Machine Learning in SQL Injection Prevention and Detection

Michelle Nguyen, Nhu Pham, Phat Tran

December 8, 2023

Abstract

In this survey, we will conduct research on how machine learning has been employed to detect data associated with SQL injection and develop vulnerability predictors. As part of our survey, we also conducted a small-scale experiment where we trained simple machine learning models using the supervised learning method and its algorithms. The goal of this experiment is to train models to distinguish between a harmless SQL query and an actual malicious injection. Through this, we hope to gain a better understanding of how machine learning can be used to add an extra layer of security to computer systems against SQL injection.

1 Introduction

Because databases are the main storage of potential confidential information online, unauthorized access can enable malicious activities such as retrieving information data, deleting tables, or other harmful actions by attackers. Furthermore, this can lead to potential damages that can cost companies millions of dollars such as preventing key programs from executing, causing downtimes. [2] SQL injections and cross-site scripting (XSS) account for 27% of all attacks on well-known web applications. [3]

The purpose of SQL Injection is to gain unauthorized access to a database by injecting code and probing the SQL query. Attackers employ SQL injections to explore databases linked to web applications or websites. It is crucial to safeguard the data within these databases from potential SQL injection threats. In the case of website fields lacking appropriate input validation, an attacker could potentially gain direct entry to the database of the associated application [4].

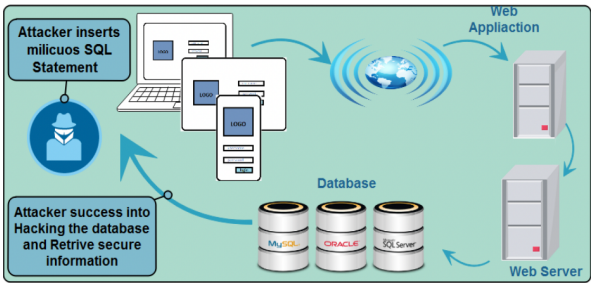


Figure 1: *Process of a SQL Injection Attack*

2 Background

2.0.1 SQL Injection Attack Types

There are numerous methods of SQL Injection attacks which can be classified into 3 different categories as Union based SQLi, error based SQLi or blind SQLi. [5]

Union based SQL injection: In SQL, the UNION operator merges results of multiple SELECT statements from two or more separate queries into a unique result set, all encapsulated in a single row. In a vulnerability application, the attackers can manipulate queries to extract database records from various tables by leveraging the UNION operator.

Error based SQL injection: In Error-based SQL injection scenarios, the operator inserts an invalid input value into the query, provoking errors within the database. When a query is detected inappropriate, an error message is returned from the database [6]. The operator then identifies queries and exerting control over the entire database

Blind SQL injection: Blind SQL injection occurs where the attacker exploits vulnerabilities in a web application's database indirectly by studying how the server reacts to injected SQL queries [7]. Even though they take more time than the

usual ways of doing SQL injection, blind SQL injection attacks can be automated to carefully map out the structure of the database and extract important data from the server. There are two ways to do blind SQL injection: one involves a series of yes/no tests (Boolean-based), and the other involves checking for differences in response times (Time-based) [8].

2.0.2 Researched related to Machine Learning in Detecting SQL Injection

Various existing research studies have been examined and included to highlight gaps in the research within this area. There is countless research that has been done using machine learning algorithms to detect the SQL injection attack.

In the research of Alghazzawi, they performed a comprehensive literature review involving 36 articles centered on the investigation of SQL injection attacks and machine learning techniques. To categorize various types of SQLI attacks, they pinpointed techniques most commonly employed. Their discoveries indicated that a limited number of studies created new datasets for SQLI attacks utilizing ML tools and methodologies. Similarly, the outcomes revealed that only a handful of studies exclusively concentrated on the utilization of mutation operators for generating adversarial SQLI attack queries [9].

In 2021, Gandhi, Patel, Sisodiya and Mishra introduced a hybrid model based on CNN-BiLSTM for the detection of SQL injection (SQLI) attacks. They conducted a comprehensive comparative analysis of various machine learning (ML) techniques employed in SQLI attack detection. The CNN-BiLSTM approach exhibited an accuracy of approximately 0.98, surpassing other ML techniques discussed in the study [10].

Similarly, in 2019, Zhang K. proposed an ML classifier for identifying SQLI vulnerabilities in PHP code [11]. They trained and evaluated multiple ML techniques, including random forest, logistic regression, support vector machine (SVM), multilayer perceptron (MLP), long short-term memory (LSTM), and CNN. The findings indicated that CNN achieved the highest precision at 0.9454 while an MLP-based model attained the top recall at 0.637 and the highest f-measure at 0.746.

In the research conducted by Kranthikumar et al in 2020, the authors compared machine learning classifications, including SVM, Gradient Boosting Algorithm, and Naive Bayes classifier, with a pattern-based classification called REGEX to detect SQL injection threats. Using a dataset of 20,474 queries, the REGEX classifier was found with better performance, achieving accuracy score of 0.97 with a computation time of 3.98 seconds. The result outperforms other machine learning techniques in the study. [12]

3 Experiments and Results

Supervised learning is a type of machine learning method that uses algorithms to train new models with given labeled data. The goal of supervised learning algorithms is to predict new, unseen data based on patterns learned during the training phase from the provided dataset. This characteristic makes it particularly well-suited for tasks where the goal is to classify or predict specific outcomes, such as distinguishing between harmless and malicious SQL queries. [13]

