Detection of SQL Injection: A Comparative Analysis of Machine Learning and Deep Learning Algorithms

Tiago Martins Ferreira
São Paulo State University (UNESP)
Bauru, Brazil
tiago.ferreira@unesp.br

Thiago José Lucas
São Paulo State Univ. of Tech. (FATEC)
Ourinhos, Brazil
thiago@fatecourinhos.edu.br

Carlos Alexandre Carvalho Tojeiro
São Paulo State University (UNESP)
Bauru, Brazil
carlos.tojeiro@unesp.br

Carlos Eduardo Silva Bertazzoli
São Paulo State University (UNESP)
Bauru, Brazil
carlos.bertazzoli@unesp.br

Alessandra de Souza Lopes
São Paulo State University (UNESP)
Bauru, Brazil
alessandra.lopes@unesp.br

Ana Caroline Silva Pontara
São Paulo State University (UNESP)
Bauru, Brazil
ana.pontara@unesp.br

***Abstract*—This study investigates the effectiveness of Machine Learning (ML) and Deep Learning (DL) models in detecting SQL Injection (SQLi) attacks. Using a labeled dataset, techniques such as TF-IDF with n-grams and SMOTE for class imbalance were applied. The models evaluated include Naive Bayes, Random Forest, Support Vector Machine (SVM), and a Bi-LSTM network. Results indicated that the Bi-LSTM achieved the best performance (accuracy of 98.15%), followed by Random Forest and SVM. The research demonstrates that DL-based approaches, combined with appropriate preprocessing, are effective for SQLi detection, surpassing traditional models in accuracy and generalization capability. It is concluded that these techniques are promising for deployment in real-world environments, provided computational costs and practical implementation challenges are taken into account.**

***Keywords*—SQL Injection, Machine Learning, Deep Learning, Bi-LSTM, TF-IDF**

I. Introduction

The growth of digital services in recent years has transformed the way individuals and organizations interact with computational systems. Web applications have become responsible for banking transactions, academic records, healthcare systems, institutional communication, and various other critical processes. This increasing digitization has amplified both the volume and sensitivity of transmitted data, making databases essential for modern applications [1]. Consequently, ensuring the integrity, confidentiality, and availability of information has become crucial for the operation of services and institutions. However, this technological advancement has brought substantial challenges in the field of information security, particularly concerning protection against unauthorized access, malicious alterations, and data leaks [2]. Studies show that the majority of web applications are still deployed without mechanisms that guarantee security. According to Edgescan [3], approximately 1 in 3 vulnerabilities detected in web applications, APIs, and infrastructure were classified as critical or high, highlighting the need for more effective protective measures. This situation is exacerbated by the lack of rigorous input validation processes, creating openings for attacks targeting database query manipulation. Such issues often reflect unintentional negligence by developers, who prioritize functionality over security [4], [5]. Additionally, the increasing adoption of technologies such as cloud computing, 5G, and the Internet of Things (IoT) has considerably expanded the attack surface, generating new exploitation vectors for malicious actors [6]. The current context, therefore, necessitates proactive, automated, and adaptive solutions for detecting and mitigating vulnerabilities in web applications.

Among the various vulnerabilities affecting web applications, SQL Injection (SQLi) stands out as one of the most recurrent and exploited over time. This type of attack occurs when user-provided inputs are insecurely incorporated into SQL statements, allowing arbitrary commands to be executed on the database. Between 2021 and 2024, SQLi attacks were associated with a series of significant security breaches on widely used platforms such as WooCommerce, BillQuick, Django, MOVEit, Fortra FileCatalyst Workflow, and Ivanti Endpoint Manager. These incidents resulted in consequences such as data theft, malware installation, compromise of confidentiality, integrity, and availability of information, authentication bypass, remote code execution, and negative impacts on reputation and regulatory compliance [7]. The problem largely lies in the absence of effective input validation mechanisms and the direct concatenation of user-supplied data in SQL statements

[8]. Even a simple login interface can become vulnerable if the data is not properly handled.

Despite numerous warnings and the existence of established best practices, such as the use of parameterized queries and prepared statements, SQLi continues to rank among the top security risks according to rankings like the OWASP Top Ten [1]. This persistence is not only due to negligence in implementing preventive measures but also to the evolution of attacks, which have become increasingly sophisticated, capable of bypassing static filters and simulating legitimate traffic [9]. Furthermore, the public and open nature of web applications allows malicious actors to exploit them extensively, including through automated bots and obfuscation techniques that disguise malicious queries as legitimate [10] apud [11].

This scenario reveals that approaches based solely on fixed rules and signatures are insufficient to handle the current complexity of attacks. Since SQLi no longer relies solely on rudimentary techniques but includes evasion strategies and exploitation of logical vulnerabilities, it becomes evident that more flexible and intelligent detection methods are required. The challenge, therefore, is not only to identify previously known patterns but also to recognize subtle, adaptive, and disguised variations of malicious behavior.

