International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V71I11P202

Volume 71 Issue 11, 18-26, November 2023

Original Article

A Supervised Ensemble Learning-Based Approach to Mitigate SQL Injection Attack on Smart City Data

Asifiqbal Sirmulla[1], M. Prabhakar[2]

[1,2]*Computer Science & Engineering, Reva University Bangalore, Karnataka, India.*

[1]*Corresponding Author : r20pcs03@cit.reva.edu.in*

Received: 25 March 2023 Revised: 18 June 2023 Accepted: 27 September 2023 Published: 04 November 2023

Abstract *- Currently, the use of internet-based applications and technologies has grown drastically. Due to this growth, a massive amount of data is generated and stored on webservers. This increased use of these technologies faces several challenges, such as data vulnerability, security threats and data privacy. SQL-based programming language is widely adopted to access these data due to its simple and efficient use. However, the attacks can use the SQL-based query injection method, where they can insert malicious queries, posing serious threats to the server. Several techniques have been presented in the past, such as blacklisting and rule-based detection methods, but these methods fail to detect SQL injection attacks due to their diversity in input queries. Thus, currently, machine learning-based schemes have gained massive attention in this field, where supervised and unsupervised methods are widely employed. However, the varying nature of SQL queries demands a stable architecture. This work presents a machine learning-based approach for SQL injection attack detection by introducing an ensemble machine learning approach where SVM, NN, discriminant and random forest classifiers are employed. The experimental analysis shows that the average accuracy performance is achieved as 95.88%, 96.23%, 96.61%, 95.11%, and 99.30% using KNN, Discriminant classifier, Naïve Bayes, SVM, and proposed ensemble classification.*

Keywords *- SQL injection, Machine learning, Ensemble learning, Malicious query prediction, SQLIA.*

1. Introduction

The world has noticed a tremendous technological revolution and growth in various fields where the Internet is considered one of the fastest-growing technologies, and it has worked as a radical element for various other technologies [1]. Due to this holistic growth, the world has emerged as a global village. Now, the internet and computer networks are considered an essential component of life. Due to the significant advantages of the World Wide Web and its accessibility, it is widely adopted in various sectors such as e-commerce, banking, e-government, health care, etc. Moreover, the latest web technologies, such as HTML5, have further accelerated the use of web technologies [2].

Moreover, these web-based technologies process several types of information along with private information for organizations, such as emails, transactions, or information related to individuals, such as shopping, social activities, etc. Many other innovative services are yet to come very shortly. As these applications are adopted for various applications, they attract more vulnerabilities, which can lead to compromise of the World Wide Web and attacks to contaminate and steal information. In general, the user submits the request to a web server with the help of Hypertext Markup Language (HTML) forms, Uniform Resource Positions (URLs) and other fields. This data is processed without checking and filtering the query [3]. Processing this unfiltered data, the attackers get a chance to perform SQL injection, which can act as faulty query input, resulting in an attack on the webserver [4].

SQL (Structured Query Language) is a widely adopted programming language used explicitly for database management. This programming language allows access to and manipulation of the database. Many WEB-based programming languages, such as PHP or JAVA, facilitate several methods to construct and execute the queries [5]. These programming languages accept the user inputs or statements, which are further concatenated to form the final query as input to the web server.

Several programming languages have been introduced to encode the user requests to construct the SQL query statement. At this stage, attackers can use SQL injection attacks where malicious code fragments are inserted in the query statements, which can malfunction the working server and execute inappropriate queries. This type of contaminated query can leak data and damage the database. For example, attackers use SQL injection to obtain login credentials and other details, which can cause a threat to private data. Attackers inject these queries into the programme as input injection, server variable injection, cookie injection etc. [6]. Below, Figure 1 depicts the process of SQL query injection.

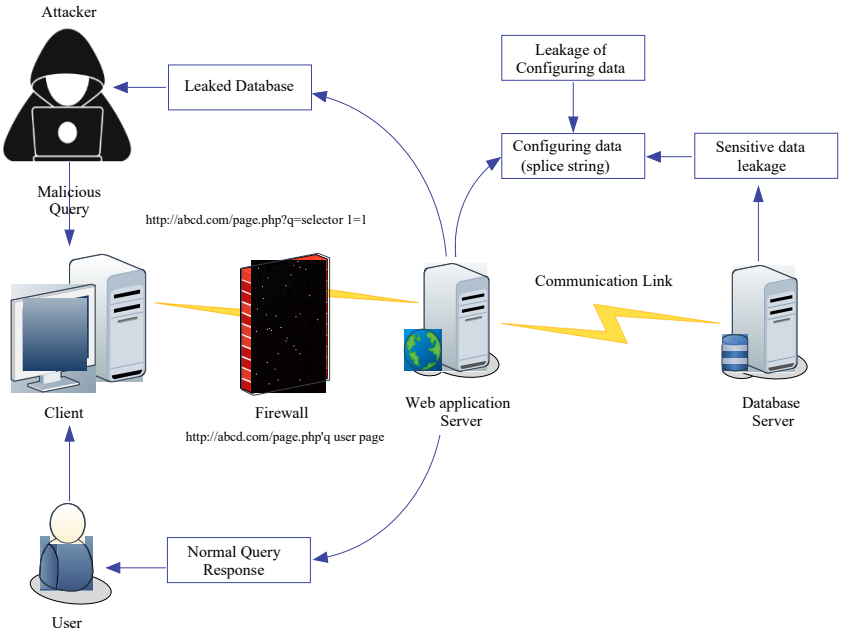

Fig. 1 SQL query injection

The current World Wide Web scenario is highly dependent on web-based servers for data storage, accessing the data from anywhere at any point in time. However, this has aggravated the vulnerability of private data to several security threats. SQL injection attacks are the most common attacks that attackers widely use, and the attack techniques constantly change as website technology advances [1]. The SQL language's structure may be changed, and it enables a variety of coding styles. As a result, several classic detection strategies, such as blocklisting and rule-based detection, cannot provide a more effective protective impact.