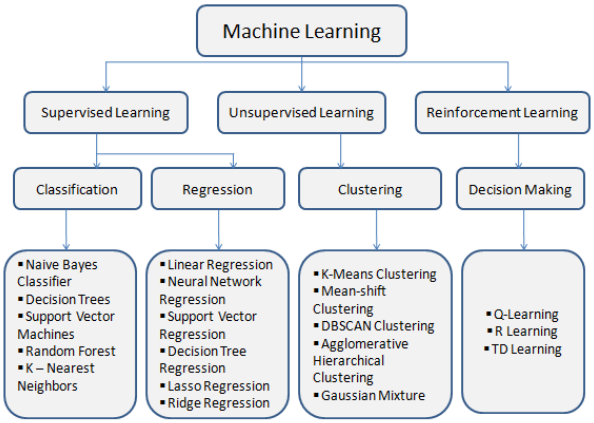


Figure 2: *Main types of machine learning model*

In this scenario, where the goal is to train a model to distinguish between harmless and malicious SQL queries, supervised learning offers distinct advantages over unsupervised and reinforced learning, mostly due to its reliable accuracy and beginner-friendly approach. On the other hand, unsupervised learning does not require known labels, providing less accuracy for prediction in this case and

poses challenges in generalizing conclusions about the models' performance. Additionally, while reinforcement learning is indeed a powerful combination of supervised and unsupervised learning, it demands a more proficient understanding and greater effort in running trial and error to draw meaningful conclusions for any underlying patterns within the dataset. [14]

Therefore, we ultimately decided to choose supervised learning as our training method for its ability to accurately generalize patterns from known samples, rendering it highly reliable for predicting outcomes in the specified dataset used for model training.

3.0.1 Standard Machine Learning Cycle [15]

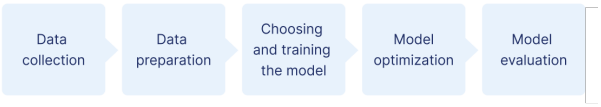


Figure 3: *Machine learning process flow*

1. **Data Collection:**
   - Gather a comprehensive dataset that includes labeled examples of input features and corresponding target outputs.
2. **Data Preparation:**
   - Process the dataset by handling missing values, outliers, and irrelevant information.
   - Split the dataset into 2 separate subsets to use for training and testing for model's performance evaluation.
   - Normalize or standardize the data points to prevent undesirable learning behavior such as overfitting or underfitting.
3. **Model Selection and Training:**
   - Choose an appropriate supervised learning algorithm based on the nature of the classification model.
   - Compare the strengths and weaknesses of each different algorithm.
   - Feed training dataset into the chosen algorithm to allow the model to learn the underlying patterns and relationships between input features and target outputs.
4. **Model Evaluation:**
   - Evaluate using metrics such as accuracy, precision, recall, and F1 score to quantify the model's effectiveness.
   - Evaluate what further steps could be taken to enhance the performance of each model

3.0.2 Data Selection

For the first step of our experiment, Data Selection, we retrieved our dataset from Kaggle, an online community under Google that aims to create a collaborative environment for data exchange between data scientists and machine learning enthusiasts to access, share and explore datasets for various projects and competitions. [15]

```
# Returns a random selection elements (25)
df.sample(25, random_state = 42)
```

	Query	Label
2111	1' where 5801 = 5801 or row (1045,7562)...	1
29588	SELECT * FROM scene WHERE apple BETWEEN '1996-...	0
28951	INSERT INTO am (ball, especially, fellow) ...	0
24360	SELECT AVG (instrument) FROM construction ...	0
12290	65	0
25935	SELECT AVG (blew) FROM stage SELECT SUM (...	0
4813	-7142')) union all select 5083,5083,...	1
27325	SELECT * FROM consonant WHERE grandfather BETW...	0
15334	idt	0
18123	7656	0
18063	francisc	0
17329	eisenberg@muchochinoenchina.mx	0
27272	SELECT * FROM quiet WHERE arm = 'detail' AND ...	0
11243	1%') and 7533 = 7533 and ('%' = '	1
21051	SELECT * FROM thought FETCH FIRST 50 PERCENT R...	0
11524	enlutada	0
26231	SELECT COUNT (rope) FROM total	0
2764	1') where 7646 = 7646 and 3580 = (...	1
16681	cr48e24n	0

Figure 4: Sample data points in dataset

Our chosen dataset comprises 30,000 entry points, each composed of two value fields: "Query", which contains the raw query string and "Label" which contains binary values (0 and 1). For the"Label" column, 0 represents harmless SQL

strings and 1 represents actual malicious SQL injection.

Due to the dataset structure and composition, we conclude that our machine learning model is a Binary Classification model. The "Label" column includes 2 different categories, which aligns with the characteristic output of a binary classification model , where it is trained to classify instances into one of two distinct categories.[18]

3.0.3 Data Processing

Data processing is an essential step in machine learning, where its objective is to improve the model's accuracy by removing any unexpected patterns, ensuring a more consistent and effective learning process. This approach is referred to as Exploratory Data Analysis (EDA for short), and includes investigating and processing tasks such as:

1. **Inspecting Data:** Examining the dataset to study its overall structure, generalizing any patterns and detecting any outliers, or anomalies that might impact the subsequent analysis.

2. **Summary Statistics:** Providing a concise overview of key quantitative measures within the dataset, such as mean, median, standard deviation, and quartiles. However, due to the nature of our model being Binary Classification (only two categories: 0 and 1), we will not inspect these statistics.

3. **Handling Missing Values:** Identifying and appropriately handling anomalies, outlier and missing data, ultimately to ensure the robustness of the dataset and prevent potential biases in subsequent model training.

4. **Visualizing Data:** In addition to Exploratory Data Analysis, data scientists may also employ Data Visualizations in their experiment. Data Visualization is a crucial aspect in this step in the sense that it can help explore, analyze and understand the structure of the dataset. They provide a quick overview of the distribution, patterns, and relationships within the dataset through a wide variety of approaches, such as histogram, scatter plot, bar plot, line chart, word cloud, etc.

