# University of Ibadan Journal of Science and Logics in ICT Research (UIJSLICTR)

**ISSN: 2714-3627**

***A Journal of the Faculty of Computing, University of Ibadan, Ibadan, Nigeria***

**Volume 15 No. 1, September 2025**

**journals.ui.edu.ng/uijslictr**
**http://uijslictr.org.ng/**
**uijslictr@gmail.com**

**University of Ibadan**
**Journal of Science and Logics in ICT Research**

# A Zero Trust Hybrid Machine Learning Algorithms for Threat Detection and Prevention with Explainable Threat Intelligence.

**[1]✉ Timadi M. E. and [2]Obasi E.C.M.**
[1,2]*Department of Computer Science and Informatics, Federal University, Otuoke, Bayelsa State.*
*timadimatiel@gmail.com, obasiec@fuotuoke.edu.ng.*

[1] https://orcid.org/0009-0002-4419-2385, [2]https://orcid.org/0009-0001-1513-9887

**Abstract**
This study presents a dual-model intelligent cybersecurity framework integrating Malware Detection and SQL Injection Detection to enhance automated threat identification and prevention. For malware detection, a Random Forest classifier was employed to analyze users activities. The model achieved an accuracy of 99.13%, precision of 98.52%, and recall of 98.56%, demonstrating exceptional reliability in differentiating malicious from benign files. The ROC curve (AUC = 0.9994) and Precision–Recall curve confirmed the model's high discriminative power, while LIME and Permutation Feature Importance analyses provided interpretability, revealing that features like MajorSubsystemVersion and SectionsMeanEntropy strongly influence classification outcomes. For SQL injection detection, a feedforward neural network (FFNN) with two dense layers (32 and 64 neurons) was implemented using three handcrafted features—query length, punctuation, and SQL keywords. The model achieved an accuracy of 99.73%, precision of 99.7%, recall of 99.95%, and F1-score of 99.8%, indicating near-perfect discrimination between malicious and benign queries. The ROC (AUC = 1.00) and Precision–Recall curves further confirmed its robustness. LIME explanations provided local interpretability by highlighting influential query attributes driving predictions. A real-time detection dashboard continuously validates every access attempt—file uploads or SQL queries—using both models in parallel. Malicious inputs are instantly flagged and blocked, ensuring proactive protection. Overall, the proposed framework combines high detection accuracy with explainable artificial intelligence (XAI) techniques, providing both transparency and reliability for modern cybersecurity defense systems.

*Keywords: Machine Learning, Cybersecurity, Malware Detection, SQL Injection, Explainable Artificial Intelligence (XAI), Random Forest, Neural Network*

## 1. Introduction

In the era of digital transformation, cyber threats have become increasingly sophisticated, posing significant risks to organizations across various sectors. Traditional security paradigms often fall short in defending against these advanced threats due to their inherently reactive nature. In the realm of cybersecurity, hybrid machine learning models have emerged as pivotal tools for effectively detecting and preventing cyber threats. The integration of these models with the Zero Trust security architecture represents a transformative approach in handling sophisticated cyber threats. Zero Trust operates on the principle of "never trust, always verify," challenging the vulnerabilities inherent in traditional security postures by ensuring comprehensive authentication and authorization for every access request within a network[14].

This paper presents a novel hybrid machine learning approach combining the strengths of Random Forest and Feed-Forward Neural Networks to address the challenges of malware and SQL injection detection, respectively. Random Forest, known for its robust ensemble learning capability, excels in malware detection by improving accuracy through its ability to handle large datasets and provide insights through decision tree-based structures [17]. Meanwhile, the Feed-Forward Neural Network, noted for its proficiency in uncovering complex patterns, enhances SQL injection detection, tackling one of the most prevalent vulnerabilities in cybersecurity [4].

Beyond detection, the explainability of these models is enhanced using Local Interpretable Model-Agnostic Explanations (LIME), which

Timadi M. E. and Obasi E. C. M. (2025). A Zero Trust Hybrid Machine Learning Algorithms for Threat Detection and Prevention with Explainable Threat Intelligence. *University of Ibadan Journal of Science and Logics in ICT Research (UIJSLICTR),* Vol. 15 No. 1, pp. 246 – 261

provides interpretability to machine learning models by approximating them with simpler models for better human comprehension [12]. This aspect is critical as it aligns with the necessity for transparency and trustworthiness in AI-driven cybersecurity systems, allowing security teams to understand and trust the decisions made by automated systems [15].

The synthesis of Zero Trust principles with explainable hybrid machine learning models represents a comprehensive strategy to mitigate ever-evolving cyber threats. By prioritizing endogenous verification and leveraging explainable AI, this study aims to lay a foundation for enhancing threat detection and prevention strategies that are both robust and transparent.

## 2. Malware Attacks and SQL Injection Attacks

Cyber threats have become increasingly sophisticated, posing significant risks to organizations across various sectors. Many reseechers have employed machine learning algorithms to mitigate the effects of cyber threats. Nnodi and Obasi [20] researched on Leveraging Artificial Intelligence for Detecting Insider Threats in Corporate Networks. Their system helps to identify abnormal user activities and flags suspicious activities in real time, providing an early warning sign for potential breaches. Obasi and Nlerum [21] worked on Intrusion Detection System for Structured Query Language Injection Attack in E-Commerce Database. Their system introduces a filter layer specifically designed to verify user inputs and mitigate known SQL injection threats, thereby enhancing the security of e-commerce platforms.

Timadi and Obasi [25] researched Integrating Zero-Trust Architecture with Deep Learning Algorithm to prevent structured query Language injection attack in cloud database. Their research contributes to the development of robust security measures for cloud databases, safeguarding sensitive information and protecting against costly data breaches. Ayush et al. [6] investigated a "Deep Learning Approach for Detection of SQL Injection Attacks Using Convolutional Neural Networks." The authors scrutinized the performance of an array of machine learning algorithms, which included Naive Bayes, Decision Trees, Support Vector Machines, and K-nearest neighbors.

Malware attacks and SQL injection attacks represent significant challenges in cybersecurity, with both having distinct mechanisms and impacts, but similar implications on data integrity, confidentiality, and accessibility. Malware, a broad term for malicious software, includes a variety of hostile or intrusive software types like viruses, worms, Trojans, ransomware, and spyware. These programs are designed to damage or disrupt systems, steal sensitive data, or gain unauthorized access to networks.

The proliferation of malware has seen an upward trend due to the increasing sophistication of attack methods, where modern malware often evades traditional detection mechanisms through techniques like polymorphism and encryption [7]. The impact of malware attacks can be severe, leading to significant financial losses, data breaches, and operational disruptions. Important defense mechanisms against malware include employing advanced machine learning and deep learning models for anomaly detection, which can adapt to new malware signatures and behaviors that traditional signature-based methods might miss [26].

SQL injection attacks (SQLI), on the other hand, primarily target web applications by manipulating improperly sanitized SQL code to execute arbitrary queries on a database [4]. This type of attack can result in unauthorized data access, data loss, or modification, severely compromising the application's security and its users' data. SQLI is particularly damaging due to its potential to exploit any application that inputs user data directly into SQL queries without proper sanitization or validation [16]. Effective prevention of SQL injection attacks involves the use of parameterized queries, stored procedures, and rigorous input validation. Additionally, machine learning models, including methods like CNN-BiLSTM for SQLI detection, have demonstrated significant performance improvements in accurately identifying potential SQLI threats in web systems [9].

Both malware and SQL injection attacks necessitate a proactive and layered defense strategy that incorporates both technological and

procedural measures to ensure comprehensive protection against ever-evolving cyber threats [3].

### *2.1. Application of Machine Learing Algorithms to Mitigate Malware Attacks and SQL Attacks.*

The application of machine learning algorithms in mitigating malware and SQL injection attacks represents a dynamic evolution in cybersecurity practices. These algorithms enhance threat detection, improve response times, and adapt to emerging threats more effectively than traditional methods.

