Journal of Sensors, IoT & Health Sciences
ISSN: 2584-2560
Vol.02(01), Mar 2024, pp.55-66

Research Article

Research on SQL Injection Attacks using Word Embedding Techniques and Machine Learning

S. Venkatramulu[1], Md. Sharfuddin Waseem[2,*], Arshiya Taneem[3], Sri Yashaswini Thoutam[4], Snigdha Apuri[5] and Nachiketh[6]

[1]Associate Professor, Department of Computer Science and Engineering, KITSW (Affiliated to Kakatiya University), Warangal, Telangana506015, India.

[2] Assistant Professor, Department of Computer Science and Engineering, KITSW (Affiliated to Kakatiya University), Warangal, Telangana506015, India.

[3-6] Students, Department of Computer Science and Engineering, KITSW (Affiliated to Kakatiya University), Warangal, India -506015

[*]Corresponding Author: Md. Sharfuddin Waseem. Email: waseem.cse@kitsw.ac.in

Received: 05/02/2024; Accepted:22/03/2024.

DOI: https://doi.org/10.69996/ jsihs.2024005

Abstract: Most of the damage done by web application attacks comes from SQL injection attacks, in which the attacker(s) can change, remove, and read data from the database servers. All three tenets of security— confidentiality, integrity, and availability—are vulnerable to a SQL injection attack. Database management systems receive their queries in the form of SQL (structured query language). It is not a new field of study, but it is still important to detect and prevent SQL injection attacks. A method of SQL injection detection based on machine learning is proposed. Feature extraction, followed by implementing various word embedding techniques like count vectorizer, TFIDF vectorizer to process the text data which can effectively represent the SQLI features is performed. Classification algorithms like Logistic Regression, SVM and Ensemble techniques like XGBoost is employed. Our goal in doing this systematic review is to find a better machine learning model to detect SQL injection attacks via implementing different word embedding techniques. The accuracy and F1-score of machine learning algorithms in terms of predicting the SQLI query has been calculated and reported in this research paper.

Keywords: SQL injection; machine learning; word embedding techniques; svm; logistic regression; xgboost.

1 Introduction

One of the most effective ways to steal data from a website's back end is using SQL Injection. These kinds of attacks enable hackers to get access to the database and steal vital data. According to the "Open Web Application Security Project,"[8] an injection attack is a method that can be used to access information or carry out unauthorized behavior. Today, most websites are breached using the SQL Injection attack technique. Up to now, a wide range of detection methods for the injecting question have been proposed; nevertheless, practical application is not possible. mainly because we employ three main techniques: prevention, detection, and rectification. Since none of the solutions offered offer a precise solution for preventing inserted queries and because there are numerous more mechanisms via which injection is conceivable, prevention is a little bit complicated.

The 2017 Equifax data breach, that resulted in the theft of personal information from over 147 million people, including names, birth dates, social security numbers, and addresses, is one instance of a SQL injection attack that occurred in real time. The attackers used an SQL injection attack to gain access to Equifax's system and steal sensitive information. They were able to exploit vulnerability in the web application's software by injecting malicious SQL statements into the input fields of the application. This allowed them to bypass authentication and gain unauthorized access to the database. The economic and financial losses resulting from the Equifax data breach were significant. The company estimated that it would spend over $400 million to deal with the fallout from the breach, including legal fees, customer notification costs, and cybersecurity enhancements. The company's stock price also plummeted, losing over 30% of its value in the weeks following the announcement of the breach.

