Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 1 April 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 77-92
DOI: https://doi.org/10.51903/jtie.v5i1.481

# IoT-Driven in the Banking Application Platforms Using a Real-Time SQL Injection Mitigative Measures

Amaka Eugenia Ngozi*[1], Oji Victor Kalu[2], Ezea Jonathan Ikechukwu[3], Okpalla Chidimma Lilian[4], Gloria Ngozi Ezeh[5]
Email: 1ngozi.festus-amaka@futo.edu.ng, ojivictorkalu@gmail.com, iykeezea@gmail.com, chidimma.okpalla@futo.edu.ng, glorian.ezeh@futo.edu.ng
Orcid: https://orcid.org/0000-0001-7816-8290, https://orcid.org/0009-0004-1938-6588, https://orcid.org/0009-0009-1481-3559, https://orcid.org/0000-0002-0560-1871, https://orcid.org/0009-0002-2142-4067
[1,2]*Department of Cybersecurity, School of Information and Communication Technology, Federal University of Technology, Owerri, Imo State, Nigeria. 500201*
[3]*Department of Information Technology, First Bank Nigeria Ltd, 35 Marina Lagos.101241*
[4]*Department of Computer Science, School of Information and Communication Technology, Federal University of Technology, Owerri, Imo State, Nigeria. 500201*
[5]*Department of Information Technology, School of Information and Communication Technology, Federal University of Technology, Owerri, Imo State, Nigeria. 500201*
*Corresponding Author

**Abstract**
*The Internet of Things (IoT) integration into banking systems has revolutionized banking operations while also posing threats, including SQL injection (SQLi) attacks. Thus, the defenses of the existing system, such as access control mechanisms, firewalls, and signature-based Intrusion Detection Systems (IDSs), failed to detect both novel and obfuscated SQLi attempts. Hence, this research developed a machine-learning-based detection framework capable of identifying SQLi attacks on IoT-driven banking platforms. The model was trained on a Random Forest (RF) classifier and evaluated in a Python environment. Streamlit was used to deploy the model for real-time prediction, while performance visualization was through the Power BI dashboard. However, the results from the model's evaluation were highly impressive, with 99.53% accuracy, 99.96% precision, and 98.78% recall. This demonstrated the model's ability to detect both known and unknown SQL patterns. However, the research concluded that combining behavioural analytics with a machine-learning approach is highly effective for securing IoT banking platforms and recommended periodic retraining using a deep-learning approach.*

**Keywords**: *Intelligent Threat Detection, IoT-Driven Banking Platform, Real-Time SQL Injection.*

## I. INTRODUCTION

IoT-based banking platforms are persistently vulnerable to SQL injection (SQLi) attacks due to a lack of real-time adaptability and a poor understanding of behavioral changes (Abougrad, 2025; Demilie & Deriba, 2022). Consequently, this platform is severely compromised, exposing sensitive financial data, since authentication is not adequately secured (Ajayi et al., 2024; Mustapha et al., 2024; Trends, 2024). Hence, this abnormality in IoT-based banking platforms disrupts operational services, biases users' mindsets, and equally destroys the reputations of various financial platforms (Sathupadi et al., 2025; Yao et al., 2024).

*Received on Januari 2026; Revised on February 2026; Accepted March 2026; Published on April 2026*

However, the dynamic SQLi protection relies on firewalls such as Web Application Firewalls (WAFs), input validation/sanitization, prepared statements, RBACC, and signature-based IDS (George & Hasan, 2025; Kariuki et al., 2025). Conversely, these approaches provide limited basic protection (Aurna et al., 2024) and require an enhanced technique for better protection (Pratama & Nugroho, 2023). In a similar vein, static and rule-based nature rendered the approaches less effective against new or obfuscated injection patterns (Rahmawati et al., 2023), provide a limited configuration in the case of input validation, and lack universal coverage in the presence of RBAC (Al-olaqi et al., 2025; Eduardo & Gutierrez, 2024). On the other hand, IDS's haste in threat detection always provides a lot of false positives, unapproved zero-day detectors, and other limitations (Hartono et al., 2024; Poongodi et al., 2025). Meanwhile, the review and pre-test of the existing security mechanisms deployed in detecting SQL injection attacks in the banking platforms were conducted, the result obtained showed that the Random Forest model had a lower recall of (55.69%) in detecting malicious queries (Demilie & Deriba, 2022; Li et al., 2024; Mustapha et al., 2024), indicating many false negatives as shown in Figure 1, where malicious queries were classified as benign.

Thus, having identified the problems of the existing system, the researchers acknowledged the need for an intelligent, behavior-driven model capable of detecting SQLi threats in real time within the dynamic, high-risk environment of IoT-enabled banking platforms, using behavioral analytics and machine learning techniques (Raharjo et al., 2024). Furthermore, the dataset titled Modified_SQL_Dataset.csv, containing SQL queries for both legitimate and injection-based attacks in the banking platform domain, was collected and pre-processed using cleaning techniques such as lowercase conversion, digit normalization, special character filtering, and whitespace reduction. Additionally, a feature-extraction approach distinguishing malicious queries from benign ones was designed (Sumarlin & Qosidah, 2024), including model training, evaluation using standard metrics, and performance comparison (Suwardi et al., 2024). Moreso, visualization using Matplotlib, seaborn, and Power BI was employed to display performance metrics, confusion matrices, and comparative model results (Oktavia et al., 2024).

