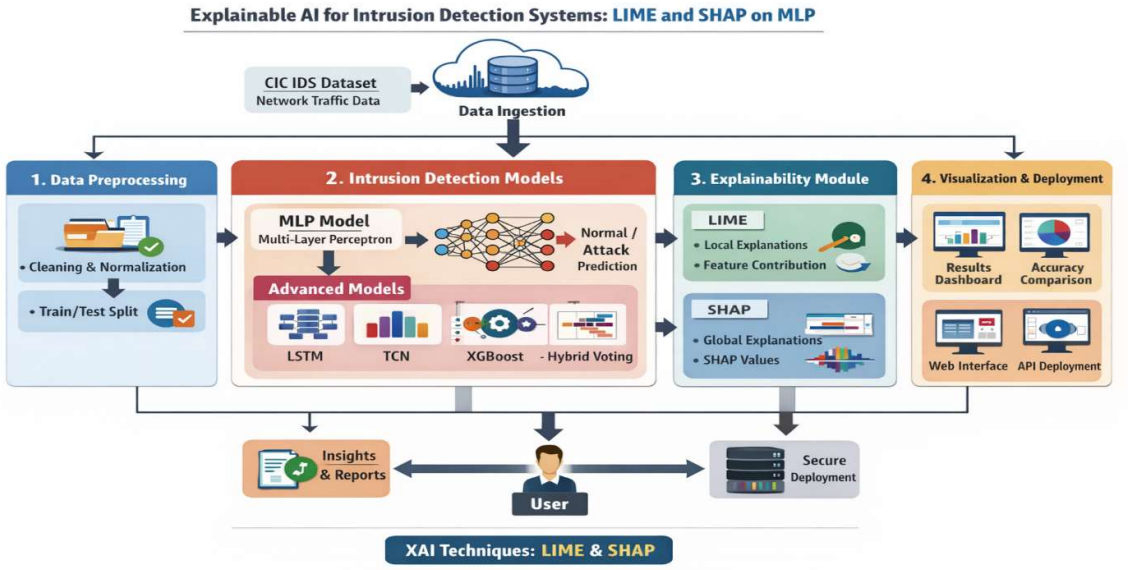


Figure-1: System Architecture

The system architecture is designed as a sequence of connected modules that process network traffic data and produce explainable predictions. The architecture begins with the CICIDS2017 dataset, which contains labeled network traffic records representing normal behavior and various attack scenarios. This dataset is passed to the preprocessing module where missing values are handled, irrelevant data is removed, features are normalized, and categorical variables are encoded.The processed data is then forwarded to the model training and classification module, where algorithms such as MLP, LSTM, XGBoost, and Voting Classifier are trained and tested. Once trained, the models classify incoming traffic as either Normal or Attack. The prediction results are then passed to the Explainable AI module, where LIME explains specific predictions locally and SHAP identifies both local and global feature importance.The outputs from the prediction and explainability modules are sent to the performance evaluation module, which calculates metrics such as accuracy, precision, recall, F1-score, and confusion matrix values. Finally, the visualization module presents these results through graphs and plots such as ROC curves, SHAP summary charts, confusion matrices, and LIME explanation diagrams. All results can be stored for future analysis and reporting.