1.1. Research Gap

The field of SQLIA detection leveraging machine learning has undoubtedly made substantial strides, yet several critical research gaps persist, necessitating further investigation and innovation. These gaps include:

- Adaptation to Evolving Attack Strategies: SQL injection attackers consistently employ novel evasion and obfuscation techniques to thwart detection. To effectively combat this, research should concentrate on devising models capable of adapting to these ever-evolving attack strategies.

- Mitigating False Positives: A common issue with machine learning-based SQL injection detection systems is their propensity to generate high rates of false positives. This phenomenon can be detrimental in real-world scenarios. Therefore, it is imperative to develop methods to curtail false positives while maintaining a high detection accuracy rate.

- Cross-Domain Applicability: The majority of existing research is domain-specific, focusing on individual applications or websites. To enhance versatility and applicability, researchers should concentrate on crafting models that can be deployed across diverse domains, ensuring a more comprehensive safeguard.

- Resource Efficiency: Many current machine learning models utilized for SQL injection detection are computationally intensive, rendering them unsuitable for resource-constrained environments. As such, the research community should explore methodologies to engineer efficient models that can be readily deployed across a spectrum of scenarios.

- Real-Time Detection Imperative: In the security realm, real-time detection of SQL injection attacks holds paramount importance for preventing potential breaches. Thus, research endeavors should be geared towards devising models and techniques capable of offering real-time detection with minimal latency, thus fortifying security protocols.

1.2. Problem Statement

Within the domain of SQLIA detection through the utilization of machine learning, the current state of the art presents a series of shortcomings characterized by an inability to adapt to evolving attack strategies, a propensity for generating false positives, domain-specificity, resource-intensive models, and an insufficiency in providing real-time detection.

The primary objective of this research endeavor is to address these critical challenges by developing an innovative and resource-efficient model that excels in real-time SQL injection attack detection, reduces false positives, exhibits adaptability across various domains, and is amenable to deployment in resource-constrained environments. This research aims to fortify the security infrastructure of applications and systems by mitigating the risks posed by SQL injection attacks.

1.3. Contributions

The research community has presented several techniques on SQL injection detection in recent years, although the detection scope is generally confined to specific subgroups of SQL injection. An effective SQL injection detection framework is of utmost importance, capable of identifying all variations of SQL injection attacks while remaining adaptable to address emerging attacks promptly. Similarly, the varying characteristics of these attacks affect the process of determining the types of attacks. Generally, combining numerous attacks weakens the traditional machine-learning process because these approaches are trained for a specific set of attributes. Thus, this work focuses on developing a new machine learning that considers different classifiers and constructs an ensemble model by employing the bagging/boosting model. Prior to this, a feature selection model is also implemented to obtain robust features to improve the training.

1.4. Organization

The remainder of this manuscript is structured as follows: Section II provides a comprehensive literature review, Section III outlines the proposed approach for SQLIA detection, and Section IV presents the results of the proposed method, including a comparative analysis to demonstrate its robustness. Finally, Section V offers concluding remarks on this research and outlines potential future avenues of study.

2. Literature Survey

This section presents the description of recent techniques of SQL injection detection by using machine learning algorithms. Li et al. [9] developed a deep machine learning-based approach, introducing a deep-forest method to detect complex SQL injection attacks. According to this approach, the deep forest approach is improved. This is achieved by concatenating the input and output of the previous layer output. This concatenation helps mitigate the feature degradation issue caused by the increased number of layers. Furthermore, the AdaBoost algorithm is also introduced to update the feature weights.

Li et al. [10] reported that traditional machine learning methods use manually defined attributes; thus, the detection performance highly depends on the feature extraction. However, numerous verities of SQL attacks are present, which vary in characteristics compared with the pre-trained models. Thus, these methods fail to detect SQL attacks. The authors have presented a long, short-term memory scheme capable of handling complex data to deal with these issues.

Kasim et al. [11] developed an ensemble approach, which includes data processing, bagging, detection and prevention phases. The data pre-processing phase uses the regex function to identify the standard attributes or queries. Afterwards, feature extraction and feature vector formulation are applied to the data. Further, a tree-based bagging model is presented to detect the attack pattern. This model classifies the attacks as simple, unified, latera and clean queries. Corresponding to each query, it suggests preventive methods to deal with different types of attacks.

Batista et al. [12] introduced a hybrid of fuzzy and neural network models to detect attacks. In this approach, the neural network generates the rules with the help of nebulous logic neurons. Generally, these networks suffer from the overfitting issue, which is resolved using the regularization model. Similarly, the fuzzy neural network is used to perform the binary classification based on the pre-defined rules. Tripathy et al. [13] presented an experimental analysis where different classifiers such as AdaBoost, Stochastic gradient descent, random forest, deep learning and Boosted tree classifiers are used to detect the SQL attacks.

