https://doi.org/10.1007/s00521-024-09429-z

ORIGINAL ARTICLE

Detecting SQL injection attacks by binary gray wolf optimizer and machine learning algorithms

Bahman Arasteh[1,6] · Babak Aghaei[2] · Behnoud Farzad[3] · Keyvan Arasteh[1] · Farzad Kiani[4] · Mahsa Torkamanian-Afshar[5]

Received: 27 July 2023 / Accepted: 15 January 2024 / Published online: 27 February 2024

Abstract

SQL injection is one of the important security issues in web applications because it allows an attacker to interact with the application's database. SQL injection attacks can be detected using machine learning algorithms. The effective features should be employed in the training stage to develop an optimal classifier with optimal accuracy. Identifying the most effective features is an NP-complete combinatorial optimization problem. Feature selection is the process of selecting the training dataset's smallest and most effective features. The main objective of this study is to enhance the accuracy, precision, and sensitivity of the SQLi detection method. In this study, an effective method to detect SQL injection attacks has been proposed. In the first stage, a specific training dataset consisting of 13 features was prepared. In the second stage, two different binary versions of the Gray-Wolf algorithm were developed to select the most effective features of the dataset. The created optimal datasets were used by different machine learning algorithms. Creating a new SQLi training dataset with 13 numeric features, developing two different binary versions of the gray wolf optimizer to optimally select the features of the dataset, and creating an effective and efficient classifier to detect SQLi attacks are the main contributions of this study. The results of the conducted tests indicate that the proposed SQL injection detector obtain 99.68% accuracy, 99.40% precision, and 98.72% sensitivity. The proposed method increases the efficiency of attack detection methods by selecting 20% of the most effective features.

Keywords Software security · SQL injection attacks · Artificial neural network · Feature selection · Binary gray wolf optimization algorithm · Accuracy

1 Introduction

Software security is one of the major quality metrics of a software product [1]. SQL injection (SQLi) is one of the most serious software security concerns that any software development team should avoid. SQLi is a web security flaw that allows an attacker to interact with database queries made by an application. SQL injection attacks are just malicious queries that change a typical SQL command into a malicious type. An attacker can directly use SQL queries to fetch information from databases to receive unlimited data and unauthorized access. In this attack, an attacker can alter or remove the data stored in the database using only a web browser. An attacker can escalate a SQL injection attack to compromise the underlying server or other back-end infrastructure or launch a denial-of-service attack in specific circumstances. Regarding the reports by

✉ Bahman Arasteh
Bahman.arasteh@istinye.edu.tr

[1] Department of Software Engineering, Faculty of Engineering and Natural Science, Istinye University, Istanbul, Turkey

[2] Department of Computer Engineering, Malekan Branch, Islamic Azad University, Malekan, Iran

[3] Department of Computer Engineering, Seraj Institute, Tabriz, Azarbaijan Province, Iran

[4] Data Science Application and Research Center (VEBIM), Fatih Sultan Mehmet Vakif University, Istanbul, Turkey

[5] Computer Engineering Department, Faculty of Engineering, Istanbul Topkapi University, 34087 Istanbul, Turkey

[6] Applied Science Research Center, Applied Science Private University, Amman, Jordan

the open worldwide application security project (OWASP), the vulnerability of SQL injection attacks is among the top 10 web application security risks [1–3].

In most web applications, filtering techniques are used to prevent these types of attacks. However many SQL injection attacks can bypass data filtering techniques. The traditional methods used in web applications cannot effectively defend the database against SQLi attacks. Therefore, a more effective method is needed to detect and prevent SQL injection attacks. Machine learning (ML) and artificial intelligence (AI) methods are used to develop the SQLi detectors; these detectors (classifiers) are created using training datasets and a classification algorithm. The SQLi detectors are classifiers that classify the malicious and non-malicious SQL queries. Different SQLi classifiers have been created using different supervised, semi-supervised, and unsupervised learning methods on different training datasets. In the first stage of the methods, an SQLi classifier is provided using the training dataset. A test dataset is used to put the classifier that was created during the training stage to the test. Insufficient accuracy, precision, error rate, and performance are the main demerits of the classifiers created by machine learning methods to detect SQLi attacks.

The variables that are fed into the machine learning models are referred to as the features of the dataset. To train an optimal model with optimal accuracy, only the effective features of the dataset should be used in the training stage. Finding the most effective features of the dataset is a combinatorial NP-complete optimization problem. Feature selection is the process of selecting the minimal and most effective features of the training dataset. The number of input variables (features) should be kept to a minimum to reduce the cost of classification model creation. Furthermore, reducing the number of features improves the model's accuracy and precision. Various metaheuristic search algorithms were used to select the optimal features of the dataset. In different feature selection problems, a metaheuristic algorithm may perform differently. The main drawbacks of the previous SQLi methods are:

- The higher error rate of the obtained results.
- Insufficient accuracy of the created SQLi detectors by the ML algorithms.
- Insufficient precision of the created SQLi detectors in detecting the SQLi attacks.
- The higher number of selected features is the other drawback of the metaheuristic algorithms used for feature selection in the SQLi detectors.
- Lower convergence speed and consequently insufficient performance of the feature selectors used in the previously suggested SQLi detectors.

Hence, the main objectives of this study are as follows:

- Enhancing the accuracy of the SQLi detection method.
- Enhancing the precision of the SQLi detection method.
- Enhancing the sensitivity of the SQLi detection Method.
- Reducing the Error rate of the SQLi detection method.
- Finding the minimum number of most effective features in the SQLi detection method.

In this paper, an effective method to detect SQLi attacks is proposed. In this method, firstly, a specific training dataset consisting of 13 features was prepared. In the second stage, due to the unique capabilities of the gray wolf optimization (GWO) algorithm, two different binary versions of it (bGWO) were developed to select the most effective features of the training dataset. The optimal datasets created by the bGWOs are used by the different machine learning algorithms to build effective SQLi detectors. The desired classification model to detect SQLi was created by the artificial neural networks (ANN) and decision tree (DT) algorithms. At the final stage, the testing stage, the performance of the suggested feature selection method was evaluated by the test data. The main contributions of this study are:

- Creating an effective numeric training dataset that includes 13 numeric features from the existing SQLi datasets.
- Developing two different binary versions of the gray wolf optimization algorithm to optimally select the features of the dataset.
- Creating an effective and efficient classification model to detect SQLi attacks.
- Increasing the efficiency of the SQLi detection methods by selecting 20% of the most effective features.

The rest of the paper structure is as follows; the previously presented SQLi methods were reviewed in Sect. 2. In Sect. 3, the proposed SQLi method was explained. In Sect. 4, the experimental platform and obtained results were discussed. The paper ends with a conclusion and future work suggestions.

2 Related works

Some researchers have come up with methods to detect and prevent SQLi attacks. In this section, the previous works are summarized in sub-categories:

2.1 Blacklist-based SQLi detection method

Traditional SQLi methods use blacklists to filter unauthorized characters. There have been numerous works completed in this field. Input validation techniques to avoid SQLi are divided into whitelist validation and blacklist validation techniques [4]. In [5], authors used penetration testing to compensate for the shortcomings of the blacklist filter defense mechanism. Like the blacklisting method, in [6], the authors used the Boyer-Moore string-matching algorithm to calculate how much a string matches the URL with an injected SQL string to determine the existence of a SQL attack. In [7], an effective automated tool is presented to prevent SQLi attacks at run time. In [8], the capabilities of the latest Blackbox web-based security scanners against SQL and XSS injection attacks were investigated. Hsiu-Chuan Huang introduced a new vulnerability scanner for web applications that uses penetration testing to prevent SQLi and detect it. In [9], a method called WAVES was suggested for testing SQLi vulnerabilities in web applications. It uses a web crawler to find points in a web application that can be used for SQLi attacks. The probe then creates attacks that target such locations based on a specific list of attack patterns, using machine learning techniques to improve the method of attack. This solution further improves penetration testing methods by using machine learning approaches to guide testing. However, like all black box testing and penetration testing methods, it cannot provide a guarantee of completeness. A detection mechanism for SQL injection attacks has been proposed in [10] that removes the value from a SQL query attribute of web pages and compares it to a specified value. In this approach, static and dynamic analysis are merged. The experiments demonstrate that the suggested strategy is straightforward and highly successful.

2.2 Static analysis method

One technique for SQL injection prevention is static analysis of SQL queries in web applications. This method focuses on validating the type of user input to reduce the SQLi probability, rather than detecting them. Carl Gould et al. [11] used the library of Java String Analysis (JSA) to validate the type of user input and prevent SQLi attacks. However, it cannot prevent SQLi attacks if the malicious input data has the correct input type or syntax. Also, the mentioned library only supported the Java language. In [12], the static analysis method was combined with automatic reasoning. This method assumes that there is no tautology in the automatic SQL query generation, and this assumption is also verified. Thus, it was an effective method for detecting SQLi attacks, but except for tautology, it could not detect other SQLi attacks. Stephen Thomas et al. [13] developed a new SQL query by collecting simple SQL statements and executing them to validate the type of user input. The method did not prevent or detect SQLi attacks directly, but it tried to prevent attacks by removing vulnerabilities in the way SQL queries were written. This method was also used for web applications written in the Java language.

2.3 Dynamic analysis method

The dynamic analysis method examines the web application's response after scanning it. Unlike the static analysis method, it can find the vulnerability of SQL injection attacks without any changes to the web application; so, it has an advantage over the static analysis method. However, the weaknesses found in the web application pages must be manually resolved by the developers. Yuji Kasuga et al. [14] introduced a method called Sania that protects against SQLi attacks. The approach collects natural queries between the client and the web application, as well as between the web application and the database. The authors of [15] proposed a web scan method with advanced features for testing and detecting intrusion into PHP-based web applications; this method can identify the web application's weak point against SQLi attacks. One disadvantage of this tool is that it only works on PHP-based web applications and will not be able to detect SQLi attacks in ASP-based web applications.

2.4 A hybrid analysis method

This method is a combination of static and dynamic analysis that provides the merits of both methods. Furthermore, it analyzes web pages and creates SQL queries to evaluate the results. William et al. [16] proposed a method called AMNESIA. This method is based on a combination of static and dynamic analysis. In the static stage, it uses the static analysis method to build models of different types of queries that an application can legally generate anywhere it has access to the database. In the dynamic step, it tracks all queries before sending them to the database and checks each query against statistically constructed models. In [17], Buehrer et al. presented a static and dynamic analysis called SQL Guard, which compared static and dynamic SQL queries generated by the user to detect SQLi attacks using a survey tree. SQLCheck is another attack injection based method. Both SQLGuard and SQLCheck methods use a hidden key to determine the user input limit when analyzed by the runtime controller.

2.5 Query profiling methods

SQLrand, which was developed in [18], is a randomization-based method of setting an instruction set that allows developers to use random instructions instead of regular SQL keywords. As a result, a proxy filter inserts queries into the database while removing keywords from normal.

Since the attacker's SQL-infused code is not made up of random instruction sets, the injectable commands lead to a syntactically incorrect query. A query-profile-based technique that might identify SQLi attacks without altering the web application was proposed in [18]. However, the web application had to be re-profiled whenever it changed.

2.6 Machine learning-based methods

Valeur et al. [19] suggested a method based on an intrusion detection system (IDS) using machine learning algorithms. To identify SQLi attacks, this technique employs numerous anomaly identification models. This method as a module is connected to the link between the database server and web applications; the queries made by the application are intercepted and sent to the attack detection system. The intrusion detection system parses the SQL queries and then compares them with the generated model during the execution of the web application to check the differences.

One of the fundamental limitations of learning-based methods is that they cannot always guarantee success in their detection abilities. Because it depends on the quality and method of training used in the presented model, it can cause many false positives and negative results. Joshi et al. [20] presented a method called the Naive Bayes algorithm, which is a classification model in machine learning based on Bayes' theorem. This method assumes that the presence of a feature in a data model is not related to the presence of other features. This method is very simple to implement but could not identify several SQL injection attacks; especially, when a particular type of SQL injection was used for the first time.

3 Suggested method

An effective strategy for detecting SQLi attacks is proposed in this research. First, a particular training dataset with 13 features was created for this technique. Due to the unique characteristics of the gray wolf optimization (GWO) algorithm, two alternative binary variants of it (bGWO) were constructed in the second stage to choose the most effective features of the dataset. The optimal datasets generated by the bGWOs are employed by various machine learning methods. The Neural Networks (NN) and Decision Tree (DT) techniques were used to generate the appropriate classification model for detecting SQLi. The performance of the suggested feature selection approach was assessed by the test data in the last stage, the testing stage. Figure 1 shows the workflow of the proposed method.