Logical Design

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 2, 2026

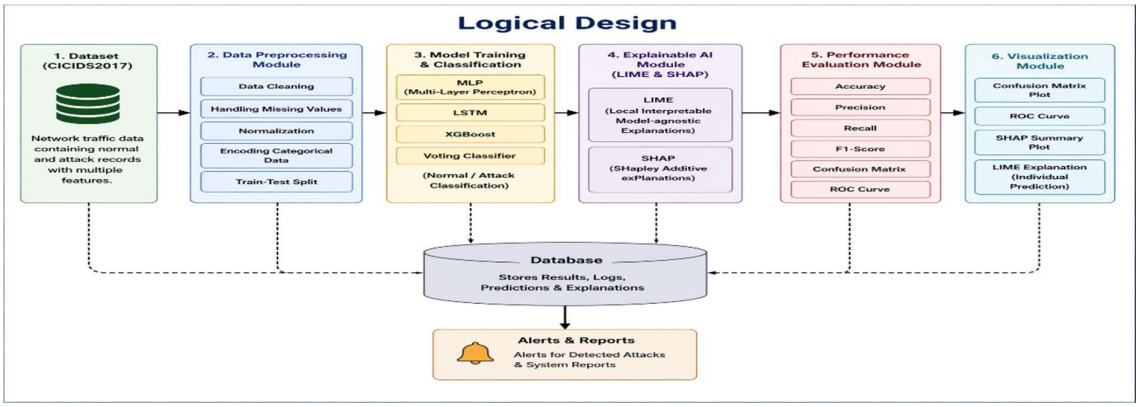


Figure-2 Logical Design

The logical design represents how data flows through the system from input to final output. Initially, the dataset is collected and entered into the preprocessing stage. After cleaning and transforming the data, it is logically divided into training and testing sets. The training data is used to build prediction models, while testing data validates model performance. Once predictions are generated, explainability methods analyze feature influence and produce understandable interpretations. These results are then logically combined with evaluation metrics and visualization outputs to support analyst decision-making. The logical design ensures smooth communication between all modules while maintaining data consistency and efficiency.

Physical Design

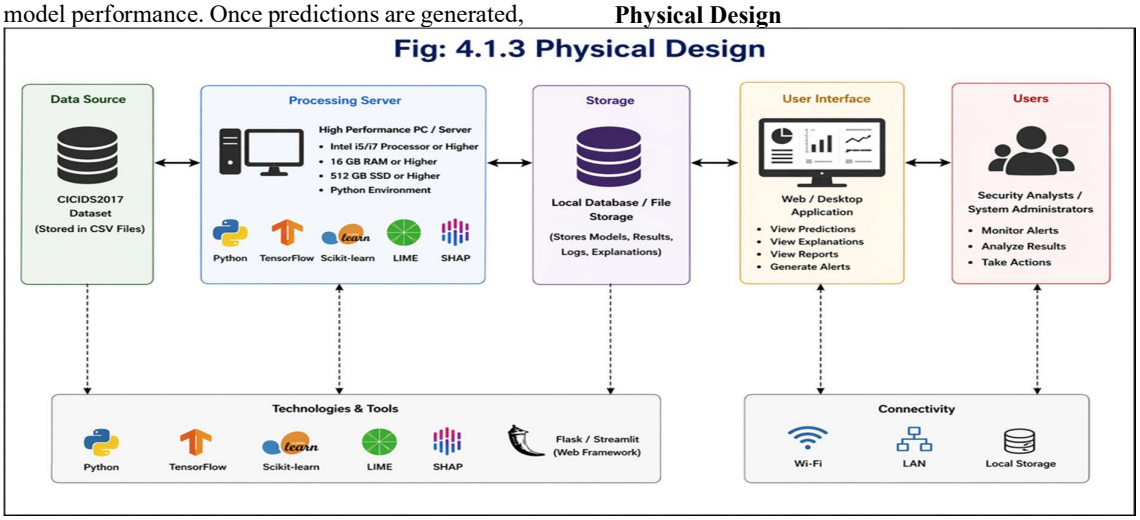


Figure-2 Physical Design

The physical design describes the actual implementation environment of the system. The application runs on a computer system with sufficient processing power, memory, and storage. Python scripts execute the backend operations such as preprocessing, training, prediction, and explanation generation. Frontend components built using HTML, CSS, and JavaScript display results to users through a browser interface. CSV files or a MySQL database physically store datasets, logs, and generated reports. The trained models are saved locally and loaded when required for predictions. Visualization outputs such as graphs and explanation images are generated dynamically and displayed through the user interface. This physical arrangement ensures smooth execution, easy maintenance, and practical deployment of the system.