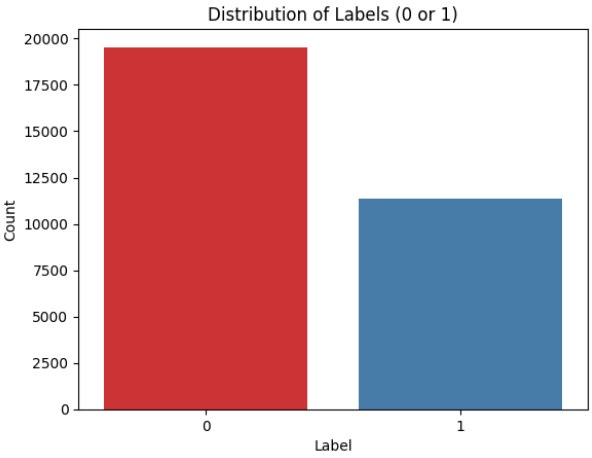


Figure 5: *Distribution of harmless SQL query and SQL injection*

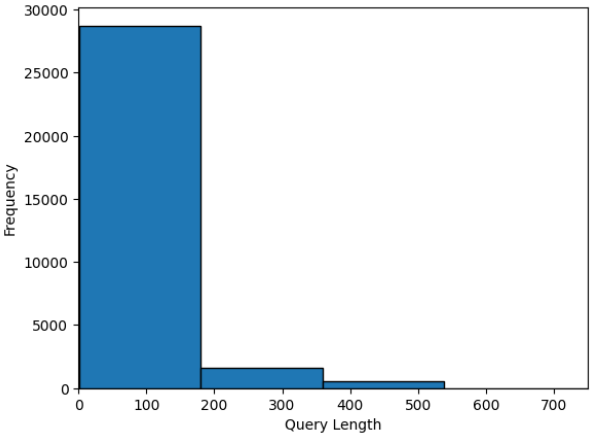


Figure 6: *Distribution in query string's length*

Through appropriate data processing and handling, we can establish a consistent yet varied dataset for the learning environment, preventing against potential risks such as overfitting, where the model becomes excessively tailored to the training data, or underfitting, where it struggles to perform well on new, unseen data due to poor generalization. Ultimately, this would mitigate inaccurate variance and improve the overall robustness and performance of machine learning models. [21]

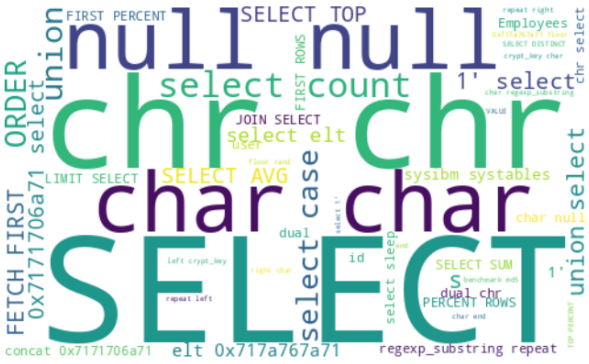


Figure 7: *Word cloud of most frequent words in harmless SQL query*

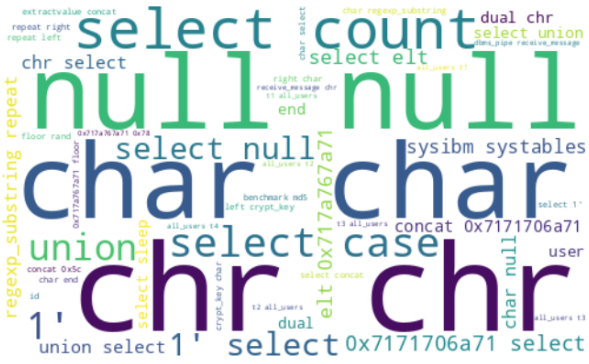


Figure 8: *Word cloud of most frequent words in SQL injection*

3.0.4 Data Splitting

Now that the dataset has undergone processing and sanitization, we proceed to the subsequent phase, Data Splitting, where the dataset is divided into two subsets: one for training and one for testing.

The training set serves as the data portion used to fit the model employing different algorithms. The testing set is the remaining portion which is used to assess the models' performance, ranking each based on accuracy and other essential metrics.

Typically, the dataset is divided in a ratio of 70:30 or 80:20, with the training set occupying the larger portion. This ensures the model is sufficiently trained to generalize well beyond the training set when confronted with new, unseen data in the testing set. [22]

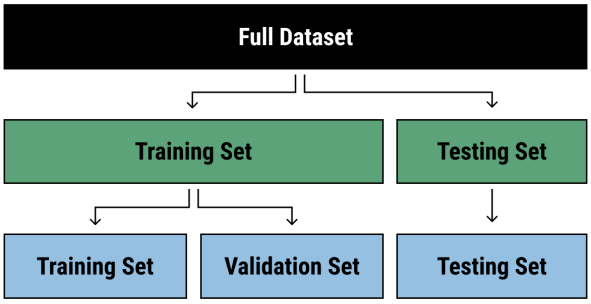


Figure 9: *2-way and 3-way splitting for dataset*

```
[ ]  # Split the data into training and testing sets (25% for testing)
     X_train, X_test, y_train, y_test
         = train_test_split(X, y, train_size = 0.75, test_size = 0.25)
```

```
[ ]  # Print dimension of the training and testing sets to console
     print(X_train.shape)    #(23189, 6594)
     print(y_train.shape)    #(23189,)
     print(X_test.shape)     #(7730, 6594)
     print(y_test.shape)     #(7730, 6594)

     (23189, 6594)
     (23189,)
     (7730, 6594)
     (7730,)
```

Figure 10: *Split dataset with sklearn.model*

Additionally, splitting the dataset into representative subsets allows us to form comprehensive insights of the model's performance, identifying potential challenges in biased behaviors. This approach helps prevent factors contributing to the model's poor performance, allowing for fine-tuning of hyperparameters and adjustments if necessary to enhance the model's confidence in real-world scenarios. [22]

3.0.5 Model Training

For our SQL injection detection models, several supervised learning algorithms might prove more effective in detecting patterns and relationships for Binary Classification. Commonly used algorithms for this type of classification model include [23][24][25]:

- Naive Bayes
- Logistic Regression
- Decision Trees
- K-Nearest Neighbors

- Support Vector Machines
- Neural Networks

These algorithms excel in uncovering patterns and relationships within training data, enabling the model to accurately classify SQL queries as either indicative or not indicative of SQL injection. Additionally, to ensure a broader coverage, we also employ additional advanced algorithms such as:

