Proceedings of the 5th International Conference on Smart Systems and Inventive Technology (ICSSIT 2023)
IEEE Xplore Part Number: CFP23P17-ART; ISBN: 978-1-6654-7467-2

SQL Injection Detection using Machine Learning and Convolutional Neural Networks

1st Jason Misquitta
Dept. of Computer Science and Engineering
Vellore Institute of Technology (VIT)
Chennai, India
jason.misquitta2020@vitstudent.ac.in

2nd Dr. Asha S
Dept. of Computer Science and Engineering
Vellore Institute of Technology (VIT)
Chennai, India
asha.s@vit.ac.in

2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT) | 978-1-6654-7467-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICSSIT55814.2023.10061019

***Abstract*—SQL Injection Attacks are a concern to database-driven online applications because they can compromise the security and integrity of information in databases. It gives hackers the potential to read, delete, modify or copy data. Exploiting the flaws in the validation of user inputs provided through login pages is the simplest technique to conduct this attack. Each login page a user accesses can help to reveal more about who they are. Feedbacks sent by the server while running a SQL command can provide details on weaknesses in the database server's validation procedure. This research paper analyses code that focuses on detecting SQL injections. Different supervised Machine Learning algorithms are used to evaluate the model. A CNN model is also created for real-time identification of whether an input is a potential attack or not.**

***Index Terms*—SQL injection, CNN, Machine Learning, user-interface, Naive Bayes, neural networks**

I. Introduction

Data is becoming more valuable every year. Organization's data leakage cause a loss of not only financial but also social standing, distrust among staff and users, the loss of a significant number of users, and sometimes even severe legal action can be taken. Web applications have multiplied in number as the Internet has grown. Numerous web programs interact with users and gather personal data from them. Thus, they maintain a connection to the database. There results in several SQL injections due to the database's high amount of valuable data being stored there, which makes it a huge attractive target for attackers [1].

The term "SQL injection attack" describes the creation of unique strings that are sent to online applications as parameters when web forms are submitted or when page requests or query strings for domain names are entered [2]. The executable statements in the SQL grammar that are frequently found in these specific strings cause web applications to mistake data for executable code, tricking servers into executing malicious SQL instructions [3].

Detection techniques include examining server logs and keeping an eye on database issues. The majority of network perimeter firewalls and intrusion detection systems (IDS) are not set up to scan HTTP traffic for malicious SQL fragments, allowing an attacker to get through network security perimeters [4]. Security solutions can incorporate Web application firewalls (WAF) to filter HTTP queries that coincide with SQL injection attempts. However, a WAF needs to be updated frequently to catch new tricks. Besides this, there are several SQL injection detection tools like Netsparker, SQLMap, Burp, Havij, etc.

Due to machine learning's ability to process data quickly, precisely, and accurately using mathematical, statistical, and econometric applications, it can easily find patterns in given samples of data resulting in its applications in security growing every day. By looking at statistics like precision, accuracy, recall, different supervised learning algorithms like K-Nearest Neighbours (KNN), Decision Tree, Support Vector Machines (SVM), Naïve Bayes can be compared and evaluated and see which one is the best for SQL injection detection.

In the last few years, Deep Learning has proved to be extremely resourceful since it can handle abundance of data. By using hidden layers in pattern recognition , deep neural networks have surpassed traditional techniques. Convolutional Neural Networks (CNN) are mostly used for processing of pixel data which assists in image recognition tasks. This makes them highly suitable for object recognition as seen in self-driving cars and facial recognition. It also serves as a great asset for computer vision. However, they can even be used for text segmentation and classification. Hence, CNN architecture is used to create a real-time user interface, where the user can input a sentence/string and the model identifies if it is an SQL injection attack or not.

II. Literature Review

While starting the research, [5] provided a lot of insight into this domain as it gives an in-depth examination of subjects like the fundamentals of SQL Injection, types, and current attacks as a case study. What makes this paper stand out is that we learn the methods being used as a base to trigger vulnerabilities in this internet-connected environment, which in turn compromises the database and discloses key secrets.

To start the modelling process, it was important to have a proper understanding of CNN. [6] provided this along with recently released studies and innovative methodologies used to create these brilliantly spectacular image recognition models. The different layers of CNN like Convolutional Layer, Pooling Layer, Full-connected layer are all explained in deep.