For malware detection, machine learning offers several robust approaches. Deep learning models such as Deep Neural Networks (DNN) have shown remarkable success in identifying malware signatures and behaviors. DNNs are particularly beneficial due to their ability to handle large amounts of unstructured data and detect hidden patterns within datasets [8]. Furthermore, machine learning can be used to develop predictive models that identify potential malware attacks by learning from previous instances, thus enhancing proactive defense mechanisms within organizations [10].

In the realm of SQL injection attacks, machine learning again proves indispensable. Various algorithms, including Support Vector Machines (SVM), Convolutional Neural Networks (CNN), and Long Short-Term Memory (LSTM), have been used to effectively detect SQL injection attempts by analyzing patterns within SQL queries. For instance, a hybrid CNN-BiLSTM model has demonstrated significant accuracy in discriminating between legitimate and malicious SQL requests, showcasing its superiority compared to traditional detection methods [9]. Stephan [24] examined "SQL Injection and Its Detection Using Machine Learning Algorithms and BERT." The manuscript advocates for the employment of machine learning strategies to augment the detection capabilities for SQL Injection attacks.

Abdalla *et al.* [1] engaged in research titled "An Efficient Model to Detect and Prevent SQL Injection Attack." They propose a model designed to detect and prevent SQL injection attacks, which employs runtime validation to ascertain the occurrence of such threats. Another approach employs a Probabilistic Neural Network (PNN), which excels in detecting novel SQL injection patterns by utilizing an optimized feature selection process and high-dimensional data analysis [3]. Obasi and Nlerum [22] developed a model for the Detection and Prevention of Backdoor Attacks using CNN with Federated Learning. Their model achieved an accuracy of 99.99% for training and 99.98 for validation. Hao *et al.* [13] conducted a study on the implementation and research of Deep Learning-Based Detection Technology for SQL Injection. Their research introduces a pioneering SQL injection attack detection strategy that leverages the capabilities of an enhanced TextCNN and Long Short-Term Memory (LSTM) networks, thereby significantly improving the recognition rate of SQL injection attacks while concurrently minimizing both false positive and false negative rates.

Maha *et al.* [18] investigated a Deep Learning Architecture for the Detection of SQL Injection Attacks Utilizing a Recurrent Neural Network Autoencoder Model. Their research proposes a novel architecture aimed at identifying SQL injection attacks through the application of a recurrent neural network autoencoder, exhibiting its efficacy in detecting SQL injection attacks with a superior level of accuracy relative to the alternative models analyzed in the research. Majid [19] advanced the field by proposing deep learning methodologies for SQL injection detection, specifically assessing Artificial Neural Networks (ANNs), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs). His investigation critically appraises the performance metrics of these three predominant neural network configurations for SQL injection attack detection, revealing that the Convolutional Neural Network consistently outperforms the others across nearly all evaluated metrics.

Ahmed *et al*. [2] conducted an investigation into a Multi-Phase Algorithmic Framework aimed at mitigating SQL Injection Attacks through the utilization of advanced Machine Learning and Deep Learning methodologies to bolster real-time Database security.

These machine learning models are not just limited to detecting known threats; they are also capable of identifying zero-day exploits and

sophisticated multi-vector attacks thanks to their adaptability and capacity for continuous learning. The deployment of these algorithms involves collecting data from network traffic, applying feature extraction techniques, and training classifiers that can act autonomously to detect anomalies indicative of a cyber threat [23].

Overall, integrating machine learning algorithms into cybersecurity frameworks enhances an organization's ability to mitigate threats while reducing the reliance on manual interventions. By leveraging the strengths of these advanced algorithms, organizations can ensure more resilient and adaptive cybersecurity defenses amidst an ever-evolving threat landscape.

### *2.2. Knowledge Gaps Addressed by the Research*

This study advances cybersecurity research by demonstrating the practical fusion of Zero Trust principles (authentication, authorization, least-privilege access) with hybrid machine learning algorithms for threat prevention. While previous studies focused either on Zero Trust policy enforcement or isolated ML detection models, this work bridges both, providing a multi-layered defense mechanism that authenticates every transaction and validates it using intelligent anomaly detection.

Unlike prior works that treat malware and SQL injection as separate research problems, this study presents a unified architecture capable of handling both malware detection and SQL injection attack prediction. By combining Random Forest (for structured binary features) and Feedforward Neural Network (for semantic SQL payload analysis), the research addresses the gap in cross-domain hybridization for cybersecurity threat modeling.

This research contributes new knowledge on how interpretability tools (LIME and Permutation Importance) can be systematically integrated into security models to make black-box predictions explainable. It demonstrates that interpretable outputs help identify critical threat indicators — such as high entropy in PE file sections or abnormal punctuation ratios in SQL queries — thereby enhancing both analyst trust and regulatory accountability in AI-driven systems. By integrating interpretability without significant accuracy loss, this study offers insight into how XAI methods can be used without compromising model performance. It addresses a key challenge in cybersecurity AI, ensuring that high-performing models remain transparent and auditable.

## 3. Methodology

This research adopted a machine learning–driven approach to detect malicious software (malware) and to identify SQL injection (SQLi) queries, supported by explainable artificial intelligence (XAI) methods for interpretability. as shown in Figure 1. For malware detection,a dataset with binary target variable labeled as legitimate (1 = legitimate, 0 = malicious). Non-numeric and irrelevant attributes were discarded. An ExtraTreesClassifier was employed to perform feature importance ranking, after which SelectFromModel reduced the dimensionality by retaining only significant predictors. A RandomForestClassifier with 50 estimators and a maximum depth of 10 was trained using an 80/20 train-test split.

For SQL injection, four datasets (sql.csv, sqli.csv, username.csv, password.csv) were integrated. Non-malicious queries (legitimate SQL statements, usernames, and passwords) were merged and labeled as non-sqli, while malicious queries were labeled as sqli. Three handcrafted features were generated:

1 . Length: the character length of the query.
2. Punctuation count: frequency of special symbols such as ', ;, --.
3. Keyword count: number of SQL-related keywords (e.g., drop, insert, select).

The numeric features were scaled using StandardScaler, and categorical labels were encoded using LabelEncoder.

A TensorFlow feedforward neural network with two hidden layers (32 and 64 units, ReLU activation) and one output unit (binary classification) was trained for 20 epochs using the Adam optimizer and binary cross-entropy loss. Model performance was evaluated using the following metrics:

1. Accuracy, Precision, Recall, and F1-score.
2. Confusion Matrix.
3. Precision–Recall Curve (PRC).

4. Receiver Operating Characteristic (ROC) Curve and Area Under the Curve (AUC).

To address the black-box nature of ML models, explainability was incorporated. Permutation feature importance was applied to rank the most influential features in the Random Forest classifier. LIME Tabular Explainer was employed to interpret individual predictions in both malware and SQLi detection models, showing how feature values contributed to a given classification outcome.

The integration of these models into a zero-trust cloud database security architecture ensures that unauthorized access is blocked, malicious activity is flagged, and sensitive data remains encrypted both in transit and at rest.

Figure 1 illustrates a Zero Trust Hybrid Machine Learning architecture that integrates machine learning (ML), explainable AI (XAI), and Zero-Trust security principles to detect and mitigate malware and SQL injection attacks while ensuring secure user access to cloud resources. The system starts with a collection of labeled data containing samples of normal (benign) and malicious activities. The dataset is divided into Train Data which is used to train the ML models and Test Data which is used to validate model performance. Before model training, data is cleaned, normalized, and transformed into a suitable format for the learning algorithms. This step ensures better accuracy and efficiency during model training. Two separate models were trained for different types of attack detection.

Random Forest Classifier (RFC) was trained to detects malware attacks. Feed Forward Neural Network (FFNN) was trained detects SQL injection attacks. The outputs were further interpreted using XAI (Explainable AI) to provide transparency, explaining why a request was classified as malicious or safe. When a user's request is made (e.g., accessing a database or application), it passes through Zero-Trust Principles which enforces "Never trust, always verify." Each access request is authenticated, authorized, and validated by the ML models before being granted. Encryption Module ensures data confidentiality and integrity, both in transit and at rest. Based on ML predictions and Zero-Trust checks, access was denied for SQL attack, granted for non-SQL attack, denied for Malware (Malign), and granted for Malware (Benign).