By integrating machine learning and behavioral analytics into IoT banking systems, this work makes a theoretical contribution to research on SQL injection detection. In practical terms, it offers an adaptive detection framework that may enhance security performance and reduce false detections in financial ecosystem results, encouraging the creation of more robust cybersecurity

*Journal of Technology Informatics and Engineering (JTIE)*
**Vol. 5 No. 1 April 2026**
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 77-92
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

standards for protecting IoT-enabled banking platforms from both a legislative and industry standpoint.

## II. RESEARCH METHOD

### *A. Research Tools/Materials*

The following tools were used to develop this research: Python (Primary development language), Scikit-learn (machine learning library), Pandas and NumPy, Matplotlib and seaborn, Apache NiFi, Power BI, MySQL/SQL DB server, and VS Code/PyCharm

### *B. Research Methodology*

Figure 1 presents the workflow stages for this research, starting with data collection, followed by data analysis and the relevant SQL queries for prediction. The collected data was pre-processed to remove noise and duplicates, and to standardize input formats. Feature extraction was also performed, converting the processed data into a suitable numeric representation for training. The trained model used the extracted features to build a capable predictive model for identifying SQL injection patterns. Lastly, the performance of the model was evaluated through the conduct of a post-test for the five trained models to ensure reliability in the model's predictions. This work combined a quantitative research method with experimental and supervised machine learning. The quantitative approach is ideal since it provides for objective measurement of model performance through numerical evaluation metrics.

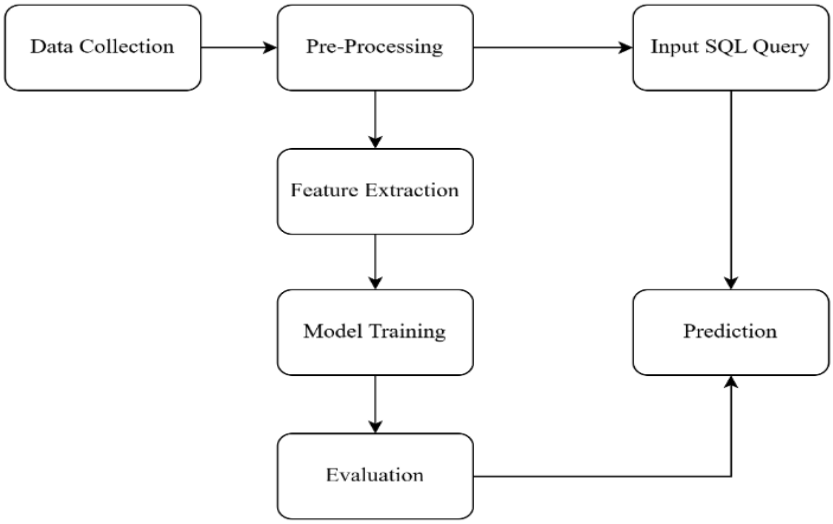


**Figure 1. Research Methodology**

### *C. Data Collection*

The dataset of 30,920 collected from SQL injection, dataset.csv, was used for this research. The dataset was obtained from a publicly available SQL injection repository. It consists of curated and

synthetic queries commonly used in SQL injection research and does not represent live banking traffic. The dataset was cleaned by removing duplicates, normalizing formatting, converting text to lowercase, and filtering special characters to ensure consistency for feature extraction. This dataset was split into 80:20 for model training and testing, respectively. Hence, 80% of 30,920 (data collected) was allotted to model training, maintaining a total of 24,736, while 20% of the total data collected (6,184), as stated above, was preserved for model testing. However, each query in the dataset was explicitly labeled as 1 for malicious and 0 for benign to enhance supervised learning and enable reliable performance evaluation during model detection.

The original dataset exhibited moderate class imbalance. To ensure fair model learning, stratified sampling was applied to create a balanced training set with equal representation of benign and malicious queries. While this improves experimental evaluation, real-world deployments may require threshold adjustment due to naturally imbalanced traffic patterns.

*D. Data pre-processing*

Raw SQL queries were filtered to enhance feature learning and minimize irrelevant variance. Thus, filtering the raw SQL queries was done in 3 different categories, such as:

1. Tokenization: each SQL query was separated into meaningful component keywords, operators, and variables, helping the system to focus on structure and intent.
2. Cleaning: special symbols, excessive whitespace, and other issues were normalized or removed.
3. Lowercasing: All elements were converted to lowercase, allowing the model to treat similar tokens equally regardless of the original casing.

*E. Feature Extraction*

The extracted features were translated into pre-processed text queries in a form the model would understand. TF-IDF Transformation (Term frequency (TF), Inverse Document Frequency (IDF)) messages were converted into vectors based on the frequency and distinctiveness of their tokens within the dataset. Hence, frequently used SQL commands (such as SELECT and FROM) were weighted differently from suspicious tokens. However, this approach highlighted anomalies in query structure and as well enabled the downstream classifier to differentiate between routine transactions and injection attempts.

*F. Model Training*

As previously noted regarding the dataset division, class balance and fairness were achieved by equally distributing the 80% of collected data across both malicious and benign queries. Nevertheless, each element contained 12,368, and the training process began after feature extraction. The process applied supervised learning to labeled, vectorized queries to identify patterns and classify SQL queries. Notably, the models trained on the extracted features were not limited to Random Forest, Logistic Regression, Naïve Bayes, Gradient Boosting, and SVM. Interestingly, each model trained adjusted its internal parameters to minimize misclassifications. TF-IDF vectorization was applied using an n-gram range of (1,2) and a maximum of 5,000 features. Stop-word removal was not applied to preserve SQL keywords. No dimensionality reduction technique was used. The TF-IDF generated vectors were directly supplied to all classifiers.