To design our own models we still needed some idea of how to go about it. Hence, we looked at some research papers that worked on machine learning and CNN models. The papers are listed and explained in the next six paragraphs.

[1] presents an SQL injection detection method that relies on Elastic-Pooling CNN. It draws a comparison with traditional methods of detection. This technique can gives fixed matrix of two-dimensional as output without any truncation of data. This efficiently recognizes the SQL injection present in web applications. It is more difficult to overpass and can identify new attacks by looking at the uneven matching characteristics.

[7] informs about a machine learning classifier that looks at PHP code and identifies if any vulnerabilities are present that could lead to SQL injection. With the aid of input validation features that were taken from source code files, classifier models were trained and evaluated using both traditional and deep learning-based machine learning techniques. The highest recall and f-measure was obtained by MLP-trained models while CNN-trained models resulted in the highest precision on ten-fold cross validations.

Based on extensive domestic and international research, [13] suggests a natural language processing model along with a deep learning framework for SQL injection detection that has nothing to do with the background rule base. By allowing the computer to automatically pick up on the language model characteristics of SQL injection assaults, the strategy can increase accuracy, decrease false alarm rates, and provide some protection against never-occurring zero-day attacks.

[14] suggests a SQL injection detection model using CNN. This model uses the high-dimensional property of injection behaviour to provide a solution. It first obtains payloads related to SQL injection attack from network flow. The proposed remedy was contrasted with the rule-matching-based method, ModSecurity, in a case study of actual real traffic. The experimental results show greater accuracy, precision, and recall rates for the CNN-based model, demonstrating its utility in attack detection and robustness against obfuscation.

[15] is collecting traffic from a Datiphy appliance node (situated between the web application host and the related database server) and the web application host. Their analysis has shown that the accuracy provided by the correlated dataset that uses algorithms like rule-based and decision trees is almost identical to that obtained with a neural network, however the performance is significantly better.

[16] analyses extraction of feature from SQL injection. The HTTP request is processed using the word2vec method. The processed samples are trained and then classified using the SVM algorithm. The method resulted in a high accuracy since it solves the problem of a high rate of leakage in rule matching.

III. Machine Learning Algorithms

In this research, we comparatively studied four machine learning algorithms. A brief explanation of each of them is given below.

A. *Naïve Bayes*

It is a group of supervised learning algorithms derived from the Bayes Theorem. It works on the assumption that attributes have no dependence on each other. This massively minimizes the cost of computation since we are only counting the class distribution. In Gaussian Naïve Bayes, the features are assumed to be Gaussian. The continuous values are are distributed according to a Gaussian distribution. This results in the formation of a bell-shaped curve that follows symmetry around the mean of the feature values. Naive Bayes was chosen as it saves a lot of time since it works very quickly. Besides, it does not need as much training data and is not sensitive to irrelevant features.[8]

B. *Support Vector Machines (SVM)*

Support Vector Machines is another type of supervised learning algorithm. A hyperplane separates the classes in feature space as far as possible. This enables us to classify the data points into 2 classes. It has a straight decision boundary. If data is not linearly-separable, a kernel function can be used to map data to a space of higher dimensionality in an implicit manner. Thus there is no need to compute each f(x) explicitly). The greater the margin of separation between the two classes, the better the classification (also called margin maximization). SVM is used for regression, classification and also outlier detection. [9] SVM was chosen as it works comparatively better when there is a big margin of separation between classes. It is more productive in high-dimensional areas and conserves memory.

C. *K-Nearest Neighbours (KNN)*

The KNN algorithm does not use the training data points for generalization; instead, it uses them during the testing phase. Its extensive use of training data points for the prediction can make it slow for large datasets [10]. The following is the core concept of the traditional KNN: To assess how similar the data points are, based on all attributes, choose a metric, such as Euclidean distance, after designing a collection of numerical features to characterise each data point. Then, based on the similarity measure, select the k closest points to a target point in the training samples, and by a majority vote of those points, classify the target point [11]. KNN was chosen as it is non-parametric hence it makes no assumptions which allows us to discover hidden relations in the data, thus providing a whole new perspective. KNN does a wonderful job in considering outliers present in the data.

