JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 9, No. 4, Desember 2024, Pp. 2064-2073

ANALYZING COMPARISON PERFORMANCE MODEL OF MACHINE LEARNING THROUGH DETECTION SQL INJECTION ATTACK

Rakha Satria Pratama*[1], Muhamad Irsan[2], Rio Guntur Utomo[3]
1. Information Technology, Informatics, Telkom University, Bandung
2. Information Technology, Informatics, Telkom University, Bandung
3. Information Technology, Informatics, Telkom University, Bandung

Article Info
Keywords: SQL Injection, Support Vector Machine, K-Nearest Neighbor, Logistic Regression, Machine Learning

Article history:
Received 11 September 2024
Revised 5 Oktober 2024
Accepted 14 November 2024
Available online 4 December 2024

DOI :
https://doi.org/10.29100/jipi.v9i4.781

* Corresponding author.
Corresponding Author
E-mail address:
rakhasatria@student.telkomuniversity.ac.id

ABSTRACT
This research aims to compare Machine Learning models that effectively detect SQL Injection attacks in security systems. The dataset was collected from the Kaggle resource published by Syed Saqlain Hussain Shah, the dataset with the highest upvotes in the SQL Injection category. The models developed include Support Vector Machine (SVM), K-Nearest Neighbor (KNN), and Logistic Regression (LR). The research process includes separating the data into 70% training and 30% test data, model training, testing model effectiveness, and implementing preventive measures against SQL Injection attacks. The research results show that the SVM model has an accuracy rate of 99.82%, precision of 99.88%, and recall (Sensitivity) of 99.34%. KNN obtained an accuracy rate of 79.28%, a precision of 98.38%, and a recall (Sensitivity) of 73.31%. LR obtained an accuracy rate of 98.99%, precision of 99.94%, and recall (Sensitivity) of 98.70%. Using a Machine Learning approach, this research improves system security against SQL Injection attacks.

.

I. Introduction

IData security has become the main focus of technological progress in the ever-growing digital era. One of the main challenges in a security context is combating increasingly sophisticated SQL Injection attacks, which can infiltrate systems in various ways [1]. SQL Injection attacks exploit security vulnerabilities by injecting malicious SQL code. The attack's impact was severe; attackers could easily access sensitive data in the database, manipulate, or even delete stored data [2]. This not only compromises the confidentiality of information but also the integrity and availability of data. Therefore, protection against SQL Injection attacks is essential in today's digital ecosystem.

As a concrete example, an attack exploiting a zero-day vulnerability in Progress Software's MoveIT Transfer product has highlighted the threat that SQL Injection vulnerabilities pose to organizations of all sizes. On May 31, 2023, Progress disclosed a critical SQL Injection vulnerability, tracked as CVE-2023-34362, which could allow an attacker to access an instance of MoveIT Transfer. A patch was released the same day, but security vendors soon reported widespread exploitation that began before the disclosure date. Microsoft later attributed the attack to a threat actor associated with the Clop ransomware group that it called “Lace Tempest.” Some of the victims of the data breach include HR software provider Zellis and the government of Nova Scotia, Canada [3].

SQL Injection attacks have been identified as one of the highest-risk threats by the Open Web Application Security Project [4]. Incidents related to SQL Injection are becoming more frequent, emphasizing the need for careful analysis to detect and prevent such attacks [5]. One of the main approaches in this analysis is to use machine learning algorithms to identify and anticipate potential SQL Injection attacks [6]. Therefore, a deep understanding of machine learning models is the key to effective SQL Injection attack detection [7].

Machine learning is a method of computer algorithms that mend mechanically via experience. Machine learning algorithms create a model that supports example data, mentioned as "training data", so as to form prevision or judgment requiring being explicitly programmed to attempt to so Machine learning algorithms are used in a great variety of uses, same as email filtering and computer vision, where it's hard or infeasible to create traditional algorithms to execute the required work [8].

Considered a subset ofAI, machine learning (ML) exhibits the experiential "learning" associated with human intelligence, while also having the capacity to learn and improve its analyses through the use of computational algorithms [9].