These machine learning schemes consider data cleaning, feature extraction, feature selection and employing the classification model steps to obtain the classification of attacks. Zhuo et al. [14] presented a deep learning approach which uses word embedding methods and a continuous bag of words (CBOW) to construct the word embedding matrix. Later, they built an LSTM-based model, which includes an input layer, word embedding layer, dense layer, activation, drop output layer, fully connected layer, sigmoid function and final output layer.

Fang et al. [15] presented a combined machine learning model based on a support vector machine and LSTM networks. First, SQL tokenization is applied, followed by the Likelihood Ratio. This generates an SQL word vector model. This model is considered a train set and fed into the LSTM model, which classifies the patterns as genuine and injected queries. Kherbache et al. [16] discussed that feature selection is considered the critical phase for anomaly-based intrusion detection by selecting the most significant attributes to improve efficiency and minimize false positives. Based on this concept, Aqsa Afroz et al. [17] presented a combined agglomerative hierarchal clustering scheme with support vector machine classification. Feature selection is performed based on the variance and grouped based on their similarities.

3. Proposed model

This section presents the proposed solution for SQL injection attack detection by using a machine learning technique where an ensemble technique is presented to reduce the misclassification problem.

Moreover, feature redundancy and dimensionality create an excessive burden on the network during training. Moreover, redundant attributes impact learning performance, degrading the overall classification accuracy.

3.1. Overview

The proposed approach is based on machine learning techniques where feature selection and ranking are performed during the pre-processing phase. Further, an ensemble of SVM, NN, discriminant analysis and random forest stacked to generate the meta-classifier.

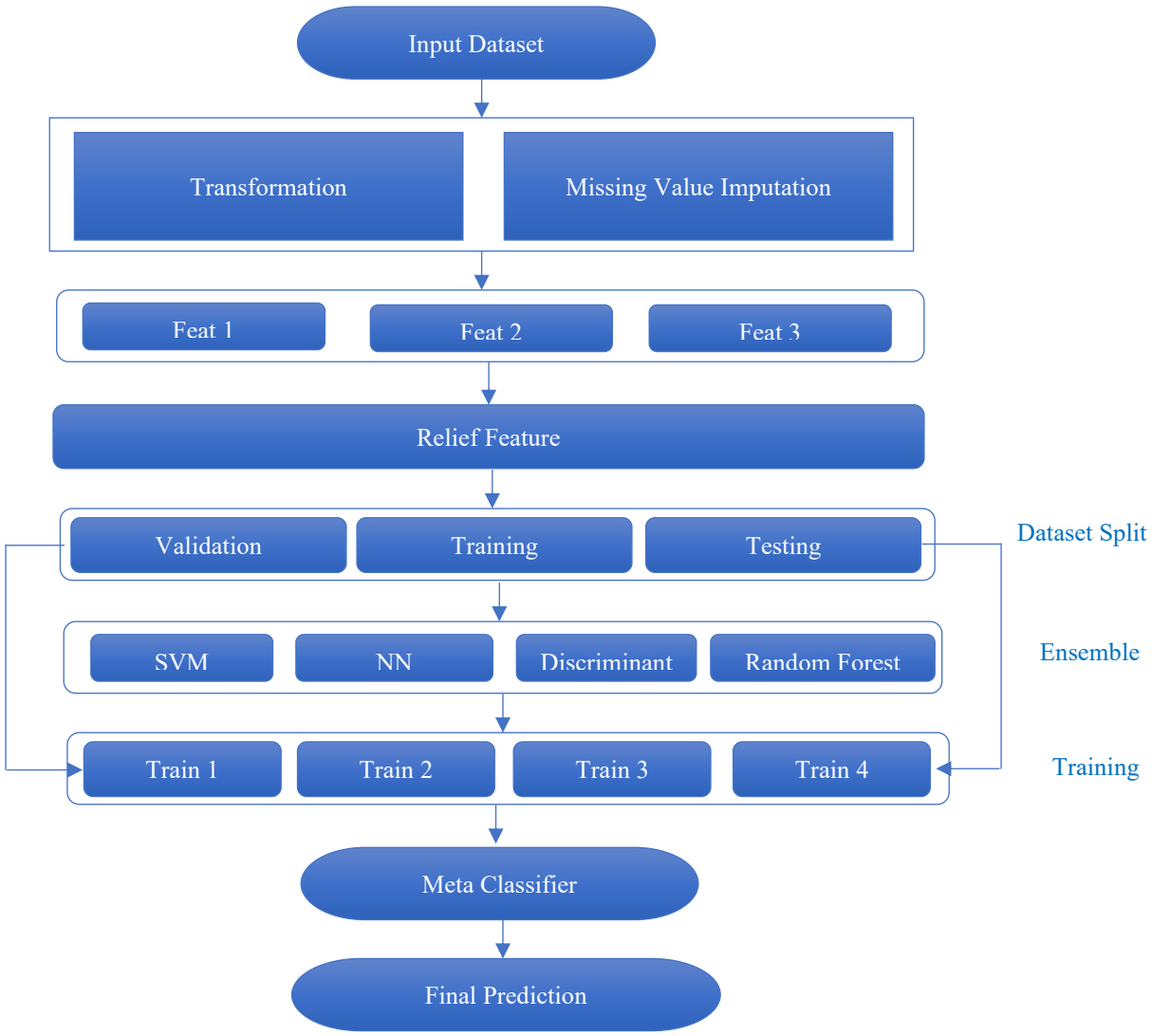

Fig. 2 The architecture of the proposed ensemble approach

According to this architecture, data is first pre-processed, where data transformation and missing value imputation are applied. This stage generates a matrix which contains features and their corresponding labels. Further, the relief feature selection method ranks the features according to their weights. In the next phase, training is performed with the help of ensemble learners, and finally, testing samples are given as input to generate the final classification.