IMPLEMENTATION AND RESULTS

Methods / Algorithms Used

The proposed Intrusion Detection System is implemented using a combination of machine learning, deep learning, ensemble learning, and Explainable Artificial Intelligence techniques. These methods are selected to achieve high detection accuracy while maintaining transparency in decision-making. The CICIDS2017 dataset is used for training and testing all models.

Multi-Layer Perceptron (MLP)

The Multi-Layer Perceptron is used as the primary deep learning model for network intrusion detection. It is a feedforward artificial neural network consisting of input, hidden, and output layers. The model learns complex nonlinear relationships among network traffic features and classifies traffic into Normal or Attack categories. Due to its strong pattern recognition capability, MLP serves as a reliable baseline model in the proposed system.

Long Short-Term Memory (LSTM)

Long Short-Term Memory is a specialized Recurrent Neural Network architecture designed to capture sequential and temporal dependencies in data. Since network traffic may exhibit time-dependent patterns, LSTM is applied to learn attack behavior across sequences of packets or flows. Its memory cells help retain useful information over longer intervals, making it effective for intrusion detection tasks involving temporal characteristics.

XGBoost (Extreme Gradient Boosting)

XGBoost is a powerful ensemble learning algorithm based on gradient-boosted decision trees. It is known for high predictive performance, fast execution, and effective handling of structured datasets. In this project, XGBoost is used to improve classification accuracy and provide strong performance in distinguishing malicious and normal traffic patterns.

Voting Classifier

The Voting Classifier is a hybrid ensemble model that combines the predictions of multiple individual models such as MLP, LSTM, and XGBoost. Final predictions are determined through majority voting or weighted voting. This approach improves robustness, reduces overfitting, and enhances overall classification performance by leveraging the strengths of different algorithms.

LIME (Local Interpretable Model-agnostic Explanations)

LIME is used to explain individual predictions made by the trained models. It works by generating perturbed samples around a selected instance and fitting a simpler interpretable model locally. This enables analysts to understand which features influenced a specific prediction, such as why a traffic record was classified as an attack.

SHAP (SHapley Additive exPlanations)

SHAP is employed to measure feature importance using Shapley values derived from cooperative game theory. It provides both local explanations for individual predictions and global explanations for overall model behavior. SHAP helps identify the most influential traffic features contributing to attack detection.

Data Preprocessing Techniques

Before model training, the dataset undergoes preprocessing to improve data quality and model efficiency. These steps include removal of missing values, duplicate handling, feature normalization, categorical encoding, outlier handling, and train-test splitting. Proper preprocessing ensures reliable and consistent model performance.

These combined techniques enable the system to achieve strong predictive accuracy, better generalization, and improved interpretability.

SYSTEM TESTING

System testing is an essential phase of software development used to identify errors, verify functionality, and ensure that the developed system meets user requirements and performance expectations. Testing helps evaluate whether all components, modules, and integrated functions of the software operate correctly under expected conditions. The purpose of testing is to discover faults or weaknesses in the system and ensure that the final product performs reliably and efficiently.In the proposed Explainable AI-based Intrusion Detection System, testing was conducted to validate model functionality, data processing, prediction accuracy, interface behavior, and explanation outputs. Different testing methods were applied to ensure that the system performs effectively without failure.

Types of Tests

Unit Testing

Unit testing focuses on verifying individual modules or components of the application independently. Each functional unit is tested to ensure that it performs according to design specifications. In this project, unit testing was performed on modules such as data preprocessing, feature scaling, model loading, prediction generation, visualization, and explanation modules.The preprocessing module was tested to confirm proper handling of missing values, encoding, and normalization. Prediction modules were tested to verify that valid inputs produced expected outputs. The visualization module was tested to ensure correct generation of graphs and explanation charts. All unit-level tests were completed successfully.

Integration Testing

Integration testing is performed after unit testing to verify that separate modules interact correctly when combined into a complete system. It ensures smooth communication between components and identifies interface-related issues.