- XGBoosst
- LightBGM,

which we hope to be able to capture any underlying complex patterns in the training dataset.

```
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred_logreg = logreg.predict(X_test)

print(f"Accuracy of LogReg: {accuracy_score(y_pred_logreg, y_test)}")
print(f"F1 Score of LogReg: {f1_score(y_pred_logreg, y_test)}")
print(f"Recall of LogReg: {recall_score(y_pred_logreg, y_test)}")
print(f"Precision Score of LogReg: {precision_score(y_pred_logreg, y_test)}")
```

```
Accuracy of Logistic Regression: 0.94
F1 Score of Logistic Regression: 0.91
Recall of Logistic Regression: 0.98
Precision Score of Logistic Regression: 0.85
```

Figure 11: *Sample code for model training (with Logistic Regression)*

```
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred_logreg = logreg.predict(X_test)

print(f"Accuracy of LogReg: {accuracy_score(y_pred_logreg, y_test)}")
print(f"F1 Score of LogReg: {f1_score(y_pred_logreg, y_test)}")
print(f"Recall of LogReg: {recall_score(y_pred_logreg, y_test)}")
print(f"Precision Score of LogReg: {precision_score(y_pred_logreg, y_test)}")
```

```
Accuracy of Logistic Regression: 0.94
F1 Score of Logistic Regression: 0.91
Recall of Logistic Regression: 0.98
Precision Score of Logistic Regression: 0.85
```

Figure 12: *Sample code for plotting confusion matrix (with Logistic Regression)*

Logistic Regression

Logistic Regression is a widely employed algorithm for Binary Classification problems, specifically designed to predict outcomes in scenarios where there is a linear relationship between the data and the label. Logistic regression uses its logistic function to predict binary outcomes, assigning each data point in either category based on the computed probability. When the calculated probability falls over a certain threshold, typically 0.5, the data instance is classified into one category; otherwise, another category if it falls below the specified threshold.

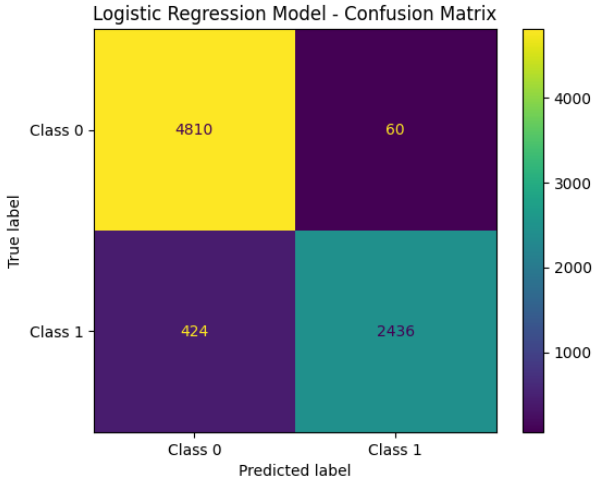


Figure 13: *Confusion Matrix for Logistic Regression*

Accuracy	Precision	Recall	F1
0.94	0.91	0.98	0.85

Table 1: *Metrics of Logistic Regression*

Naive Bayes

Naive Bayes is a generative learning algorithm that operates under the assumption that the features of the dataset are conditionally dependent regardless of the given label. Essentially, it does not determine which features are most important to differentiate between labels, unlike discriminative learning algorithms. Despite its simplicity and "naivety" in assumption, Naive Bayes classifier demonstrates remarkably high efficiency, particularly in handling classification tasks in real-time scenarios or resource-constrained environments.

Accuracy	Precision	Recall	F1
0.82	0.99	0.68	0.80

Table 2: *Metrics of Naive Bayes*

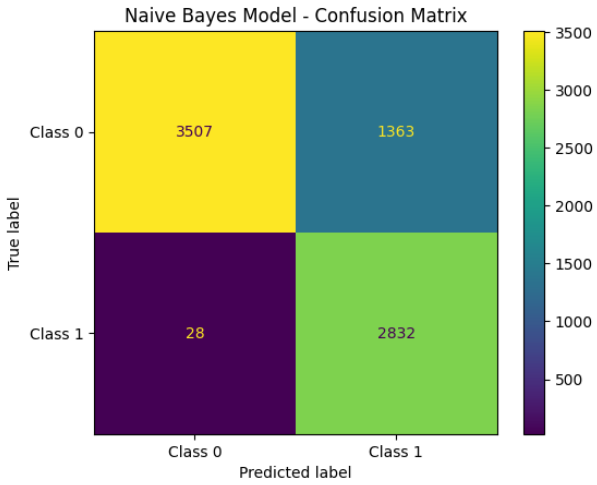


Figure 14: *Confusion Matrix of Naive Bayes*

K-Nearest Neighbor (KNN)

K-Nearest Neighbor (or KNN for short) is also another uncomplicated yet intuitive algorithm that categorizes data points and determines its label based on the local patterns of other similar data points, or nearest neighbors. The k- variable determines the number of nearest neighbors that should be taken into account when evaluating each data point. While offering simplicity and efficiency, K-Nearest Neighbor can also be rather computationally expensive as it requires memory to store all the training data. [26]

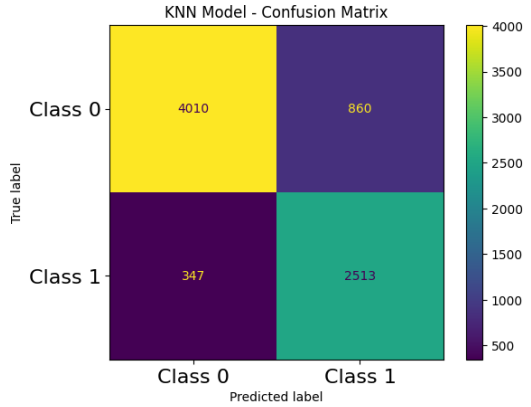


Figure 15: *Confusion Matrix of K-Nearest Neighbor*

Accuracy	Precision	Recall	F1
0.84	0.81	0.75	0.88

Table 3: *Metrics of K-Nearest Neighbor*

Decision Tree

Decision tree is a versatile algorithm capable of capturing complex relationships and discerning non-linear patterns. However, decision trees are often prone to overfitting, especially when the tree overgrows and becomes extremely tailored to the training dataset. To address this challenge when using decision trees, it is advisable to explore ensemble methods such as Random Forests, which is essentially a collection of decision trees.