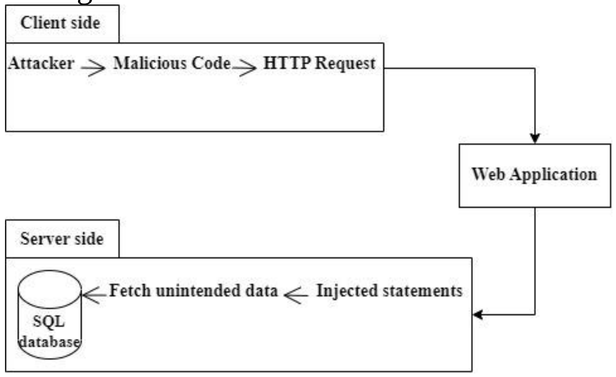


Figure 1: Flow of SQL Injection attack

Examples of malicious SQL queries and attacks that can be used to exploit security vulnerabilities in web applications:

Tautology attack: It involves the injection of SQL tokens into a conditional query statement that is always considered to be true. For example:

"SELECT * FROM employee_record WHERE emp_id = '102' and Password ='aa' OR '1'='1".

Union-based attack: This type of attack uses the UNION operator to combine the results of two or more SELECT statements. For example:

“SELECT username, password FROM users UNION ALL SELECT credit_card_number, expiration_date FROM credit_cards;”

Time-based attack: This type of attack uses time delays to extract data from the database. For example, the following query can be used to extract the first character of the admin password:

“SELECT IF (SUBSTRING (password,1,1) = ‘a’, SLEEP (10),null) FROM users WHERE username = ‘admin’;”

Blind attack: This type of attack does not return any data to the attacker, but it can be used to determine if a vulnerability exists in the web application. For example, the following query can be used to determine if the user table exists:

“SELECT * FROM users WHERE username = ‘admin’ AND 1=0;”

Piggy-Backed Queries: With the use of a query delimiter, such as ";," an unauthorized user can exploit the database in this type of attack by appending a new SQL query to the original one.

2 Related Works

SQL Injection detection is much easier with the use of machine learning algorithms. Following them, a classification system determines if the subsequent traffic is a SQL Injection or plain text. Naive Bayes Classifier, Passive Aggressive Classifier, Support Vector Machine, Convolutional Neural Network are some of the machine learning classification algorithms employed here. Finally, convolutional neural networks (CNN) were opted to be used in the detection of SQL Injection attacks [1][7].

In article [2][3][16], the author has attempted to present a variety of word embedding techniques, along with the models and techniques employed by those methods, to make them more accessible, the information presented here is by no means exhaustive and should be treated instead as an introduction. The primary focus of this study is to evaluate how well an algorithm for generating word embeddings performs. They have devised several algorithms that let us extract useful information from text data and store it as vectors in more conventional models. Initially, we talked about the traditional approaches to textual representation. The acquired results not only corroborate the hypothesis that Word2Vec performs well on very high dimensional text like web documents, but they also demonstrate that it can capture the true semantics of the web content.

Based on our findings, tokenization is the most fficient method of pre-processing data, and it also required the least amount of computational time. After examining the outcomes, Light GBM was shown to be superior to the other boosting methods, with an accuracy of 99.51 percent for SQLi and 99.59 percent for XSS [4][11].

Binh Ahn Pham, Vinitha Hannah Subburaj [5], they examine SQL injection protection methods. Machine learning algorithms detected SQLi assaults in this investigation. ML algorithms learn from data and infer interesting findings. ML algorithms detected harmful code from SQL code and human input. This research's machine learning model prevents such attacks. This research article calculates and reports the SQLi attack prediction accuracy of machine learning methods.

In machine learning classification algorithms, the Naive Bayes Classifier and Gradient Boosting Classifier are specifically applied to the problem. The proposed classifier recognizes objects by combining a Role Based Access Control mechanism with the Naive Bayes machine learning method. Using test scenarios produced by the three SQLIA attacks, the proposed model is assessed (comments, union, and tautology). The success percentage for the Naive Bayes classifier machine learning model is 92.8%. The Naive Bayes Classifier and the Gradient Boosting Classifier are used to the issue as machine learning classification techniques. The use of numerous simple classifiers to reduce error and increase precision is at the heart of the ensemble learning approach, which is thought to yield more reliable results. As a result, the ensemble learning technique known as Gradient Boosting Classifier was chosen for application to the SQL Injection classification problem [6][10][12][14].