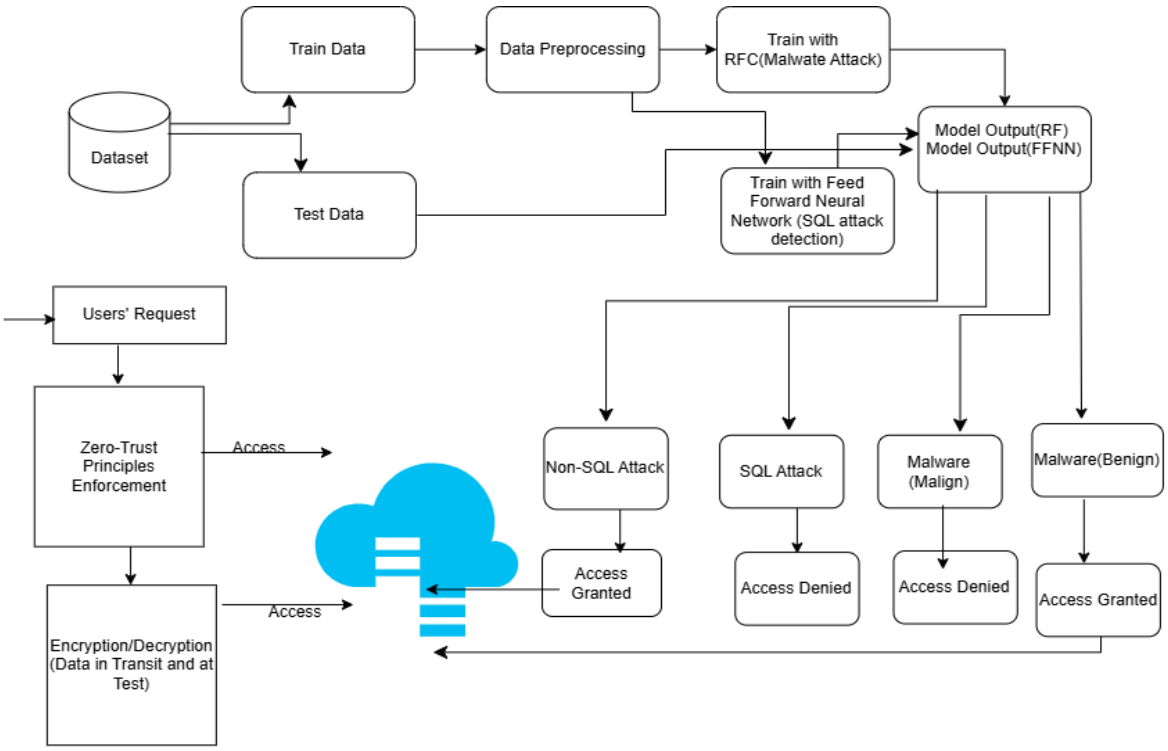


**Figure 1: Architecture of the Zero Trust Hybrid Machine Learning Model with Explainable Threat Intelligence**

The architecture demonstrates a secure, intelligent, and explainable cybersecurity framework where ML algorithms detects threats, XAI provides transparency, Zero-Trust enforces strict access control, and Encryption ensures data security.

## 4. Results and Discussion

### *4.1. Results*

The evaluation demonstrates that both models achieved exceptional performance in their respective domains. For Malware Detection, the Random Forest classifier's high precision and recall indicate strong resilience to false positives and false negatives. Features such as high entropy in PE file sections and unusual subsystem versions emerged as strong indicators of malware, aligning with known obfuscation and packing strategies. Legitimate files with large version information sizes acted as counter-balancing features, reducing overfitting. For SQL Injection Detection, The neural network exhibited near-perfect classification accuracy, demonstrating that simple yet well-chosen handcrafted features can effectively discriminate between malicious and benign queries. The high recall score is particularly significant in cybersecurity, where undetected attacks can compromise entire systems.

Both models shows continuous verification in which every access attempt (SQL query or file upload) undergoes validation, regardless of prior authentication as seen in figure 2. The dashboard displays two real-time detection panels: The Malware Detection Panel is on the Left. The Orange/Yellow (Malicious Hashes) represents data or file hashes that the Random Forest classifier has flagged as malicious. The Green (Normal Hashes) represents safe/benign hashes identified by the classifier. The interwoven color bands suggest continuous scanning, where every new hash is immediately analyzed. Malicious samples were differentiated from normal ones in real time, preventing malware execution or infiltration. SQL Injection Detection Panel is on the Right. Yellow (SQL Injection Attempts) queries were detected as malicious (SQLi). Blue/Green (Normal Queries) queries were deemed safe. Similar to malware detection, every SQL query is tokenized, passed to the FFNN, and instantly classified. Normal queries (e.g., SELECT * FROM employees WHERE id=5) remain unblocked, while malicious payloads are denied.

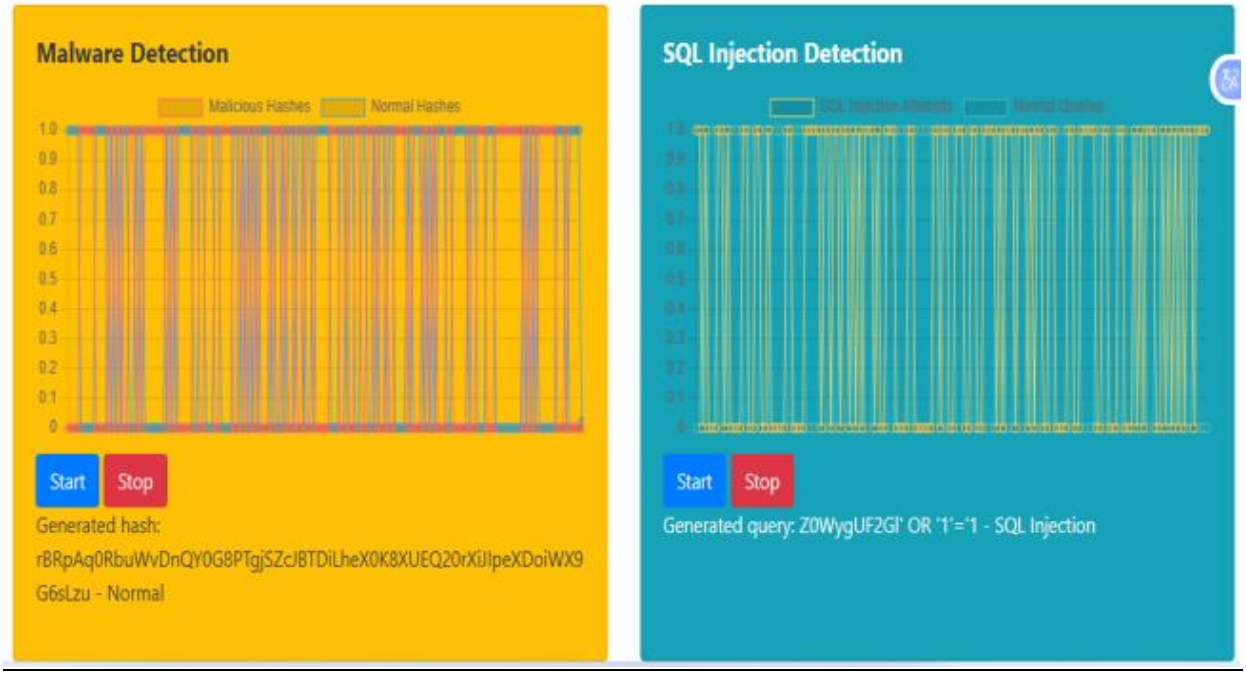


**Figure 2: Malware and SQL Detection for Malign and Benign Queries**

For malware detection, the Receiver Operating Characteristic (ROC) Curve in figure 3 shows how well the classifier distinguishes between classes (malware vs. benign), and the AUC (Area Under the Curve) score summarizes its overall performance. Accuracy measures the proportion of total predictions (both benign and malware) that the model classified correctly. With 99.13%, the model is very effective at distinguishing malware from benign files. Figure 4 shows the Precision-Recall curve. Recall measures the ability of the model to correctly detect actual malware samples. A 98.56% recall recorded means the model detected almost all malware samples, with very few missed (false negatives). This is critical in cybersecurity: missing a malware (false negative) can be more dangerous than mistakenly flagging a benign file. Precision measures how many of the samples predicted as malware were actually malware. A 98.52% precision means that when the model flags a file as malware, it is almost always correct. False alarms (false positives) are very low. The ROC curve plots the tradeoff between the True Positive Rate (Recall) and False Positive Rate at various thresholds. The AUC (Area Under Curve) is 0.9994 which is nearly perfect. This shows that the model separates malware from benign files with extremely high discriminative power.