### *G. Evaluation and Testing*

An 80:20 train-test split was used across all classifiers: 20% of the dataset was set aside for testing, and 6,184 queries were used to validate the detection system's effectiveness. The key metrics used include accuracy for overall correctness, precision to measure the proportion of detected SQL injections that were malicious, recall to capture the proportion of actual attacks identified, and, lastly, the F1-Score, which balances precision and recall. Similarly, pinpointing the models' strengths and weaknesses in a real-world application leveraged insights gained from the distribution of True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN). On the other hand, Matplotlib was used to present the results of the confusion matrices and graphs. Hyperparameter tuning was conducted using 5-fold cross-validation on the training data. Final performance metrics were averaged over multiple runs to ensure consistency and reduce variance. The same evaluation protocol was applied uniformly to all models. The overall structure of the proposed detection framework and the evaluation pipeline is illustrated in Figure 2.

As shown in Figure 2, the model's architecture comprises multiple layers designed to effectively detect and mitigate SQL injection attacks. Hence, the detection layer employs preprocessing, feature extraction, and behavioral analytics, and is supported by machine learning models, such as logistic regression, SVM, Gradient Boosting, and Random Forest, which are combined via a voting classifier. Furthermore, the processed results were stored in the database layer, visualized for monitoring, and later passed to the mitigation layer to prevent future attacks. Thus, Figure 3 shows the SQL injection mitigation measures.

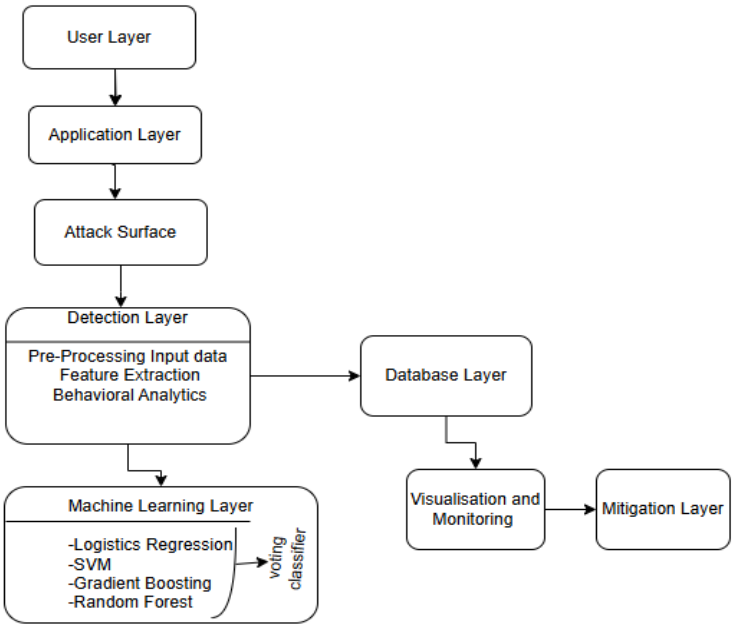


**Figure 2. Model Architecture**

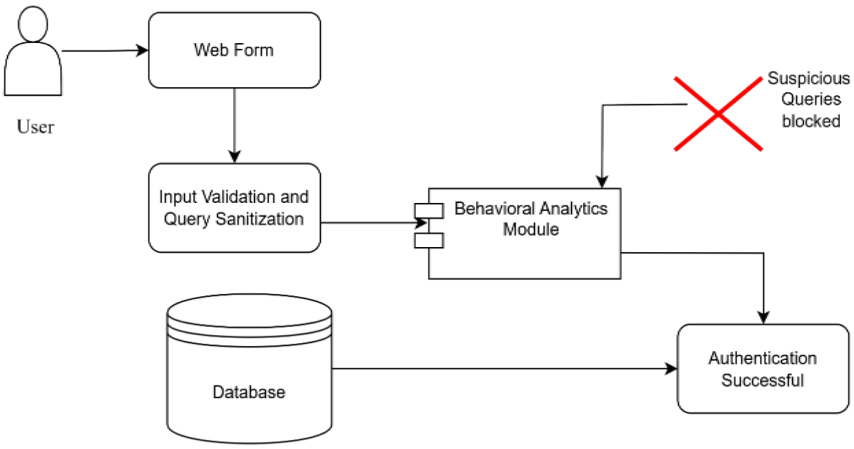


**Figure 3. SQL-Injection Mitigative Pattern**

A strategic layered defense was used as a mitigating measure to detect and prevent malicious queries that attempt to access the database. Thus, capturing user input as shown in the design was the primary approach to enable effective validation and sanitization to remove harmful characters. Similarly, the user input was successfully tokenized and normalized to extract meaningful features effectively. The machine learning model analyzed these features, while the Random Forest classifier was trained to effectively distinguish benign from malicious queries. Nonetheless, the trained model blocked and logged the detected malicious queries, while the legitimate queries proceeded to the database. Interestingly, the trained model maintained continuous learning on

new data to effectively monitor and report, thereby ensuring adequate protection against SQL injection attacks.