The reason for using machine learning for SQL injection is that SQL injection attacks are a major vulnerability in web applications and pose a significant threat to privacy and financial security. Traditional signature-based detection systems are no longer reliable as attackers constantly come up with new types of SQL injections. Machine learning algorithms can be trained to detect patterns and anomalies in incoming communication, allowing for the classification of SQL injection attacks versus normal text. By utilizing supervised learning methods and training the algorithms with a dataset, machine learning models can achieve high accuracy in detecting SQL injection attacks. Additionally, machine learning allows for the detection of innovative, never-before-seen attacks, making it a valuable tool in the field of cybersecurity [10].

The purpose of this research is to develop a robust detection system for SQL Injection attacks using machine learning. The main contribution is a detailed evaluation of various machine learning algorithms' performance in this specific context. This study is unique because it provides a comprehensive comparison of different models, including Support Vector Machine (SVM), Logistic Regression (LR), and K-Nearest Neighbor (KNN), in the detection of SQL Injection attacks. The findings of this research are expected to advance the field by identifying the most effective algorithms for this purpose.

Support Vector Machine (SVM) is particularly notable for its superior performance in this context. SVM is a classification algorithm that determines decision boundaries by maximizing the margin between different classes. The power of an SVM stems from its ability to learn data classification patterns with balanced accuracy and reproducibility [11]. This makes it highly effective in detecting SQL Injection attacks with high accuracy. In comparison, Logistic Regression (LR), which models the probability of class membership, and K-Nearest Neighbor (KNN), which classifies based on the proximity of data points, also show high accuracy but are slightly outperformed by SVM.

Several studies have evaluated the performance of machine learning models in detecting SQL Injection attacks. In 2022, Triloka and colleagues researched a large SQL Injection dataset consisting of 30,904 rows of data. They utilized various algorithms, including Support Vector Machine (SVM), Logistic Regression (LR), and K-Nearest Neighbor (KNN). The research results showed that SVM achieved the highest accuracy at 99.77%. LR had an accuracy of 99.60%, while KNN reached 99.70% [5]. This comparison demonstrates the superiority of SVM in detecting SQL Injection attacks.

By leveraging machine learning-based approaches, detection systems can proactively identify and mitigate attack attempts before they cause damage or compromise data integrity. This will provide better data protection and prevent potentially harmful data leaks. Implementing machine learning-based SQL Injection attack detection is crucial in securing the increasingly complex and vulnerable online environment against cyber threats. By continuously developing and improving these detection models, we can reduce the risk of attacks and ensure the safety of our data in this evolving digital era.

To enhance system security and reduce the risk of SQL Injection attacks, other factors that can affect the performance of machine learning models need to be considered. For example, selecting the right features to include in the model is crucial. These features should differentiate between normal and suspicious behavior or potential attacks. Additionally, careful data preprocessing is necessary to eliminate noise and ensure dataset cleanliness.

Furthermore, collaboration among researchers, security practitioners, and industry players is essential in addressing these complex security challenges. Exchanging information and experiences can help develop more effective and innovative solutions to combat cyberattacks. Companies are expected to implement effective and efficient SQL Injection attack detection solutions by addressing these aspects comprehensively. Thus, they can enhance the security and resilience of their systems against evolving and increasingly complex cybersecurity threats in today's digital landscape.

II. RESEARCH METHODOLOGY

This research has formed a framework that will be used as a reference for this research. Figure 1 is a framework used as a general illustration of system design in research. There are several stages of activities, such as dataset search, data splitting, data processing, prediction accuracy, Precision, Recall (Sensitivity), and evaluation.

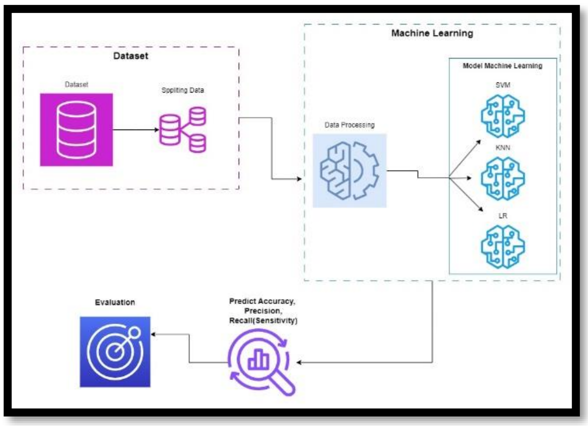
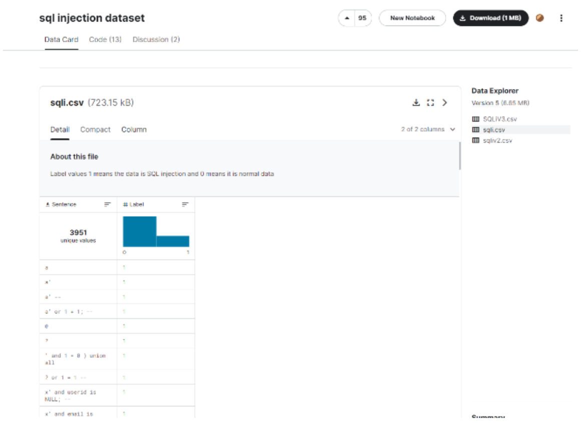