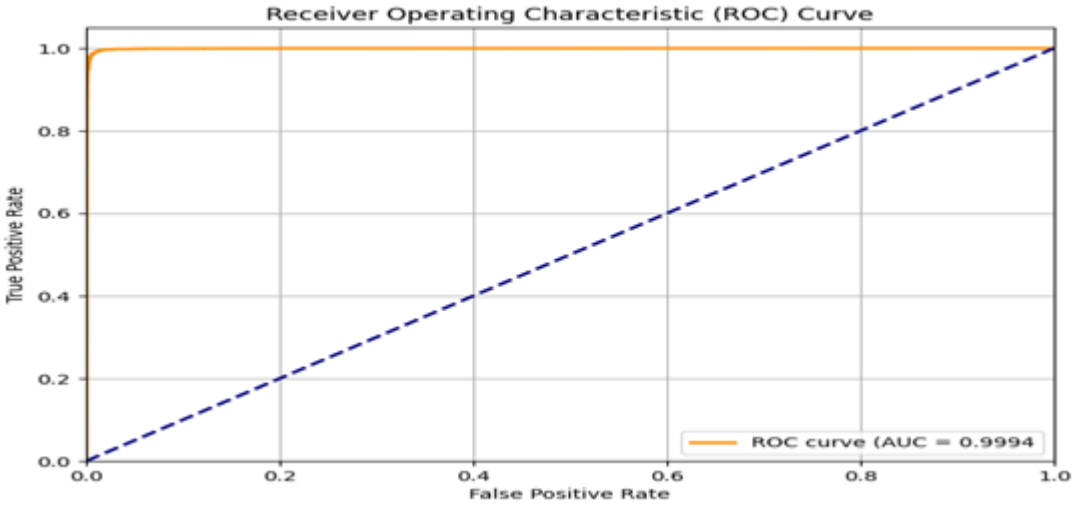


**Figure 3: Receiver Operating Characteristics (ROC) Curve**

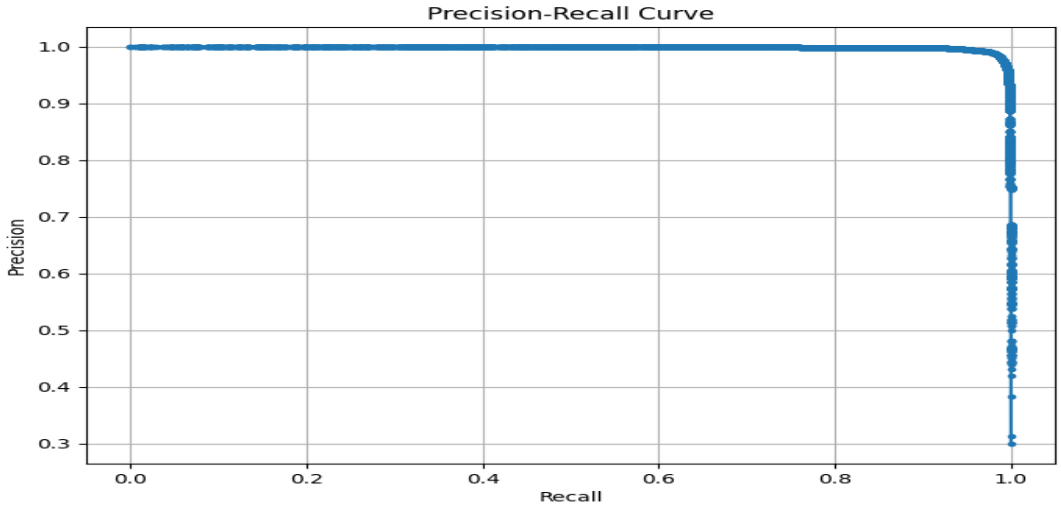


**Figure 4: Precision-Recall Curve**

The confusion matrix in Figure 5 provides a clear summary of the classification results of the RandomForestClassifier for malware detection. A total of 19,222 malicious files were correctly identified as malicious (True Positives), while 123 malicious files were wrongly classified as legitimate (False Negatives). Similarly, 8,144 legitimate files were correctly predicted as legitimate (True Negatives), whereas 121 legitimate files were misclassified as malicious (False Positives).

From these results, the model achieved an overall accuracy of 99.13%, meaning that the majority of predictions were correct.

Accuracy = (TP+TN) / (TP + TN +FP + FN) = (19222+8144) / (19222 + 8144 + 121 + 123) =99.13%

The recall value of 98.56% indicates that the model successfully detected almost all malware samples, with very few missed.

Recall = TP / (TP + FN) = 19222 / (19222 + 123) = 98.56%

The precision value of 98.52% shows that when the model flagged a file as malicious, it was almost always correct, with very few false alarms.

Precision = TP / (TP +FP) = 19222 + (19222 + 121) = 98.52%

The ROC curve complements these results by showing an Area Under the Curve (AUC) of 0.9994, which suggests that the model has an excellent ability to distinguish between malicious and legitimate files across different thresholds. The Precision–Recall curve also remains close to the upper-right corner, confirming that the model maintains both high precision and high recall even in the presence of possible class imbalance.

The confusion matrix, ROC curve, and Precision–Recall curve together demonstrate that the proposed model is highly effective in detecting malware, with minimal false positives and false negatives. However, the presence of 123 false negatives indicates that some malicious files were missed, which in cybersecurity contexts could pose significant risks.

Figure 6 shows the LIME explanations for an individual prediction for Malware detection. The chart is a Local Interpretable Model-agnostic Explanations (LIME) visualization, which explains the predictions of the machine learning model implemented in the `PE_xai_with_metrics.py` code for a specific instance. The model is a `RandomForestClassifier` trained on a dataset to classify PE (Portable Executable) files as either "Malicious" or "Legitimate." LIME approximates the model's behavior locally around a single data point using a simpler, interpretable model (e.g., a linear model) and highlights the features that most influence the prediction.

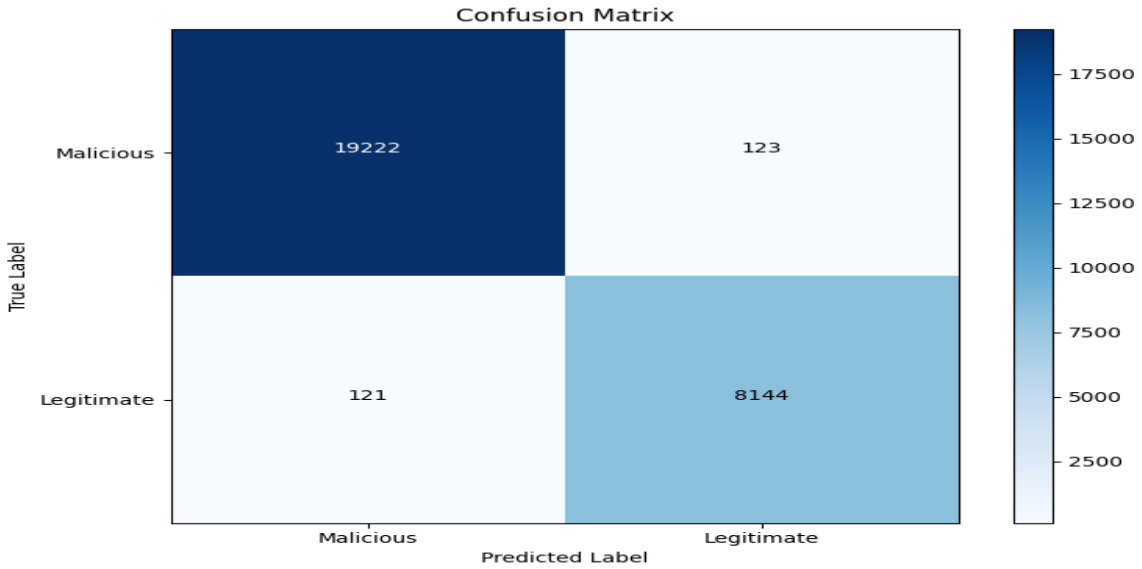


**Figure 5: Confusion Matrix of the Random Forest Model**

The model predicts a 71% probability of "Malicious" and 29% probability of "Legitimate" for the analyzed file, indicating it is flagged as likely malicious. The central section shows the decision pathway with features that contributed to the classification. Features highlighted in blue (left) push the decision toward "Malicious". Features in orange (right) push toward "Legitimate". The importance of each feature is visually proportional to its horizontal bar. Features such as MajorSubsystemVersion, Subsystem, SectionsMinEntropy, Machine, DLLCharacteristics, Characteristics, and ImageBase all contributed to the "Malicious" outcome. Features such as SizeOfOptionalHeader, SectionsMaxEntropy, ResourcesMaxEntropy, and VersionInformationSize provided some evidence toward "Legitimate" but with less influence.