3.2. Feature Selection

Data dimension plays a vital role in this field of machine learning techniques. High-dimensional data requires huge processing time and consumes more resources. Moreover, redundancy in the data leads to degrade the classifier performance. This work has adopted a relief-based feature selection algorithm [8]. It computes the statistics for each attribute and uses these parameters to estimate the 'relevance' of the f attribute. These features are represented as feature weights as $W[A]$ Weight of feature A, which ranges from -1 to +1. Below algorithm 1 shows the pseudocode for relief-based feature selection.

Input: feature vector and its corresponding class values (in our attack attributes and their classes are considered as 0 and 1 for attack and normal query)

$n \leftarrow$number of training instances present in the dataset

$a \leftarrow$ number of attributes

Parameters: $m \leftarrow$ denotes the random training samples which are used to update W

Step 1: initialize the weights as $W[A] := 0$

Step 2: for i=1 to m do

Step 3: select a random target form instance R_i

Step 4: identify the nearest hit and miss from the instances

Step 5: for $A = 1\ to\ a$ dor

// compute and update the weights

$$W[A] = W[A] - \frac{diff(A, R_i, H)}{m} + \frac{diff(A, R_i, M)}{m}$$

Step 6: end for

Step 7: end for

Step 8: return the updated weighted feature scores W and ranked attributes.

According to this approach, the feature selection process is iterated through numerous training instances and selects the random target from R_i instances. Further, this is used to update the feature weights W by considering the target and neighbouring instances. At this stage, the algorithm computes the nearest neighbour instances of the target, where the nearest to the same class is known as the nearest hit. (H) and the other instance is known as nearest miss which is known as nearest miss (M). In the last step, computed weights are updated based on the difference between targets. For discrete attributes, $diff$ is defined as:

$$diff(A, I_1, I_2) = \begin{cases} 0, if\ value\ (A, I_1) = value\ (A, I_2) \\ 1, if\ otehrwise \end{cases} \quad (1)$$

3.3. Classifiers Used for Ensemble Learning

As mentioned before, traditional machine learning fails to provide detection for varied attack types; thus, a robust scheme or architecture is required which can handle different types of attacks which are dynamic in nature of attack (i.e., handling the SQL injection query which has different types of SQL injections). In order to improve the detection accuracy, an ensemble learning approach is introduced, which uses SVM, NN, DT and RF. These classifiers used the stacking approach to formulate the ensemble model of classification. This section briefly describes these techniques to adopt in SQL injection detection.

3.4. Support Vector Machine

The SVM is considered a promising technique in the field of machine learning. This is a type of supervised learning scheme which is widely adopted for binary classification problems. The SVM classifier focuses on identifying the best hyperplane in such a way that it separates all data points from one class to another class. Here, the best hyperplane represents an SVM with the highest margin between two classes of the input data. Below given, Figure 2 depicts the actual and hyperplane-separated data.

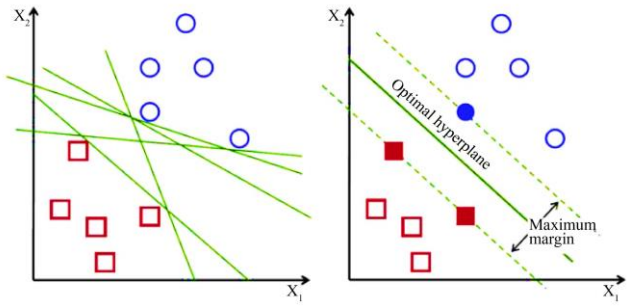

Fig. 3 SVM classifier

According to this approach, the SQL injection dataset is represented as x_j and their corresponding category (SQL attack query or not) are denoted as y_j. Let d be the dimension of the dataset; thus, the training set is represented as $x_j \in R^d$. For this condition, the hyperplane can be represented as:

$$f(x) = x'\beta + b = 0 \quad (2)$$

$\beta \in R^d$ and b are the real numbers.

This approach aims at identifying the best hyperplane which separates the attack regular queries based on their attributes. It focuses on finding the β and b, which are used to minimize the $\|\beta\|$ such a way that for all data points (x_j, y_j),

$$y_j f(x_j) \geq 1 \quad (3)$$

Here, support vectors are x_j on the boundary of the plane where $y_i f(x_i) = 1$. Generally, minimizing the $\|\beta\|$ is the main aim of this approach. The classification can be obtained as follows:

$$Class(z) = sign(z\hat{\beta} + \hat{b}) = sign\left(\hat{f}(z)\right) \quad (4)$$

$\left(\hat{f}(z)\right)$ is the classification score and denotes the distance from the decision boundary.