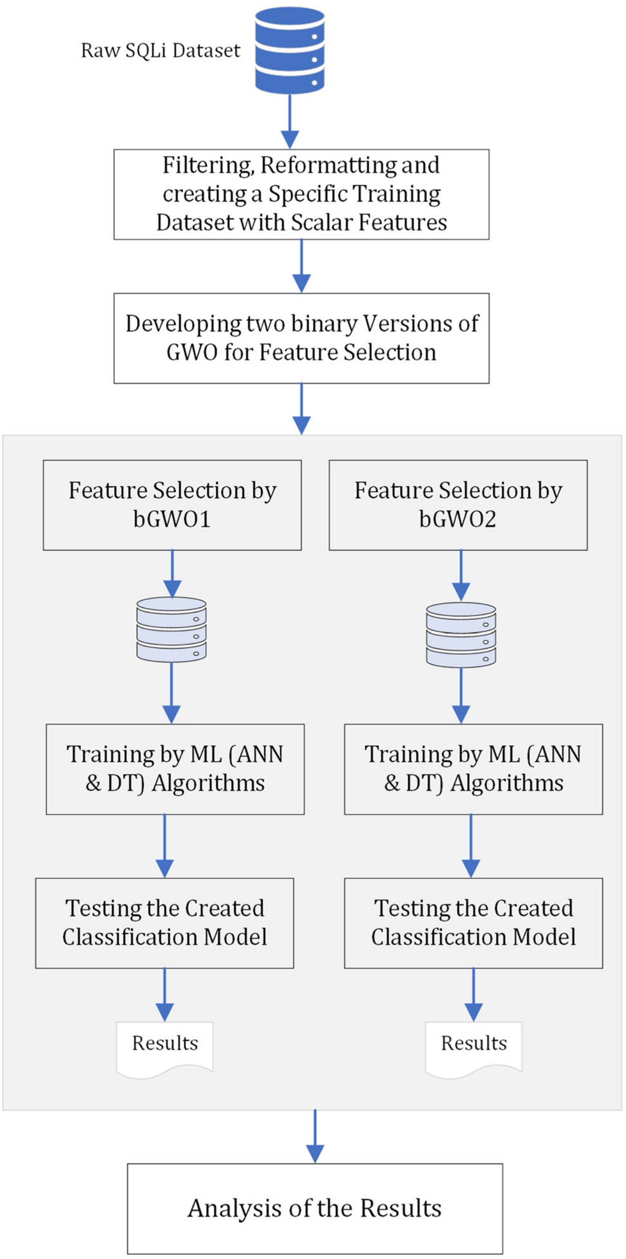

Raw SQLi Dataset
Filtering, Reformatting and creating a Specific Training Dataset with Scalar Features
Developing two binary Versions of GWO for Feature Selection
Feature Selection by bGWO1
Feature Selection by bGWO2
Training by ML (ANN & DT) Algorithms
Training by ML (ANN & DT) Algorithms
Testing the Created Classification Model
Testing the Created Classification Model
Results
Results
Analysis of the Results

Fig. 1 Workflow of the suggested method

3.1 Dataset creation

In this section of the paper, the required dataset to train the supervised machine learning algorithm is explained. As seen in Fig. 2, the SQL query profiling process is used to create SQLi dataset.

The experiment system consists of some web applications with a frontend, HTTP APIs, and MySQL server backend. The SQLi traffics (normal and SQLi queries) have been generated using SQL query generation application. The resulting MySQL traffic between the web app server and the database server is captured. These collected

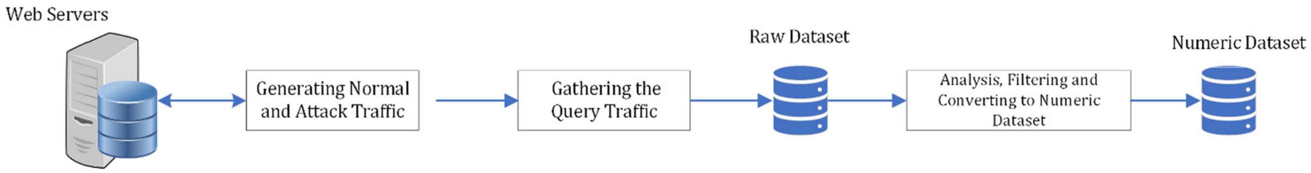

Web Servers
Generating Normal and Attack Traffic
Gathering the Query Traffic
Raw Dataset
Analysis, Filtering and Converting to Numeric Dataset
Numeric Dataset

Fig. 2 The process of SQLi dataset creation

raw data sets were then processed and correlated to create a separate dataset containing features (labeled records). Table 1 shows the structure of the records in the collected raw dataset. This dataset comprises attacks and normal queries for a web application. These queries were implemented by different SQL structures and complexities. There are simple and complex nested queries in the dataset. Each row in the selected dataset is a normal or malicious SQL query performed in a database of a real-world web application. Each record in the dataset consists of SQL keywords, fragmented text, single quotes, semicolons, comments, intermediate data, and so on.

The raw dataset was filtered, cleaned, and converted to a numeric dataset. This study used a dataset that includes 1027 extracted unique SQL queries (normal and malicious) retrieved from a textual dataset. This dataset includes 1027 records in such a way that each record indicates a SQL query. In this dataset, 554 records are malicious queries, and 473 records are normal queries. The labeling of dataset records is expressed in binary values of 0 (normal SQL query) or 1 (malicious SQL query). Each row of the original dataset consists of two text columns; the first column includes the source of the SQL query, and the second column indicates its label.

Creating an effective numeric training dataset that includes 13 numeric features is the first contribution of this study. The homogeneity of features in a dataset makes machine learning algorithms perform better in terms of accuracy and precision. In the first stage of the proposed methods, the standard SQLi datasets that include thousands SQLi queries were analyzed, and 13 numeric effective features were extracted. In this stage of this study, the selected original dataset was analyzed, and the 13 features were extracted from the source code of each query. All these 13 features are numeric. Twelve features are independent, and the final feature is dependent and indicates the label of the query. The query is normal when its label is zero; the malicious queries are labeled by one. All extracted features have numeric values. Length of query, number of nested queries, number of constants, number of punctuations, and number of logical operators are some of the extracted features. Finally, the created dataset includes 1027 records; each record (each row) includes 13 features extracted from the source code of the queries. The homogeneity of features in a dataset makes machine learning algorithms perform better in terms of accuracy and precision. Table 2 shows the structure of the created dataset. Table 3 describes the features of the created training dataset.

3.2 Feature selection by gray wolf optimization algorithm

After creating the scalar (numeric) training dataset, in the second stage, two different binary versions of the GWO algorithm were developed. The developed two bGWOs algorithms were used for feature selection. Indeed, the optimal features of the training dataset were selected before creating the desired classification model by the machine learning algorithms. The gray wolf optimization algorithm is a metaheuristic algorithm inspired by the hierarchical structure and social behavior of gray wolves during hunting [21]. It is an algorithm based on population and is simply capable of generalizing to large-scale problems. All the members have a precise hierarchy, and they have certain tasks. In each group of wolves to hunt, there are four levels, which are modeled as a pyramidal structure: the leader wolves are called the alpha (α) group, which can be male or female. These wolves dominate the flock (group). Beta (β) wolves are assisted by alpha wolves in the decision-making

Table 1 Format of the records in the raw training dataset

select * from customer where cu_id = 1 or ’a’ = ’a’	1
SELECT * FROM customer UNION SELECT * clear FROM result ORDER BY habit	2
select customer1.cu_id, customer1.name, customer1.family, factor.fact_id, factor.fdate, factor.amount from (select * from customer where cu_id = @cust_id) as customer1 inner join factor on customer1.cu_id = factor.cust_id;	1
select name, family, tell, sum1 from customer inner join (select cust_id, sum(amount) as sum1 from payment group by cust_id) as payment1 on customer.cu_id = payment1.cust_id where cu_id = @cu_id;	1

Table 2 The structure of the created numeric datasets with 13 numeric features

Length	Nesting	Unionnum	Constantnum	Orconstant	Spacifical character	Type	Andnum	“Num	Num	Null num	() num	Attack
30	2	0	4	0	15	1	1	0	0	0	4	NO
30	2	0	4	0	15	1	1	0	0	0	4	NO
30	2	0	4	0	15	1	1	0	0	0	4	NO
30	2	0	4	0	15	1	1	0	0	0	4	NO
30	2	0	4	0	15	1	1	0	0	0	4	NO
30	2	0	4	0	15	1	1	0	0	0	4	NO
30	2	0	4	0	15	1	1	0	0	0	4	NO
30	2	0	4	0	15	1	1	0	0	0	4	NO
30	2	0	4	0	15	1	1	0	0	0	4	NO
30	2	0	4	0	15	1	1	0	0	0	4	NO
30	2	0	4	0	15	1	1	0	0	0	4	NO
30	2	0	4	0	15	1	1	0	0	0	4	NO
30	2	0	4	0	15	1	1	0	0	0	4	NO
30	2	0	4	0	15	1	1	0	0	0	4	NO
30	2	0	4	0	15	1	1	0	0	0	4	NO
30	2	0	4	0	15	1	1	0	0	0	4	NO
30	2	0	4	0	15	1	1	0	0	0	4	NO
30	2	0	4	0	15	1	1	0	0	0	4	NO
30	2	0	4	0	15	1	1	0	0	0	4	NO

Table 3 The structure of the created numeric datasets with 13 numeric features

Feature	Description
1	Length of SQL Query
2	Nesting Level Query
3	Num. of union Operator in the SQL Query
4	Num. of Constant Value in the SQL Query
5	Num. of OR Operator in the SQL Query
6	Num. of Specific Character in the SQL Query
7	Type of SQL Query in the SQL Query
8	Num. of AND Operator in the SQL Query
9	Num. of double quotation Character (“) in the SQL Query
10	Num. of double quotation Character (‘) in the SQL Query
11	Num. of Null Value in the SQL Query
12	Num. Parenthesis in the SQL Query
13	Class (Attack or Not Attack)

process and are also susceptible to being selected instead. Delta (δ) wolves are lower than beta wolves and include old predators. Omega (ω) wolves have the lowest rank in the hierarchy pyramid, which is the least right than the rest of the group; they eat and are not involved in the decision-making process. GWO algorithm consists of three main steps: Tracking and Approaching, encircling, and attacking. In the GWO, Eqs. 1 and 2 were used to mathematically model the encircling behavior of the gray wolves.

$$\vec{D} = \left|\vec{C}\cdot\vec{X}_p(t) - \vec{X}(t)\right| \tag{1}$$

$$\overrightarrow{X}(t+1) = \overrightarrow{X}_p(t) - \overrightarrow{A}\cdot\overrightarrow{D} \tag{2}$$

In Eqs. 1 and 2, t represents the current repetition, and $\overrightarrow{A}$ and $\overrightarrow{C}$ the coefficient vectors, and $\overrightarrow{X}$ the position vector of a gray wolf, the $\overrightarrow{A}$ and $\overrightarrow{C}$ vectors are calculated by Eqs. 3 and 4:

$$\overrightarrow{A} = 2\overrightarrow{a}\cdot\overrightarrow{r}_1 - \overrightarrow{a} \tag{3}$$

$$\overrightarrow{C} = 2\cdot\overrightarrow{r}_2 \tag{4}$$

In Eqs. 3 and 4, $\overrightarrow{a}$ vector components are linearly reduced from 2 to 0 during repetition and $\overrightarrow{r}_1$ and $\overrightarrow{r}_2$ vectors are random vectors in the interval $[0, 1]$. To see the impact of the relationships mentioned above, a two-dimensional space with possible locations is shown in Fig. 3. A gray wolf with a location (X, Y) can update its location according to the position of the prey (X^*, Y^*). The locations of the gray wolves were updated based on the (X*, Y*). The Algorithm iteratively converges to the best solution (location of prey). The best solution indicates the minimum number of attributes of the dataset that have

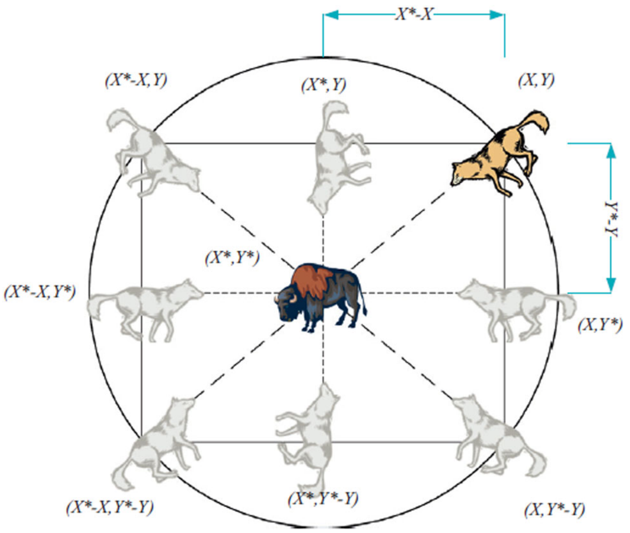


Fig. 3 Two-dimensional view of position vectors and their next possible locations [17]

maximum effect on the SQLi detection. Consider the vector $\vec{A}$ and $\vec{C}$ values and the current location, different positions are around the best response. For instance, it is possible to reach a place $(X^{*} - X, Y^{*})$ according to.$\vec{A} = (1, 0)$ and $\vec{C} = (1, 1)$. The two random vectors $\vec{r}_1$ and $\vec{r}_2$ allow wolves to access all positions. Abstractly, it can be noted that a gray wolf can randomly update its position according to the relationships listed around the prey.