Figure 7 shows a bar chart titled "Permutation Feature Importance (Global)". This plot is a global explanation of the model, meaning it shows which features were most important to the model's overall performance. The length of each bar represents the average decrease in the model's accuracy when that feature's values are randomly shuffled. A longer bar indicates a more important feature. The MajorSystemVersion feature has the highest permutation importance, indicating it's the most critical feature for your model's predictions. The model's accuracy would drop the most if MajorSystemVersion values were randomized. This suggests that the size of the executable file is a very strong predictor of whether it's malicious or legitimate.

ResourcesMinEntropy and SectionsMeanEntropy are also quite important, while DllCharacteristics is the least important of the listed features. The LIME explainer indicates the model’s 71% "Malicious" prediction is driven by high values of `MajorSubsystemVersion`,

`SectionsMaxEntropy`, and `SizeOfOptionalHeader`, with `VersionInformationSize` and `ResourcesMaxEntropy` providing some counterevidence toward "Legitimate."

This aligns with the `RandomForestClassifier`’s feature importance and the code’s feature selection process.

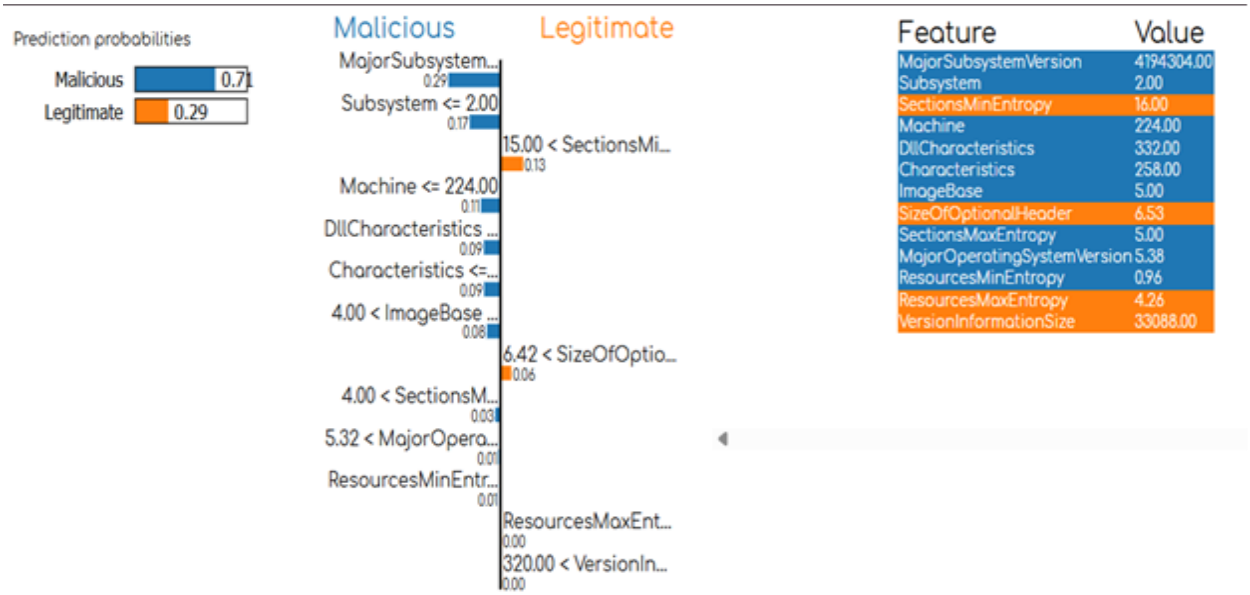


**Figure 6: LIME explanations for an individual prediction for Malware detection**

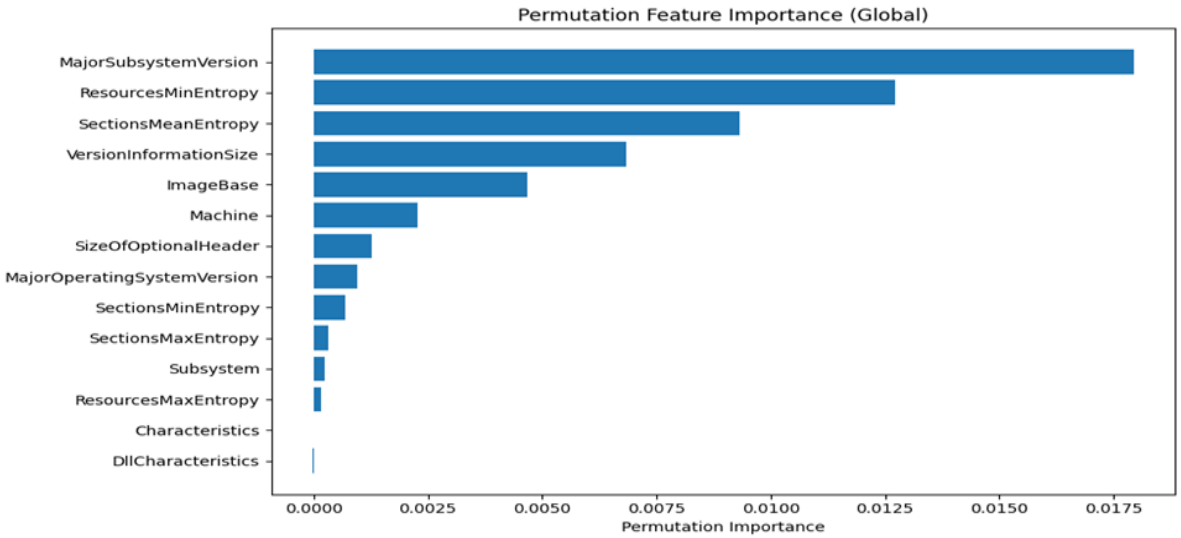


**Figure 7: Permutation Feature Importance**

For the SQL detection, labels were binary encoded to distinguish between malicious SQLi queries and non-SQLi queries (which include regular SQL, usernames, and passwords). The label column was label-encoded, turning categories into numeric values. Numerical features (Length, punctuation, keyword) were standardized using StandardScaler for normalized input to the model. A simple feedforward neural network was built using TensorFlow: Input layer for three features, Two hidden fully connected (Dense) layers with ReLU activation (32 and 64 units), Output layer with one unit (binary output logits). Figure 8 shows the Precision–Recall (PR) curve which visually summarizes our model's ability to balance between precision (avoiding false positives) and recall (avoiding false negatives) for SQL injection detection. The X-axis (Recall) measures how many actual positive cases (SQL injection attacks) were correctly identified. A higher value (closer to 1) means the model misses fewer attacks. Y-axis (Precision) measures how many predicted positive cases were actually correct. A higher value (closer to 1) means the model makes fewer false alarms. The curve is almost flat near Precision = 1.0, meaning your model maintains very high precision across nearly all recall levels. The overall shape (a sharp upper-right curve) indicates excellent model performance, nearly perfect discrimination between attack and non-attack cases.