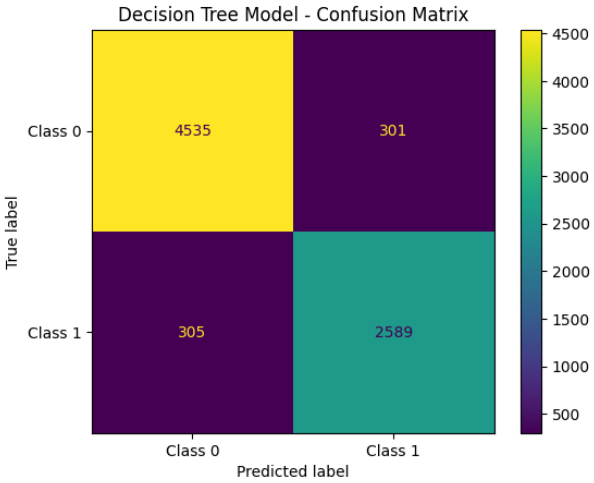


Figure 16: *Confusion Matrix of Decision Tree*

Accuracy	Precision	Recall	F1
0.92	0.90	0.88	0.90

Table 4: *Metrics of Decision Tree*

Random Forest

Random Forest, or Random Decision Forest, is a popular ensemble method with robust perfomance used for classification, regression and various other machine learning tasks. It achieves better diversity by training multiple decision trees on different subsets within the training dataset, then determines the final prediction by aggregating the results from these trees using a voting mechanism.

This approach allows Random Forest to enhance the model's generalization capabilities and mitigates the risk of overfitting that individual trees might encounter. [28] lization.

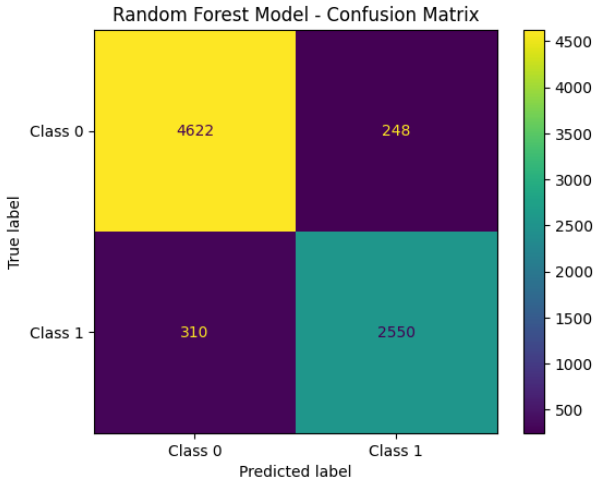


Figure 17: *Confusion Matrix of Random Forest*

Accuracy	Precision	Recall	F1
0.93	0.89	0.91	0.90

Table 5: *Metrics of Decision Tree*

Support Vector Machines (SVM)

Support Vector Machines (or SVM for short) is another powerful algorithm that specializes in scenarios with high-dimensional spaces. It is designed for both classification and regression tasks, but particularly excels in binary classification problems. This is because SVM's algorithm enables it to identify a clear margin between different labels within the training dataset, accomodating nonlinear, complex and high-dimensional relationships where other algorithms might struggle.

Neural Network

Neural Network excels at capturing both linear and nonlinear patterns, allowing it to model intricate and complex relationships within the training data. However, it can be susceptible to overfitting, particularly when dealing with limited data or complex architectures. Regularization techniques such as dropout or L2 regularization can be employed to address this biased behavior when training the model. Additionally, similar to Decision Tree and Random Forest, ensemble methods such as model averaging or stacking should also be considered in order to enhance generalization.

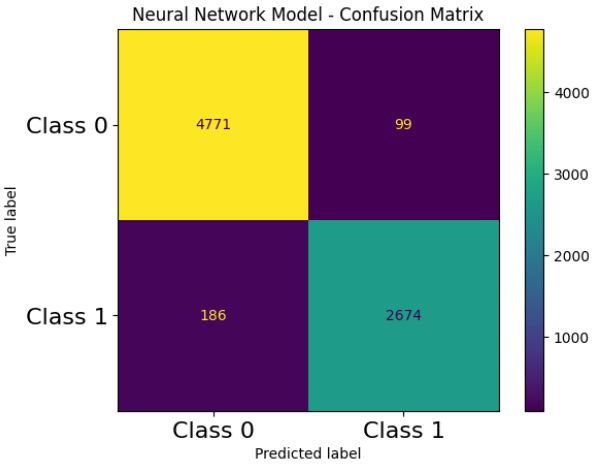


Figure 18: *Confusion Matrix of Neural Network*

Accuracy	Precision	Recall	F1
0.96	0.93	0.96	0.95

Table 6: *Metrics of Neural Network*

XGBoost

XGBoost, abbreviated from Extreme Gradient Boosting, is one of the more advanced machine learning algorithms, highly effective in constructing a robust model by amalgamating predictions from multiple weak learners. By doing so, XGBoost can eliminate the weaknesses of individual models , such as decision trees, by focusing on the correction of errors, minimizing the overall prediction error.

Accuracy	Precision	Recall	F1
0.94	0.86	0.97	0.91

Table 7: *Metrics of XGBoost*

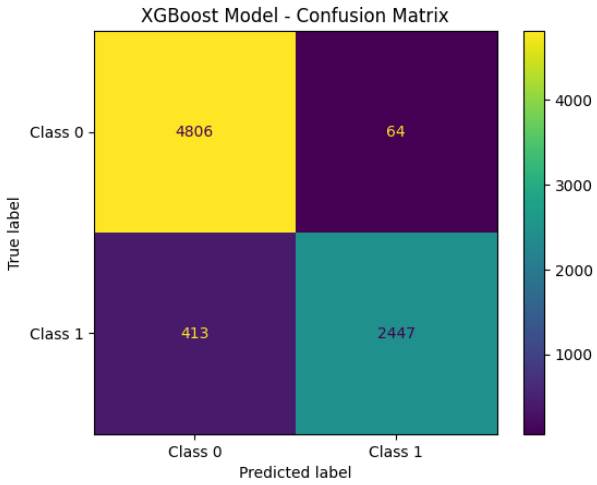


Figure 19: *Confusion Matrix of XGBoost*

LightGBM

LightGBM is another advanced gradient boosting framework, specifically designed to handle large and high-dimensional datasets. LightGBM utilizes histogram-based learning, which accelerates up the training process by grouping continuous feature values. Additionally, it also employs a leaf-wise tree growth strategy, prioritizing splits that lead to more significant reductions in prediction errors.