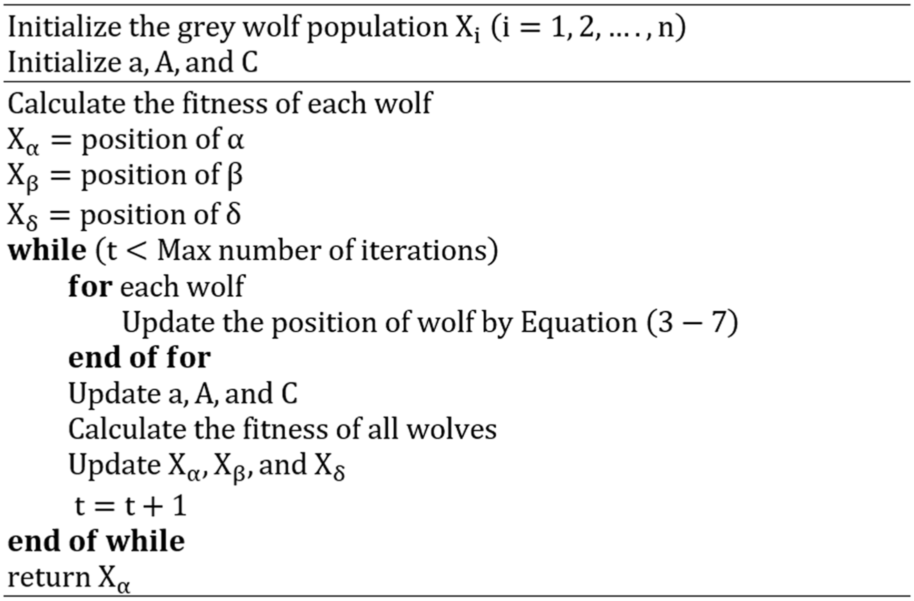
Gray wolves can detect the location of the prey and surround them. The alpha wolves are often the ones that hunt; however, beta and delta wolves occasionally join them. Alpha, beta, and delta wolves are more aware of the probable location of prey; their locations are updated by Eqs. 5, 6, and 7. This allows us to mathematically replicate the hunting behavior of gray wolves. Figure 4 illustrates how a search agent updates its position according to Alpha, Beta, and Delta in the two-dimensional search space. The Pseudo-code of the GWO algorithm is presented in Algorithm 1.

$$\vec{D}_{\alpha} = \left|\vec{C}_1 \cdot \vec{X}_{\alpha} - \vec{X}\right|, \vec{D}_{\beta} = \left|\vec{C}_2 \cdot \vec{X}_{\beta} - \vec{X}\right|, \vec{D}_{\delta} = \left|\vec{C}_3 \cdot \vec{X}_{\delta} - \vec{X}\right| \tag{5}$$

$$\vec{X}_1 = \vec{X}_{\alpha} - \vec{A}_1 \cdot \left(\vec{D}_{\alpha}\right), \vec{X}_2 = \vec{X}_{\beta} - \vec{A}_2 \cdot \left(\vec{D}_{\beta}\right), \vec{X}_3 = \vec{X}_{\delta} - \vec{A}_3 \cdot \left(\vec{D}_{\delta}\right) \tag{6}$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \tag{7}$$

Algorithm 1 Gray Wolf Optimization Pseudo-code



```
Initialize the grey wolf population $X_i$ $(i = 1, 2, ...., n)$
Initialize a, A, and C
Calculate the fitness of each wolf
$X_{\alpha}$ = position of $\alpha$
$X_{\beta}$ = position of $\beta$
$X_{\delta}$ = position of $\delta$
while (t < Max number of iterations)
    for each wolf
        Update the position of wolf by Equation (3 − 7)
    end of for
    Update a, A, and C
    Calculate the fitness of all wolves
    Update $X_{\alpha}$, $X_{\beta}$, and $X_{\delta}$
    t = t + 1
end of while
return $X_{\alpha}$
```

3.3 Binary GWO

The Continuous Gray Wolves Optimization (OGWO) algorithm constantly shifts their positions to anywhere in the state space. In some special issues, such as selecting features, solutions are limited to binary values of zero and one which creates a binary version [18]. In the method, the wolf Equation is to update a function of three positional

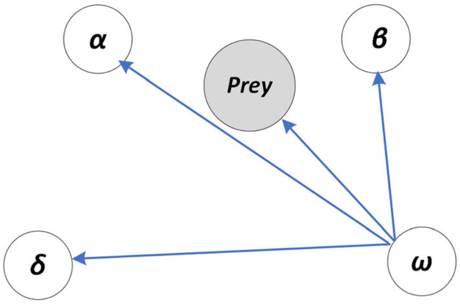


Fig. 4 Updating positions in GWO

vectors, namely, $\overrightarrow{X}_{\propto}, \overrightarrow{X}_{\beta}\, and\, \overrightarrow{X}_{\delta}$. It attracts each wolf to the first of the top three solutions. The set of solutions at any given time is binary. To update the positions of a given wolf, following the original gray wolf algorithm, while keeping binary constraints based on Eq. 7, one of the following two approaches can be used:

First approach: In this approach, Eq. 7 can be formulated as Eq. 8. In Eq. 8, Crossover$(\overrightarrow{x}_1, \overrightarrow{x}_2, \overrightarrow{x}_3)$ is a suitable shift between x, y, and z solutions. The binary vectors $\overrightarrow{x}_1, \overrightarrow{x}_2 and \overrightarrow{x}_3$ effectively move wolves toward alpha, beta, and delta wolves. The vectors $\overrightarrow{x}_1, \overrightarrow{x}_2 and \overrightarrow{x}_3$ are calculated using Eq. 9.

$$\overrightarrow{X}_i^{t+1} = \text{Crossover}(\overrightarrow{x}_1, \overrightarrow{x}_2, \overrightarrow{x}_3) \tag{8}$$

$$\overrightarrow{X}_1^d = \begin{cases} 1\ if\left(X_\alpha^d + \text{bstep}_\alpha^d\right) \geq 1 \\ \qquad 0otherwise \end{cases} \tag{9}$$

So, the vector $\overrightarrow{X}_1^d$ is the position of the alpha wolf and $bstep_\alpha^d$ a binary step in the d dimension. $bstep_\alpha^d$ is calculated based on Eq. 10. Here, *rand* is a random number based on a uniform distribution between zero and one. $cstep_\alpha^d$ is the step size being a continuous value for d dimension and is calculated using the Sigmoid function (Eq. 11).

$$bstep_\alpha^d = \begin{cases} 1 & if\left(X_\alpha^d + \text{cstep}_\alpha^d\right) \geq rand \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

$$cstep_\alpha^d = \frac{1}{1 + e^{-10(A_1^d D_\alpha^d - 0.5)}} \tag{11}$$

A_1^d and D_α^d are calculated using Eqs. 1 and 3. To calculate X_2^d and X_3^d which are the positions of beta and delta wolves, respectively, Eq. 12 is used. So, the $\overrightarrow{X}_2^d$ vector positions are beta wolves and $bstep_\beta^d$ a binary step in the d dimension. $bstep_\beta^d$ can be calculated based on Eq. 13.

$$\overrightarrow{X}_2^d \begin{cases} 1 & if\left(X_\beta^d + \text{bstep}_\beta^d\right) \geq 1 \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

$$bstep_\beta^d = \begin{cases} 1 & if\left(X_\beta^d + cstep_\beta^d\right) \geq rand \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

Again, here, *rand* is a random number based on a uniform distribution between zero and one. $cstep_\beta^d$ is the step size being the continuous value for the d dimension. It is calculated using the Sigmoid function indicated by Eq. 14. Here, also, A_2^d and D_β^d respectively using Eqs. 1 and 3. So that the $\overrightarrow{X}_3^d$ is vector positions of delta wolf (calculated by Eq. 15) and $bstep_\delta^d$ are a binary step in the d dimension. $bstep_{\beta\delta}^d$ is calculated based on Eq. 16.

$$cstep_\beta^d = \frac{1}{1 + e^{-10(A_2^d D_\beta^d - 0.5)}} \tag{14}$$

$$\overrightarrow{X}_3^d = \begin{cases} 1 & if\left(X_\delta^d + \text{bstep}_\delta^d\right) \geq 1 \\ 0 & \text{otherwise} \end{cases}. \tag{15}$$

$$bstep_\delta^d = \begin{cases} 1\ if\left(X_\delta^d + \text{cstep}_\delta^d\right) \geq rand \\ \qquad 0otherwise \end{cases} \tag{16}$$

In Eq. 16, a random number *rand* is based on a uniform distribution between zero and one and $cstep_\delta^d$ the step size is a continuous value for the d dimension and is calculated using the Sigmoid function (Eq. 17). In this regard, also, $A_3^d and D_\delta^d$ are calculated using Eqs. 1 and 3. X_d is calculated by Eq. 18. In this relationship, a_d, b_d and c_d are binary values for the first, second, and third parameters in dimension d. The X_d is the output of Crossover in dimension d and *rand* is a random number from a uniform distribution in the range of zero and one. The pseudocode of the bGWO1 is explained in Algorithm 2.

$$cstep_\delta^d = \frac{1}{1 + e^{-10(A_3^d D_\delta^d - 0.5)}} \tag{17}$$

$$X_d = \begin{cases} a_d & \text{if rand } < \frac{1}{3} \\ b_d & \text{if} \frac{1}{3} \leq rand \leq \frac{2}{3} \\ c_d & otherwise \end{cases} \tag{18}$$

Algorithm 2 The binary gray wolf optimization algorithm–the first approach

```
input: n is the number of grey wolves in the pack
output: $X_\alpha$ optimal grey wolf binary position
  1. Initialize a population of n wolves positions at random $\in [0, 1]$
  2. Find the $\alpha, \beta, \delta$ solutions based on fitness
  3. while stopping criteria do
     foreach wolf$_i$ $\in$ pack do
         Calculate $x_1, x_2, x_3$ using Equations (8 − 12)
         $X_i^{t+1} \leftarrow$ crossover among $x_1, x_2, x_3$
     end of for
     Update A, and C
     Evaluate the positions of individual wolves
     Update $\alpha, \beta, \delta$
  end of while
```

Second Approach: To develop the second version of the bGWO, only the updated gray wolf position vector is forced to be binary. Equation 9 in the first approach is replaced by Eq. 19. So a random number *rand* is between zero and one. X_d^{t+1} is the updated binary position in the dimension d and in repetition t, as well as the sigmoid function calculated by Eq. 20. The pseudocode of the bGWO2 is explained in Algorithm 3.

$$X_d^{t+1} = \begin{cases} 1 & \text{if sigmoid}\left(\left(\dfrac{x_1 + x_2 + x_3}{3}\right) \geq \text{rand}\right. \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

$$\text{sigmoid}(a) = \frac{1}{1 + e^{-10(x-0.5)}} \tag{20}$$

Algorithm 3 The binary gray wolf optimization algorithm - the second approach

```
input: n is the number of grey wolves
output: $X_\alpha$ Optimal grey wolf binary position
  1. Initialize the positions of n wolves randomly $\in [0, 1]$
  2. Find the $\alpha, \beta, \delta$ wolves based on fitness
  3. while stopping criteria do
     foreach Wolf$_i$ $\in$ pack do
        Update Wolf$_i$ position to a binary position according to Equations (21)
     end of for
     Update a, A, and C
     Evaluate the positions of individual wolves.
     Update $\alpha, \beta, \delta$.
  end of while
```

3.4 Adaption of binary gray wolf optimizer

In this section, both the first and second approaches of binary GWO algorithms are adapted in feature selection for classification problems. According to the multiplication or permutation principle, for a feature vector of size n, there are $n2^n$ different choices of features, which creates a very large and tedious space, and all these 2^n choices must be thoroughly explored. Therefore, the binary gray wolf algorithm is adapted to search the matching feature space to find the best combination of features. It is remembered

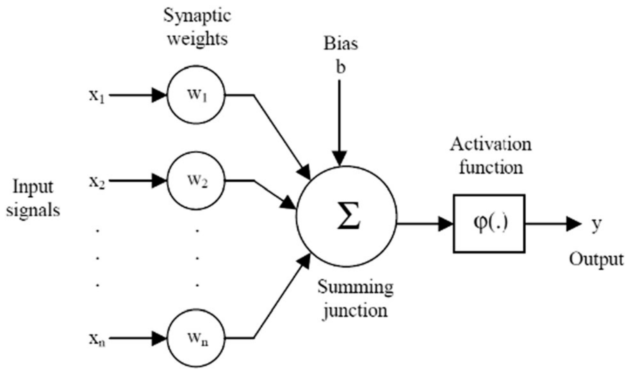


Fig. 5 Mathematical model of artificial neurons

that the best combination of features is the combination with maximum classification efficiency and minimal selection of the number of features. The Fitness Function, which is used to evaluate the individual positions of gray wolves in the binary gray wolf algorithm, is obtained by Eq. 21.

$$\text{Fitness} = \alpha\gamma_R(D) + \beta\frac{|C-R|}{|C|} \tag{21}$$

In Eq. 21, $\gamma_R(D)$ is the classification quality of the set of features that is determined relative to the decision$D$. Also, C is the total number of features, and R is the number of selected features, and the ratio $\frac{|C-R|}{C}$ is the ratio of unselected features to the total number of features. On the other hand, α and β are two parameters corresponding to the importance of classification quality and the number of selected features so that$\alpha \in [0,1], \beta = 1-\alpha$. The fitness function maximizes the classification quality$\gamma_R(D)$. The ratio of unselected features to the total number of features is indicated by$\frac{|C-R|}{C}$. The fitness function is transformed into a minimization function by substituting the error rate for the classification quality and using the number of selected features instead of the number of unselected features (Eq. 22).

$$\text{Fitness} = \alpha E_R(D) + \beta\frac{|R|}{|C|} \tag{22}$$

In Eq. 22, $E_R(D)$ is the error rate for classifying the feature set, $|R|$ is the number of selected features, and $|C|$ is the total number of features. Also, α and β are constants to control the classification accuracy and reduce the number of selected features, so that;$\alpha \in [0,1], \beta = 1-\alpha$.