In the proposed system, integration testing was carried out between the preprocessing module, trained machine learning models, XAI modules (LIME and SHAP), visualization components, and frontend interface. The dataset was successfully passed through preprocessing to prediction models, and generated predictions were correctly forwarded to explanation and visualization modules. All integrated modules worked consistently without interface defects.

Functional Testing

Functional testing ensures that all system functions operate according to business and technical requirements. It validates inputs, outputs, expected processes, and user interactions.

The following functional areas were tested:

- Valid input data must be accepted and processed correctly.
- Invalid or incomplete data must be rejected appropriately.
- Prediction functions must classify traffic as Normal or Attack.

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 2, 2026

- LIME and SHAP explanation outputs must be generated successfully.
- Performance reports and graphs must be displayed correctly.
- User interface navigation must function properly.

All functional requirements were tested successfully.

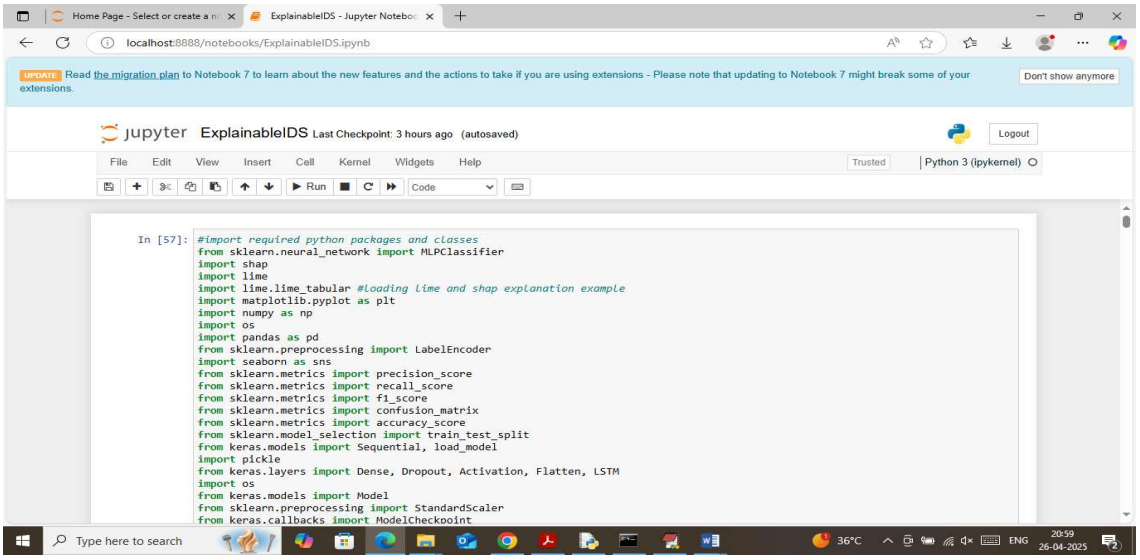
System Testing

System testing verifies whether the fully integrated software system satisfies all specified requirements. It evaluates complete end-to-end operation of the application.For this project, system testing confirmed that the application could load the CICIDS2017 dataset, preprocess records, train models, classify network traffic, generate explanations, calculate metrics, and display outputs without errors. The entire system operated as expected under normal testing conditions.