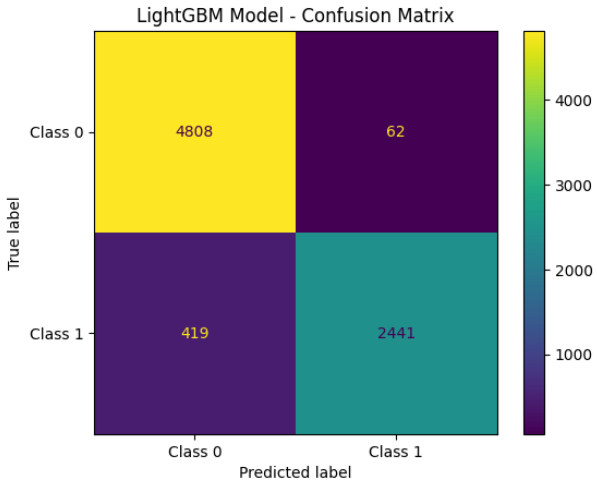


Figure 20: *Confusion Matrix of LightBGM*

Accuracy	Precision	Recall	F1
0.94	0.85	0.98	0.91

Table 8: *Metrics of LightBGM*

3.0.6 Model Evaluation

When evaluating the models' performance, we use the following quantitative assessing metrics: accuracy, precision, recall and F1 score. In general, the overall consensus is that higher scores across these metrics indicate a more effective and well-performing model.

For the following metrics, we can compute the score using the extracted data from each model's confusion matrix.[29]

- TP: True Positive
- FP: False Positive
- TN: True Negative
- FN: False Negative

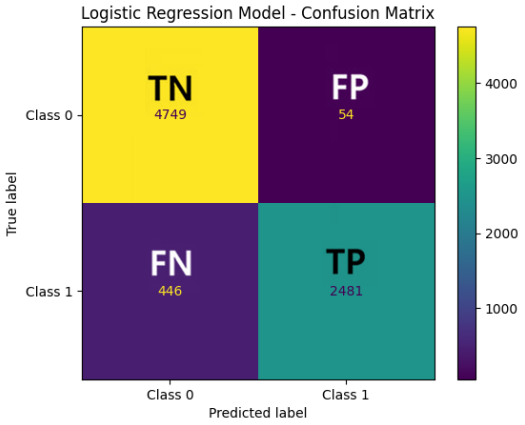


Figure 21: *Reading a Confusion Matrix*

Precision

Precision is the ratio of correctly predicted positives to the total predicted positives, and can be calculated using the following formula:

$$Precision = \frac{TP}{TP + FP}$$

A high precision score indicates that the model was able to predict more relevant instances than irrelevant ones.

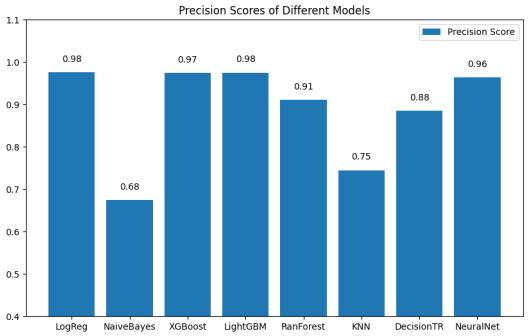


Figure 22: *Precision score across different models*

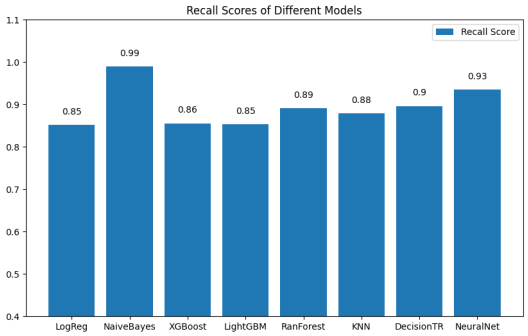


Figure 24: *Recall score across different models*

Accuracy

Accuracy is the ratio of correctly predicted instances to the total number of instances in a dataset, and can be calculated using the following formula:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

A higher accuracy score indicates better overall correctness, reflecting the model's ability to make accurate predictions across the entire dataset.

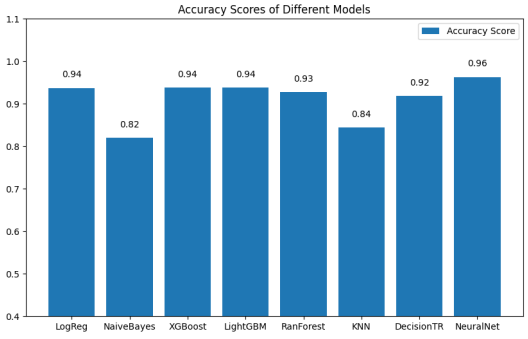


Figure 23: *Accuracy score across different models*

Recall

Recall is the ratio of true positives to the sum of true positives and false negatives, and can be calculated using the following formula:

$$Recall = \frac{TP}{TP + FN}$$

Higher recall score indicates that a model is able to correctly identify all relevant instances from the dataset more than irrelevant ones.

F1 Score

F1 score is the harmonic mean of precision and recall, and can be calculated using the following formula:

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

It is particularly useful when there is an uneven class distribution, with a higher F1 score indicating a better balance between precision and recall.