3.5 Creating the classifier

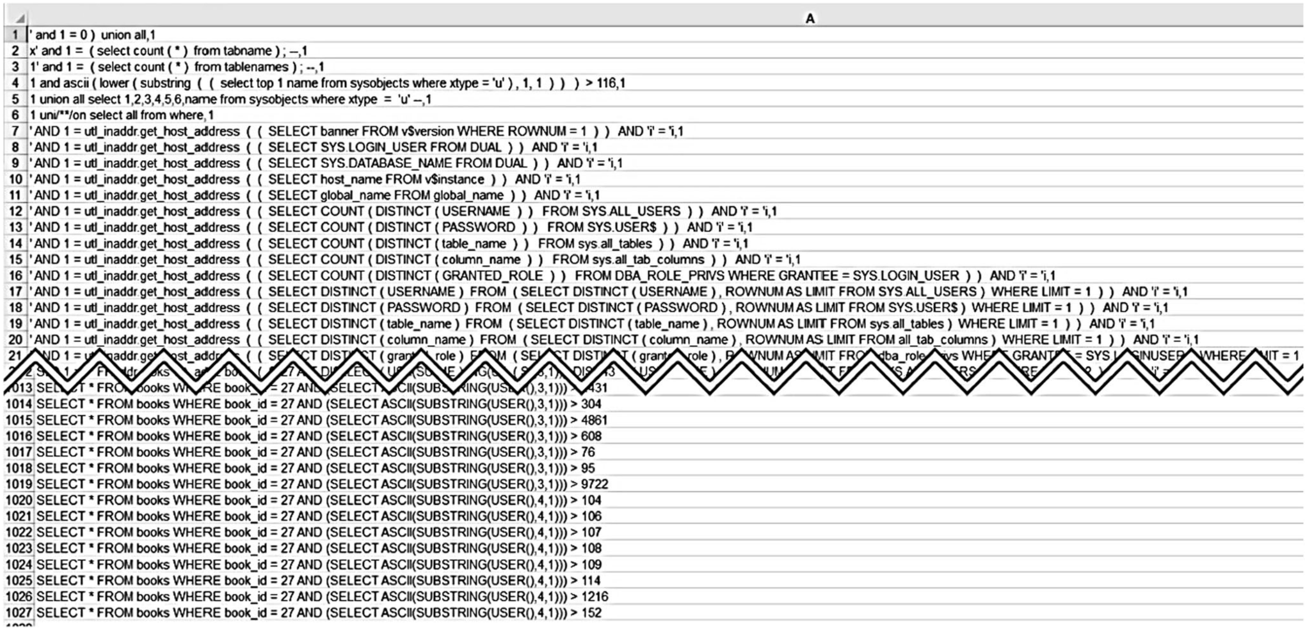
After creating the optimal training dataset that includes optimal features, artificial neural network (ANN) and decision tree (DT) algorithms were used to train the classification model. ANN structures were created from the biological neural systems and the human brain itself. Information processing units, which are called artificial neurons, are the basis of the operation of an ANN. they are the simplified models of brain cells and biological neurons. An artificial neuron can have multiple inputs, but only one output. Artificial neurons used in neural networks are nonlinear and usually provide continuous outputs. Figure 5 shows the mathematical model of a neuron, which is the basis for the design of artificial neural networks.

In Fig. 5, multiple input signals from the external environment are represented by the set $\{x_1, x_2, x_3, \ldots, x_n\}$. In this study, the input signals are the values of the features in the training dataset. Also, the weighting performed by the synaptic connections of the network is implemented on the artificial neuron as a set of synaptic weights $\{w_1, w_2, w_3, \ldots, w_n\}$. In the next step, the relevance of each input of the neuron $\{x_i\}$ is calculated by multiplying them by the corresponding synaptic weight $\{w_i\}$. Input signals $(x_1, x_2, x_3, \ldots, x_n)$ are signals (dataset features) that come from the external environment (train dataset). Input signals were optimized by the bGWO to increase the computational efficiency of learning algorithms. Synaptic weights $(w_1, w_2, w_3, \ldots, w_n)$ are used to determine the weight of each feature in the SQLi dataset. Specifically, each feature in the dataset $\{x_i\}$ at the input of synapse j connected to the neuron is multiplied by the synaptic weight $\{w_i\}$. Linear adder $(\sum)$ sums all the weighted input features to produce an activation voltage.

Activation Threshold or Bias (b) is a variable used to determine the appropriate threshold. The result produced by the linear adder has a stimulus value toward the output of the neuron. The activation function (φ) is to limit the output of the neuron to a suitable range of values. Typically, the normalized amplitude of the output of a neuron is written with the closed interval [0,1] or with the closed interval [− 1,1]. The output signal (y) contains the final value produced by the neuron by a particular set of input features that can be used as input to other connected neurons. Equation 23 was used to calculate the amount of output signal. In this regard, the activation function is such

Table 4 Hardware and software specifications of the implementation environment

HW/SW	Specification
Central processing unit (CPU)	Intel Core i7
Frequency	3.4 GHz
RAM	8 GB
Operating system (OS)	Microsoft co. Windows 10
Software platform	MATLAB 2020b



	A
1	' and 1 = 0) union all,1
2	x' and 1 = (select count (*) from tabname) ; –,1
3	1' and 1 = (select count (*) from tablenames) ; --,1
4	1 and ascii (lower (substring ((select top 1 name from sysobjects where xtype = 'u') , 1, 1))) > 116,1
5	1 union all select 1,2,3,4,5,6,name from sysobjects where xtype = 'u' –,1
6	1 uni/**/on select all from where,1
7	'AND 1 = utl_inaddr.get_host_address ((SELECT banner FROM v$version WHERE ROWNUM = 1)) AND 'i' = 'i,1
8	'AND 1 = utl_inaddr.get_host_address ((SELECT SYS.LOGIN_USER FROM DUAL)) AND 'i' = 'i,1
9	'AND 1 = utl_inaddr.get_host_address ((SELECT SYS.DATABASE_NAME FROM DUAL)) AND 'i' = 'i,1
10	'AND 1 = utl_inaddr.get_host_address ((SELECT host_name FROM v$instance)) AND 'i' = 'i,1
11	'AND 1 = utl_inaddr.get_host_address ((SELECT global_name FROM global_name)) AND 'i' = 'i,1
12	'AND 1 = utl_inaddr.get_host_address ((SELECT COUNT (DISTINCT (USERNAME)) FROM SYS.ALL_USERS)) AND 'i' = 'i,1
13	'AND 1 = utl_inaddr.get_host_address ((SELECT COUNT (DISTINCT (PASSWORD)) FROM SYS.USER$)) AND 'i' = 'i,1
14	'AND 1 = utl_inaddr.get_host_address ((SELECT COUNT (DISTINCT (table_name)) FROM sys.all_tables)) AND 'i' = 'i,1
15	'AND 1 = utl_inaddr.get_host_address ((SELECT COUNT (DISTINCT (column_name)) FROM sys.all_tab_columns)) AND 'i' = 'i,1
16	'AND 1 = utl_inaddr.get_host_address ((SELECT COUNT (DISTINCT (GRANTED_ROLE)) FROM DBA_ROLE_PRIVS WHERE GRANTEE = SYS.LOGIN_USER)) AND 'i' = 'i,1
17	'AND 1 = utl_inaddr.get_host_address ((SELECT DISTINCT (USERNAME) FROM (SELECT DISTINCT (USERNAME) , ROWNUM AS LIMIT FROM SYS.ALL_USERS) WHERE LIMIT = 1)) AND 'i' = 'i,1
18	'AND 1 = utl_inaddr.get_host_address ((SELECT DISTINCT (PASSWORD) FROM (SELECT DISTINCT (PASSWORD) , ROWNUM AS LIMIT FROM SYS.USER$) WHERE LIMIT = 1)) AND 'i' = 'i,1
19	'AND 1 = utl_inaddr.get_host_address ((SELECT DISTINCT (table_name) FROM (SELECT DISTINCT (table_name) , ROWNUM AS LIMIT FROM sys.all_tables) WHERE LIMIT = 1)) AND 'i' = 'i,1
20	'AND 1 = utl_inaddr.get_host_address ((SELECT DISTINCT (column_name) FROM (SELECT DISTINCT (column_name) , ROWNUM AS LIMIT FROM all_tab_columns) WHERE LIMIT = 1)) AND 'i' = 'i,1
21	'AND 1 = utl_inaddr.get_host_address ((SELECT DISTINCT (granted_role) FROM (SELECT DISTINCT (granted_role) , ROWNUM AS LIMIT FROM dba_role_privs WHERE GRANTEE = SYS.LOGINUSER) WHERE LIMIT = 1
22	[illegible]
1013	SELECT * FROM books WHERE book_id = 27 AND (SELECT ASCII(SUBSTRING(USER(),3,1))) > 431
1014	SELECT * FROM books WHERE book_id = 27 AND (SELECT ASCII(SUBSTRING(USER(),3,1))) > 304
1015	SELECT * FROM books WHERE book_id = 27 AND (SELECT ASCII(SUBSTRING(USER(),3,1))) > 4861
1016	SELECT * FROM books WHERE book_id = 27 AND (SELECT ASCII(SUBSTRING(USER(),3,1))) > 608
1017	SELECT * FROM books WHERE book_id = 27 AND (SELECT ASCII(SUBSTRING(USER(),3,1))) > 76
1018	SELECT * FROM books WHERE book_id = 27 AND (SELECT ASCII(SUBSTRING(USER(),3,1))) > 95
1019	SELECT * FROM books WHERE book_id = 27 AND (SELECT ASCII(SUBSTRING(USER(),3,1))) > 9722
1020	SELECT * FROM books WHERE book_id = 27 AND (SELECT ASCII(SUBSTRING(USER(),4,1))) > 104
1021	SELECT * FROM books WHERE book_id = 27 AND (SELECT ASCII(SUBSTRING(USER(),4,1))) > 106
1022	SELECT * FROM books WHERE book_id = 27 AND (SELECT ASCII(SUBSTRING(USER(),4,1))) > 107
1023	SELECT * FROM books WHERE book_id = 27 AND (SELECT ASCII(SUBSTRING(USER(),4,1))) > 108
1024	SELECT * FROM books WHERE book_id = 27 AND (SELECT ASCII(SUBSTRING(USER(),4,1))) > 109
1025	SELECT * FROM books WHERE book_id = 27 AND (SELECT ASCII(SUBSTRING(USER(),4,1))) > 114
1026	SELECT * FROM books WHERE book_id = 27 AND (SELECT ASCII(SUBSTRING(USER(),4,1))) > 1216
1027	SELECT * FROM books WHERE book_id = 27 AND (SELECT ASCII(SUBSTRING(USER(),4,1))) > 152

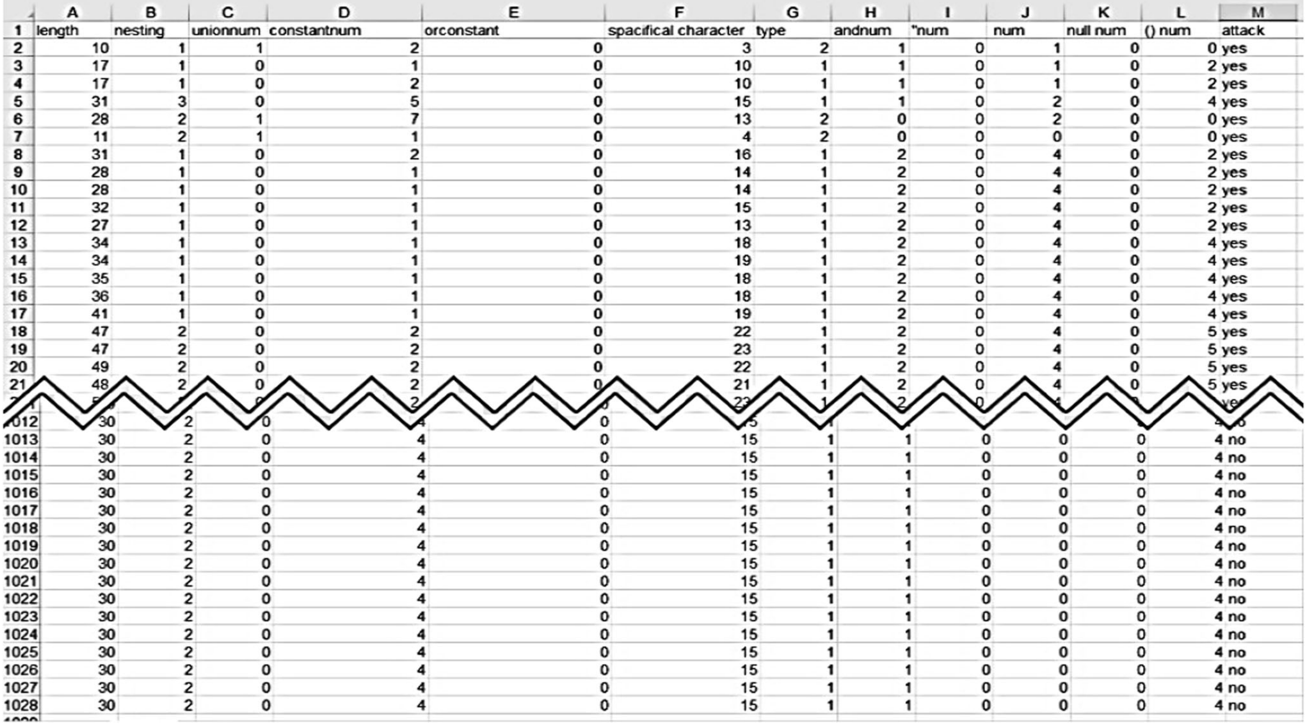
Fig. 6 A clipping of a dataset including normal and malicious queries

that the function is φ. $\varphi : \mathbb{R} \to \mathbb{R}$ in the space of real numbers and explains the neuronal output. The activation function in an artificial neuron acts so that the output of the neuron (y) in a neural network is between special values (usually between 0 and 1, or between − 1 and 1). The activation function can be bounded by Eq. 24.