Given the persistence of SQL Injection and the limitations of traditional defense methods, the scientific community and industry have increasingly invested in machine learning (ML) and deep learning (DL) approaches as more effective alternatives. These techniques enable the analysis of large volumes of data and the identification of anomalous patterns based on features extracted directly from SQL queries, even when they do not match previously known signatures [12]. Unlike static mechanisms, ML-based models can be trained with real examples of attacks and legitimate queries, learning to distinguish between them autonomously and adaptively [6].

One of the main advantages of these approaches is their generalization capability, allowing the detection of novel attacks or sophisticated variants. Additionally, techniques such as TF-IDF vectors with n-grams or semantic embeddings like Word2Vec have been explored to capture syntactic and semantic nuances in SQL statements, enriching model inputs and improving accuracy [8]. However, model performance can be impacted by issues such as class imbalance, as attack instances often represent a minority in datasets. In this context, oversampling algorithms like SMOTE have been used to enhance the representation of the minority class and improve model generalization [13], [14].

The evolution of techniques also includes the application of deep models, such as LSTM networks and stacked architectures, which can capture contextual dependencies and temporal patterns in token sequences [15]. Recent studies indicate that the combined use of preprocessing techniques, data balancing, and DL models can significantly increase detection rates and reduce false positives, offering a promising approach for SQLi detection in real environments [16]. Thus, a comparative analysis between traditional ML models and more recent DL approaches presents a relevant strategy to identify more effective and applicable solutions for web application security.

Despite advances in detection models, the practical adoption of ML and DL techniques still faces significant challenges. One of the main obstacles is dataset imbalance, where attack records constitute a minority of samples. To address this issue, techniques like SMOTE are widely employed to generate synthetic instances of the minority class, improving model generalization capabilities [13], [14]. Another critical aspect is the vector representation of SQL queries. Strategies such as TF-IDF with n-grams are effective in pattern extraction but may fail to capture the underlying semantics of SQL statements, which compromises the distinction between benign and malicious queries [6]. Models such as Bi-LSTM networks, on the other hand, can process these sequences contextually but require greater computational resources and are sensitive to hyperparameter configurations [15]. Factors such as feature selection, prevention of overfitting, and feasibility in production environments also impact performance. Therefore, careful construction of the data pipeline, class balancing, and appropriate architecture choice are essential for the success of these approaches [16].

Considering these challenges, a comparative evaluation of different ML and DL models applied to SQLi detection is necessary. Current literature exhibits gaps regarding joint analysis of traditional algorithms, deep neural networks, and data preparation techniques in a single experiment [6]. Models such as Random Forest and SVM are recognized for robustness and strong performance with balanced datasets and well-defined vectors. Architectures like Bi-LSTM, however, can identify subtler patterns, advantageous against obfuscated attacks, though they require higher computational effort and careful parameter tuning [15]. In this context, this study proposes a systematic comparison of different models and preprocessing strategies, such as SMOTE and TF-IDF with n-grams, to evaluate which combinations offer the best balance between performance and practical feasibility. This approach aims to contribute to the scientific community by providing guidance for informed decisions in developing automated SQLi detection mechanisms. The study combines experimental rigor with critical result analysis to pursue safer and more efficient applications in real-world web system environments.

In this context, this study is justified by the relevance of

SQL Injection as a threat and the growing sophistication of attack techniques, which often overcome traditional signature-based detection mechanisms. The use of ML and DL techniques emerges as a promising alternative, as it enables the identification of complex patterns and anomalous behavior in SQL queries not captured by conventional approaches. However, despite the broad application of these technologies in related cybersecurity fields, their use in SQLi detection still lacks systematic investigations regarding effectiveness and applicability in realistic scenarios. Therefore, this work is justified by the need to compare different modeling strategies in a structured manner, providing technical support for more effective decisions in protecting database-based systems.

Based on this motivation, the following hypothesis is formulated: more sophisticated DL models, such as Bi-LSTM, when combined with appropriate preprocessing techniques, perform better in detecting SQL Injection compared to traditional ML models. To test this hypothesis, the present study was designed with the general objective of investigating and comparing the performance of various ML and DL models in SQLi detection, applied to a labeled set of SQL queries.

Specifically, the study aims to: (i) evaluate the effectiveness of data preprocessing techniques, such as SMOTE for class balancing and TF-IDF vectorization with n-grams, in improving model performance; (ii) implement and analyze traditional ML models, such as Naive Bayes, Random Forest, and Support Vector Machine, in classifying SQL statements as legitimate or malicious; (iii) explore the capability of a DL model based on a Bi-LSTM architecture for SQLi detection; and (iv) perform a systematic comparison between models, considering performance metrics, observed limitations, and practical applicability in production environments. The study structure is organized to ensure experiment replicability and critical analysis of results in the context of existing literature.