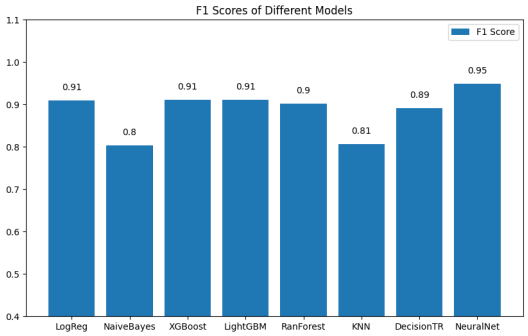


Figure 25: *F1 score across different models*

ROC Curves

ROC (Receiver Operating Characteristic) curve displays the relationship between true positive rate and false positive rate across different threshold values, providing a visual assessment of a model's performance.

The AUC (Area Under the ROC Curve) is a numerical measure of the ROC curve's performance. The higher AUC score indicates a better ability of the model to distinguish between positive and negative instances, with a perfect classifier having an AUC of 1.0.

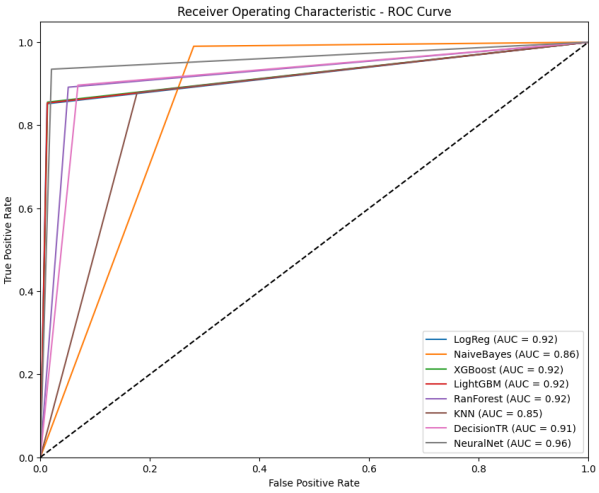


Figure 26: *ROC Curve across different models*

3.0.7 Further Improvement

To further enhance the accuracy of our models, we can explore more techniques to improve our data processing and splitting processes.

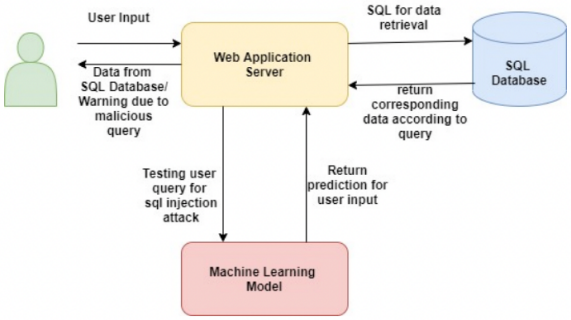
Instead of a two-way split of the selected dataset, we can consider a three-way split, consisting of training, validating and testing sets. Here, the validating set can be used to assess the model's performance before actually running with the testing set. This way, we can fine-tune any hyperparameters if necessary and further minimize any risks in overfitting and underfitting on unseen data.

For Data Processing, we can consider resampling techniques, such as undersampling and oversampling, which would allow us to address class imbalance in datasets. Undersampling would reduce the number of entries in the majority class (in this case, harmless SQL queries), while oversampling increases the number of entries in the minority class, by generating SQL injection based on the known labels.

4 Conclusion

SQL Injection is one of the oldest and most well-known attacks that targets databases on web applications. It is the most effective method for stealing data from backend databases, especially from relational databases by MySQL. With SQL Injection, the attacker can obtain and employ command statements to modify database schemes of sensitive and private information. [1] Identifying potential SQL injection is essential to ensure the security and integrity of web applications and their corresponding data, which was made easier through the rapid growth of machine learning.

Although experts have attempted to apply numerous solutions to solve the SQL injection problem, there still remain issues with this because they do not fully fix the underlying problem and leave room for limitations. Studies done by SQL injection Attacks (SQLIA) by Gartner Group on over 300 internet websites have shown most of them could be vulnerable to SQLIA. [31] Some of the more well-known companies that were on the list are Travelocity, FTD.com, and Guess Inc. One possible solution that can address SQLIA is improving programming techniques, such as escaping single quotes, limiting character length of input, sanitizing queries and filtering exception messages. [32]



With the rapid growth of machine learning and its application in today's technology, cybersecurity engineers and data scientists can consider integrating machine learning into the system's infrastructure, enhancing its security by automating the process of detecting potential SQL injections. Additionally, machine learning algorithms can adapt and evolve over time by continuously learning from new data, allowing professionals to stay ahead of emerging threats and deploy proactive measures against them.

5 References

[1] K. Wei, M. Muthuprasanna and Suraj Kothari, "Preventing SQL injection attacks in stored procedures," Australian Software Engineering Conference (ASWEC'06), Sydney, NSW, Australia, 2006, pp. 8 pp.-198, doi: 10.1109/ASWEC.2006.40.

[2] Mohammed Nasereddin, Ashaar ALKhamaiseh, Malik Qasaimeh Raad Al-Qassas (2023) A systematic review of detection and prevention techniques of SQL injection

[3] 2017 2nd International Conference on Knowledge Engineering and Applications (ICKEA), London, UK, 2017, pp. 55-59, doi: 10.1109/ICKEA.2017.8169902. [4] Halfond, William G., Jeremy Viegas, and Alessandro Orso. "A classification of SQL-injection attacks and countermeasures." Proceedings of the IEEE international symposium on secure software engineering. Vol. 1. IEEE, 2006.

[5] Lawal, M. A., Abu Bakar Md Sultan, and Ayanloye O. Shakiru. "Systematic literature review on SQL injection attack." International Journal of Soft Computing 11.1 (2016): 26-35.

[6] Singh, G., Kant, D., Gangwar, U., Singh, A.P. (2015). SQL Injection Detection and Correction Using Machine Learning Techniques.

[7] G. Yiğit and M. Arnavutoğlu, "SQL Injection Attacks Detection Prevention Techniques," vol. 9, no. 5, 2017, DOI: 10.7763/IJCTE.2017.V9.1165.

[8] F. Mavituna, "Deep blind sql injection," White Paper, 2008.