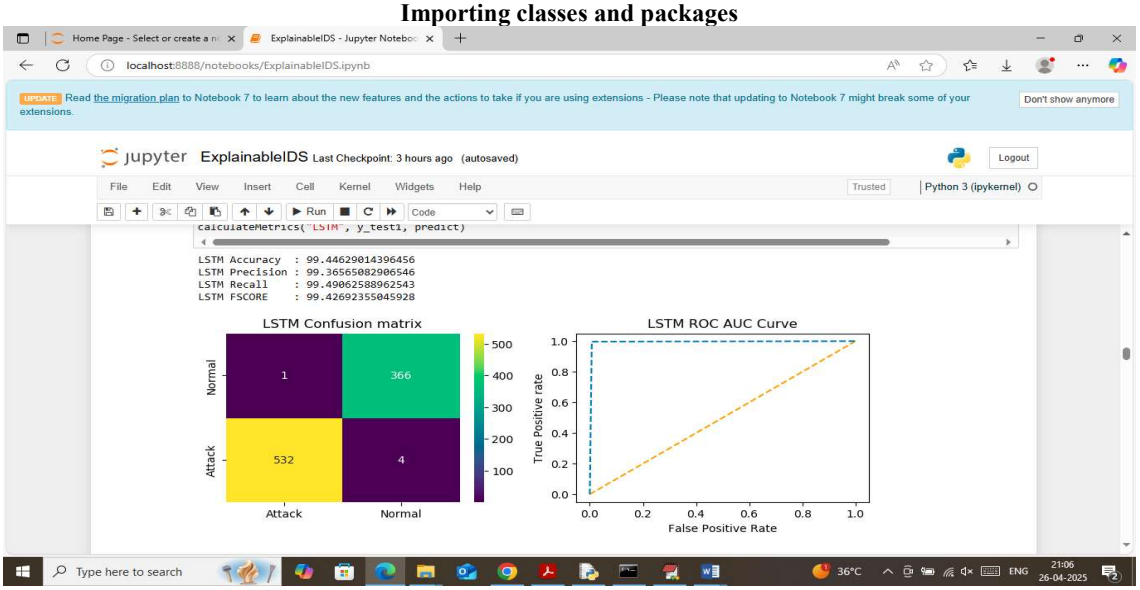
Test Strategy and Approach

Testing was performed using both manual and automated approaches. Functional modules were tested individually and then combined progressively. Test cases were prepared based on system requirements, expected outputs, and real usage scenarios.