However, kernel functions are applied if a simple hyperplane is not obtained. These functions are polynomial, radial basis (Gaussian) and Multilayer perceptron or sigmoid (neural network). These functions are expressed as:

Polynomial function:

$$G(x_1, x_2) = (1 + x_1 x_2)^p$$

Radial basis function (Gaussian):

$$G(x_1, x_2) = \exp(-\|x_1 - x_2\|^2) \quad (5)$$

Multilayer function:

$$G(x_1, x_2) = \tanh(p_1 x_1 x_2 + p_2)$$

3.5. Discriminant Analysis

The discriminant is a supervised classification scheme which uses Gaussian distribution to generate the data corresponding to the classes. This task is accomplished in two phases: the training phase and the prediction phase. In the training phase, the fitting function focuses on estimating the Gaussian distribution parameter for each class. Similarly, the prediction phase considers these trained models and finds the class with the smallest misclassification to identify the class of the input test sample. In the training phase, it considers the input SQL data and their corresponding category, which are processed through a multivariate normal distribution. During the fitting process in linear discriminant analysis, it measures the sample mean of each class is, and later sample covariance is measured by subtracting the sample mean of each class. In this phase, it constructs a weighted classifier as:

$$\hat{\mu}_k = \frac{\sum_{n=1}^{N} M_{nk}\, x_n}{\sum_{n=1}^{N} M_{nk}} \quad (6)$$

On the other hand, this approach focuses on minimizing the classification cost for prediction. This function can be expressed as:

$$\hat{y} = \arg\min_{y=1,..K} \sum_{k=1}^{K} \hat{P}(k|x) C(y|k) \tag{7}$$

Where $\hat{y}$ denotes the predicted class, K total number of classes (in our case 2 classes), $\hat{P}(k|x)$ is the posterior probability of class k for x observations and $C(y|k)$ is the cost of classification. The posterior probability is expressed as:

$$P(x|k) = \frac{1}{((2\pi)^d |\Sigma_k|)^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x - \mu_k) \sum_{k}^{-1} (x - \mu_k)^T\right) \tag{8}$$

Based on the probability, the category for input attributes can be classified based on the highest discriminant value as $Pr(Y = k|X = x)$

3.6. Neural Network Classifier

The neural network is a supervised learning algorithm which contains several neurons as units. These neurons are used to convert the input data into some output via the layered arrangement of neurons. Each layer performs some specific function and then passes to the next layer. This process is known as the feed-forward process. Along with data passing, weights are also applied to train the neural network during the learning phase. Figure 4 below shows the architecture of the neural network.

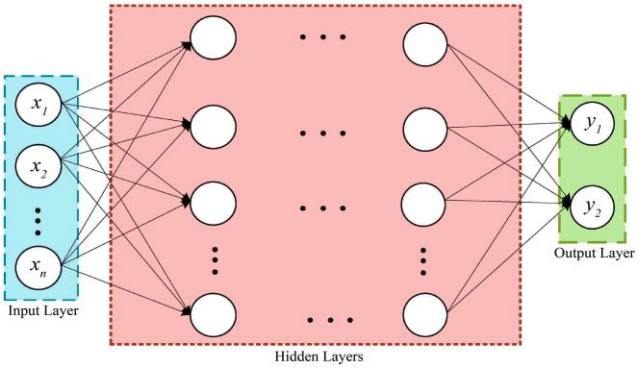

Fig. 4 Neural network classification

3.7. Random Forest

Random forest is a supervised learning scheme which constructs multiple solutions in the form of decision trees, and to obtain the output of classification, the class which is selected the maximum times by most trees is considered the final classification.

Algorithm 1: Random forest classification algorithm

Inputs: pre-processed data, corresponding labels, splitting ratios, hyper parameters (estimators, depth, sample leaf)
Output: predicted classes, classification performance

Step 1: function $Search(estimators, depth, leaf)$
Step 2: generate the dictionary based on estimators, depth, sample leaf
Step 3: initialize the hyper parameter optimizer
Step 4: return the best estimator
Step 5: generate c bootstrap samples
Step 6: for i=1 to c do
Randomly sample the training data D
Create a root node
$BuildTree()$
Step 7: end
Step 8: BuildTree()
Step 9: If N contains attributes only belonging to one class
Step 10: Return
Step 11: Else
Step 12: Randomly pick the p% of attributes where the higher possibility for splitting
Step 13: Select F features from the list based on the highest information gain
Step 14: Generate f child nodes from these attributes
Step 15: procedure
$RFClassifier(train, test, train\ class, test\ class)$
Step 16: use $search$ function to obtain the best estimator
Step 17: Apply $modelfit$ with train and test attributes
Step 18: Predict the labels based on input test data
Step 19: Return predicted class

4. Results and Discussion

This section presents the discussion of the proposed approach and presents the discussion about the outcome of the proposed ensemble approach. The proposed approach is implemented using Python 3.8 and MATLAB2021a installed on the Windows platform.

4.1. Dataset Details

The proposed approach is implemented on publicly available data taken from Kaggle's website [7]. This dataset contains 34,048 entries in two columns. The first column represents the sentences used as normal queries or SQL injection attack queries, and the second column represents the corresponding labels as 1 or 0. Here, 1 represents the "attack query", and 0 represents the "normal query". In this dataset, a total of 22,305 positive and 11,781 negative samples are present, divided into training and testing ratios to obtain the system's classification performance.