Four Ensemble Machine Learning algorithms—GBM, AdaBoost, XGBM, and LGBM—were used. The models produce nearly error-free outcomes. LGBM has the best accuracy, precision, recall, and f1 values. The LGBM had 0.120761 FPR and 0.007 RMSE. AdaBoost has the worst accuracy, precision, recall, and f1 values. AdaBoost has a 0.009 FPR [9].

SVM is one of the most widely used algorithms in the field of machine learning, generally used for classification problems. It works admirably well, although not as well as ensemble models. It can handle many features and is effective in high-dimensional spaces. This is so that classes may be more easily separated using SVM, which transforms the input space into a higher-dimensional space using a kernel function. [13][15].

3 Proposed System

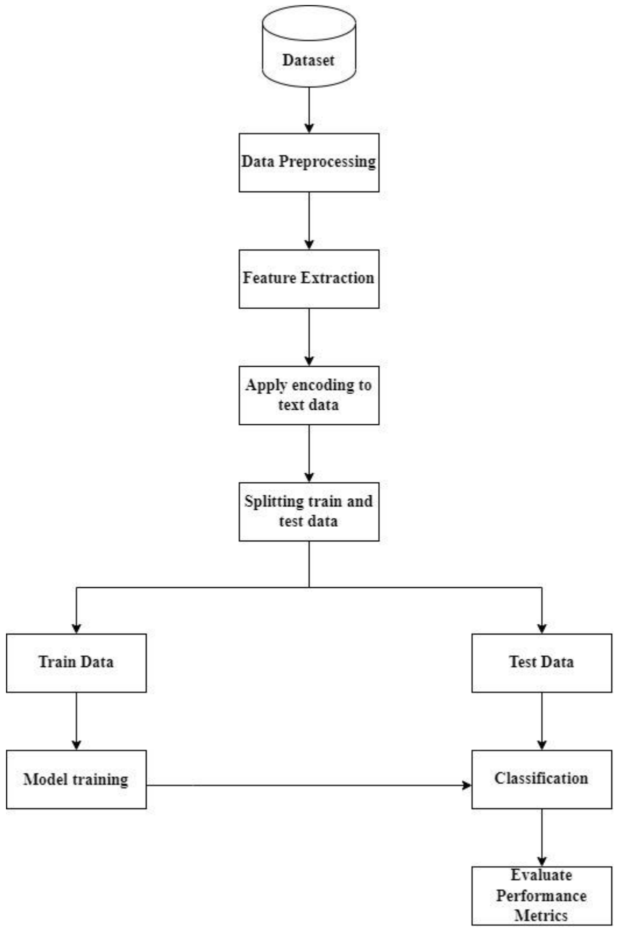
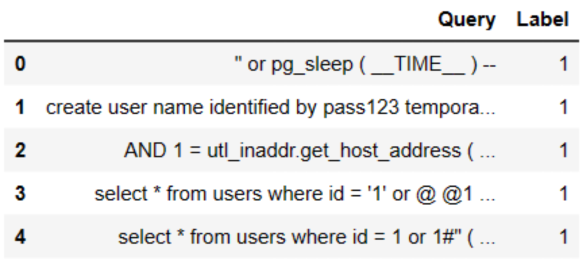


Figure 2: Methodology

DATASET

Source: Kaggle | SQL Injection

Dataset contains two columns Query, and Label. There are 30919 rows in the data set. Query column contains a combination of SQLI queries, Genuine SQL queries, and plain text. Label column contains '1' representing SQLI query and '0' representing it can be a genuine SQL query or plain text.