## III. RESULT

The atprimary goal was to train the machine learning model and evaluate its performance in detecting SQL injection attacks using a labeled dataset. It is important to note that results were obtained under controlled experimental conditions using the evaluated dataset. While the model demonstrates strong detection capability, these findings do not guarantee identical performance in real-world IoT banking environments, where traffic diversity and class imbalance may vary. Hence, the results obtained were presented thus:

### *A. Data Preparation and Model Development*

1. Data Preprocessing

The preprocessed data transformed the raw SQL queries into a suitable structural format for machine learning. The collected data from the dataset were initially cleaned to remove redundant symbols, convert text to lowercase, and normalize numerical values. This phase of data preprocessing, however, ensured smooth downstream analysis by removing inconsistencies and noise found in the raw data. Moreover, the application of tokenization splits SQL queries into smaller, more meaningful units to ensure that the model identifies recurring malicious patterns, including piggybacked, tautology-, statement-, and comment-based injections.

2. Feature Extraction

The transformed SQL queries were represented numerically using TF-IDF vectorization, and the frequently used terms in malicious injection. These frequently used terms include “OR”, “UNION”, and “DROP”, and were appropriately assigned higher weights due to their disproportionate appearances in the attack queries. However, this process maintained the model's focus on the most significant linguistic patterns that differentiate malicious from benign queries. The extracted feature aided model training by providing highly informative representations of the queries.

3. Model Training and Evaluation

A supervised learning technique was used to train the model on a labeled dataset. The training involved teaching the model to understand and recognize different types of SQL injection attempts. The dataset was split into two categories, with 80% for training and 20% for testing. The models trained on the extracted features were Random Forest, Logistic Regression, Naïve Bayes, Gradient Boosting, and SVM. The model was adjusted during training to internal

parameters to minimize miscalculations, while hyperparameter tuning improved accuracy, precision, and recall. Notably, another achievement in the training was optimizing the models for real-time detection of malicious query structures.

Nonetheless, the trained models were tested on the dataset's 20% test set, and performance was evaluated in real-world scenarios, using the standard evaluation metrics of accuracy, precision, recall, and F1-score. These metrics, as calculated for precision, identified how the frequencies of detected injections were maliciously ascertained, while the metrics for recall identified the proportion of actual attacks successfully detected. The models tested include Logistic Regression, Linear Support Vector Machine (SVM), Gradient Boosting Classifier, Naive Bayes (MultinomialNB), and Random Forest Classifier.

*B. Machine Learning Model Performance*

1. Logistic Regression

Logistic Regression is a linear model that predicts probabilities of class membership. In this experiment, it achieved an accuracy of 99.32%, meaning that out of every 100 queries, fewer than 1 were misclassified, as shown in Figure 4.

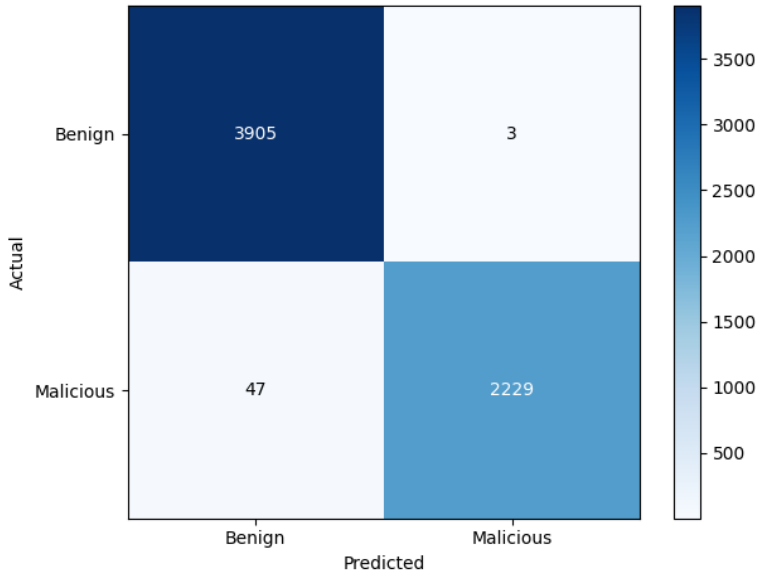


**Figure 4. Logistic Regression**

2. Linear Support Vector Machine (SVM)

The Linear SVM finds the best hyperplane to separate benign from malicious queries. It achieved an accuracy of 99.42%, slightly outperforming Logistic Regression as depicted on Figure 5.

*Journal of Technology Informatics and Engineering (JTIE)*
**Vol. 5 No. 1 April 2026**
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 77-92
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