[9] Alghawazi M, Alghazzawi D, Alarifi S. Detection of sql injection attack using machine learning techniques: a systematic literature review. J Cybersecur Privacy. 2022;2(4):764–77.

[10] Gandhi N, Patel J, Sisodiya R, Doshi N, Mishra S. A CNN-BiLSTM based approach for detection of SQL injection attacks. In: 2021 international conference on computational intelligence and knowledge economy. 2021. p. 378–383.

[11] Zhang K. A machine learning based approach to identify SQL injection vulnerabilities. In: 2019 34th IEEE/ACM international conference on software engineering and automation. 2019. p. 1286–1288.

[12] B. Kranthikumar, and R. L. Velusamy, "SQL injection detection using REGEX classifier Journal of Xi'an University of Architecture Technology," vol.12, 2020, pp. 800-909.

[13] D. A. Kindy and A. -S. K. Pathan, "A survey on SQL injection: Vulnerabilities, attacks, and prevention techniques," 2011 IEEE 15th International Symposium on Consumer Electronics (ISCE), Singapore, 2011, pp. 468-471, doi: 10.1109/ISCE.2011.5973873.

[14] J. Abirami, R. Devakunchari, and C. Valliyammai, "A top web security vulnerability sql injection attack—survey," in 2015 Seventh International Conference on Advanced Computing (ICoAC). IEEE, 2015, pp. 1–9.

[15] Ahmed, A. Shachi, M. (2021, November). SQL Injection Dataset. Retrieved November 11, 2023 from https://www.kaggle.com/datasets/sajid576/sql-injection-dataset/data.

[16] S. S. A. Krishnan, A. N. Sabu, P. P. Sajan, and A. L. Sreedeep, "SQL Injection Detection Using Machine Learning," vol 11, no. 3, pp. 300–310.

[17] L. Erdődi, Å. Å. Sommervoll, and F. M. Zennaro, "Journal of Information Security and Applications Simulating SQL injection vulnerability exploitation using Q-learning reinforcement learning agents," J. Inf. Secur. Appl., vol. 61, no. July, p. 102903, 2021, DOI: 10.1016/j.jisa.2021.102903.

[18] Azman, Muhammad Amirulluqman, Mohd Fadzli Marhusin, and Rossilawati Sulaiman. "Machine learning-based technique to detect SQL injection attack." Journal of Computer Science, 2021.

[19] S. Miller and C. Busby-Earle, "The impact of different botnet flow feature subsets on prediction accuracy using supervised and unsupervised learning methods," International Journal of Internet Technology and Secured Transactions, vol. 5, no. 2, pp. 474–485, 2016.

[20] A. Joshi and V. Geetha, "SQL Injection detection using machine learning," 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), Kanyakumari, India, 2014, pp. 1111-1115, doi: 10.1109/ICCICCT.2014.6993127.

[21] P. Kumar and R. K. Pateriya, "A survey on SQL injection attacks, detection and prevention techniques," 2012 Third International Conference on Computing, Communication and Networking Technologies (ICCCNT'12), Coimbatore, India, 2012, pp. 1-5, doi: 10.1109/ICCCNT.2012.6396096.

[22] S. Mishra, "SQL injection detection using machine learning," 2019.

[4] B. Kranthikumar, and R. L. Velusamy, "SQL injection detection using REGEX classifier. Journal of Xi'an University of Architecture Technology," vol.12, 2020, pp. 800-909.

[23] U. Farooq, "Ensemble Machine Learning Approaches for Detection of SQL Injection Attack," Tehnički glasnik, vol.15, , 2021, pp. 112-120.

[24] Akinsola, J E T Oludele, Awodele A., Idowu Kuyoro, Shade. (2020). SQL Injection Attacks Predictive Analytics Using Supervised Machine Learning Techniques. International Journal of Computer Applications Technology and Research. 9. 139-149. 10.7753/IJCATR0904.1004.

[25] Saidu Aliero, Muhammad Ardo, Abdulhamid Ghani, Imran Atiku, Mustapha. (2016). Classification of Sql Injection Detection And Prevention Measure. IOSR Journal of Engineering. Volume 6. 06-17.

[26] Wei Zhang, Yueqin Li, Xiaofeng Li, Minggang Shao, Yajie Mi, Hongli Zhang, Guoqing Zhi, "Deep Neural Network-Based SQL Injection Detection Method", Security and Communication Networks, vol. 2022, Article ID 4836289, 9 pages, 2022. https://doi.org/10.1155/2022/4836289

[27] Jianwei Hu, Wei Zhao, and Yanpeng Cui. 2020. A Survey on SQL Injection Attacks, Detection and Prevention. Proceedings of the 2020 12th International Conference on Machine Learning and Computing (ICMLC '20). Association for Computing Machinery, New York, NY, USA, 483–488. https://doi.org/10.1145/3383972.3384028

[28] Hasan, Musaab Balbahaith, Zayed Tarique, Mohammed. (2019). Detection of SQL Injection Attacks: A Machine Learning Approach. 1-6. 10.1109/ICECTA48151.2019.8959617.

[29] S. O. Uwagbole, W. J. Buchanan and L. Fan, "Applied Machine Learning predictive analytics to SQL Injection Attack detection and prevention," 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Lisbon, Portugal, 2017, pp. 1087-1090, doi: 10.23919/INM.2017.7987433.

[30] Jemal, Ines Cheikhrouhou, Omar Hamam, Habib Mahfoudhi, Adel. (2020). SQL Injection Attack Detection and Prevention Techniques Using Machine Learning. International Journal of Applied Engineering Research. 569-580.

[31] Alghawazi M, Alghazzawi D, Alarifi S. Detection of SQL Injection Attack Using Machine Learning Techniques: A Systematic Literature Review. Journal of Cybersecurity and Privacy. 2022; 2(4):764-777. https://doi.org/10.3390/jcp2040039

[32] K. Wei, M. Muthuprasanna, and S. Kothari, "Preventing sql injection attacks in stored procedures," in Software Engineering Conference, 2006. Australian. IEEE, 2006, pp. 8–pp.