4.2. Performance Measurement

Performance evaluation relies on four key parameters: accuracy, sensitivity, specificity, and F-measure. These metrics are derived from the confusion matrix analysis, which encompasses true positives, false positives, true negatives, and false negatives. Below the table shows the representation of the confusion matrix.

Table 1. Confusion matrix

	Positive	**Negative**	**Total**
Positive	T_p	F_p	$T_p + F_p$
Negative	F_N	T_N	$F_N + T_N$
Total	$T_P + F_N$	$T_P + T_N$	

The detection accuracy is computed based on the values obtained as mentioned in the confusion matrix, such as true positive, false positive, false negative, and true positive. The accuracy can be computed as:

$$Accuracy = \frac{T_P + T_N}{T_P + T_N + F_P + F_N} \quad (1)$$

Similarly, the sensitivity performance is computed with the help of a confusion matrix. The sensitivity can be expressed as:

$$Sensitivity = \frac{T_P}{T_P + F_N} \quad (2)$$

The specificity can be computed as:

$$Specificity = \frac{T_N}{T_N + F_P} \quad (3)$$

The F-measure can be computed as:

$$F - measure = \frac{2 \times T_P}{2 \times T_P + F_N + F_P} \quad (4)$$

4.3. Comparative Analysis

In this section, we conduct a comparative analysis to assess the performance of the proposed ensemble classifier in relation to other classification methods. The confusion matrix of the proposed approach is presented in the table below.

Table 2. Confusion matrix by using the proposed classifier

	Positive	Negative
Positive	1298	3
Negative	11	1290

Based on this confusion matrix, the other performance parameters, such as precision, recall, f-measure and ROC, are also measured. The obtained performance is demonstrated in below given table with a weighted average.

Table 3. Detailed classification performance for the proposed approach

Class	TP	FP	Pr	Rec	F1	ROC
1	0.995	0.009	0.991	0.995	0.993	0.986
2	0.990	0.004	0.995	0.990	0.992	0.986
Avg.	0.993	0.007	0.993	0.993	0.993	0.986

Similarly, a support vector machine classification scheme is applied. The obtained confusion matrix is presented in below given table.

Table 4. Confusion matrix by using SVM classifier

	Positive	Negative
Positive	1255	46
Negative	81	1220

Table 5. Detailed classification performance for SVM classifier

Class	TP	FP	Pr	Rec	F1	ROC
1	0.964	0.062	0.939	0.964	0.951	0.904
2	0.937	0.035	0.963	0.937	0.950	0.904
Avg.	0.951	0.048	0.951	0.951	0.951	0.904

Later, the performance of the Naïve Bayes classifier is also measured for the given dataset and obtained the confusion matrix as follows:

Table 6. Confusion matrix by using Naïve Bayes classifier

	Positive	Negative
Positive	1263	38
Negative	24	1277

The detailed classification performance for both classes is presented in below given table.

Table 7. Detailed classification performance for Naïve Bayes classifier

Class	TP	FP	Pr	Rec	F1	ROC
1	0.970	0.018	0.981	0.970	0.976	0.952
2	0.981	0.029	0.971	0.981	0.976	0.952
Avg.	0.976	0.023	0.976	0.976	0.976	0.952

Similarly, a discriminant analysis classifier is implemented, and the obtained confusion matrix is presented in below given table.

Table 8. Confusion matrix by using discriminant analysis classifier

	Positive	Negative
Positive	1290	11
Negative	87	1214

Based on this confusion matrix, the statistical performance parameters are obtained, which are presented in below given table.

Table 9. Detailed classification performance for discriminant classifier

Class	TP	FP	Pr	Rec	F1	ROC
1	0.9915	0.066	0.9368	0.991	0.963	0.925
2	0.9331	0.008	0.9910	0.933	0.961	0.925
Avg.	0.962	0.0377	0.963	0.962	0.962	0.925

Finally, the performance for the K-nearest neighbourhood classifier is measured, and the obtained confusion matrix is presented in below given table.

Table 10. Confusion matrix by using knn classifier

	Positive	Negative
Positive	1256	45
Negative	62	1239

Furthermore, the detailed performance of the classifier is demonstrated in below given table. This experiment contains the performance analysis for each class present in the dataset.

Table 11. Detailed classification performance for knn classifier

Class	TP	FP	Pr	Rec	F1	ROC
1	0.965	0.047	0.953	0.965	0.959	0.919
2	0.952	0.034	0.965	0.965	0.958	0.919
Avg	0.958	0.041	0.9590	0.958	0.958	0.919

The comprehensive classifier performance evaluation is conducted, considering metrics such as Kappa, MAE, RMSE, RAE, RRSE, and Accuracy. We benchmark the performance of our proposed approach against various

established classification methods, including k-nearest neighbourhood, discriminant analysis, Naïve Bayes, and support vector machine classification. Below, the table 13 shows the summarised performance of the proposed approach and its comparison with other existing classifiers. The comparative study shows that the overall accuracy is obtained as 95.88%, 96.23%, 96.61%, 95.11%, and 99.30% using KNN, Discriminant Analysis, Naïve Bayes, SVM, and Proposed classifier, respectively.