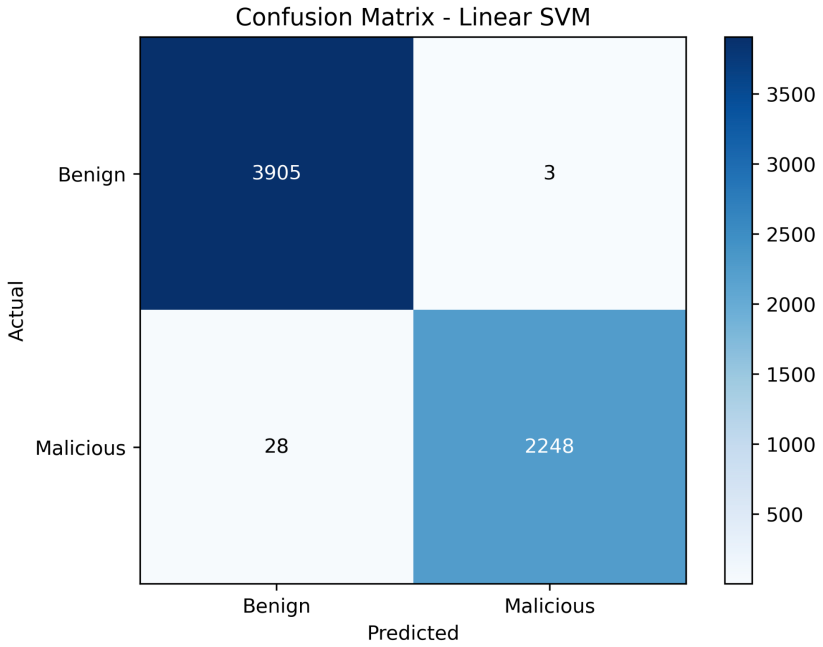


**Figure 5. Linear SVM**

3. Gradient Boosting Classifier

Gradient Boosting builds decision trees sequentially, where each tree tries to correct the errors of the previous one. It reached an accuracy of 98.79% as captured in Figure 6

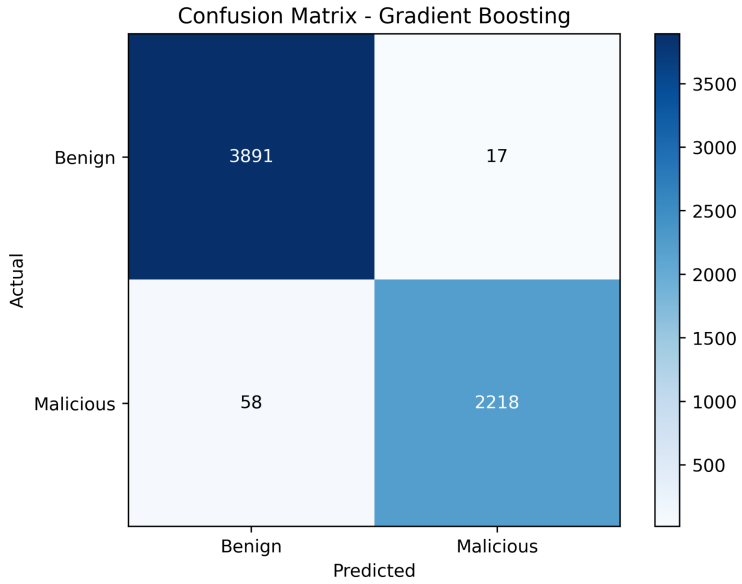


**Figure 6. Gradient Boosting Classifier**

4. Naive Bayes (MultinomialNB)

Naive Bayes uses probabilities based on independence assumptions among features. It achieved an accuracy of 97.91%, which is still good but lower than other models tested, as shown in Figure 7.

**Table 1. Model's Post-test results**

| No | Model | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|---|
| 1 | Logistic Regression | 0.991915 | 0.998656 | 0.979350 | 0.988909 |
| 2 | Linear SVM | 0.994987 | 0.998667 | 0.987698 | 0.993152 |
| 3 | Gradient Boosting | 0.987872 | 0.992394 | 0.974517 | 0.983374 |
| 4 | Naive Bayes | 0.988195 | 0.995056 | 0.972759 | 0.983781 |
| 5 | Random Forest | 0.995472 | 0.999112 | 0.988576 | 0.993816 |
| 6 | Voting Classifier | 0.994987 | 0.998667 | 0.987698 | 0.993152 |

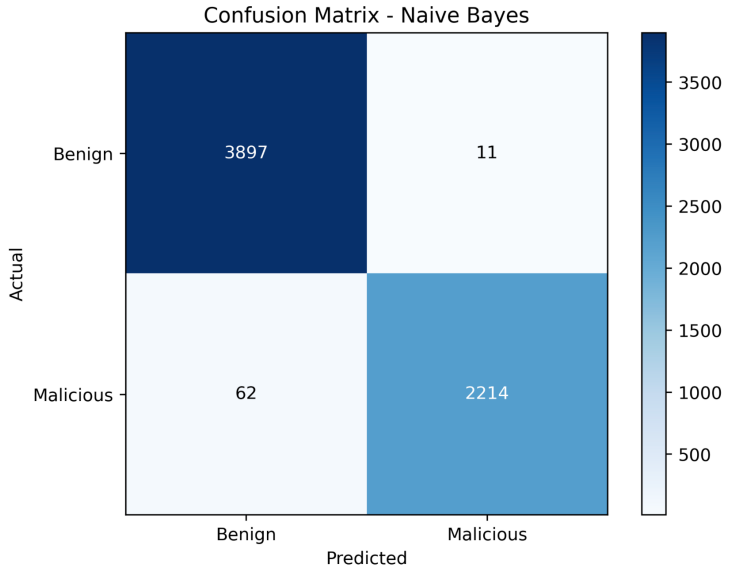


**Figure 7. Naive Baye**

5. Random Forest Classifier

Random Forest builds multiple decision trees and takes a majority vote for classification. It achieved the highest accuracy as demonstrated on Figure 8.