	Query	Label
0	" or pg_sleep (__TIME__) --	1
1	create user name identified by pass123 tempora...	1
2	AND 1 = utl_inaddr.get_host_address (...	1
3	select * from users where id = '1' or @ @1 ...	1
4	select * from users where id = 1 or 1#" (...	1

Figure 3: Overview of dataset

DATA PREPROCESSING

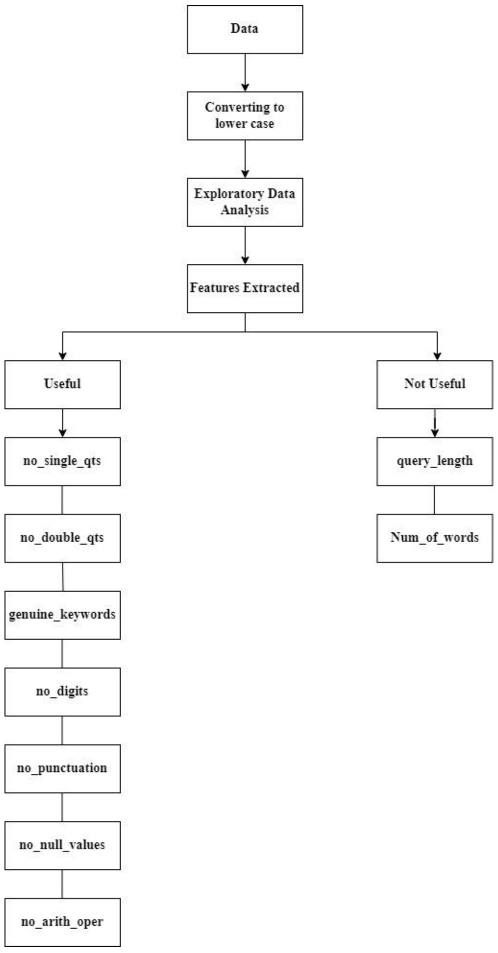
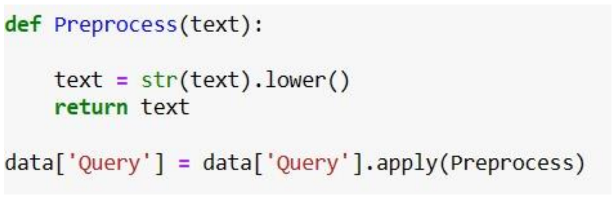


Figure 4: Data preprocessing steps

Queries are a composition of special characters, punctuations, digits, and alphabets. These are all the features that differentiate between the normal SQL query and SQLI query. So, we will not remove any null characters or punctuations as they are helpful in classification.

As the first step, we convert every character to lower-case, and perform exploratory data analysis to fetch important features present in the data followed by analyzing the important extracted features.



```
def Preprocess(text):

    text = str(text).lower()
    return text

data['Query'] = data['Query'].apply(Preprocess)
```

Figure 5: Preprocess code

We plot word clouds, which is a data visualization technique to convert text to numerical values based on the frequency of words occurring in the query.

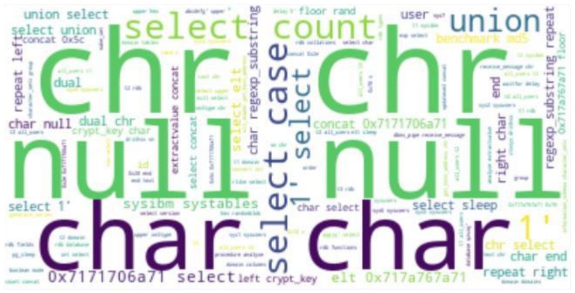


Figure 6: Keywords in Label-1

We understood that keywords like null, char, chr, select, count, and case are frequently repeated for the label-1 query.

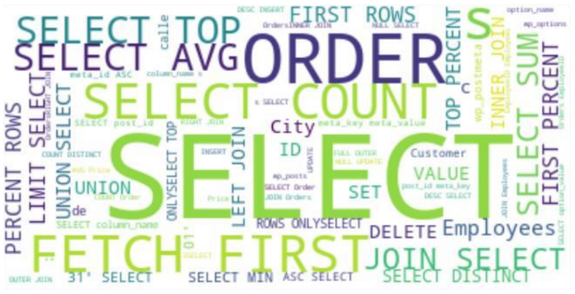


Figure 7: Keywords in Label-0

Keywords like order, top, union, limit, avg, join, and count are prominent for the label-0 query.