II. Theoretical Background

SQL Injection is a widely recognized attack technique that exploits vulnerabilities in web applications to manipulate SQL queries sent to the database. The attack occurs when user-provided inputs are not properly validated, allowing malicious commands to be inserted into queries and executed on the application backend [8], [17]. As a result, the attacker can access confidential data, modify or delete information, and even execute administrative commands. The origin of this vulnerability is often associated with the direct use of user input in SQL statements without secure mechanisms such as parameterized queries or Object-Relational Mapping (ORM). Although this threat has been known for decades, it remains among the top risks identified by OWASP due to its ease of execution and potential impact on data integrity and system security [18].

SQL Injection attacks are commonly categorized into three groups: In-band SQLi, Inferential SQLi, and Out-of-band SQLi. The In-band type is the most direct, using the same communication channel to send the malicious payload and receive results. Within this category, Error-based attacks exploit error messages to gain information about the database structure, while Union-based attacks use the UNION clause to extract additional data [18]. Inferential SQLi, also known as Blind SQLi, does not return data directly but allows information to be inferred by observing application behavior. It is divided into Boolean-based, which uses logical conditions to test hypotheses, and Time-based, where the response is inferred through intentional delays using functions like SLEEP() [19]. Finally, Out-of-band SQLi is used when traditional channels are unreliable, leveraging external means such as HTTP requests to an attacker-controlled server to transmit data indirectly. Each technique has distinct characteristics and requires specific mitigation strategies, making understanding them essential for the development of effective detection and prevention mechanisms [18], [20].

SQLi prevention traditionally involves secure coding practices, such as using Prepared Statements, Stored Procedures, and rigorous input validation. However, the complexity of applications and the constant search for new vulnerabilities make automated detection an active area of research.

A. Machine Learning in Threat Detection

Machine Learning (ML) is a subfield of artificial intelligence that aims to develop algorithms capable of learning automatically from data without relying on explicit programming. This adaptive capability makes ML a valuable tool in cybersecurity, especially in dynamic and complex environments such as virtual attack defense systems [21]. By processing large volumes of data, such as network traffic logs, security events, and user behavior patterns, ML models can identify anomalous patterns and anticipate malicious behavior [16]. One advantage of these approaches is their applicability across different paradigms: supervised, unsupervised, and reinforcement learning, each with specific purposes in information security contexts [6]. Additionally, ML-based systems have demonstrated efficiency in detecting previously unknown threats, such as zero-day attacks, using pattern recognition and anomaly detection techniques [6].

The main ML algorithms employed in this study include:

- **Naive Bayes:** a widely used probabilistic method based on Bayes' theorem, assuming independence between input attributes. Its simple model, known as the "independent feature model," facilitates the classification of large text volumes efficiently, though with limitations in representing attribute dependencies [22].

- **Random Forest:** an ensemble learning algorithm composed of multiple decision trees. Each tree performs an individual classification, and the final result is determined by majority voting. This approach provides greater stability and generalization, especially useful for detecting structural variations in malicious SQL queries [22].
- **Support Vector Machine:** employs a mathematical approach that seeks to construct a hyperplane with the largest possible margin between different classes. It is particularly effective for binary classification tasks and high-dimensional environments, widely adopted for separating legitimate and malicious patterns [23].

B. Deep Learning and Recurrent Neural Networks (RNNs)

Deep Learning (DL) is a subarea of ML that leverages deep neural network architectures to identify complex patterns in large volumes of data. Its strength lies in automatically extracting relevant representations from raw data without manual feature engineering. This approach has shown significant results in cybersecurity, particularly in scenarios where attacks involve nonlinear patterns and temporal features that are difficult to capture with traditional methods [24]. Recurrent Neural Networks (RNNs) are particularly suitable for sequence analysis, as they consider temporal dependencies in data, such as the succession of commands in an SQL query. In intrusion detection systems, RNNs and their variants can identify anomalous behavior over time, even when masked by normal activity, which is essential for detecting persistent and sophisticated attacks like APTs [6].

1) Long Short-Term Memory (LSTM): LSTM networks represent an evolution of traditional RNNs, addressing short-range problems such as vanishing gradients. They are effective in identifying long-term patterns, maintaining memory of relevant past events for classifying new inputs. This characteristic makes LSTM models particularly useful for detecting distributed SQL Injection, such as Blind SQLi attacks [6].

2) Bidirectional LSTM (Bi-LSTM): The Bi-LSTM architecture extends the LSTM model by allowing sequences to be processed in both directions, past-to-future and future-to-past, providing a richer contextual understanding. This capability is crucial for SQL query analysis, where token positions influence meaning. Studies show that combining Bi-LSTM with vectorization techniques such as TF-IDF is effective for detecting malicious patterns in SQL queries [20].

C. Text Preprocessing and Data Balancing Techniques

The success of predictive models heavily depends on proper data preparation. In the context of SQLi detection, queries must be transformed into numerical representations understandable by algorithms. Preprocessing includes steps such as text cleaning, removal of irrelevant words, and tokenization, followed by vectorization techniques like TF-IDF. Additionally, since attack occurrences are often less frequent than legitimate queries, balancing methods are essential to prevent classification bias. One of the most widely used is SMOTE, which increases the minority class proportion by generating new synthetic samples based on existing ones [17]. This strategy enhances model generalization and avoids biased predictions, especially when combined with robust classification algorithms.