The figure 5 illustrates the average performance of various classifiers alongside our proposed ensemble approach. Average performance is evaluated based on accuracy, ROC, F-Measure, recall, and precision.

The comparative analysis reveals that our proposed approach attains an average performance of 99.315% precision, 99.305% recall, 99.305% F-Measure, 98.61% ROC, and 99.3% accuracy.

Table 12. Performance evaluation of proposed approach against various established classification methods

	Kappa	MAE	RMSE	RAE	RRSE	Accuracy
KNN	0.917	0.041	0.202	12.1	4.9	95.8
Discriminant analysis	0.924	0.037	0.194	13.2	5.1	96.2
Naïve Bayes	0.952	0.023	0.154	20.9	6.4	96.6
SVM	0.902	0.048	0.220	10.244	4.5	95.1
Proposed	0.986	0.006	0.083	72.2	12.0	99.30

Table 13. Comparative analysis of proposed approach against various established classification methods

Classifier		TPR	FPR	Precision	Recall	FMeasure	ROC	Accuracy
KNN	Class 1	0.9564	0.0477	0.9530	0.9654	0.9591	0.9194	95.88
	Class 2	0.9523	0.0346	0.9650	0.9523	0.9586	0.9194	
Discriminant Analysis	Class 1	0.9915	0.0669	0.9368	0.9915	0.9634	0.9252	96.23
	Class 2	0.9391	0.0085	0.9910	0.9331	0.9612	0.9252	
Naïve Bayes	Class 1	0.9708	0.0184	0.9814	0.9708	0.9760	0.9529	96.61
	Class 2	0.9816	0.0292	0.9711	0.9818	0.9763	0.9529	
SVM	Class 1	0.9646	0.0623	0.9394	0.9646	0.9518	0.9046	95.11
	Class 2	0.9377	0.0354	0.9637	0.9377	0.9505	0.9046	
Proposed classifier	Class 1	0.9955	0.0094	0.9910	0.9955	0.9932	0.9861	99.30
	Class 2	0.9906	0.0045	0.9953	0.9906	0.9929	0.9861	