Extracted features = [query_len, num_word_query, no_single_qts, no_double_qts, no_punctn, no_single_cmnt, no_double_cmnt, no_white_space, no_percent, no_log_optr, no_arith_oprtr, no_null_val, no_hexdec_val, no_alphabet, no_digits, genuine_keywords]

4 Extracting Features Using Word Embedding Techniques

Count Vectorizer: Count Vectorizer, a useful function of Python's scikit-learn module. It is used to convert a text into a vector depending on how often each word appears across the whole text. This adjustment is made using the frequency analysis method. This is useful when we need to turn each word in texts into a vector and a matrix of Count Vectorizer is generated. Each column denotes each distinct word, while a row denotes each passage of text from the document.

TF-IDF Vectorizer**:** The acronym "TF-IDF" stands for "Term Frequency Inverse Document Frequency". One explanation of this term is to determine how significant a group of words or a corpus of words are to a certain text. A word's meaning develops in direct proportion to how often it appears in text, although this influence is lessened by the word's widespread use across the dataset. We import the necessary modules, in addition to Create a corpus that contains the strings you have collected from various texts after gathering them from various texts. To get the TF-IDF values, we then use the fit_transform () method.

The formula to calculate the TF-IDF score for a term in a document is:

$$TF - IDF(t,d) = TF(t,d) * IDF(t)$$

$$IDF(t) = \ loge(\frac{\text{Total number of documents}}{\text{Number of documents with term t}})$$

$$TF(t,d) = \ \frac{\text{Number of times term t appears in document d}}{\text{Total number of terms in document d}}$$

Bag-Of-Words**:** A text word occurrence frequencies can be captured by the BoW model. Bag of words does not care what order words appear. One can think of each individual word as a characteristic or a variable. Now, a feature vector will be built for each individual document. Given that there are seven distinct words, each feature vector will have seven dimensions. A group of N words in a row makes up an N-Gram. Number of words in the sequence (here denoted by the integer N). We call it a unigram, for instance, if N is equal to 1. If N=2, then it is a bigram.

5 Performance Metrics

F-1 Score:

The F1 score is the harmonic mean of recall and precision. It is typically employed for performance comparison between two classifiers. It gives us the best estimate of performance for the given models.

$$f1 - score = 2 * (\frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}) \quad (1)$$

$$Precision = \frac{\text{TP}}{\text{TP + FP}} \quad (2)$$

$$Recall = \frac{\text{TP}}{\text{TP + FN}} \quad (3)$$

Accuracy:

The performance of the model across all classes is measured by accuracy. When all classes are given the same weight, it is beneficial.

$$Accuracy = \frac{True\ positives + True\ negatives}{\text{True positives + False positives + True negatives +False negatives)}} \quad (4)$$

ALGORITHMS

We have implemented the models on Logistic Regression, SVM, and XGBoost.