D. *Decision Tree*

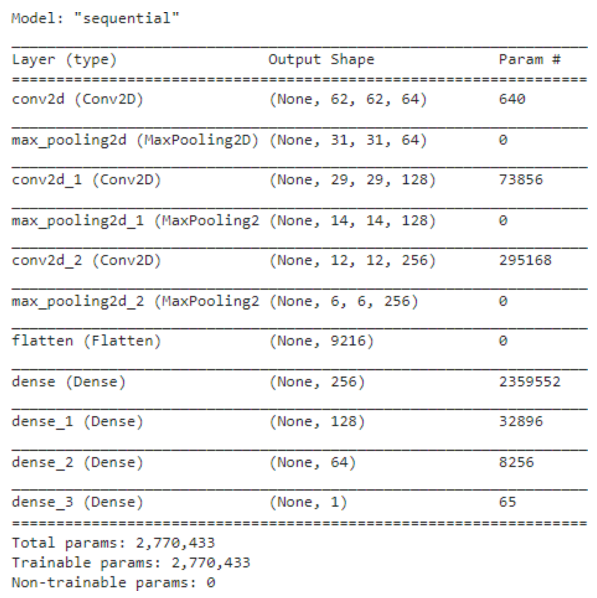
Decision Tree is a non-parametric form of supervised learning. We try to create a model that predicts the next target by learning from information deduced from previous data parameters. It is used for regression and classification. However, it is highly suited to multi-class classification on a dataset. The dataset run through a tree, with a decision to be made at the various nodes. This helps in visualization. A plus side is that it can handle both numerical and categorical data. Even if the dataset has an error or is missing values, it is not a problem. However, biased trees can be created if few classes dominate. Hence, balancing is required. [12] Decision Tree was chosen as the amount of pre-processing required here is very little. No normalization or scaling is required. Missing values also have no effect.

IV. Methodology

All the coding is done in Python. We first imported all the 5 dataset files from different GitHub repositories. They are text files consisting of possible SQL injections along with benign strings of data. There were totally 5234 records of data in all the datasets combined, out of which 3816 records were benign data and 1418 were SQL injection data.

We pre-processed each file individually in order to check missing values, noisy data, and other inconsistencies. For example, at the end of some lines, '\n' was present so it had to be removed. '%20' had to be removed from some records. In many records, there were spacing issues, hence '=' was replaced by ' = ', similarly '))' was replaced by ')) ' and so on. All of this was done to make the data compatible for the models to analyze.