*Journal of Technology Informatics and Engineering (JTIE)*
**Vol. 5 No. 1 April 2026**
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 77-92
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

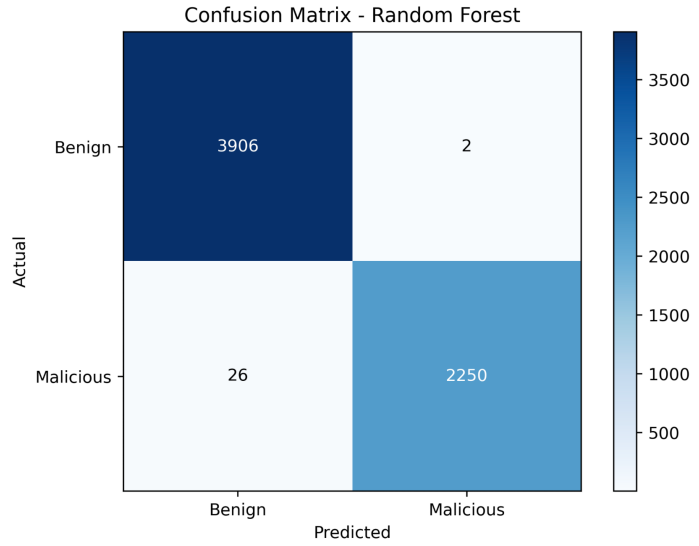


**Figure 8. Random Forest Classifier**

6. Training Voting Classifier (Ensemble)

These results show that combining multiple models (Logistic Regression, Linear SVM, Gradient Boosting, Naïve Bayes, and Random Forest) into an ensemble produced a balanced and reliable detection mechanism, as shown in Figure 9.

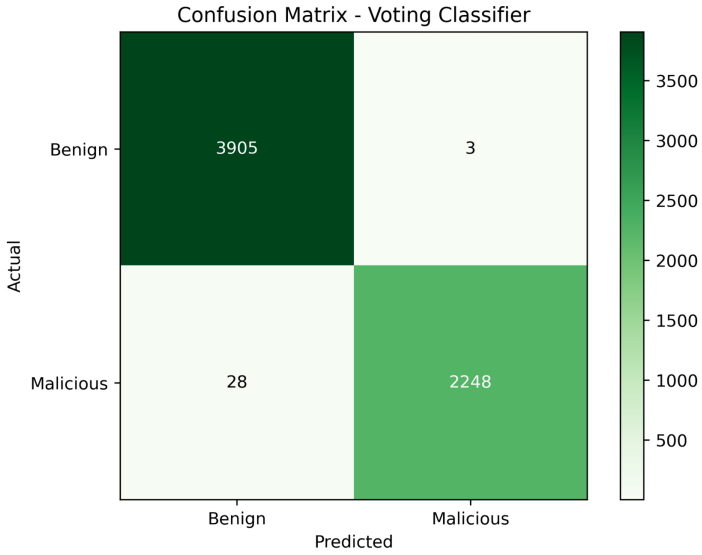


**Figure 9. Training Voting Classifier (Ensemble)**

*C. Models Comparison Results*

The ensemble method used to compare the five models (Logistic Regression, Linear Support Vector Machine (SVM), Gradient Boosting Classifier, Naive Bayes (MultinomialNB), Random Forest Classifier) was Voting. To evaluate and compare the performance of five machine learning algorithms in detecting SQL injection attacks using a labeled dataset

The post-test results show the performance of all trained models, evaluated using accuracy, precision, recall, and F1-score. From Table 1 and Figure 10, Random Forest emerged as the best-performing model with an accuracy of 99.55%, precision of 99.91%, recall of 98.86%, and F1-score of 99.38%, making it the most balanced and reliable classifier for detecting SQL injection attacks. An accuracy of 99.50% was obtained using Linear SVM and a voting classifier, indicating robust results from both ensemble methods and margin-based learning. On the other hand, the 99.19% accuracy achieved by Logistic Regression demonstrated strong performance. In comparison, the 98.80% accuracies obtained from Gradient Boosting and Naïve Bayes maintained high performance but were slightly lower than that of Logistic Regression.

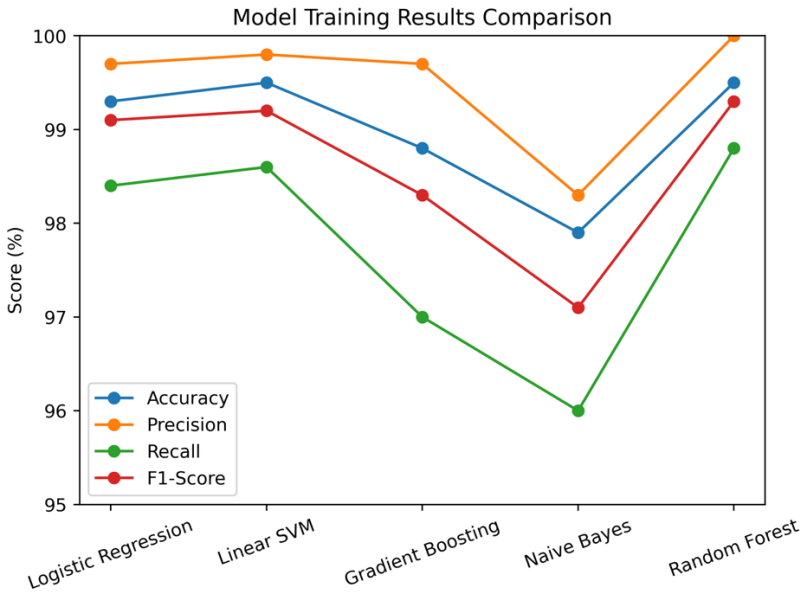


**Figure 10. Model's Graph Results**

The findings of this study indicate that although the Voting Classifier combined multiple models, it did not significantly outperform Random Forest, since Random Forest is itself an ensemble of decision trees. Additionally, the base learners relied on similar TF-IDF features, resulting in correlated predictions. Therefore, Random Forest offers comparable performance with lower computational complexity for real-time deployment.