1) TF-IDF (Term Frequency-Inverse Document Frequency): The TF-IDF technique calculates the importance of a term within a document, considering its frequency and rarity across other texts. In SQLi detection, it highlights tokens that frequently appear in attacks but are rare in legitimate queries. This vectorization model is efficient and widely used in text classification tasks, making it a suitable choice for semantically representing SQL queries [17].

2) SMOTE (Synthetic Minority Over-sampling Technique): SMOTE generates new minority class instances by interpolating between real samples. This approach helps balance the dataset, reducing prediction bias. When applied alongside classifiers such as Random Forest or neural networks, it significantly improves model accuracy and F1-score [13], [14].

D. Related Work

The study by [20], entitled “A Novel SQL Injection Detection Using Bi-LSTM and TF-IDF”, proposes a SQL Injection detection model based on a combination of TF-IDF and Bi-LSTM neural networks, addressing the limitations of traditional blacklist-based methods. The study assumes that malicious SQL queries can be distinguished by specific linguistic patterns, even when attempts are made at syntactic obfuscation. Through textual preprocessing, TF-IDF vectorization, and sequential modeling with Bi-LSTM, the model achieved strong performance metrics, including 99% accuracy and F1-score, according to the authors. The research emphasizes the importance of semantic vector representation combined with deep models that exploit token sequences. Additionally, it highlights that bidirectional networks can capture contextual relationships that traditional models, such as Naive Bayes or SVM, cannot. The similarity with the present work lies in the choice of the Bi-LSTM model and the use of TF-IDF as a vectorization technique, demonstrating methodological alignment and reinforcing the benefits of this hybrid approach. However, the present analysis advances by also incorporating data balancing techniques using SMOTE and performing a comparison with classical ML models, providing a broader view of the relative performance across approaches.

The study by [25], entitled “Detecting SQL Injection Web Attacks Using Ensemble Learners and Data Sampling”, investigates SQL Injection attack detection in highly imbalanced scenarios using ensemble classifiers such as Random Forest, XGBoost, and CatBoost. The novelty of the research lies in handling highly imbalanced datasets, including attack occurrence rates below 1:100,000, simulating real network traffic conditions. To address this challenge, the authors employed random undersampling and evaluated model performance using the AUC metric. The study shows that ensemble-based classifiers consistently outperform traditional methods such as Naive Bayes or Logistic Regression in highly sparse attack scenarios. The relevance of this work lies in its practical and realistic approach, prioritizing model robustness in production environments. In the present article, this perspective is complemented by the use of SMOTE for oversampling and a comparative analysis of different ML and DL models, including Bi-LSTM. The combination of these studies underscores the importance of considering class imbalance as a critical dimension in modern intrusion detection systems.

In the article "Classification of SQL Injection Attacks using Ensemble Learning SVM and Naïve Bayes", [17] propose a SQL Injection detection model based on the combination of SVM and Naive Bayes classifiers through ensemble learning. Using a dataset composed of five attack types (error-based, union-based, boolean-based, time-based, and benign), the authors performed a multi-level classification approach to differentiate attack types. The model combination achieved superior accuracy (92.9%) compared to individual application of SVM (93.98%) and Naive Bayes (73.50%). This result highlights the potential of ensemble learning in improving model robustness and generalization in environments with high attack variability. The present study also employs models such as SVM and Naive Bayes, in addition to Random Forest, and conducts a structured performance comparison. However, it differentiates itself by integrating a Deep Learning approach based on Bi-LSTM, as well as applying modern balancing and textual vectorization techniques, expanding analytical scope and applicability in real-world, imbalanced datasets.

The study by [26], "An Investigation of Machine Learning Algorithms for High-bandwidth SQL Injection Detection Utilizing BlueField-3 DPU Technology", analyzes the feasibility of executing SQLi detection models on DPUs (Data Processing Units), targeting high-traffic datacenter environments. The novelty lies in evaluating 20 machine learning algorithms with a focus on inference time and accuracy. The Passive Aggressive classifier achieved notable performance with 99.78% accuracy and 0.3μs latency per sample, demonstrating suitability for embedded real-time inspection systems. Although the study does not directly address DL models or textual vectorization techniques, it stands out for emphasizing computational efficiency and feasibility of deployment on specialized hardware. The present article complements this perspective by focusing on accuracy and robustness in imbalanced environments with large text volumes, exploring techniques such as SMOTE, TF-IDF, and Bi-LSTM networks. Thus, while [26] contributes to discussions on deployment and infrastructure performance, the present study deepens the analysis of algorithm performance relative to linguistic characteristics of the data, forming a complementary set of approaches.