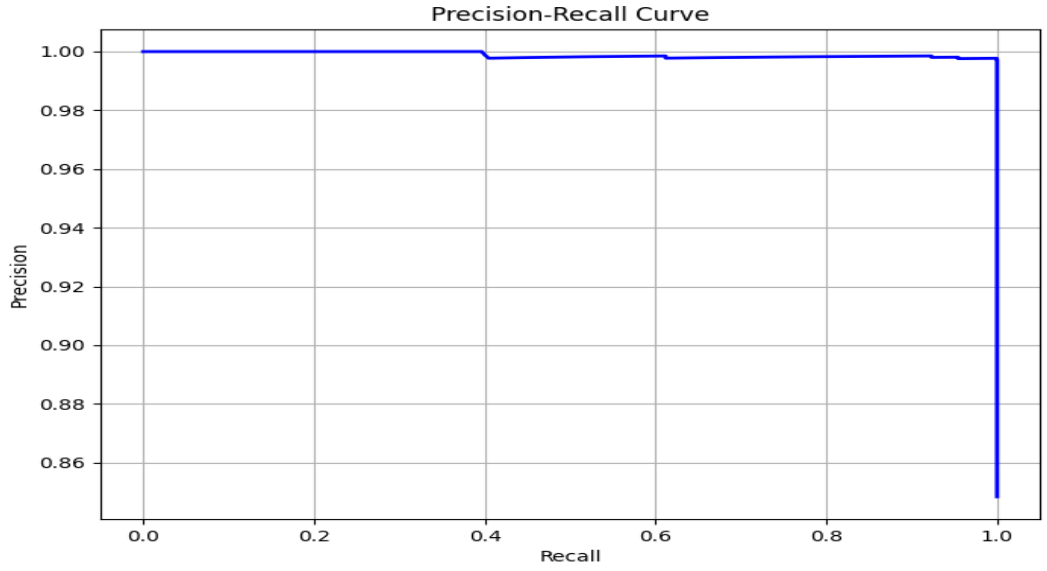


**Figure 8: Precision-Recall Curve**

Figure 9 shows the Receiver Operating Characteristic (ROC) Curve, a key performance visualization for binary classifiers like our SQL injection detection model. The X-axis (False Positive Rate – FPR) shows the fraction of normal (non-sqli) samples incorrectly classified as attack,

Figure 10 shows the confusion matrix which is used to evaluate the performance of a machine learning model for SQL injection (SQLi) attack detection. The model classifies inputs into two categories: SQLi (attack) and Non-SQLi (normal or benign traffic) with 385 samples correctly identified as non-SQLi , 6 samples of non-SQLi incorrectly predicted as SQLi, 1 SQLi sample incorrectly predicted as non-SQLi, and 2186 SQLi samples correctly identified.

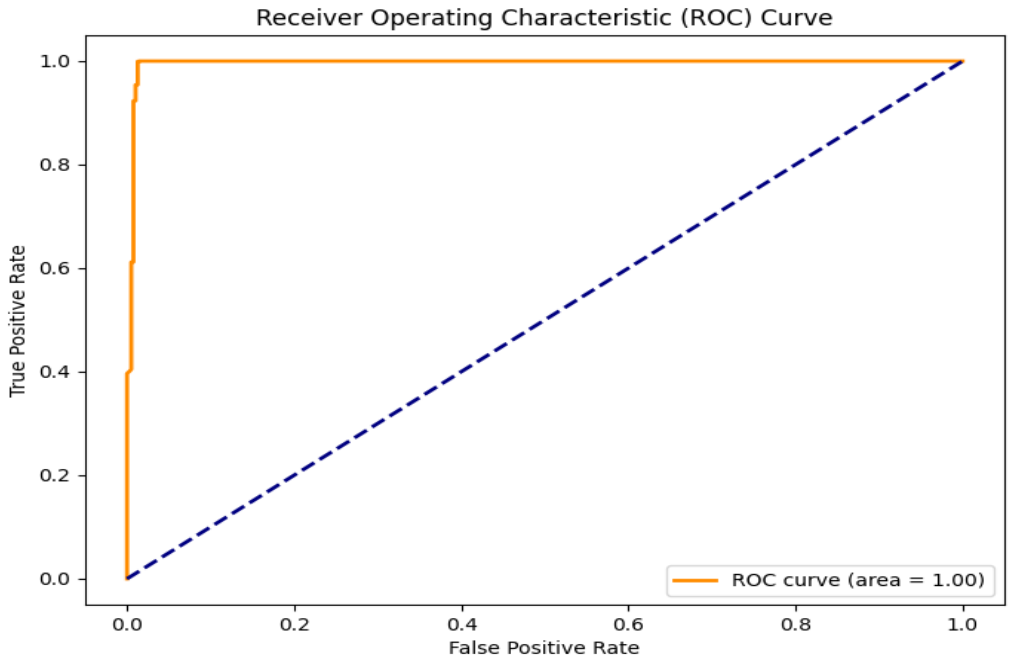


**Figure 9: Receiver Operating Characteristic (ROC) Curve**

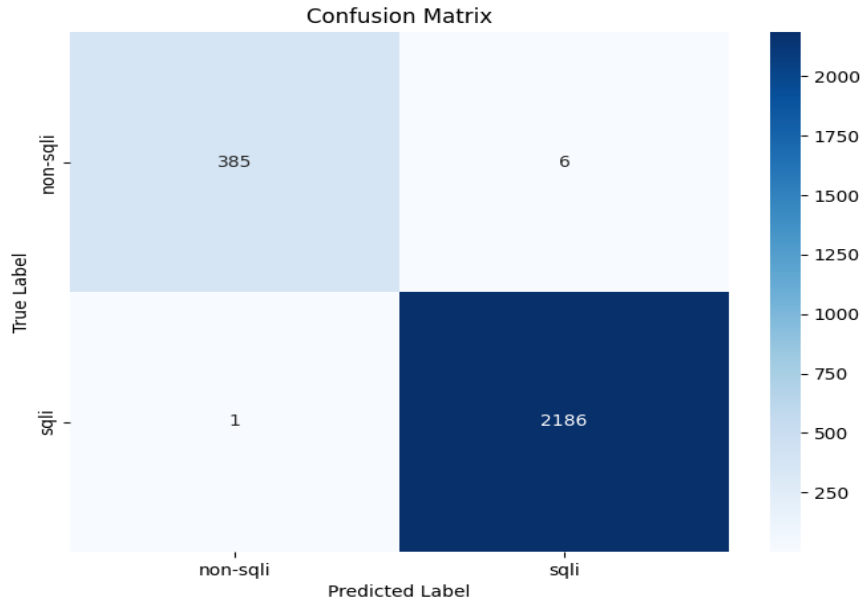


**Figure 10: Confusion Matrix of the Model**

The model achieved an overall accuracy of 99.73%, meaning that the majority of predictions were correct.

Accuracy = (TP+TN) / (TP + TN +FP + FN) = (2186+385) / (2186 + 385 + 6 + 1) =99.73%

The recall value of 99.95% was achieved by the model indicating that the model successfully detected almost all SQLi attacks, missing only one.
Recall = TP / (TP + FN) = 2186 / (2186 + 1) = 99.95%

The precision value of 99.7% was obtained showing that when the model predicted SQLi, it was 99.7% accurate.
Precision = TP / (TP +FP) = 2186 + (2186 + 6) = 99.7%

F1 score of 99.8% was achieved as follows:
F1 Score = 2 * (Precision * Recall) / (Precision + Recall) = 99.8%. This shows excellent balance between identifying attacks and minimizing false alarms

The model integrated LIME to explain model predictions locally as shown in Figure 11.Local Interpretable Model-agnostic Explanations (LIME) visualization explains the model's prediction for a specific input query by approximating its behavior locally with a simpler model. The visualization is divided into two main sections:

i. Prediction Probabilities: Displays the model's confidence scores for the two classes: "non-sqli" and "sqli." with 0% probability for non-sqli and 100% probability for sqli. This indicates the model is fully confident that the input query is an SQL injection ("sqli").
ii. Feature Importance: Lists the features and their contributions to the "sqli" prediction, with thresholds and values indicating their impact.