$$y = \varphi(\sum\nolimits_{j=1}^{n} w_j x_j + b) \tag{23}$$

$$\lim_{t \to +\infty} \varphi(t) = m\, and \lim_{t \to -\infty} \varphi(t) = n (m \neq n) \tag{24}$$

A decision tree is a hybrid data structure of the graph and tree structures; Where the internal node represents the



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	length	nesting	unionnum	constantnum	orconstant	spacifical character	type	andnum	"num	num	null num	() num	attack
2	10	1	1	2	0	3	2	1	0	1	0	0	yes
3	17	1	0	1	0	10	1	1	0	1	0	2	yes
4	17	1	0	2	0	10	1	1	0	1	0	2	yes
5	31	3	0	5	0	15	1	1	0	2	0	4	yes
6	28	2	1	7	0	13	2	0	0	2	0	0	yes
7	11	2	1	1	0	4	2	0	0	0	0	0	yes
8	31	1	0	2	0	16	1	2	0	4	0	2	yes
9	28	1	0	1	0	14	1	2	0	4	0	2	yes
10	28	1	0	1	0	14	1	2	0	4	0	2	yes
11	32	1	0	1	0	15	1	2	0	4	0	2	yes
12	27	1	0	1	0	13	1	2	0	4	0	2	yes
13	34	1	0	1	0	18	1	2	0	4	0	4	yes
14	34	1	0	1	0	19	1	2	0	4	0	4	yes
15	35	1	0	1	0	18	1	2	0	4	0	4	yes
16	36	1	0	1	0	18	1	2	0	4	0	4	yes
17	41	1	0	1	0	19	1	2	0	4	0	4	yes
18	47	2	0	2	0	22	1	2	0	4	0	5	yes
19	47	2	0	2	0	23	1	2	0	4	0	5	yes
20	49	2	0	2	0	22	1	2	0	4	0	5	yes
21	48	2	0	2	0	21	1	2	0	4	0	5	yes
22	[illegible]	[illegible]	[illegible]	2	[illegible]	23	1	2	0	4	[illegible]	[illegible]	yes
1012	30	2	0	4	0	15	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	4	no
1013	30	2	0	4	0	15	1	1	0	0	0	4	no
1014	30	2	0	4	0	15	1	1	0	0	0	4	no
1015	30	2	0	4	0	15	1	1	0	0	0	4	no
1016	30	2	0	4	0	15	1	1	0	0	0	4	no
1017	30	2	0	4	0	15	1	1	0	0	0	4	no
1018	30	2	0	4	0	15	1	1	0	0	0	4	no
1019	30	2	0	4	0	15	1	1	0	0	0	4	no
1020	30	2	0	4	0	15	1	1	0	0	0	4	no
1021	30	2	0	4	0	15	1	1	0	0	0	4	no
1022	30	2	0	4	0	15	1	1	0	0	0	4	no
1023	30	2	0	4	0	15	1	1	0	0	0	4	no
1024	30	2	0	4	0	15	1	1	0	0	0	4	no
1025	30	2	0	4	0	15	1	1	0	0	0	4	no
1026	30	2	0	4	0	15	1	1	0	0	0	4	no
1027	30	2	0	4	0	15	1	1	0	0	0	4	no
1028	30	2	0	4	0	15	1	1	0	0	0	4	no

Fig. 7 Part of the converted dataset to the numeric type

preparation of the feature, and each branch represents the test result and each leaf node represents the class label. Also, the paths from the head to the leaves show the classification rules. Typically for decision analysis, decision trees are regularly used in data mining to help identify a strategy that is most likely to achieve a goal.

4 Experiments platform and results

4.1 Experiment platform

To evaluate the proposed method, firstly, the required scalar (numeric) training dataset was created. Then, the proposed feature selection bGWO algorithms were implemented in MATLAB. The ANN and DT machine learning algorithms were implemented in two forms; In the first form, the machine learning algorithms were implemented without using selective features. In the second form, the machine learning algorithms were implemented using feature selection algorithms. Indeed, in the experiments, two different classification models were created. In the first form of implementation, the ANN and DT train the classification model using the dataset with all features. In the second implementation, the machine learning algorithms invoke the bGWO for selecting the optimal features; then, the machine learning algorithms train the classification model using the dataset with optimal features. The hardware and software specifications of the implementation environment are listed in Table 4.

4.2 Datasets

In this study, the SQLi training dataset has been used to train and test the ANN and DT algorithms.

In this study, 70% of the dataset was used for training the classification model and 30% of the dataset was used for testing the created classification model. In the test stage, a subset (30%) of the training dataset was also used. The selected training and test datasets have uniform distribution and consist of normal and malicious queries. Figure 6 shows the raw dataset that includes the source code of the SQL queries (normal and malicious).

These features are defined based on the important features in SQL injection and a weight is assigned to each feature based on their importance. In this research, nominal features such as the length (number of words) of the query, the number of UNION commands, the number of special characters, the number of AND operators, the number of parentheses, and other features have been used. To improve the performance of the classification models, it is more desirable to use numerical features in the training dataset. Hence, the extracted nominal features from the raw dataset were converted into numerical features. Figure 7 shows the numeric form of the training dataset. As shown in Fig. 7, features are selected for each query, and each row of the dataset indicates the numerical features of a query. Also, the last column of this table indicates that each row (query) is a normal or malicious query. In this research, the converted dataset has been used to train and test the machine learning algorithms. Table 5 shows the specifications of the numeric training dataset.

To implement the proposed method, a program was implemented in the MATLAB programming environment version 2020b. Accuracy, precision, and sensitivity are the classification criteria that were used to evaluate the proposed method; these criteria were applied as a fitness function of the binary gray wolf algorithm. The parameters of the developed binary gray wolf algorithm have been calibrated experimentally during the experiments. Table 6 shows the best values of the bGWO parameters.

Table 6 The value of the bGWO parameters

Parameter name	Value
No. of agents (wolf)	12
Dimension	No. of elected features
Iteration	100
α	0.9
β	0.1

4.3 Evaluation criteria and research questions

The proposed algorithm is implemented in two scenarios named bGWO1 and bGWO2. The scenarios are performed on a dataset to determine the efficiency criteria for the neural network and the decision tree algorithms. The

Table 5 Numeric dataset specifications

Dataset type	Total number of samples	Number of normal samples	Number of malicious samples
Healthy and malicious query dataset	1027	473	554

Table 7 Results of ANN without feature selection

ANN accuracy	ANN error rate	ANN sensitivity	ANN precision
0.9708	0.0292	0.9856	0.9514
0.9773	0.0227	0.9867	0.9673
0.9805	0.0195	0.9893	0.9660

Table 8 Results of DT without feature selection

DT accuracy	DT error rate	DT sensitivity	DT precision
0.8939	0.1061	0.8213	0.9440
0.9123	0.0877	0.9346	0.8830
0.9329	0.0671	0.9294	0.9251

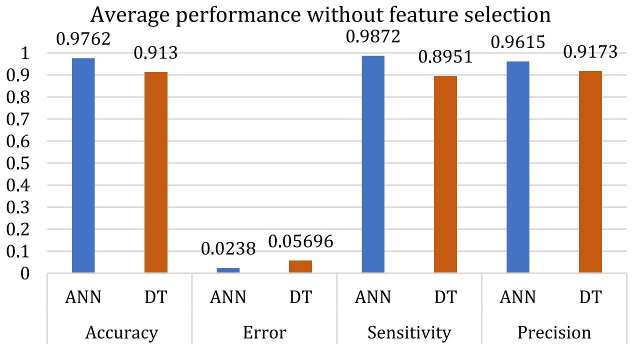


Fig. 8 The average performance of different learning algorithms without feature selection in SQLi detection problem

performance criteria that were used in this study are as follows:

- Accuracy
- Precision
- Sensitivity
- Error Rate
- Number of selected features

These criteria were applied as a fitness function of the binary gray wolf algorithm. Extensive experiments were conducted on the created data set that answers the research questions as follows:

- RQ1: How effective is the use of the proposed first feature selector (bGWO1) on the accuracy precision and error rate of the SQLi detectors?
- RQ2: How effective is the use of the proposed second feature selector (bGWO2) on the accuracy precision and error rate of the SQLi detectors?
- RQ3: What is the effect of using the proposed feature selector in reducing the number of features of the training dataset?
- RQ4: What is the stability of the proposed feature selectors as a stochastic-based method?
- RQ5: What is the success rate of the proposed feature selectors in finding the most effective features in the SQLi dataset?

In this study, two series of ML experiments have been performed; in the first experiment, the ANN and DT were applied to the whole dataset (13 features). In the second experiment, the ANN and DT were applied in the filtered dataset (with the selective features by the proposed bGWO). The results of the experiments have been analyzed to evaluate the effectiveness of the proposed feature selector in the SQLi detectors' performance.

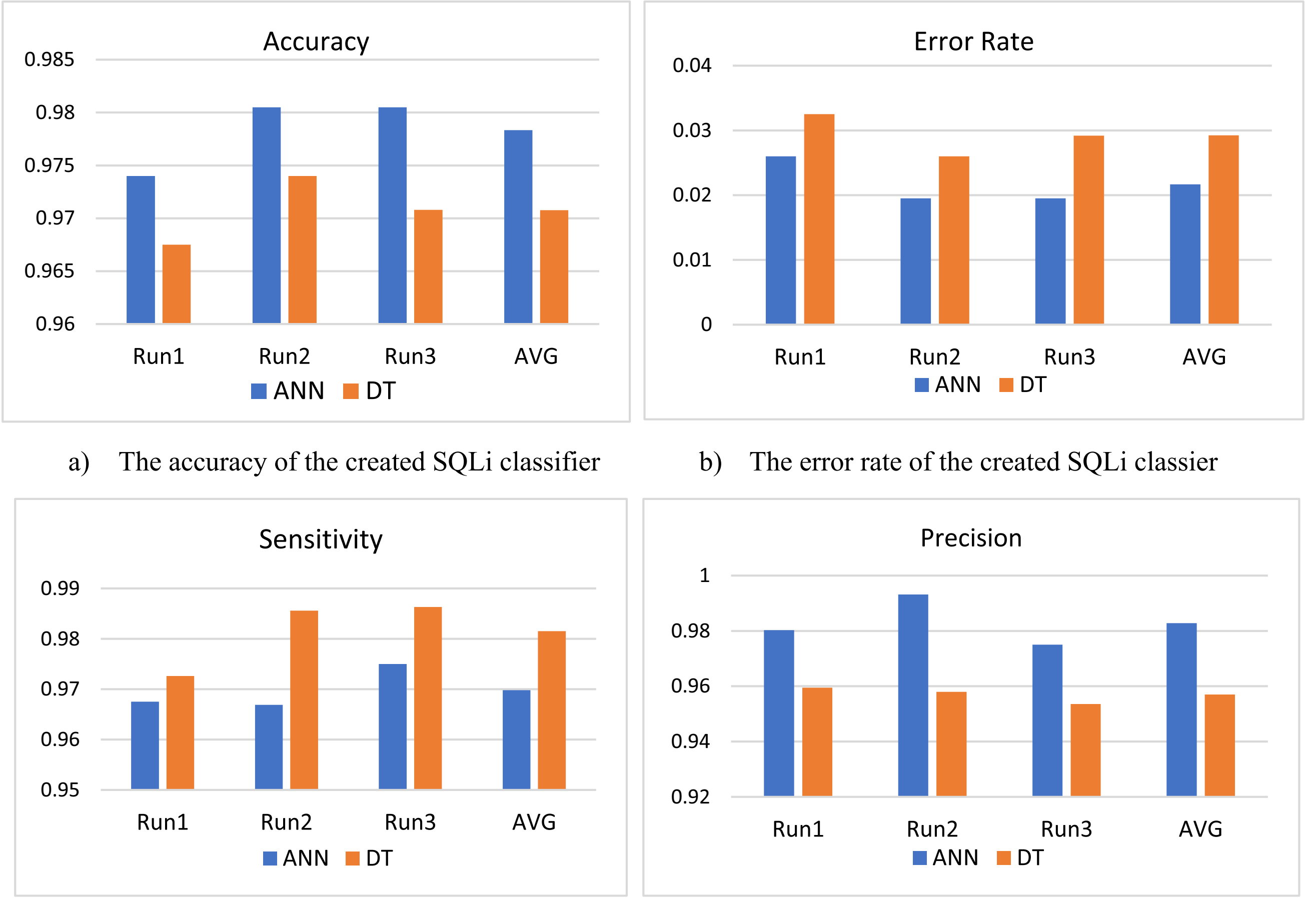
4.4 Results and discussion

4.4.1 SQLi detectors without feature selector

In the first series of experiments, the ANN and DT have been used to create SQLi classifier using the generated numeric train dataset. In these experiments, all features of the dataset have been used in the training stage. The results of ANN and DT efficiency without the proposed features selector in 3 times executions have been shown in Tables 7 and 8. The average performance of the SQLi classifiers created by ANN and DT without feature selectors is shown in Fig. 8. The created SQLi classifier by ANN has higher performance than the classifier created by the DT algorithms in terms of accuracy, error rate, sensitivity, and precision.