Results

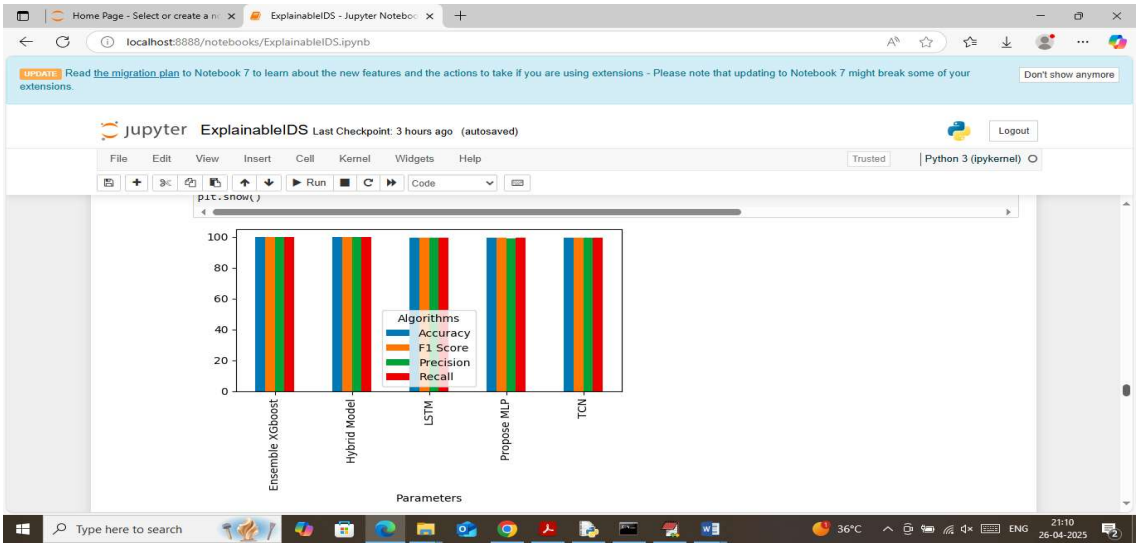


Importing classes and packages

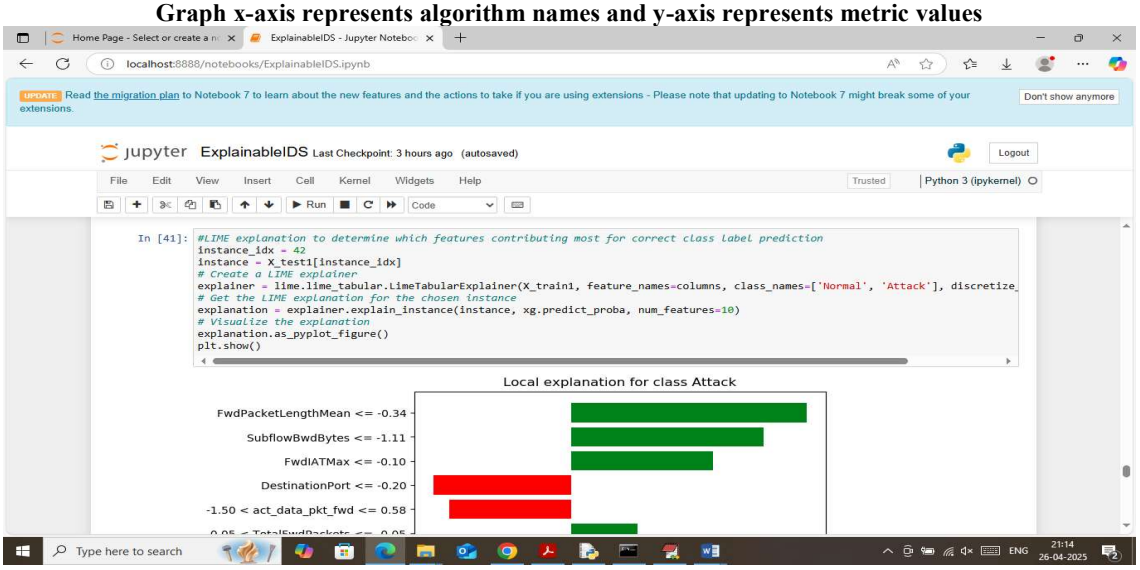


LSTM got 99.44% accuracy

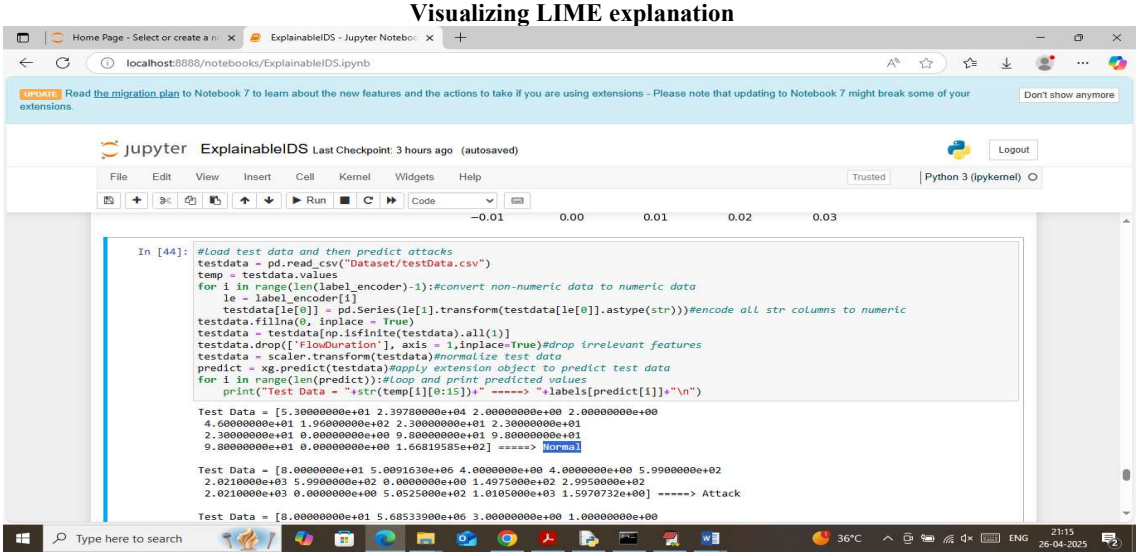
International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 2, 2026