The article by [19], "Random Decision Forest Approach for Mitigating SQL Injection Attacks", proposes a Random Forest-based approach to identify malicious SQL queries through structural analysis of sentences. The study defines a scoring system for query components (keywords, special characters, and logical patterns), assigning weights according to associated risk. This heuristic modeling is subsequently used as input to the Random Forest algorithm. The model achieved 95% accuracy and 97% precision, reinforcing the effectiveness of decision trees in binary detection problems. The similarity with the present work lies in the adoption of Random Forest as one of the evaluated techniques; however, here this technique is applied with TF-IDF vectorization and compared with other classical and neural models. The main limitation of [19] is the use of manually defined features, which may restrict model generalization. The present study advances by adopting NLP techniques for automatic feature extraction, promoting greater adaptability and scalability.

The study by [22], "Semantic Query-Featured Ensemble Learning Model for SQL-Injection Attack Detection in IoT-Ecosystems", proposes a machine learning solution with semantic feature extraction from SQL queries using Word2Vec (CBOW and Skip-Gram). After semantic vector extraction, the study applies attribute selection and normalization to reduce redundancy and improve accuracy. The final classifier is a heterogeneous ensemble with nine algorithms, including Random Forest, Naive Bayes, and SVM, resulting in 98% accuracy and an AUC of 0.999. The study highlights the application of semantic vectors in IoT environments, considering computational constraints and data imbalance. Conceptually, the approach aligns with the present study, which also applies balancing methods and supervised learning to textual SQL datasets. However, the present work differentiates itself by using TF-IDF with n-grams instead of embeddings and evaluating Bi-LSTM performance to capture contextual dependencies. Both studies demonstrate complementary advances, with the present article more focused on model comparison and textual preprocessing strategies.

III. Methodology

This section details the methodology employed for SQL Injection attack detection, including the dataset description, data preprocessing steps, implementation of Machine Learning and Deep Learning models, and evaluation metrics. This study was conducted entirely with free software: all experiments were run on a Debian 12 Linux server using the scikit-learn library in Python. The objective is to provide a clear and replicable experimental process.

A. Dataset

For this study, the "Biggest SQL Injection Dataset" available on Kaggle [27] was used. This dataset consists of SQL queries categorized as benign or malicious (containing SQL Injection). The dataset was chosen for its breadth and representativeness of real SQLi attack scenarios and is widely used in research. The original dataset has a considerable volume of data, which is crucial for robust training of Machine Learning and Deep Learning models.

B. Data Preprocessing

Data preprocessing is crucial to ensure SQL queries are suitable for model training. The following steps were performed:

1) Cleaning and Normalization: Initially, SQL queries underwent basic cleaning, including removal of irrelevant characters and whitespace normalization. Although the dataset was relatively clean, this step ensures input consistency.

2) Text Vectorization (TF-IDF with N-grams): To convert textual SQL queries into a numeric format understandable by ML and DL algorithms, TF-IDF (Term Frequency-Inverse Document Frequency) was applied. The TfidfVectorizer from the scikit-learn library was configured to consider unigrams and bigrams (`ngram_range=(1,2)`). Including bigrams allows capturing sequential word patterns frequently found in SQLi attacks (e.g., "OR 1=1", "UNION SELECT"), which is more effective than analyzing isolated words. `max_features` was set to $10,000$ to limit the vocabulary to the 10,000 most relevant terms, reducing dimensionality and noise.

3) Handling Class Imbalance: The SQL Injection dataset is inherently imbalanced, with significantly more benign queries than malicious ones. To mitigate bias towards the majority class and ensure effective attack detection (minority class), SMOTE was applied. SMOTE was used on the training set $(X_{\text{train}}, y_{\text{train}})$ to generate synthetic minority class examples, balancing class distribution before model training. This step was essential for improving recall and precision in attack detection.

C. Machine Learning and Deep Learning Models

For comparative analysis, four distinct models were implemented and evaluated:

1) Naive Bayes: A simple yet effective probabilistic classifier commonly used in text classification tasks. MultinomialNB from scikit-learn was employed, suitable for term-count data (e.g., TF-IDF vectors).

2) Random Forest: An ensemble learning algorithm based on decision trees. RandomForestClassifier from scikit-learn was used, known for robustness and capability to handle high dimensionality and complex feature interactions.

3) Support Vector Machine: A powerful classification algorithm seeking the optimal hyperplane to separate classes. SVC from scikit-learn was configured with a linear kernel, suitable for high-dimensional data like TF-IDF vectors.

4) Bidirectional Long Short-Term Memory: An advanced RNN architecture implemented using TensorFlow/Keras. The Bi-LSTM model captures sequential and contextual dependencies in SQL queries. The architecture consisted of:

- **Embedding Layer:** Converts input tokens into dense vectors of embedding_dim $= 128$, capturing semantic relationships among words.
- **Two Bidirectional LSTM Layers:** The first with 64 units and `return_sequences=True` to stack LSTMs, and the second with 32 units. Bidirectional layers process sequences both forwards and backwards, capturing full query context.
- **Dense Layer:** Hidden layer with 64 neurons and ReLU activation to learn high-level representations.
- **Dropout Layer:** With rate 0.5 to reduce overfitting.
- **Output Layer:** Single unit with Sigmoid activation for binary classification (0 benign, 1 malicious).