4.4.2 The effects of bGWO1 on the SQLi detectors (RQ1)

This subsection of the paper is related to the RQ1. An extensive series of experiments have been conducted to respond to RQ1. In this experiment, the proposed bGWO1 algorithm was used to select the optimal features in the created numeric dataset. The filtered training dataset by the bGWO1 was used to train the ANN and DT; then the created SQLi classification models were tested by the test data. Test data were selected from the created numeric train dataset. Figure 9 shows the effects of the bGWO1 algorithm on the performance of the created SQLi classifier by ANN and DT. The accuracy of the created SQLi classifier using the ANN using bGWO1 algorithm, as the feature selector, is about 0.9783, whereas this figure is about 0.9707. Furthermore, the error rate of the created classifier by ANN and DT with the bGWO1 are respectively 0.0216 and 0.0292. In contrast, the sensitivity of the DT and



a) The accuracy of the created SQLi classifier

b) The error rate of the created SQLi classier

c) The sensitivity of the created SQLi classier

d) The precision of the created SQLi classifier

Fig. 9 The performance of the created SQLi classifier created by ANN and DT using the bGWO1 feature selection algorithm

Table 9 The selected features by bGWO1 in best fitness value

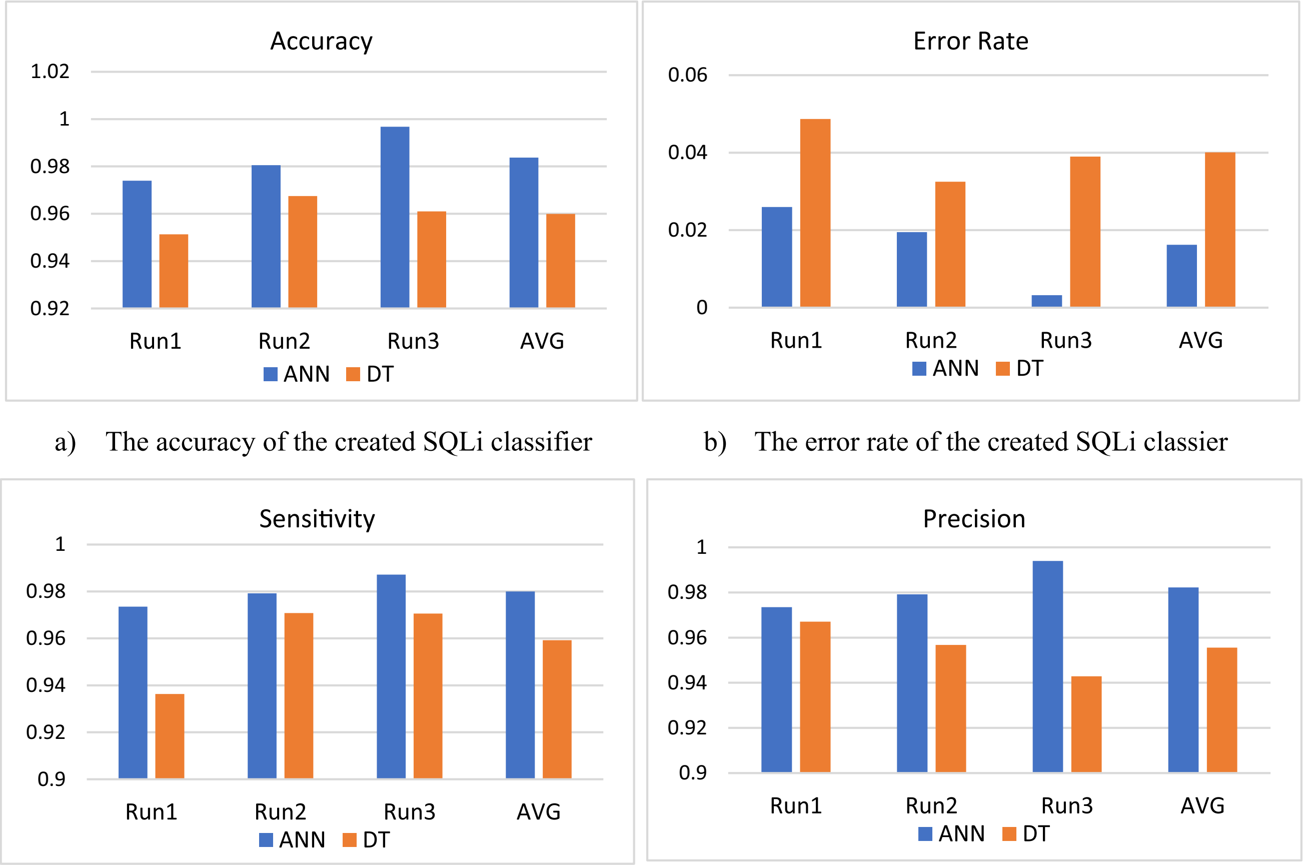
	Number of features		Selected features	
	bGWO1 + ANN	bGWO1 + DT	bGWO1 + ANN	bGWO1 + DT
Run1	4	6	[3, 4, 8]	[1, 3, 4, 8]
Run2	3	7	[3, 4]	[3–5, 7, 8]
Run3	4	6	[2, 4]	[1, 2, 4, 5, 8]

bGWO1 is higher than the sensitivity of the ANN and bGWO1. Table 9 shows the number of features selected by the bGWO1 algorithm in the best execution (best fitness value) during 10 times executions. The selected features by bGWO1 in the ANN are 4, 3, and 4 in three executions; the number of selected features by bGWO1 are 6, 7, and in three executions. The lower the number of selected features, the higher the performance of the ML algorithms.

Regarding the results shown in Fig. 9 and Table 9, the performance of the created SQLi detection by ANN and bGWO1 is higher than the performance of the created SQLi detector by the DT and bGWO1. The selected features by bGWO1 for the ANN and DT have overlapped. The fitness of the selected features by the bGWO1 was evaluated using the test stage. The accuracy, error rate, sensitivity, and precision of the trained model were considered as the selected feature's fitness.

4.4.3 The effects of bGWO2 on the SQLi detectors (RQ2)

In this subsection, the second research question (RQ2) is answered. In the second series of experiments, the developed bGWO2 was used to select optimal features before the training step. The developed bGWO2 algorithm was used along with the ANN and DT for creating SQLi classifier. Figure 10 shows the effect of the bGWO2 algorithm



a) The accuracy of the created SQLi classifier

b) The error rate of the created SQLi classier

c) The sensitivity of the created SQLi classier

d) The precision of the created SQLi classifier

Fig. 10 The performance of the created SQLi classifier created by ANN and DT using the bGWO2 feature selection algorithm

Table 10 The selected features by bGWO2 in best fitness value

	Number of features		Selected features	
	bGWO2 + ANN	bGWO2 + DT	bGWO2 + ANN	bGWO2 + DT
Run1	2	3	[2, 4]	[1, 3, 4]
Run2	3	3	[3, 4]	[1, 4]
Run3	2	3	[2, 4]	[1, 4, 8]

on the performance of the ANN and DT in the SQLi classification. The accuracy of the SQLi detector that was created by ANN and bGWO2 is about 0.9837; this figure for the DT and bGWO2 is about 0.9599. The error rate of the SQLi detector created by bGWO2 and ANN is about 0.0162 which is lower than the error rate of the SQLi detector created by DT and bGWO2. Similarly, the SQLi detector created by the ANN and bGWO2 has higher sensitivity and precision.

4.4.4 The effects of bGWO in reducing the number of features (RQ3)

Table 10 shows the selected features by bGWO2 in the best execution (best fitness value) during 10 times executions. The average number of selected features by bGWO2 is lower than the average number of selected features by bGWO1. Figure 11 shows the number of selected features by bGWO1 and bGWO2 in different runs. The results indicate that the bGWO2 has higher performance than the bGWO1 in terms of the number of selected features.

Figure 12 indicates the number of features in the raw and filtered dataset. The raw training dataset includes 12 features. The developed bGWO1 selects 7 effective

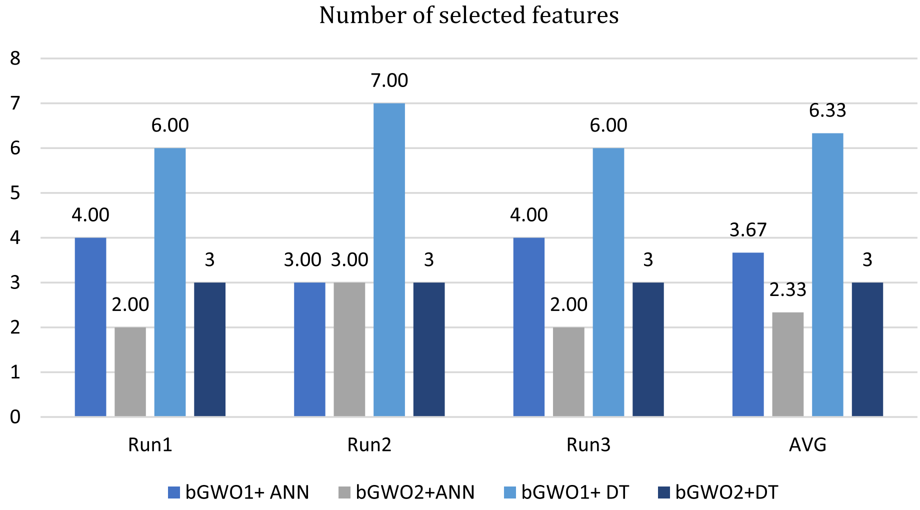


Fig. 11 The number of selected features by the bGWO1 and bGWO2 in different runs

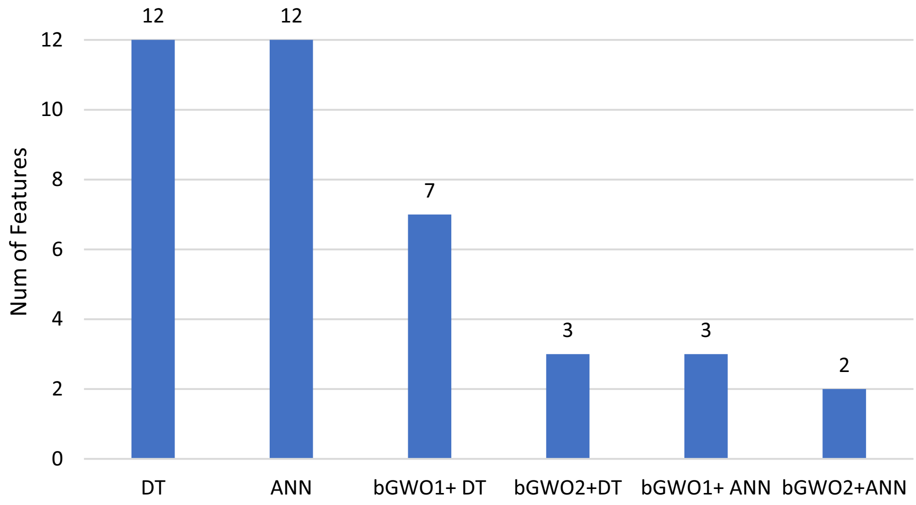


Fig. 12 The number of selected features in the SQLi dataset by different ML algorithms

Table 11 Evaluation of methods in terms of efficiency and feature selection in the best case

Methods	Features	Selected features
DT	12	All
ANN	12	All
bGWO1 + DT	7	[3–5, 7, 8]
bGWO2 + DT	3	[1, 4]
bGWO1 + ANN	3	[3, 4]
bGWO2 + ANN	2	[2, 4]

features (traits) when combined with the DT algorithm. The bGWO1 selects 3 features when combined with the ANN. The minimum number of features was selected by bGWO2 and ANN. Although the least number of features selected by the ANN and bGWO2, it has the best performance in terms of accuracy, sensitivity, and precision. On average, by picking 20% of the most efficient traits, the suggested strategy improves the efficacy of attack detection systems. Overall, the SQLi detection model created by the combination of the ANN and bGWO2 has higher performance than the other algorithms.

According to the obtained results, the first and second approaches of the binary GWO algorithm, especially the second approach of the algorithm, are better than the other methods in terms of accuracy, sensitivity, and precision detection. Also, in terms of feature selection takes less time. Consequently, it minimizes the detection time of SQL injection attacks. Table 11 shows the selected features by the bGWO1 and bGWO2 in the best case (fitness).