After pre-processing, we combined all of the 5 datasets into one single dataset using pandas' data frame method (python module). Next, we split the dataset into test and train. We split the benign data into 3072 records for training and 744 for testing. The SQL injection data was split into 1128 records for training and 290 for testing. Lastly, we fed them into the different machine learning algorithms and analyzed the results from each.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 62, 62, 64)        640
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 31, 31, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 29, 29, 128)       73856
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 12, 12, 256)       295168
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 9216)              0
_________________________________________________________________
dense (Dense)                (None, 256)               2359552
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 65
=================================================================
Total params: 2,770,433
Trainable params: 2,770,433
Non-trainable params: 0
```

Fig. 1. CNN Architecture

Finally we created a CNN model (with Adam Optimizer) for SQL injection detection. The learning rate applied was 0.001. We then imported gradio and used it to run a real-time user interface where the user can input a sentence/string and the CNN model identifies if it is an SQL injection attack or not.

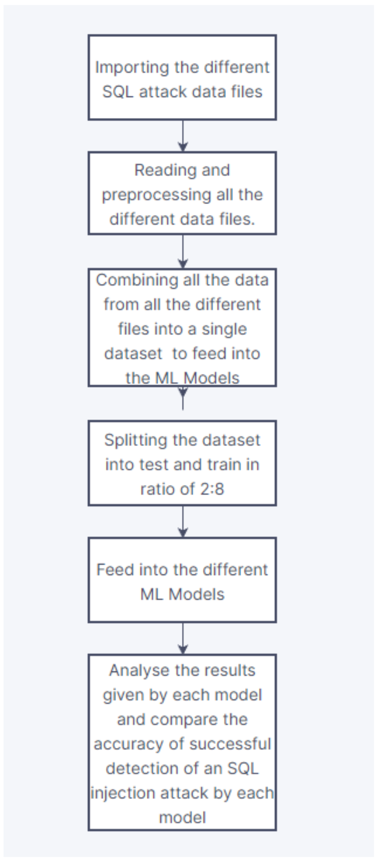

Importing the different SQL attack data files
Reading and preprocessing all the different data files.
Combining all the data from all the different files into a single dataset to feed into the ML Models
Splitting the dataset into test and train in ratio of 2:8
Feed into the different ML Models
Analyse the results given by each model and compare the accuracy of successful detection of an SQL injection attack by each model

Fig. 2. Sequence of events in methodology

V. Comparative Analysis

For comparative analysis, we look at 4 metrics : **accuracy, precision, recall and F-score**

- Accuracy tells us what percentage of the total number of data instances are correctly classified data instances.
- Precision measures what percentage of predicted positive instances are true.
- Recall indicates what percentage of actual positive instances are true.
- F-score gives an optimal blend of precision and recall as it is the harmonic mean of those two metrics.