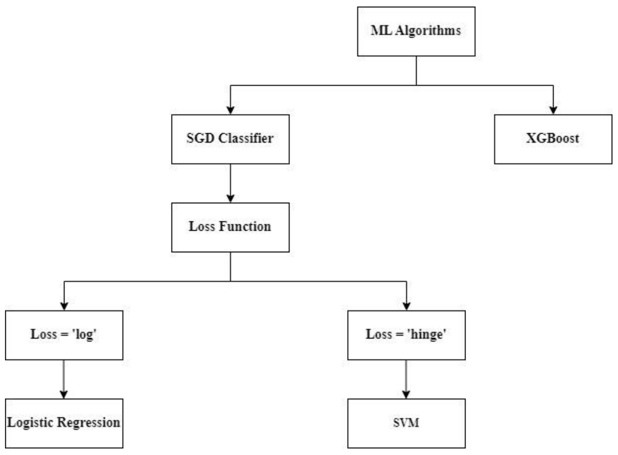


Figure 8: Machine Learning algorithms

Support Vector Machine: The Support Vector Machine (SVM) algorithm's objective is to generate the most basic classification boundary or line for n-dimensional space. This is done so that we can assign the new data to the appropriate family promptly and accurately. The main objective of choosing SVM is, it can handle many features and is effective in high-dimensional spaces and robust to outliers.

$$F(x) = sign(w^T x + b) \tag{5}$$

Where, w is the weight vector, b is the bias term, and sign() is the sign function, which returns -1 for negative values and +1 for positive values. F(x) is the predicted class label for input sample x.

Logistic regression: It ranks second in terms of popularity to the supervised learning approach. It can be used to create precise predictions about the likelihood that a given event will occur (0/1, True/False, Yes/No). Its advantages are simplicity, efficiency, robustness, and interpretability.

$$P(\frac{y=1}{x}) = \frac{1}{1+e^{-z}} \tag{6}$$

Where, p(y= 1| x) is the conditional probability of the outcome variable y being equal to 1, the exponential function is called exp(), and the linear combination of predictor variables is called z.

XGBoost: The supervised learning technique known as gradient boosting combines the forecasts of several weaker, simpler models to anticipate a target variable more correctly. The key features are high performance and speed. It is designed to be scalable, and can handle very large datasets with millions of examples and features.

SGD Classifier: A linear classifier with SGD training is called SGD-Classifier. In a nutshell, gradient descent is used to minimize the cost function. It is computationally efficient and can handle large datasets with many features, making it a good choice for big data applications. It can handle different types of data, such as numerical and categorical features, can be used for both binary and multi-class classification tasks.

Loss Function calculates the difference between an estimated value and the true value. For, Logistic Regression the loss = ‘log’ and for SVM loss = ‘hinge’

- Log Loss: It establishes the degree to which the forecast probability corresponds to the real or actual value given by log-loss.

$$logloss_i = [y_i \ln p_i + (1 - y_i) \ln (1 - p_i)] \tag{7}$$

Where n is the total number of observations, yi denotes the binary result for observation i and pi denotes the probability of observation i falling into positive class.

- Hinge Loss: Machine learning classifiers are trained using the loss function known as hinge loss. For "maximum-margin" classification, particularly for support vector machines.

$$l(y) = max(0, 1 - \mathrm{t} * \mathrm{y}) \tag{8}$$

where n denotes the total number of observations, t denotes the binary result, and y denotes the observation i predicted score.

5.1 Results

The train-test split is used to measure how well machine learning algorithms work in prediction-based algorithms and applications. We consider two different train-test split ratio data. One set with the Training ratio as 70% data, whereas the Test ratio as 30% data and another set, with 80% of the data for training and 20% are for testing.

As a result, 30% and 20% of our data are utilized for testing, and the remaining 70% and 80% are used for training data, with the test size defined as 0.3 and 0.2, respectively. Also, we set the random state to 0 so that the data is divided at random into these two datasets.

Table 1: Performance measures for 70:30 split data

Word Embedding	No.	Algorithm	F1score	Accuracy
Unigram Count vectorizer	1	LR	0.989	0.992
	2	XGB	0.992	0.994
	3	SVM	0.991	0.993
Unigram Tfidf	4	LR	0.979	0.985
	5	XGB	0.986	0.990
	6	SVM	0.980	0.985
Bigram Count vectorizer	7	LR	0.917	0.942
	8	XGB	0.915	0.942
	9	SVM	0.920	0.944
Bigram Tfidf	10	LR	0.914	0.940
	11	XGB	0.869	0.907
	12	SVM	0.919	0.944