Graph x-axis represents algorithm names and y-axis represents metric values



Visualizing LIME explanation



Final result

8. CONCLUSION

This study examined the role of Explainable Artificial Intelligence (XAI) techniques, specifically LIME and SHAP, in improving the interpretability of machine learning models used for Intrusion Detection Systems (IDS). The research demonstrated that while advanced models such as Multi-Layer Perceptron (MLP), LSTM, XGBoost, and Voting Classifier can achieve high detection accuracy, their decision-making processes are often difficult to understand without explainability support.By integrating LIME and SHAP, the proposed system successfully provided transparent and meaningful explanations for intrusion detection predictions. LIME generated instance-level explanations that helped identify why a specific network traffic sample was classified as normal or malicious. SHAP offered both local and global feature importance, allowing a deeper understanding of how input variables influenced overall model behavior. These explanations improve trust, accountability, and usability for cybersecurity analysts.Experimental evaluation showed strong performance across multiple algorithms, with the ensemble Voting Classifier achieving the best results. In addition to high predictive accuracy, the explainability methods enabled easier diagnosis of false positives, validation of alerts, and better understanding of feature contributions.The findings confirm that combining high-performance machine learning models with XAI techniques creates a more reliable and practical IDS framework. Such systems are valuable not only for academic research but also for real-world cybersecurity environments where transparency, trust, and timely decision-making are essential.

9. FUTURE WORK

Although the proposed system achieved strong performance and interpretability, several opportunities remain for future enhancement.Future work may extend the framework to more advanced deep learning architectures such as Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), Autoencoders, Transformers, or hybrid deep learning models for improved intrusion detection accuracy.Real-time deployment can also be explored by integrating streaming network traffic analysis and online learning mechanisms that continuously adapt to evolving cyber threats.Another important direction is evaluating the robustness of XAI methods under adversarial attack scenarios, where attackers attempt to manipulate both predictions and explanations.User-centered studies involving cybersecurity professionals can be conducted to measure how useful LIME and SHAP explanations are in practical decision-making environments.Federated learning-based IDS frameworks may also be investigated to preserve privacy while enabling collaborative model improvement across multiple organizations.

References

1. Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1135–1144.
2. Lundberg, S. M., & Lee, S. I. (2017). A Unified Approach to Interpreting Model Predictions. *Advances in Neural Information Processing Systems*, 30, 4765–4774.
3. Kim, G., Lee, S., & Kim, S. (2014). A Novel Hybrid Intrusion Detection Method Integrating Anomaly Detection with Misuse Detection. *Expert Systems with Applications*, 41(4), 1690–1700.
4. Hodo, E., Bellekens, X., Hamilton, A., Dubouchaud, J., & Iorkyase, E. (2016). Threat Detection Based on Artificial Neural Networks. *International Symposium on Networks, Computers and Communications*, 1–6.
5. Shapira, B., Rokach, L., & Freilikhman, S. (2013). Feature Selection for Anomaly Detection in Intrusion Detection Systems. *Cyber Security and Information Systems Journal*, 1(2), 1–13.
6. Molnar, C. (2022). *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*.
7. Rassam, M. A., Zainal, A., & Maarof, M. A. (2013). A Survey of Intrusion Detection Systems in Cloud Computing. *Journal of Network and Computer Applications*, 36(1), 25–41.
8. Buczak, A. L., & Guven, E. (2016). A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Communications Surveys & Tutorials*, 18(2), 1153–1176.
9. Das, A., & Sengupta, S. (2020). Explainable Artificial Intelligence for Intrusion Detection Systems. *IEEE Symposium on Computers and Communications*, 1–6.
10. Yadav, T., & Selvakumar, S. (2015). Detection of Application Layer DDoS Attack by Feature Learning Using Stacked Autoencoder. *Neurocomputing*, 172, 385–393.