4.4.5 Evaluating the stability of the proposed method (RQ4)

The first research question is related to the stability of the proposed feature-selecting algorithm. The stability of the

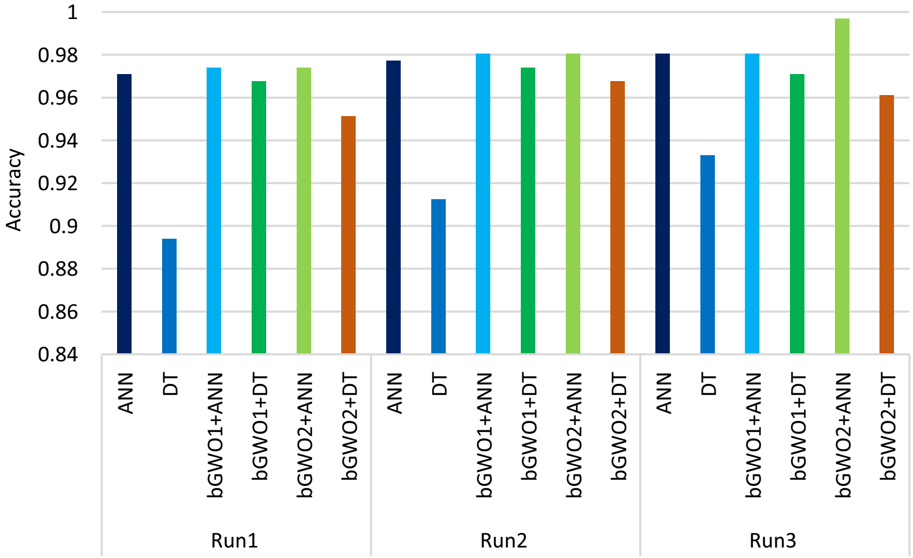


Fig. 13 Accuracy of the SQLi detector created by different methods with and without feature selection during different runs

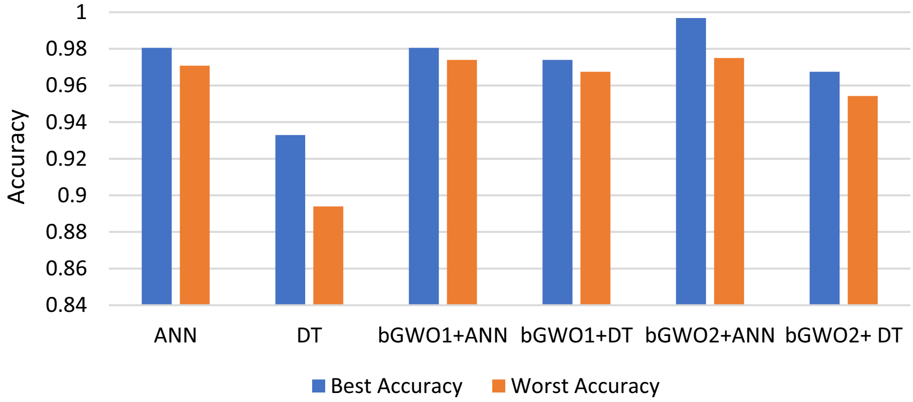


Fig. 14 The best and worst accuracy value of the SQLi detectors created by different methods

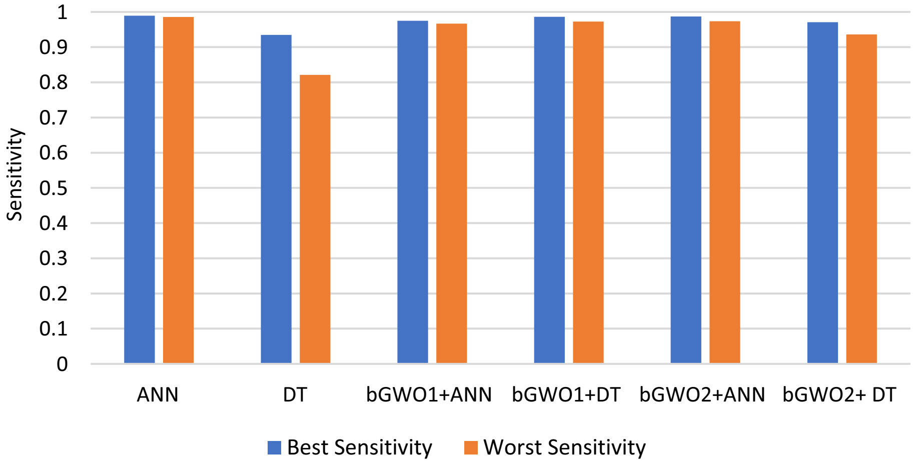


Fig. 15 The best and worst sensitivity value of the SQLi detectors created by different methods

metaheuristic algorithms (stochastic-based algorithms) should be evaluated based on their best, worst, and average outputs during different executions. To evaluate the best, worst, and average outputs of the proposed method, the proposed bGWO was executed at different times in the same condition. Figure 13 shows the accuracy of the SQLi classifier created by different methods in three runs. Overall, the performance of the DT and ANN without the proposed features selection is lower than the performance of the DT and ANN with the proposed features selection.

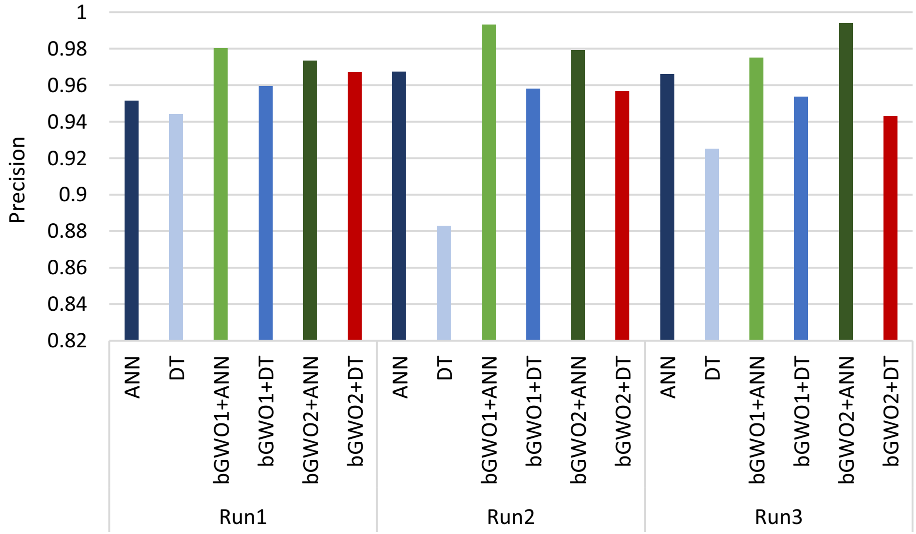


Fig. 16 Precision of the SQLi detector created by different methods with and without feature selection during different runs

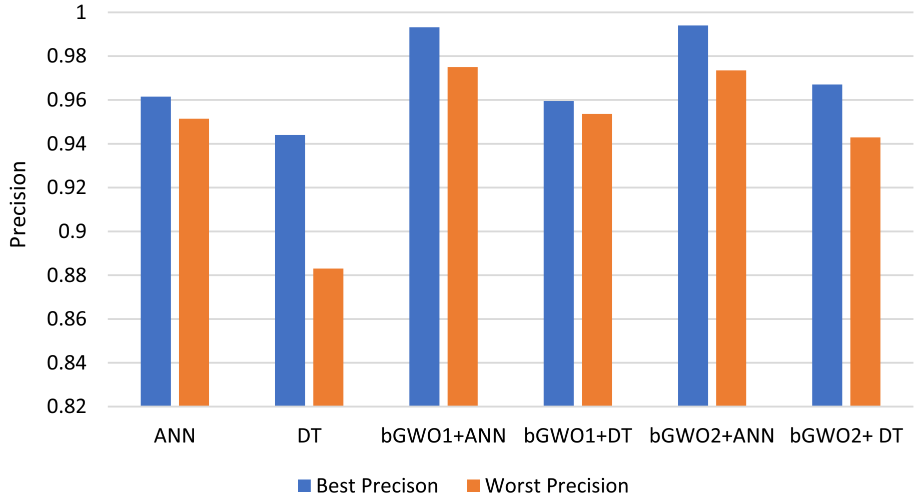


Fig. 17 The best and worst precision value of the SQLi detectors created by different methods

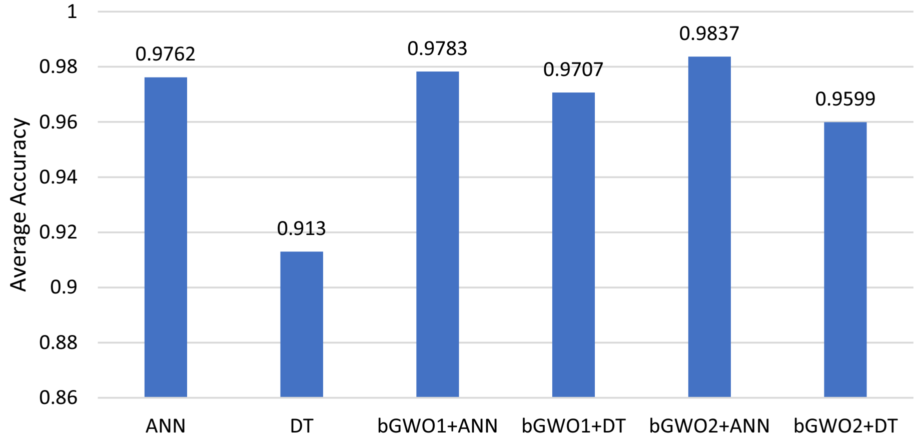


Fig. 18 The average accuracy of the created SQLi detectors created by different methods

The results of experiments confirm that the developed binary GWO, as feature selection, improves the accuracy of the created SQLi detector by the DT and ANN. When using bGWO2 together with the ANN, the accuracy is higher than the other methods.

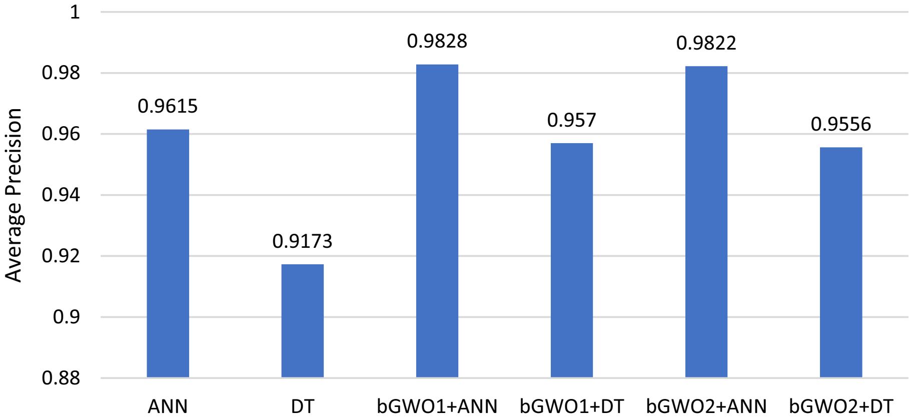


Fig. 19 The average precision of the created SQLi detectors created by different training algorithms

Figure 14 shows the best and worst accuracy of the SQLi detectors created by different machine learning algorithms with and without feature selection algorithms. As shown in Fig. 14, ANN has higher accuracy than the other methods in worst and best cases. Figure 15 shows the sensitivity of the created SQLi detector by different algorithms in the best and worst cases. Regarding the results, ANN has higher sensitivity than the other methods when it is used along with bGWO2.

Figure 16 shows the precision of the created SQLi detectors by different algorithms during three runs. As shown in Fig. 16, ANN has a higher precision when the training dataset is filtered by bGWO1 and bGWO2. DT, as a machine learning algorithm, has lower precision than the other algorithms. The precision of the model created by DT and bGWO (as feature selector) is higher than the original DT. Overall, the created model by ANN has higher precision than the model created by DT. Figure 17 shows the precision of the feature selector created by different algorithms in the best and worst cases. The best precision of bGWO1 + ANN and bGWO2 + ANN are 0.9932 and 0.9940, respectively; these figures in the worst cases are 0.9750 and 0.9735. Indeed, the created models by the filtered ANN are similar.

The filtered dataset by bGWO1 includes three features, whereas the filtered dataset by bGWO2 includes two features. The lower the number of features, the higher the performance. Furthermore, using the bGWO algorithm reduces the difference between the accuracy of the created classifiers in the best and worst case. Similar results have been obtained for the accuracy and sensitivity criteria. Indeed, machine learning algorithms have similar performance in different runs and consequently, they have higher stability. As shown in Figs. 13 and 16, bGWO has a higher impact on the performance of the ANN than the DT. The optimal results have been generated by the combination of ANN and bGWO2 in terms of accuracy and precision. Overall, in the SQLi classification problem, the combination of bGWO1 and ANN is superior to the other algorithms.

Figure 18 shows the average accuracy of the classification models created by different algorithms on the raw and filtered train dataset. The average accuracy of the created models by the filtered ANN (bGWO2 + ANN) is higher than the algorithms. Both proposed feature selection algorithms (bGWO1 and bGWO2) have higher effects on the accuracy of the models generated by ANN. Similarly, the average precision of the classification models generated by filtered ANN (bGWO and ANN) is higher than the precision of the created models by the other algorithms. Figure 19 indicates the average precision of the created SQLi detection by different training algorithms.