Table 2: Performance measures for 80:20 split data

Word Embedding	No.	Algorithm	F1score	Accuracy
Unigram Count vectorizer	1	LR	0.988	0.991
	2	XGB	0.990	0.992
	3	SVM	0.988	0.991
Unigram Tfidf	4	LR	0.973	0.980
	5	XGB	0.987	0.991
	6	SVM	0.977	0.983
Bigram Count vectorizer	7	LR	0.924	0.947
	8	XGB	0.920	0.945
	9	SVM	0.927	0.949
Bigram Tfidf	10	LR	0.916	0.942
	11	XGB	0.886	0.920
	12	SVM	0.924	0.947

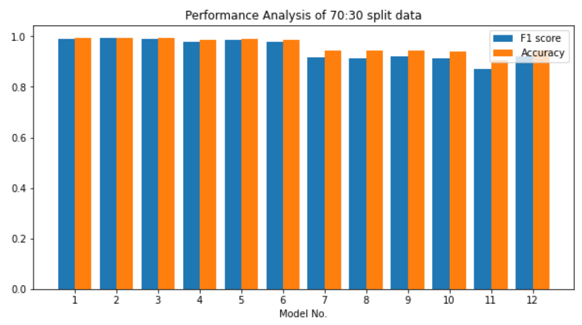


Figure 9: Performance Analysis of 70:30 split data

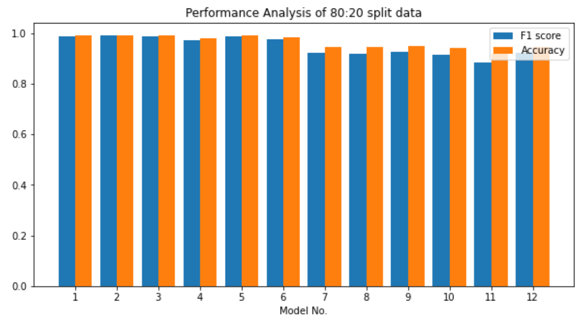


Figure 10: Performance Analysis of 80:20 split data

Table 3: Run time analysis of algorithms

Avg. Run time Analysis (in seconds)		
N-gram Model	**Unigram**	**Bigram**
LR	**1**	**1**
SVM	**1**	**1**
XGB	**3.5**	**4.5**

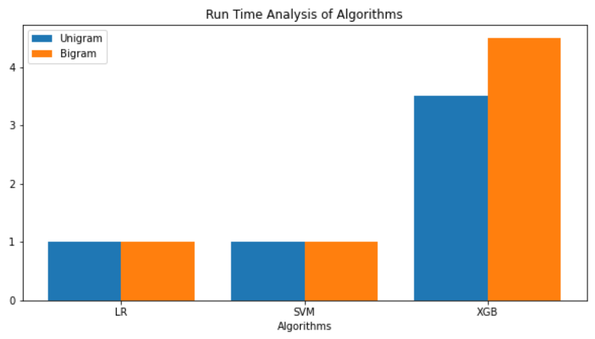


Figure 11: Analyzing run time analysis

6. Conclusion

In the modern scenario, the World Wide Web has proven to be the most dependable and popular medium for corporate information sharing and communication. The security in online applications cannot be guaranteed. Due to their accessibility, they are vulnerable to several flaws, and if these flaws are not fixed, they could have negative effects. One attack type that is simple to execute but difficult to detect is SQL Injection. This could lead to theft, the disclosure of private information, or the loss of property.

This research effort has produced a novel method for detecting SQLi attacks utilizing word encoding techniques and machine learning algorithms. Our dataset includes legitimate, fraudulent SQL queries and plain text. We suggested a reliable methodology for differentiating plain text and regular queries from SQL injection attack queries. After evaluating the results, XGBoost algorithm with a unigram count vectorizer encoding of 70:30 split data ratio, gave us the best model with an F1-score of 0.992 and accuracy of 0.994. It has greater runtime as compared with other machine learning algorithms used. As numerous simple classifiers are used, ensemble learning techniques are said to produce results with higher accuracy.