In the confusion matrix, we have four measures :

- True Positives (TP) - The number of records correctly classified as attacks
- True Negatives (TN) - The number of records correctly classified as benign

- False Positives (FP) - The number of records wrongly classified as attacks
- False Negatives (FN) - The number of records wrongly classified as benign

Accordingly, the metrics are calculated as :

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F - score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4}$$

Metrics for each Machine Learning Algorithm :

All metrics are rounded to the 3rd decimal place

A. *Naive Bayes*

Accuracy : 0.975
Precision : 0.912
Recall : 1.0
F-score : 0.954

B. *Support Vector Machines*

Accuracy : 0.826
Precision : 1.0
Recall : 0.316
F-score : 0.480

C. *K- Nearest Neighbours*

Accuracy : 0.865
Precision : 0.708
Recall : 0.801
F-score : 0.752

D. *Decision Tree*

Accuracy : 0.841
Precision : 0.617
Recall : 1.0
F-score : 0.763

Fig.4 was generated by visualizing the accuracies of the 4 machine learning algorithms. Similarly, the F-scores have been visualized and displayed in Fig.5. We used the library scikit-learn to import GaussianNB, SVC, KNeighborsClassifier and tree. These are the ready made functions to implement the respective algorithms. After passing the training data and testing data to each function, we received the prediction. The prediction was then passed to a confusion matrix that generated the above 4 metrics. Accordingly, the graphs were made.

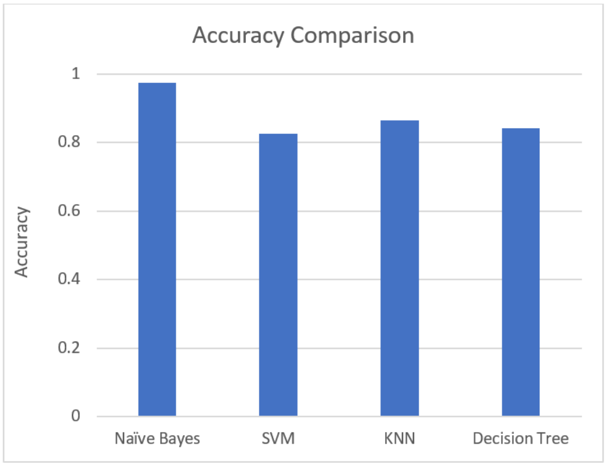


Fig. 3. Bar graph of Accuracy of different ML models

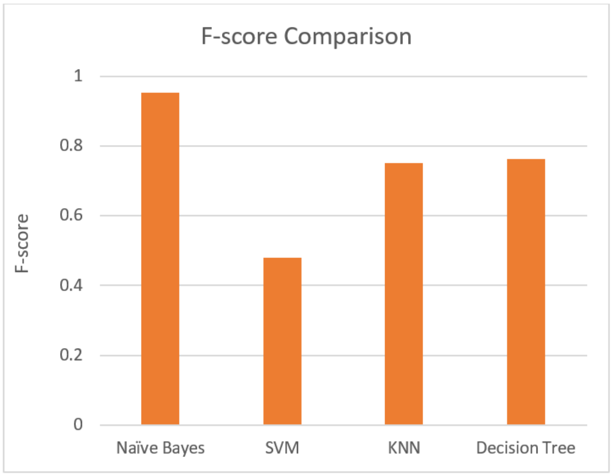


Fig. 4. Bar graph of F-score of different ML models

TABLE I
STATISTIC METRICS OF THE 4 ML MODELS

Algorithms	Accuracy	Precision	Recall	F-score
Naive Bayes	0.975	0.912	1.0	0.954
Support Vector Machines	0.826	1.0	0.316	0.480
K-Nearest Neighbours	0.865	0.708	0.801	0.752
Decision Tree	0.841	0.617	1.0	0.763

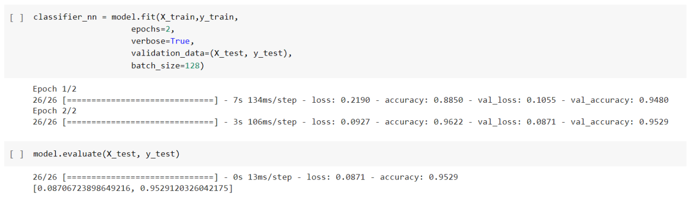
VI. DISCUSSION

Among all the algorithms, Naïve Bayes has the highest accuracy of 0.975 while Support Vector Machines (SVM) has the least accuracy of 0.826

Highest precision is shown by SVM. It is exactly 1.0 . Lowest precision is 0.617 which is by Decision Tree.

However, Decision Tree gives the highest recall of exactly 1.0 while SVM gives a very low recall of 0.316.

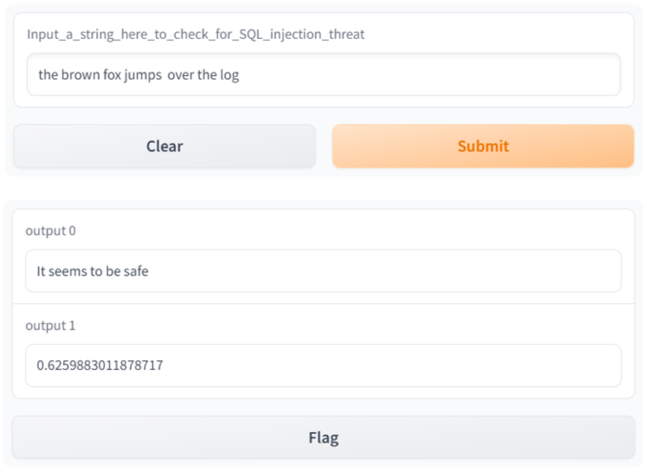
Naïve Bayes provides the highest F-score of 0.954 while the lowest is 0.48 by SVM.