Table 12 The success rate of different methods

Methods	Success rate
DT	0.8795
ANN	0.9866
bGWO1 + DT	0.9917
bGWO2 + DT	0.9863
bGWO1 + ANN	0.991
bGWO2 + ANN	0.9802

4.4.6 Evaluating the success rate of the proposed method (RQ5)

To evaluate the success rate of the proposed method. Another series of experiments have been conducted. In these experiments, the average accuracy of the created SQLi detectors by ANN, DT, bGWO1 + ANN, bGWO1 + DT, bGWO2 + ANN, and bGWO2 + ANN have been evaluated. The success rate is the probability of reaching the optimal solution (highest accuracy in SQLi detection) in these 10 executions. The probability of reaching the highest accuracy by the created different SQLi detectors indicates the success rate of that method. Table 12 shows the calculated success rate of different

Table 13 The average running time of the training stage in the SQLi creation with and without feature selectors

Methods	Training by the dataset with all Features	Feature selection using GWO1	Feature selection using GWO2	Training by the dataset with selective features
DT	0.1057 s	9.8611 s	11.6230 s	0.0184 s
ANN	3.6185 s	845.2082s	930.5194 s	1.0520 s

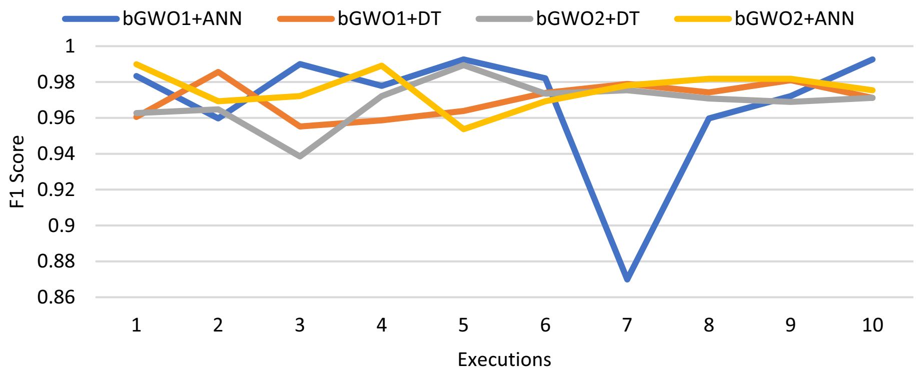


Fig. 20 The F1 score of different SQLi detection methods during 10 times executions

Table 14 The average F1 score of different method during 10 times executions

bGWO1 + ANN	bGWO1 + DT	bGWO2 + DT	bGWO2 + ANN
0.970588	0.972775	0.96874	0.97606

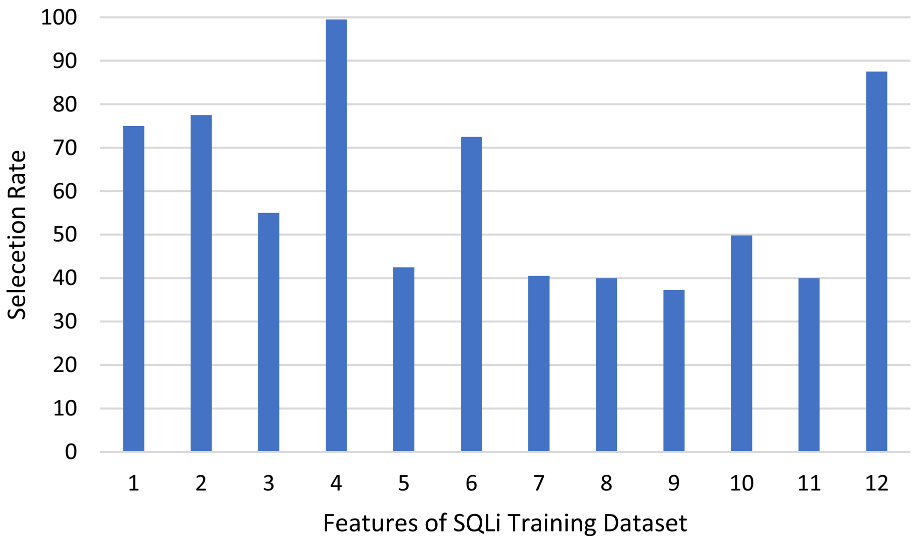


Fig. 21 The selection rate of the features in the SQLi dataset by the proposed bGWO during 10 times executions

methods. As shown in Table 12, bGWO1 + DT and bGWO1 + ANN have the higher success rate and DT has the lowest success rate. Table 13 shows the average running time of the training stage using DT and ANN using the dataset with all features (12 features) and the average running time of the training stage using the dataset with selective features. The running time of the proposed feature selectors (bGWO1 and bGWO2) has been shown in Table 13. Each training stage was executed 10 times, and the average running time was calculated. The bGWO was executed 10 times and each execution includes 100 iterations.

The F1 score of each method during 10 executions has been indicated in Fig. 20. Furthermore, Table 14 shows the average F1 score of the methods (bGWO1 + ANN, bGWO1 + DT, bGWO2 + DT and bGWO2 + ANN) during 10 times executions.

Furthermore, Fig. 21 shows the average selection rate of the features in the SQLi training dataset by the proposed bGWO during 10 times executions. Features 4, 12, 2, 1, 6 and 3, respectively, are the most selected features by the bGWO (Fig. 21). Indeed, these features have the highest effect on the performance of the created SQLi detectors. These features are as follows:

- Num. of Constant Value in the SQL Query
- Num. Parenthesis in the SQL Query
- Nesting Level Query
- Length of SQL Query
- Num. of Specific Character in the SQL Query
- Num. of union Operator in the SQL Query
- Num. of double quotation Character (') in the SQL Query

5 Conclusion and future works

Machine learning techniques were used to create the SQLi attack detection models. The performance of machine learning-based attack detectors depends on the training dataset and algorithm. An effective strategy for detecting SQL injection attacks has been provided in this study. To choose the most effective characteristics of the dataset, two binary variants of the Gray-Wolf algorithm were constructed. The test results show that the suggested SQL injection detection techniques have 99.68% accuracy, 99.40% precision, and 98.72% sensitivity. By picking 20% of the most effective traits, the suggested strategy improves the efficacy of attack detectors. The limited number of selected effectivefeatures is one of the main merits of the proposed methods. Obtaining the higher value of accuracy, precision, and sensitivity is the other merit of the method. Providing similar results during different executions is the other advanteg of this study. The machine learning and heuristic algorithms exploited in [22–26] can be used to develop effective SQLi detection models. Investigating the performance of the SQLi detectors using other advanced learning and deep learning methods [27] is considered as a future study. Evaluating the chaos based methods [28] in the performance of the bGWOA is suggested as another future study. The developed bGWOA can be used in the resiliency improvement problems [29].

Author contributions The proposed method was designed by BA. The designed algorithm was implemented and coded by BF and checked by KA. The implemented method was adapted and benchmarked by BA and BF. The generation of the raw SQLi dataset was performed by BA and BA. The data and results analysis were performed by BA and BA. The manuscript of the paper was written by BA and BA. The paper was proofread by BA, FK and MT-A.

Funding Open access funding provided by the Scientific and Technological Research Council of Türkiye (TÜBİTAK).

Data availability The data relating to the current study are available on Google. drive and can be freely accessed by the following link: https://drive.google.com/drive/folders/194tzSPS03D4Ylt1L30qv21meCwpnG1Jt?usp=sharing.

Declarations

Conflict of interest The authors declare that no funds, grants, or other support were received during the preparation of this manuscript. The authors have no relevant financial or non-financial conflict of interest.

Ethical approval The data used in this research do not belong to any other person or third party and were prepared and generated by the researchers themselves during the research. The data of this research will be accessible to other researchers.

References

1. Marashdeh, Z, Suwais, K, Alia, M (2021) A Survey on SQL Injection attacks: detection and challenges. In: Proceedings of the 2021 international conference on information technology (ICIT), Amman, Jordan, pp 957–962
2. Huang H-C, Zhang Z-K, Cheng H-W, Shieh SW (2017) Web application security: threats, countermeasures, and pitfalls. Comput (Long Beach Calif) 50(6):81–85. https://doi.org/10.1109/MC.2017.183
3. Ibarra-Fiallos S, Higuera JB, Intriago-Pazmino M, Higuera JRB, Montalvo JAS, Cubo J (2021) Effective filter for common injection attacks in online web applications. IEEE Access 9:10378–10391. https://doi.org/10.1109/ACCESS.2021.3050566
4. Hu H (2017) Research on the technology of detecting the SQL injection attack and non-intrusive prevention in WEB system. AIP Conf Proc. https://doi.org/10.1063/1.4982570
5. Tian W, Yang J-F, Xu J, and Si G-N (2012) Attack model based penetration test for SQL injection vulnerability. In: 2012 IEEE 36th annual computer software and applications conference workshops, pp. 589–594. https://doi.org/10.1109/COMPSACW.2012.108.
6. Buja G, Jalil KBA, Ali FBHM, and Rahman TFA (2015) Detection model for SQL injection attack: an approach for preventing a web application from the SQL injection attack. In: ISCAIE 2014-2014 IEEE symposium on computer applications and industrial electronics, pp 60–64. https://doi.org/10.1109/ISCAIE.2014.7010210.

7. Masri W, Sleiman S (2015) SQLPIL: SQL injection prevention by input labeling. Secur Commun Netw 8(15):2545–2560. https://doi.org/10.1002/sec.1199
8. Parvez M, Zavarsky P, and Khoury N (2015) Analysis of effectiveness of black-box web application scanners in detection of stored SQL injection and stored XSS vulnerabilities. In: 2015 10th international conference for internet technology and secured transactions (ICITST), pp 186–191. https://doi.org/10.1109/ICITST.2015.7412085.
9. Huang Y-W, Huang S-K, Lin T-P, and Tsai C-H (2003) Web application security assessment by fault injection and behavior monitoring. In: Proceedings of the twelfth international conference on World Wide Web–WWW, p 148. https://doi.org/10.1145/775152.775174.
10. Lee I, Jeong S, Yeo S, Moon J (2012) A novel method for SQL injection attack detection based on removing SQL query attribute values. Math Comput Model 55(1–2):58–68. https://doi.org/10.1016/j.mcm.2011.01.050
11. Gould C, Su Z, and Devanbu P (2004) JDBC checker: a static analysis tool for SQL/JDBC applications. In: Proceedings 26th international conference on software engineering, vol 26, pp 697–698. https://doi.org/10.1109/ICSE.2004.1317494.
12. Wassermann G and Su Z An analysis framework for security in Web applications. SAVCBS 2004 Specif. Verif. Component-Based Syst, p 70
13. Thomas S and Williams L (2007) Using automated fix generation to secure SQL statements. Softw Eng Secur Syst 2007. SESS '07 ICSE Work. 2007. Third Int. Work, p 9
14. Kosuga Y, Kono K, Hanaoka M, Hishiyama M, and Takahama Y (2007) Sania: syntactic and semantic analysis for automated testing against SQL injection. In: Twenty-third annual computer security applications conference (ACSAC 2007), pp 107–117. https://doi.org/10.1109/ACSAC.2007.20.
15. Bashah Mat Ali A, Yaseen Ibrahim Shakhatreh A, Syazwan Abdullah M, Alostad J (2011) SQL-injection vulnerability scanning tool for automatic creation of SQL-injection attacks. Procedia Comput Sci 3:453–458. https://doi.org/10.1016/j.procs.2010.12.076
16. William WG and Orso A (2005) AMNESIA: analysis and monitoring for NEutralizing SQL-injection attacks. In: Proceedings of the 20th IEEE/ACM international conference on automated software engineering
17. Buehrer GT, Weide BW, and Sivilotti PAG (2005) Using parse tree validation to prevent SQL injection attacks. In: Proceedings of the 5th international workshop on software engineering and middleware–SEM, p 106. https://doi.org/10.1145/1108473.1108496.
18. Park JC and Noh BN (2007) SQL injection attack detection: profiling of web application parameter using the sequence pairwise alignment. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics), vol 4298, pp 74–82. https://doi.org/10.1007/978-3-540-71093-6_6.
19. Valeur F, Mutz D, and Vigna G (2005) A learning-based approach to the detection of SQL attacks. Lect Notes Comput Sci 3548. In: Detection of intrusions and malware, and vulnerability assessment: second international conference, DIMVA 2005. Proceedings, pp 123–140. doi: https://doi.org/10.1007/11506881_8.
20. Joshi A and Geetha V (2014) SQL Injection detection using machine learning. In: 2014 International conference on control, instrumentation, communication and computational technologies (ICCICCT), no 2, pp 1111–1115. https://doi.org/10.1109/ICCICCT.2014.6993127
21. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007
22. Keshtgar A, Arasteh B (2017) Enhancing software reliability against soft-error using minimum redundancy on critical data. Int J Comput Netw Inf Secure 9:51. https://doi.org/10.5815/ijcnis.2017.05.03
23. Zadahmad M, Arasteh B, Yousefzadeh Fard P (2011) A pattern-oriented and web-based architecture to support mobile learning software development. Procedia Soc Behav Sci 28:194–199. https://doi.org/10.1016/j.sbspro.2011.11.037
24. Bouyer A, Arasteh B, Movaghar A (2007) A new hybrid model using case-based reasoning and decision tree methods for improving speedup and accuracy. In: IADIS international conference of applied computing 2007.
25. Arasteh B, Abdi M, Bouyer A (2022) Program source code comprehension by module clustering using a combination of discretized gray wolf and genetic algorithms. Adv Eng Softw 173:103252. https://doi.org/10.1016/j.advengsoft.2022.103252
26. Arasteh B, Pirahesh S, Zakeri A, Arasteh B (2014) Highly available and dependable e-learning services using grid system. Procedia-Soc Behav Sci 143:471–476. https://doi.org/10.1016/j.sbspro.2014.07.519
27. Arasteh B (2022) Clustered design-model generation from a program source code using chaos-based metaheuristic algorithms. Neural Comput 1:23. https://doi.org/10.1007/s00521-022-07781-6
28. Mendonça YVS, Vinueza PG, Diego CP (2022) The role of technology in the learning process: a decision tree-based model using machine learning. Emerg Sci J. https://doi.org/10.28991/ESJ-2022-SIED-020
29. Arasteh B, Miremadi SG, Rahmani AM (2014) Developing inherently resilient software against soft-errors based on algorithm level inherent features. J Electron Test 30:193–212. https://doi.org/10.1007/s10836-014-5438-8