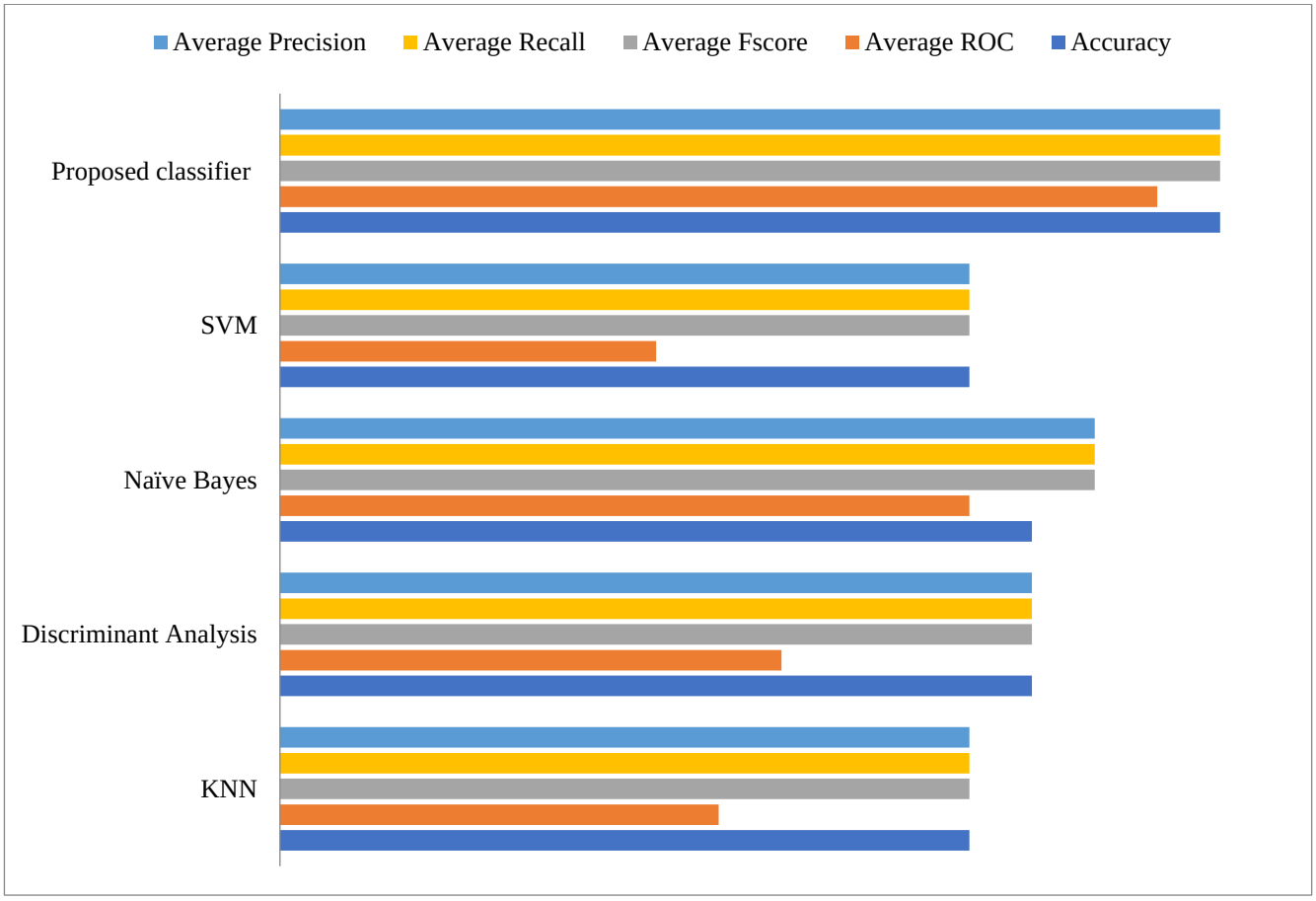

Fig. 5 Average performance of various classifiers and proposed ensemble approach

5. Conclusion

According to the recent studies presented by OWASP, an injection attack is one of the most vulnerable attacks out of the top ten security dangers. SQL injection, on the other hand, is one of the most common forms of injection assaults. SQL injection may cause significant network damage due to its different kinds and rapid changes, culminating in data leaking and website paralysis. Detecting SQL injection represents a complex challenge owing to the diversity of attack payloads, the wide range of attack strategies, and the multitude of attack vectors. How to successfully defend against SQL injection attacks has been the focus and frontier of online security in recent years. Thus, this approach focuses on machine learning-based techniques and presents an ensemble machine learning approach which considers neural network, random forest, SVM and discriminant analysis as learners. The performance of the proposed approach is measured in terms of Precision, Recall, F-Measure, ROC, and Accuracy, which are obtained as 0.99315, 0.99305, 0.99305, 0.9861, and 99.30, respectively.

References

[1] Tomás Sureda Riera et al., "A New Multi-Label Dataset for Web Attacks CAPEC Classification Using Machine Learning Techniques," *Computers and Security*, vol. 120, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[2] Ayush Falor et al., "A Deep Learning Approach for Detection of SQL Injection Attacks Using Convolutional Neural Networks," *Proceedings of Data Analytics and Management*, pp. 293-304, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[3] O.P. Voitovych, O.S. Yuvkovetskyi, and L.M. Kupershtein, "SQL Injection Prevention System," *International Conference Radio Electronics and Info Communications (UkrMiCo), IEEE*, pp. 1-4, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[4] William G.J. Halfond, and Alessandro Orso, "AMNESIA: Analysis and Monitoring for Neutralizing SQL-Injection Attacks," *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering*, pp. 174-183, 2005. [CrossRef] [Google Scholar] [Publisher Link]

[5] Chrystian Byzdra, and Grzegorz Kozieł, "Analysis of the Defending Possibilities Against SQL Injection Attacks," *Journal of Computer Sciences Institute*, vol. 13, pp. 339-344, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[6] Haifeng Gu et al., "DIAVA: A Traffic-Based Framework for Detection of SQL Injection Attacks and Vulnerability Analysis of Leaked Data," *IEEE Transactions on Reliability*, vol. 69, no. 1, pp. 188-202, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[7] Syed Saqlain Hussain Shah, SQL Injection Dataset, Kaggle. [Online]. Available: https://www.kaggle.com/datasets/syedsaqlainhussain/sql-injection-dataset

[8] Ryan J. Urbanowicz et al., "Relief-Based Feature Selection: Introduction and Review," *Journal of Biomedical Informatics*, vol. 85, pp. 189-203, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[9] Qi Li et al., "A SQL Injection Detection Method Based on Adaptive Deep Forest," *IEEE Access*, vol. 7, pp. 145385-145394, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[10] Qi Li et al., "LSTM-Based SQL Injection Detection Method for Intelligent Transportation System," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 4182-4191, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[11] Ömer Kasim, "An Ensemble Classification-Based Approach to Detect Attack Level of SQL Injections," *Journal of Information Security and Applications*, vol. 59, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[12] Lucas Oliveira Batista et al., "Fuzzy Neural Networks to Create an Expert System for Detecting Attacks by SQL Injection," *The International Journal of Forensic Computer Science - IJoFCS*, vol. 13, no. 1, pp. 8-21, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[13] Dharitri Tripathy, Rudrarajsinh Gohil, and Talal Halabi, "Detecting SQL Injection Attacks in Cloud Saas Using Machine Learning," *IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS), IEEE*, pp. 145-150, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[14] Z. Zhuo et al., "Long Short-Term Memory on Abstract Syntax Tree for SQL Injection Detection," *IET Software*, vol. 15, no. 2, pp. 188-197, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[15] Yong Fang et al., "WOVSQLI: Detection of SQL Injection Behaviors Using Word Vector and LSTM," *Proceedings of the 2nd International Conference on Cryptography, Security and Privacy*, pp. 170-174, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[16] Meriem Kherbache, Kamal Amroun, and David Espes, "A New Wrapper Feature Selection Model for Anomaly-Based Intrusion Detection Systems," *International Journal of Security and Networks*, vol. 17, no. 2, pp. 107-123, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[17] Aqsa Afroz et al., "An Algorithm for Prevention and Detection of Cross Site Scripting Attacks," *SSRG International Journal of Computer Science and Engineering,* vol. 7, no. 7, pp. 8-18, 2020. [CrossRef] [Publisher Link]