Features are derived from the SQL injection dataset and processed by the leng, cal_puncndop, and cal_keyword functions in the code. A wrapper function converts Keras model output logits to probability arrays compatible with LIME. The LIME explainer was initialized with training data and feature names. The model attempts to explain an instance that the model misclassifies. If none found, it explains a random instance.

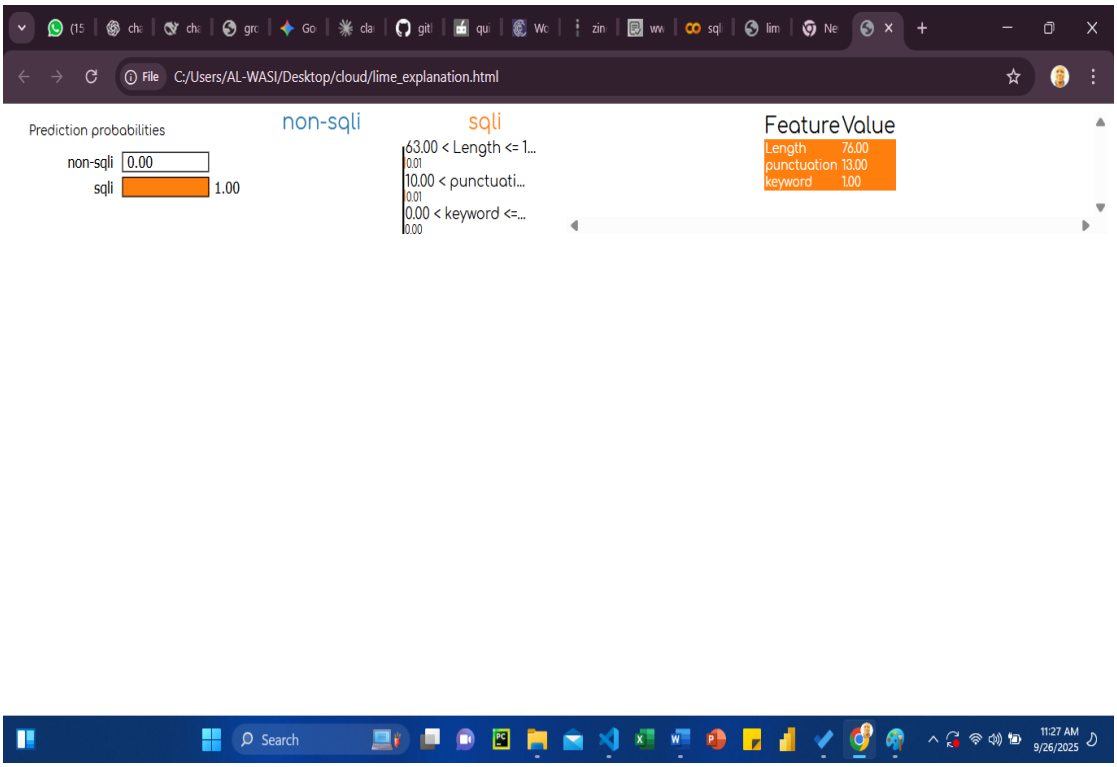


**Figure 11: LIME Explainer**

### *4.2. Discussion of Results*

The evaluation results demonstrate that both models—Random Forest (for malware detection) and Feedforward Neural Network (for SQL Injection detection)—achieved exceptional classification performance in their respective domains. The two models operate in a continuous verification framework where every data access attempt, either in the form of a Portable Executable (PE) file upload or an SQL query, is verified in real-time before execution. This approach aligns with modern Zero Trust security architecture, ensuring that no data transaction is trusted by default, thereby providing a proactive defense mechanism against cyberattacks.

The visual analytics dashboard (Figure 2) effectively showcases this operational flow, distinguishing between benign and malicious traffic through color-coded panels. The real-time color transitions (yellow/orange for malicious and green/blue for benign) validate that detection occurs instantaneously—malware is blocked prior to execution and SQL injection attempts are denied before compromising the database. This dual-model structure therefore represents an intelligent intrusion prevention system combining static malware analysis with real-time web input sanitization. The Random Forest classifier achieved an overall accuracy of 99.13%, demonstrating superior predictive capacity and robustness. The AUC score of 0.9994 from the ROC curve indicates the model's nearly perfect discriminative ability between malware and benign files across varying thresholds. Similarly, the Precision (98.52%) and Recall (98.56%) metrics reinforce that the classifier not only detects nearly all malware samples but also produces minimal false alarms—an essential quality in security environments where false positives can lead to unnecessary quarantining of legitimate files or service disruptions.

The Precision–Recall curve (Figure 4) remaining close to the upper-right corner further confirms the balance between sensitivity and specificity. The high recall score suggests that the model rarely misses malware, while the high precision indicates that almost all flagged files are indeed malicious. This tradeoff balance is crucial in threat detection, as a single false negative could result in system compromise, data exfiltration, or persistent infiltration. From the confusion matrix (Figure 5), out of 27,610 total samples, 19,222 malicious files were correctly classified (True Positives), while only 123 were missed (False Negatives). Similarly, 8,144 legitimate files were correctly identified (True Negatives) with 121 false positives. These numbers reinforce the model's reliability and operational readiness. However, the 123 false negatives represent undetected malware instances—critical in cybersecurity since undetected threats may later propagate or evolve. This emphasizes the need for complementary defenses, such as behavioral monitoring or sandboxing, to address any residual blind spots. The integration of Explainable Artificial Intelligence (XAI) techniques—LIME and Permutation Feature Importance—provides interpretability to the otherwise opaque Random Forest model.

The LIME explanation (Figure 6) revealed that features such as MajorSubsystemVersion, Subsystem, SectionsMinEntropy, and Machine were strong indicators of maliciousness. High entropy values in PE sections often indicate packed or obfuscated code—a typical trait in malware that attempts to evade signature-based detection.

Conversely, features like VersionInformationSize and ResourcesMaxEntropy pushed predictions toward "Legitimate," suggesting that large and consistent metadata structures are typical of authentic, well-compiled applications.

The Permutation Feature Importance (Figure 7) confirmed MajorSystemVersion as the most influential feature globally, followed by ResourcesMinEntropy and SectionsMeanEntropy. This correspondence between local (LIME) and global (Permutation) interpretability strengthens confidence in the model's internal logic. Importantly, such insights enable security analysts to understand why specific files are flagged, supporting traceability and regulatory compliance. The malware detection model's high performance suggests it can effectively replace or complement traditional signature-based antivirus systems. While conventional methods rely on known hash patterns, this model identifies structural and statistical anomalies in PE files, making it resilient against zero-day attacks and metamorphic malware. The combination of entropy and subsystem analysis provides a more generalized detection mechanism adaptable to unseen threats.

The Feedforward Neural Network (FFNN) used for SQL Injection detection was trained on engineered features—query length, punctuation

density, and SQL keyword frequency—standardized through StandardScaler. Despite the model's simplicity, it achieved remarkable performance metrics: Accuracy: 99.73%, Precision: 99.7%, Recall: 99.95%, F1-Score: 99.8%

These metrics indicate near-perfect classification capability. The model's Recall of 99.95% means almost every SQL injection attempt was correctly detected, missing only one attack in the entire dataset. Such reliability is crucial since a single undetected SQLi vulnerability can expose entire databases to unauthorized data retrieval or corruption. Figures 8 and 9 depict both the Precision–Recall and ROC curves for the SQLi detection model. The PR curve maintains a precision value near 1.0 across all recall levels, confirming consistent accuracy even under varied decision thresholds. The ROC curve further demonstrates a strong separation between positive (attack) and negative (benign) samples, reinforcing the model's excellent True Positive Rate and minimal False Positive Rate. These outcomes suggest the FFNN efficiently generalizes from its training data, avoiding overfitting while maintaining operational reliability in live environments.

The confusion matrix (Figure 10) shows: True Positives (TP): 2186; True Negatives (TN): 385; False Positives (FP): 6; False Negatives (FN): 1. This distribution underscores that the classifier rarely misclassifies benign queries as attacks (minimizing user disruption) and almost never misses actual SQLi attempts. The combination of a near-zero false negative rate and a low false positive count reflects high precision with operational trustworthiness—key for automated intrusion detection systems (IDS) that run continuously in production environments.