The model was compiled with the `adam` optimizer and `binary_crossentropy` loss, using `accuracy` as a metric. Training was conducted over 10 epochs, with `batch_size` 32 and `validation_split` 0.2. EarlyStopping with `patience=3` was applied to monitor `val_loss` and halt training if no improvement occurred, preventing overfitting and optimizing training time.

D. Evaluation Metrics

Model performance was assessed using the following classification metrics on the test set $(X_{\text{test}}, y_{\text{test}})$:

- **Accuracy:** Proportion of correct predictions (true positives + true negatives) out of total samples. General performance metric.
- **Precision:** Proportion of true positives among predicted positives. Indicates ability to avoid false positives.
- **Recall:** Proportion of true positives among actual positives. Indicates ability to identify all real attacks.
- **F1-Score:** Harmonic mean of precision and recall. Useful when there is imbalance or a need to balance both.
- **Classification Report:** Table summarizing precision, recall, and F1-score for each class, along with support (number of actual occurrences in the test set).

All metrics were calculated using `accuracy_score` and `classification_report` functions from scikit-learn.

Table I
CLASSIFICATION MODEL PERFORMANCE ON TEST SET

Model	Accuracy	Prec. (C0)	Recall (C0)	F1 (C0)	Prec. (C1)	Recall (C1)	F1 (C1)
Naive Bayes (NB)	0.9216	0.90	0.95	0.92	0.95	0.90	0.92
Random Forest (RF)	0.9805	0.96	1.00	0.98	1.00	0.97	0.98
Support Vector Machine	0.9739	0.95	1.00	0.97	1.00	0.95	0.97
Bi-LSTM	0.9815	0.97	1.00	0.98	1.00	0.97	0.98

IV. RESULTS

This section presents results obtained from applying various ML and DL models for SQL Injection detection. Performance metrics for each model are compared, discussing implications in light of theoretical foundations and related work, as well as observed limitations.

A. Model Performance

Models were evaluated using accuracy, precision, recall, and F1-score on the test set. Table I summarizes overall model performance, with detailed classification reports..

1) Naive Bayes: Despite its simplicity, Naive Bayes achieved 0.9216 accuracy. This result is notable, considering the model benefits from SMOTE preprocessing and TF-IDF with n-grams. Precision and recall for both classes (benign and attack) were balanced, indicating reasonable pattern distinction. However, as expected, more complex models outperformed it.

2) Random Forest: Random Forest showed significantly higher performance with 0.9805 accuracy. This corroborates literature on the effectiveness of ensemble methods for complex classification. High precision and recall for the attack class (1.00 and 0.97) indicate excellent capability to identify real attacks (low false negatives) and correctly classify attacks (low false positives). Its robustness in handling high-dimensional data and reduced susceptibility to overfitting contributed to this outcome.

3) Support Vector Machine: SVM also demonstrated robust performance with 0.9739 accuracy. Like Random Forest, SVM benefited from TF-IDF n-gram vectorization, enabling a feature space where classes are more separable. High precision and recall for both classes confirm SVM's effectiveness in SQL Injection detection, reinforcing its relevance in binary classification problems.

4) Bi-LSTM: The Bi-LSTM achieved the highest accuracy at 0.9815. This marginal advantage over Random Forest highlights the ability of bidirectional recurrent networks to capture complex contextual dependencies in text sequences. Its strong performance, with precision and recall of 0.97 and 1.00 for benign, and 1.00 and 0.97 for attack class, suggests Deep Learning excels in understanding SQL query semantics, crucial for detecting sophisticated SQLi attacks. Bi-LSTM's ability to learn feature representations directly from data without extensive manual feature engineering is a significant advantage.

V. DISCUSSION

Results clearly demonstrate the superiority of advanced models (Random Forest, SVM, Bi-LSTM) over Naive Bayes for SQL Injection detection, highlighting the importance of algorithms capable of capturing complex relationships.

Class imbalance handling with SMOTE was critical for success. Significant performance improvement after SMOTE application shows that balancing training classes is essential to prevent bias toward the majority class and failure in detecting the minority class, which is of greater interest in cybersecurity.

Including n-grams in TF-IDF also improved performance, especially for traditional ML models. Considering word sequences allows the vectorizer to capture syntactic and semantic patterns characteristic of SQLi attacks, such as UNION SELECT or OR 1=1, which would be less evident analyzing isolated words.

Although Bi-LSTM had the highest accuracy, the difference from Random Forest was minimal. For this dataset and preprocessing, ensemble learning models are competitive alternatives to Deep Learning models, especially given lower computational complexity and shorter training times. However, for more complex attacks or larger, diverse datasets, Bi-LSTM's ability to learn hierarchical, long-term representations may provide a pronounced advantage.