```
[ ]  classifier_nn = model.fit(X_train,y_train,
                               epochs=2,
                               verbose=True,
                               validation_data=(X_test, y_test),
                               batch_size=128)

     Epoch 1/2
     26/26 [==============================] - 7s 134ms/step - loss: 0.2190 - accuracy: 0.8850 - val_loss: 0.1055 - val_accuracy: 0.9480
     Epoch 2/2
     26/26 [==============================] - 3s 106ms/step - loss: 0.0927 - accuracy: 0.9622 - val_loss: 0.0871 - val_accuracy: 0.9529

[ ]  model.evaluate(X_test, y_test)

     26/26 [==============================] - 0s 13ms/step - loss: 0.0871 - accuracy: 0.9529
     [0.08706723898649216, 0.9529120326042175]
```

Fig. 5. Evaluation of CNN model

The CNN model gave a very high accuracy of 95.29% within just 2 epochs as seen in Fig.5. This indicates that the model is a successful implementation. Accuracy can further be increased using more epochs and hyperparametric tuning.



Input_a_string_here_to_check_for_SQL_injection_threat

the brown fox jumps over the log

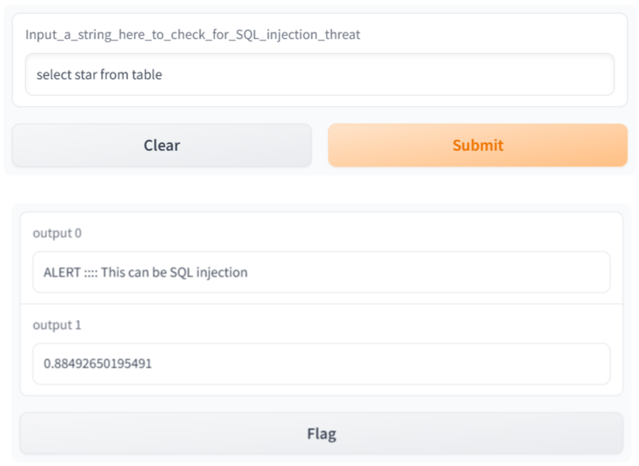
output 0

It seems to be safe

output 1

0.6259883011878717

Fig. 6. Model detects a text as safe



Input_a_string_here_to_check_for_SQL_injection_threat

select star from table

output 0

ALERT :::: This can be SQL injection

output 1

0.88492650195491

Fig. 7. Model detects a text as a possible attack

VII. Conclusion

After looking at all the parameters, we can conclude that Naïve Bayes is the best algorithm for SQL injection detection as it gave the highest accuracy and highest f-score. The worst algorithm seems to be Support Vector Machines (SVM) as it reported the lowest accuracy, recall and f-score.

Since our CNN model is very successful, we used it to run a real-time user interface to detect if a text is a possible SQL injection or not. The working user interface is shown in Fig.6 and Fig.7. for both the cases. We can add this code into Web browser extension, so that this will Save, and alert the user in Real time mode about any of those attacks. This model can also be included in Email/Whatsapp etc, i.e., when any of the unknown /suspicious link is there, instead of directly opening the link in the browser, our model will first check the website and then notify if any of the SQL injected codes are there or not.

References

[1] Xie, X., Ren, C., Fu, Y., Xu, J., Guo, J. (2019). Sql injection detection for web applications based on elastic-pooling cnn. IEEE Access, 7, 151475-151481.
[2] Kumar, A., Binu, S. (2018). Proposed Method for SQL Injection Detection and its Prevention. International Journal of Engineering Technology, 7(2.6), 213.
[3] Alghawazi, M., Alghazzawi, D., Alarifi, S. (2022). Detection of SQL Injection Attack Using Machine Learning Techniques: A Systematic Literature Review. Journal of Cybersecurity and Privacy, 2(4), 764-777.
[4] Tajpour, A., zade Shooshtari, M. J. (2010, July). Evaluation of SQL injection detection and prevention techniques. In 2010 2nd International Conference on Computational Intelligence, Communication Systems and Networks (pp. 216-221). IEEE.
[5] Devi, R., Venkatesan, R., Raghuraman, K. (2016). A study on SQL injection techniques. Int. J. Pharm. Technol., 8(4), 22405-22415.
[6] O'Shea, K., Nash, R. (2015). An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458.
[7] Zhang, K. (2019, November). A machine learning based approach to identify SQL injection vulnerabilities. In 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE) (pp. 1286-1288). IEEE.
[8] Rish, I. (2001, August). An empirical study of the naive Bayes classifier. In IJCAI 2001 workshop on empirical methods in artificial intelligence (Vol. 3, No. 22, pp. 41-46).
[9] Zhang, Y. (2012, September). Support vector machine classification algorithm and its application. In International conference on information computing and applications (pp. 179-186). Springer, Berlin, Heidelberg.
[10] Song, Y., Liang, J., Lu, J., Zhao, X. (2017). An efficient instance selection algorithm for k nearest neighbor regression. Neurocomputing, 251, 26-34.
[11] Yao, Z., Ruzzo, W. L. (2006, March). A regression-based K nearest neighbor algorithm for gene function prediction from heterogeneous data. In BMC bioinformatics (Vol. 7, No. 1, pp. 1-11). BioMed Central.
[12] Patel, H. H., Prajapati, P. (2018). Study and analysis of decision tree based classification algorithms. International Journal of Computer Sciences and Engineering, 6(10), 74-78.
[13] Chen, D., Yan, Q., Wu, C., Zhao, J. (2021). Sql injection attack detection and prevention techniques using deep learning. In Journal of Physics: Conference Series (Vol. 1757, No. 1, p. 012055). IOP Publishing.
[14] Luo, A., Huang, W., Fan, W. (2019, June). A CNN-based Approach to the Detection of SQL Injection Attacks. In 2019 IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS) (pp. 320-324). IEEE.
[15] Ross, K. (2018). SQL injection detection using machine learning techniques and multiple data sources.
[16] Chen, Z., Guo, M. (2018). Research on SQL injection detection technology based on SVM. In MATEC web of conferences (Vol. 173, p. 01004). EDP Sciences.