7. Future Scope

Exploratory Data Analysis (EDA) has a significant impact on how well models work. We can find different ways to analyze newer features and can draw more inferences and insights. To increase the accuracy and reliability, a larger and more diverse dataset must be used to evaluate the experimental model. In addition, a more robust and advanced methodology like deep learning would be considered, which would enhance the efficiency and performance of the generated model. We can also try with different word encoding methods like fastText, GloVe and combine them with NLP techniques and find the best model among them.

The comparison of new datasets and word embedding methods, however, will help researchers identify more trends and develop better embedding algorithms that may be more effective in text classification.

Acknowledgement: Not Applicable.

Funding Statement: The author(s) received no specific funding for this study.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

[1] S.S. Anandha Krishnan1, Adhil N Sabu, Priya P Sajan, A.L. Sreedeep "SQL Injection Detection Using Machine Learning" Revista Geintec, 2021

[2] Neha Kulkarni, Dr. Ravindra Vaidya and Dr. Manasi Bhate "A comparative study of Word Embedding Techniques to extract features from Text," *Turkish Journal of Computers and Mathematics*, 2021

[3] B.M.Ajose-Ismail, O.V. Abimbola and S.A. Oloruntoba, "Performance Analysis of Different Word Embedding Models for Text Classification," *International Journal of Scientific Research and Engineering Development*, vol. 3, no.6, 2020.

[4] P.Ravi , K.Sai Prasad, M. Lasya and N.Ram Akhilesh, "Detection of Web Attacks using Ensemble Learning," *International Research Journal of Engineering and Technology*, vol. 8, no.7, 2021

[5] Binh Ahn Pham and Vinitha Hannah Subburaj "An Experimental Setup for Detecting SQLi Attacks using Machine Learning Algorithms," *Journal of The Colloquium for Information Systems Security Education*, vol.8, no. 1, 2020

[6] Anamika Joshi and V. Geetha, "SQL Injection Detection using Machine Learning," *International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT),* Kanyakumari, India, 2014

[7] Kevin Ross "SQL Injection Detection Using Machine Learning Techniques and Multiple Data Sources," 2019 Masters Thesis, San Jose State University.

[8] The Open Web Application Security Project (OW ASP). The Ten Most Critical Web Application Security Risks 2010. https://www.owasp.org/index.php/Top_10_2013

[9] Umar Farooq “Ensemble Machine Learning Approaches for Detection of SQL Injection Attacks,” *Technical Journal* , vol.15, 2021

[10] Tareek Pattewar, Hitesh Patil, Harshada Patil, Neha Patil, Muskan Taneja et al., “Detection of SQL Injection using Machine Learning: A Survey,” *International Research Journal of Engineering and Technology*, vol.6, no.11, 2019

[11] Sonali Mishra “SQL Injection Detection Using Machine Learning” 2019 Master’s Thesis San Jose State University.

[12] Gradient Boosting https://bradleyboehmke.github.io/HOML/gbm.html

[13] Understanding Support Vector Machine (SVM) algorithm from examples- https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

[14] J. Brownlee, ‘Start With Gradient Boosting, Results from Comparing 13 Algorithms on 165 Datasets’, March 30,2018.[Online]Available: https://machinelearningmastery.com/start-with-gradient-boosting/

[15] Zhuang Chen, Min Guo and Lin zhou. “Research on SQL injection detection technology based on SVM,” *MATEC Web of Conferences*, vol. 173, no.01004,2018.

[16] S. Mumtaz, C. Rodriguez, B. Benatallah, M. Al-Banna and S. Zamanirad. “Learning Word Representation for the Cyber Security Vulnerability Domain.” The 2020 International Joint Conference on Neural Networks - IJCNN 2020, Glasgow, UK, 2020.