A. Limitations

- **Dataset Dependence:** Model performance is intrinsically tied to dataset quality and diversity. Although the Biggest SQL Injection Dataset is comprehensive, generalization to completely new attacks or highly obfuscated techniques may be limited.
- **Bi-LSTM Computational Cost:** Despite EarlyStopping optimization, Bi-LSTM training required significantly more processing time compared to traditional ML models, which may be important in resource-limited environments.
- **Interpretability:** Deep Learning models like Bi-LSTM are often considered "black boxes", complicating understanding of decision-making. In security contexts where

explainability is desired for forensic analysis or threat understanding, this is a limitation.

- **SMOTE on Text Data:** While effective, SMOTE applied directly to vectorized (TF-IDF) or sequence data may generate semantically invalid or noisy synthetic examples. More sophisticated NLP balancing approaches, such as class weighting or text generation techniques, could be explored in future work.

Overall, results confirm the effectiveness of ML and DL approaches for SQL Injection detection, particularly Random Forest and Bi-LSTM. Data preprocessing, especially class balancing and n-gram vectorization, was important for optimizing model performance. Observed limitations highlight avenues for future research and improvement.

VI. Conclusion

This article investigated and compared the performance of different ML (Naive Bayes, Random Forest, SVM) and DL (Bi-LSTM) models in detecting SQL Injection attacks in SQL queries. The study demonstrated the efficacy of these approaches, particularly when combined with preprocessing techniques such as SMOTE for class balancing and TF-IDF with n-gram vectorization.

The proposed objectives were achieved. Experimental analysis showed class imbalance handling is crucial for model robustness, significantly improving performance metrics, particularly detection of the minority class (SQLi attacks). Including n-grams in TF-IDF vectorization also enhanced model capability to capture complex contextual patterns in queries.

In terms of performance, ensemble learning (Random Forest) and Deep Learning (Bi-LSTM) consistently outperformed Naive Bayes. Random Forest and Bi-LSTM achieved 0.9805 and 0.9815 accuracy, respectively, demonstrating exceptional ability to identify SQL Injection attacks with high precision and recall. Although Bi-LSTM had a slight accuracy advantage, Random Forest proved a highly competitive alternative with lower computational complexity.

Key contributions include empirical validation of ML and DL approaches for SQLi detection on a comprehensive dataset, demonstration of critical importance of class

Acknowledgments

The authors are grateful to State Center for Technological Education “Paula Souza” (CEETEPS) for research conditions at the Defensive Cybersecurity and Artificial Intelligence Laboratory (Detect.AI), and to Research Support Foundation of São Paulo State (FAPESP), Brazil grants #2023/12830-0.

References

[1] OWASP Foundation, “OWASP Top Ten: The Ten Most Critical Web Application Security Risks,” https://owasp.org/www-project-top-ten/, 2024, accessed: 2025-05-30.

[2] S. Samtani, H. Chen, M. Kantarcioglu, and B. Thuraisingham, “Explainable artificial intelligence for cyber threat intelligence (xai-cti),” *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 4, pp. 2149–2150, 2022.

[3] “Edgescan vulnerability statistics report 2025,” Edgescan, Tech. Rep., 2025, 10th Edition. [Online]. Available: https://www.edgescan.com/stats-report

[4] T. Scholte, D. Balzarotti, and E. Kirda, “Have things changed now? an empirical study on input validation vulnerabilities in web applications,” *Comput. Secur.*, vol. 31, no. 3, p. 344–356, May 2012. [Online]. Available: https://doi.org/10.1016/j.cose.2011.12.013

[5] N. Kaur and P. Kaur, “Input validation vulnerabilities in web applications,” *Journal of Software Engineering*, vol. 8, pp. 116–126, 2014. [Online]. Available: https://scialert.net/abstract/?doi=jse.2014.116.126

[6] N. Mohamed, “Artificial intelligence and machine learning in cybersecurity: a deep dive into state-of-the-art techniques and future paradigms,” *Knowledge and Information Systems*, 2025. [Online]. Available: https://doi.org/10.1007/s10115-025-02429-y

[7] A. Paul, V. Sharma, and O. Olukoya, “Sql injection attack: Detection, prioritization prevention,” *Journal of Information Security and Applications*, vol. 85, p. 103871, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S221421262400173X

[8] N. D. Bobade, V. A. Sinha, and S. S. Sherekar, “A diligent survey of sql injection attacks, detection and evaluation of mitigation techniques,” in *2024 IEEE International Students’ Conference on Electrical, Electronics and Computer Science (SCEECS)*, 2024, pp. 1–5.