The LIME visualization (Figure 11) illustrates how the neural network's decisions are grounded in interpretable feature contributions. For example, a query may be flagged as SQLi due to high keyword frequency (e.g., use of "UNION," "SELECT," or "DROP"), excessive punctuation, or anomalous query length. The model's 100% confidence in predicting a malicious query indicates a clear separation in feature space between normal and attack samples. Such interpretability enhances analyst trust and supports model debugging, auditing, and compliance verification.

Table 1 shows the comparative analysis of both models.

**Table 1: Comparative Analysis of RF Model and FFNN Model**

| Matrics | Malware Detection (Random Forest) | SQL Injection Detection (FFNN) |
|---|---|---|
| Accuracy | 99.13% | 99.73% |
| Precision | 98.52% | 99.7% |
| Recall | 98.56% | 99.95% |
| F1 Score | 98.54% | 99.8% |
| AUC | 0.9994 | 1.0 |

Table 1 shows that the Neural Network slightly outperforms the Random Forest, likely due to the simplicity and distinctiveness of SQL injection features. The Random Forest still performs exceptionally well despite dealing with more complex and heterogeneous malware data.

Both models demonstrate low error rates, strong generalization, and high interpretability when enhanced with XAI methods.

## 5. Conclusion

The study demonstrates the successful application of machine learning and explainable AI in enhancing cybersecurity defenses. The Random Forest model effectively distinguishes malware from legitimate PE files with strong interpretability, while the Feedforward Neural Network delivers near-perfect performance in identifying SQL injection attacks. Together, they provide a robust, interpretable, and automated framework capable of supporting real-time security validation across multiple threat vectors.

### References


1. Abdalla, H., Eltyeb, S. A., Elsamani, A., Ali, E., & Abdallah, R. E. (2022). An efficient model to detect and prevent SQL injection attack. https://doi.org/10.54388/jkues.v1i2.141

2. Ahmed, A. A., Atta, B., & Stahl, F. T. (2022). Multi-phase algorithmic framework to prevent SQL injection attacks using improved machine learning and deep learning to enhance database security in real-time. https://doi.org/10.1109/SIN56466.2022.9970504

3. Alarfaj, F. K., & Khan, N. A. (2023). Enhancing the performance of SQL injection attack detection through probabilistic neural networks. Applied

Sciences, 13(7), 4365. https://doi.org/10.3390/app13074365

4. Alghawazi, M., Alarifi, S., & Alghazzawi, D. (2022). Detection of SQL injection attack using machine learning techniques: A systematic literature review. Journal of Cybersecurity and Privacy, 2(4), 764–777. https://doi.org/10.3390/jcp2040039

5. Alshammari, M. (2023). Deep learning approaches to SQL injection detection: Evaluating ANNs, CNNs, and RNNs. https://doi.org/10.1117/12.3012620

6. Ayush, F., Manav, H., Henil, V., Priyank, M., & Deepa, K. (2022). A deep learning approach for detection of SQL injection attacks using convolutional neural networks. 293–304. https://doi.org/10.1007/978-981-16-6285-0_24

7. Bijitha, C. V., & Nath, H. V. (2021). On the effectiveness of image processing based malware detection techniques. Cybernetics and Systems, ahead-of-print(ahead-of-print), 615–640. https://doi.org/10.1080/01969722.2021.2020471

8. Chaudhary, H., Shah, P., Detroja, A., & Prajapati, P. (2020). A review of various challenges in cybersecurity using artificial intelligence. 829–836. https://doi.org/10.1109/iciss49785.2020.9316003

9. Gandhi, N., Mishra, S., Doshi, N., Patel, J., & Sisodiya, R. (2021). A CNN-BiLSTM based approach for detection of SQL injection attacks. 378–383. https://doi.org/10.1109/iccike51210.2021.9410675

10. Gorment, N. Z., Selamat, A., Cheng, L. K., & Krejcar, O. (2023). Machine learning algorithm for malware detection: Taxonomy, current challenges, and future directions. IEEE Access, 11, 141045–141089. https://doi.org/10.1109/access.2023.3256979

11. Gorgulu, A., & Kakisim, (2024). A deep learning approach based on multi-view consensus for SQL injection detection. International Journal of Information Security. https://doi.org/10.1007/s10207-023-00791-y

12. Hasan, M. M. (2024). Understanding model predictions: A comparative analysis of SHAP and LIME on various ML algorithms. Journal of Scientific and Technological Research, 5(1), 17–26. https://doi.org/10.59738/jstr.v5i1.23(17-26).eaqr5800

13. Hao, S., Yuejin, D., & Qi, L. (2023). Deep learning-based detection technology for SQL injection research and implementation. Applied Sciences. https://doi.org/10.3390/app13169466

14. Kaur, J., Miah, M., Hasan, S., Goffer, M., Barikdar, C., Orthi, S., & Hassan, J. (2024). Advanced cyber threats and cybersecurity innovation – Strategic approaches and emerging solutions. Journal of Computer Science and Technology Studies, 5(3), 112–121. https://doi.org/10.32996/jcsts.2023.5.3.9

15. Korade, D. (2024). Unlocking machine learning model decisions: A comparative analysis of LIME and SHAP for enhanced interpretability. Journal of Electrical Systems, 20(2s), 598–613. https://doi.org/10.52783/jes.1480

16. Kumar, P., & Pateriya, R. K. (2012). A survey on SQL injection attacks, detection and prevention techniques. 1–5. https://doi.org/10.1109/icccnt.2012.6396096

17. Li, Z., Zhu, H., Liu, H., Song, J., & Cheng, Q. (2024). Comprehensive evaluation of Mal-API-2019 dataset by machine learning in malware detection. International Journal of Computer Science and Information Technology, 2(1), 1–9. https://doi.org/10.62051/ijcsit.v2n1.01

18. Maha, A., Daniyal, M., Alghazzawi, D., & Alarifi, S. (2023). Deep learning architecture for detecting SQL injection attacks based on RNN autoencoder model. Mathematics. https://doi.org/10.3390/math11153286

19. Majid, Alshammari. (2023). 6. Deep learning approaches to SQL injection detection: evaluating ANNs, CNNs, and RNNs. doi: 10.1117/12.3012620.

20. Nnodi, J. T., & Obasi, E. C. M. (2025). Leveraging artificial intelligence for detecting insider threats in corporate networks. University of Ibadan Journal of Science and Logics in ICT Research, 13(1), 130–144.

21. Obasi, E., & Nlerum, P. (2020). Intrusion detection system for structured query language injection attack in e-commerce database. International Journal of Scientific and Research Publications, 10(8), 446–453. https://doi.org/10.29322/IJSRP.10.08.2020.P10455

22. Obasi, E. C. M., & Nlerum, P. A. (2023). A model for the detection and prevention of backdoor attacks using CNN with federated learning. University of Ibadan Journal of Science and Logics in ICT Research, 10(1), 9–21.

23. S, V., Taneem, A., Thoutam, S. Y., Apuri, S., & Md, S. W. (2024). Research on SQL injection attacks using word embedding techniques and machine learning. Journal of Sensors, IoT &

Health Sciences, 2(1), 55–64. https://doi.org/10.69996/jsihs.2024005

24. Stephan, L. (2023). SQL injection and its detection using machine learning algorithms and BERT. Lecture Notes in Computer Science, 3–16. https://doi.org/10.1007/978-3-031-28975-0_1

25. Timadi, M. E., & Obasi, E. C. M. (2025). Integrating zero-trust architecture with deep learning algorithm to prevent structured query language injection attack in cloud database. University of Ibadan Journal of Science and Logics in ICT Research, 13(1), 52–62.

26. Yan, S., Ren, J., Sun, L., Yu, Q., Wang, W., & Zhang, W. (2023). A survey of adversarial attack and defense methods for malware classification in cyber security. IEEE Communications Surveys & Tutorials, 25(1), 467–496. https://doi.org/10.1109/comst.2022.3225137.