Thus, integrating machine learning with behavioral analytics substantially enhances SQL injection detection in IoT-enabled banking platforms. Among the evaluated models, the Random Forest classifier showed superior performance, indicating a strong ability to capture complex, non-linear attack patterns typical of modern SQL injection techniques. This outcome directly supports the research's objective of improving real-time detection, particularly for previously

unseen attacks. While earlier studies have reported improvements using machine learning-based intrusion detection, Random Forest models in those works often suffered from high false-negative rates; the improved recall observed here therefore represents an important methodological advancement in the IoT banking context.

From a theoretical standpoint, the results reinforce the growing shift away from static, rule-based security mechanisms toward behavior-driven anomaly detection models. Practically, the proposed framework offers a viable and scalable approach for financial institutions seeking to strengthen database security and protect sensitive information. Nevertheless, the use of a single dataset and offline evaluation limits the extent to which the findings can be generalized. Future studies should consider incorporating diverse real-world datasets and online or deep learning techniques to further improve robustness against evolving SQL injection attacks.

## IV. DISCUSSION

The model was evaluated in an offline experimental environment using a pre-collected and labeled dataset. While this setup enabled controlled performance assessment, it does not fully replicate the dynamic, latency-sensitive conditions of real-time IoT banking systems.

The study relied on a single primary dataset containing labeled SQL queries. Although the dataset was balanced and carefully preprocessed, using a single dataset may limit the model's exposure to diverse attack variations and real-world query distributions.

The framework was not empirically validated within a live IoT banking deployment environment. Therefore, factors such as real-time system overhead, scalability at high transaction volumes, and integration constraints were not measured experimentally.

## V. CONCLUSION AND RECOMMENDATION

This study developed and evaluated a machine learning-based SQL injection detection framework. By integrating behavioral analytics with supervised learning techniques, the system effectively distinguished between benign and malicious SQL queries. Among the evaluated models, the Random Forest classifier achieved the best performance, recording 99.55% accuracy, 99.91% precision, and 98.86% recall, demonstrating strong capability in identifying both known and previously unseen attack patterns.

The results confirm that behavior-driven machine learning approaches provide a more adaptive and scalable alternative to traditional rule-based and signature-based security mechanisms. The framework offers practical value for enhancing database protection in financial systems, reducing false detections, and improving overall resilience against evolving SQL injection threats.

Thus, given the offline evaluation setting and reliance on a single dataset, further validation in live IoT banking environments is recommended to fully establish scalability and operational robustness.

Further research is recommended, using larger and more diverse datasets drawn from a real IoT-based banking environment, to determine how well the model generalizes beyond the current experimental setup. It would also be valuable to explore online or incremental learning techniques to enable the system to adapt to new and evolving SQL injection patterns over time. In addition, investigative deep learning models would be recommended to provide further insight into complex query structures. Lastly, evaluating the framework on a live banking platform is recommended to better understand its scalability, latency, and operational impact.

## REFERENCES

AbouGrad, H., & Sankuru, L. (2025). Online Banking Fraud Detection Model: Decentralized Machine Learning Framework to Enhance Effectiveness and Compliance With Data Privacy Regulations. *Mathematics*, *13*(13), 2110. https://doi.org/10.3390/math13132110

Ajayi, A. M., Omokanye, A. O., Olowu, O., Adeleye, A. O., Omole, O. M., & Wada, I. U. (2024). Detecting Insider Threats in Banking Using AI-Driven Anomaly Detection with a Data Science Approach to Cybersecurity. *International Journal of Advanced Computer Science and Applications*, *15*(6), 1097–1106. https://doi.org/10.14569/ijacsa.2024.01506116

Al-Olaqi, M., Al-Gailani, A., & Rahman, M. M. H. (2025). Comprehensive Study of SQL Injection Attacks Mitigation Methods and Future Directions. *Journal of Cyber Security and Risk Auditing*, *2025*(4), 347–365. https://doi.org/10.63180/jcsra.thestap.2025.4.11

Aurna, N. F., Hossain, D., Ochiai, H., Taenaka, Y., & Khan, L. (2024). Banking Malware Detection: Leveraging Federated Learning with Conditional Model Updates and Client Data Heterogeneity. *Proceedings of the International Conference on Information Systems Security and Privacy (ICISSP)*, 309–319. https://doi.org/10.5220/0012409700003648

Demilie, W. B., & Deriba, F. G. (2022). Detection and Prevention of SQLI Attacks and Developing Compressive Framework Using Machine Learning and Hybrid Techniques. *Journal of Big Data*, *9*, 148. https://doi.org/10.1186/s40537-022-00678-0

Eduardo, C. E. C. G. (2024). Branchless Banking: The Role of Fintech Technologies and the Internet of Things (IoT) in the Disruption of the Traditional Banking Model. *International Journal of Multidisciplinary Research and Growth Evaluation*, *5*(5), 586–591. https://doi.org/10.54660/.ijmrge.2024.5.5.586-591

George, Z. H., & Hasan, T. (2025). Assessing the Influence of Cybersecurity Threats and Risks on the Adoption and Growth of Digital Banking: A Systematic Review. *Finance and Banking Review*, *1*(1), 226–257. https://doi.org/10.63125/fh49gz18