[9] V. Felmetsger, L. Cavedon, C. Kruegel, and G. Vigna, “Toward automated detection of logic vulnerabilities in web applications,” in *Proceedings of the 19th USENIX Security Symposium*. Washington, DC, USA: USENIX Association, Aug. 2010, pp. 10–10. [Online]. Available: https://www.usenix.org/legacy/event/sec10/tech/full_papers/Felmetsger.pdf

[10] T. Y. Khashirova, I. I. Mamuchiev, M. I. Mamuchieva, M. I. Ozhiganova, A. D. Kostyukov, and I. Shumeiko, “Assessment of information security in integrated systems,” in *2021 International Conference on Quality Management, Transport and Information Security, Information Technologies (ITQMIS)*, 2021, pp. 201–205.

[11] I. I. Mamuchiev and L. A. Moskalenko, “The use of data labels for access control by means of ols,” in *Perspective – 2018: Proceedings of the International Scientific Conference of Students, Postgraduates, and Young Scientists*, A. M. Kumykov *et al.*, Eds. Nalchik, Russia: Kabardino-Balkarian State University, 2018, Conference Proceedings, p. 50, 100 copies.

[12] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM Comput. Surv.*, vol. 41, no. 3, Jul. 2009. [Online]. Available: https://doi.org/10.1145/1541880.1541882

[13] R. Hallman, J. Bryan, G. Palavicini Jr, J. Divita, and J. Romero-Mariona, “Ioddos — the internet of distributed denial of service attacks: A case study of the mirai malware and iot-based botnets,” 04 2017.

[14] E. Bertino and N. Islam, “Botnets and internet of things security,” *Computer*, vol. 50, no. 2, pp. 76–79, 2017.

[15] W. Wu, H. Fouzi, B. Benamar, S. Sidi-Mohammed, and S. Ying, “Deep learning-based stacked models for cyber-attack detection in industrial internet of things,” *Neural Computing and Applications*, 2025. [Online]. Available: https://doi.org/10.1007/s00521-025-11418-9

[16] J. P. Bharadiya, “Machine learning in cybersecurity: Techniques and challenges,” *European Journal of Technology*, vol. 7, no. 2, pp. 1–14, 2023, doctor of Philosophy Information Technology, University of the Cumberlands, USA.

[17] M. M. Ibrohim and V. Suryani, “Classification of sql injection attacks using ensemble learning svm and naïve bayes,” in *2023 International*

Conference on Data Science and Its Applications (ICoDSA), 2023, pp. 230–236.

[18] V. K. Chauhan, A. Kumar, S. Kumar, T. Singh, and R. K. Dwivedi, “Utilizing an ensemble classification method to assess the severity of sql injection attacks and xss,” in *2023 12th International Conference on System Modeling Advancement in Research Trends (SMART)*, 2023, pp. 133–139.

[19] P. Aggarwal, A. Kumar, K. Michael, J. Nemade, S. Sharma, and P. K. C, “Random decision forest approach for mitigating sql injection attacks,” in *2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, 2021, pp. 1–5.

[20] I. Ghozali, M. F. Asy’ari, S. Triarjo, H. M. Ramadhani, H. Studiawan, and A. M. Shiddiqi, “A novel sql injection detection using bi-lstm and tf-idf,” in *2022 7th International Conference on Information and Network Technologies (ICINT)*, 2022, pp. 16–22.

[21] M. Ozkan-Okay, E. Akin, Aslan, S. Kosunalp, T. Iliev, I. Stoyanov, and I. Beloev, “A comprehensive survey: Evaluating the efficiency of artificial intelligence and machine learning techniques on cyber security solutions,” *IEEE Access*, vol. 12, pp. 12 229–12 256, 2024.

[22] G. M and P. H B, “Semantic query-featured ensemble learning model for sql-injection attack detection in iot-ecosystems,” *IEEE Transactions on Reliability*, vol. 71, no. 2, pp. 1057–1074, 2022.

[23] D. M. Abdullah and A. M. Abdulazeez, “Machine learning applications based on SVM classification: A review,” *Qubahan Academic Journal*, vol. 1, no. 2, pp. 81–90, apr 2021. [Online]. Available: https://journal.qubahan.com/index.php/qaj/article/view/50

[24] N. Mohamed, A. Oubelaid, and S. Almazrouei, “Staying ahead of threats: A review of ai and cyber security in power generation and distribution,” *International Journal of Electrical and Electronics Research*, vol. 11, 03 2023.

[25] R. Zuech, J. Hancock, and T. M. Khoshgoftaar, “Detecting sql injection web attacks using ensemble learners and data sampling,” in *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2021, pp. 27–34.

[26] K. Tasdemir, R. Khan, F. Siddiqui, S. Sezer, F. Kurugollu, and A. Bolat, “An investigation of machine learning algorithms for high-bandwidth sql injection detection utilizing bluefield-3 dpu technology,” in *Proceedings of the 36th IEEE International Conference on System-on-Chip (SOCC 2023)*. Santa Clara, CA, USA: IEEE, 2023, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/10256777

[27] GambleRyu, “Biggest SQL Injection Dataset,” https://www.kaggle.com/datasets/gambleryu/biggest-sql-injection-dataset, 2025, kaggle dataset.