Gottipati, K. C. (2024). 5G-Driven IoT in Banking: Revolutionizing Real-Time Transaction Processing. *International Journal of Fintech (IJFT)*, *3*(1), 1–14. https://doi.org/10.5281/zenodo.14258588

Hartono, B., Silalahi, F. D., & Muthohir, M. (2024). Transformers in Cybersecurity: Advancing Threat Detection and Response Through Machine Learning Architectures. *Journal of Technology Informatics and Engineering*, *3*(3), 382–396. doi:10.51903/jtie.v3i3.211

Hidayat, M. S., Aziz, F., & Mansur, R. (2025). Digital Ethics and AI Transparency: Comparative Analysis of AI Recommendation Systems Across E-Commerce Platforms. *Journal of Management and Informatics*, *4*(1), 178–199. doi:10.51903/jmi.v4i1.178

Kakolu, S., Faheem, M. A., & Aslam, M. (2023). AI-Enabled Intrusion Detection Systems in IoT Networks: Advancing Defense Mechanisms for Resource-Constrained Devices. *International Journal of Science and Research Archive*, *9*(1), 752–769. https://doi.org/10.30574/ijsra.2023.9.1.0316

Kariuki, P., Oluwatoyin, L., Lauda, M., & Goyayi, J. (2025). Internet of Things on Banking Processes in South Africa : A Systematic Reflection on Innovations, Opportunities and Challenges. *Digital Business*, *5*(1), 100097. https://doi.org/10.1016/j.digbus.2024.100097

Kumbhar, S. B., Mundkar, N. K., Ohol, A. A., Patel, G. A., & Dhanake, S. A. (2024). Enhancing Financial Security- An Intrusion Detection Approach. *International Journal of Novel Research and Development*, *9*(5), b535–b540. https://ijnrd.org/papers/IJNRD2405168.pdf

Li, X., Wang, Q., Fan, C., Zhan, W., & Zhang, M. (2024). A New Malicious Code Classification Method for the Security of Financial Software. *Computer Systems Science and Engineering*, *48*(3), 773–792. https://doi.org/10.32604/csse.2024.039849

Mustapha, A. A., Udeh, A. S., Ashi, T. A., Sobowale, O. S., Akinwande, M. J., & Oteniara, A. O. (2024). Comprehensive Review of Machine Learning Models for SQL Injection Detection in E-Commerce. *World Journal of Advanced Research and Reviews*, *23*(1), 451–465. https://doi.org/10.30574/wjarr.2024.23.1.2004

Ogun, J. O. (2024). Advancements in Automated Malware Analysis: Evaluating the Efficacy of Open-Source Tools in Detecting and Mitigating Emerging Malware Threats to US Businesses. *International Journal of Science and Research Archive*, *12*(2), 1958–1964. https://doi.org/10.30574/ijsra.2024.12.2.1488

Poongodi, E., Mahesh, P., Venkatesh, P., & Venkateshwarlu, R. (2025). Fraud Detection in Banking Data Using Machine Learning. *Journal of Engineering Sciences*, *16*(05), 653–659. https://doi.org/10.36893/jes.2025.v16i05.070

Pratama, A., & Nugroho, B. (2025). Effectiveness and Reliability of Artificial Intelligence in Fraud Detection: A Mixed-Method Study on Financial Audit. *Journal of Management and Informatics*, *4*(1), 168–185. doi:10.51903/jmi.v4i1.168

Sathupadi, K., Achar, S., Bhaskaran, S. V., & Faruqui, N. (2025). BankNet: Real-Time Big Data Analytics for Secure Internet Banking. *Big Data and Cognitive Computing*, *9*(2), 24. https://doi.org/10.3390/bdcc9020024

Sumarlin, T., & Qosidah, N. (2025). Optimizing Sales and Inventory Management With Machine Learning: Applications of Neural Networks and Random Forest. *Journal of Management and Informatics*, *4*(2), 35–48. doi:10.51903/jmi.v4i2.35

Supriadi, C., Wahyudi, W., & Priyadi, A. (2025). Decentralized AI on the Edge: Implementing Federated Learning for Predictive Maintenance in Industrial IoT Systems. *Journal of Technology Informatics and Engineering*, *4*(2), 317–336. doi:10.51903/jtie.v4i2.281

Suwardi, R., Fadli, A., & Putri, V. (2025). Model Training and Performance Evaluation Using Standard Metrics in Predictive Analytics. *Journal of Management and Informatics*, *4*(2), 57–74. doi:10.51903/jmi.v4i2.57

Taufik, M., Aziz, M. S., & Fitriana, A. (2025). Hybrid Explainable AI (XAI) Framework for Detecting Adversarial Attacks in Cyber-Physical Systems. *Journal of Technology Informatics and Engineering*, *4*(1), 157–171. doi:10.51903/jtie.v4i1.295

Trends, E. (2024). Goranin, N., Hora, S. K., & Čenys, H. A. (2024). A Bibliometric Review of Intrusion Detection Research in IoT: Evolution, Collaboration, and Emerging Trends. *Electronics*, *13*(16), 3210. https://doi.org/10.3390/electronics13163210

Yao, S., Liu, D., Guo, Z., Zhang, Z., & Hu, J. (2024). Spotting Sneaky Scammers: Malicious Account Detection From a Chinese Financial Platform. *Electronics*, *13*(23), 4742. https://doi.org/10.3390/electronics13234742