Fig. 1. System Design Framework

2.1 Dataset

This dataset was collected from Kaggle resources published by Syed Saqlain Hussain Shah [12]. It was selected based on several criteria. Firstly, the dataset aligns perfectly with the research objectives of training a security model or algorithm to detect and prevent SQL injection attacks. It contains a comprehensive collection of SQL injection scripts that cover various attack vectors, which is crucial for developing a robust detection system. Furthermore, the dataset has received the highest upvotes among similar datasets, with 95 upvotes and 6612 downloads as of November 27, 2023. This community validation indicates the dataset's reliability and quality, reflecting its acceptance and trust within the research community. Additionally, Syed Saqlain Hussain Shah, the dataset creator, is a recognized expert in the field, adding further credibility. The dataset's thorough documentation and proven utility in previous studies make it an invaluable resource for our research.

Considering these points, we believe this dataset is the best and most relevant choice for our study, ensuring the development of an effective and reliable SQL injection detection model.



About this file

Label values 1 means the data is SQL injection and 0 means it is normal data

Sentence	Label
3951 unique values	0 1
a	1
a'	1
a' --	1
a' or 1 = 1; --	1
@	1
?	1
' and 1 = 0) union all	1
? or 1 = 1 --	1
x' and userid is NULL; --	1
x' and email is	1

Fig. 2. SQL Injection Datasets

Figure 2 is an example of an SQL Injection dataset image. This dataset is in CSV format and has more than 30,000 scripts. In this dataset, there are two types of scripts: the first is a SQL Injection script, and the second is a standard script that has no potential to attack a database. This dataset has two columns: the first column is the sentence column, which contains scripts, and the second is the label column. In the label column is the number 0, which states the script is standard, and the number 1, which states the SQL Injection script can potentially attack a database. This dataset was collected from Kaggle resources published by Syed Saqlain Hussain Shah. This dataset was taken based on the highest upvotes among other SQL Injection datasets. Upvotes on this dataset reached 95 and were downloaded 6612 times when accessed on November 27, 2023. This dataset contains SQL injection scripts compiled to predict SQL injection attacks. The aim of using this dataset is to train a security model or algorithm that can detect and prevent hacking attempts using SQL injection in the system.

The following dataset will have two subordinate stages: data preprocessing and splitting. The data preprocessing process is a stage that is carried out manually. In the data preprocessing stage, essential steps are carried out, including verifying the SQL Injection dataset obtained, identifying and eliminating redundant or duplicate data entries, and resetting the data format to ensure the data is well-structured and ready for further processing. Carry on. In this phase, if the data being analyzed does not meet the standards or criteria set, a decision will be made to repeat the research process at the dataset collection stage to ensure that the quality is maintained and the analysis results are valid. Then, the data splitting stage is where the data is separated into two main parts: training data and test data. In the context of this research, the dataset will be separated by 70% of the total available data as training data, which will be used to 'teach' or 'train' the model. Meanwhile, the remaining 30% of the total data is defined as test data, which acts as a validation set to test the effectiveness of the trained model. This 70-30 split is based on best practices in machine learning research, which suggest that a larger portion of the data should be used for training to ensure the model learns effectively, while a sufficient portion is reserved for testing to evaluate the model's performance. This balance helps in preventing overfitting and ensures that the model can generalize well to new, unseen data. This separation process is essential to ensure that the model can learn from multiple examples and then test on data it has never seen before, thereby evaluating its performance more objectively and accurately.

2.2 Data processing

Preprocessing is basically implemented on any raw big data before using any kinds of classification or identification [13]. This stage is a critical stage in the research process, which is carried out after the separation of data into training data and test data has been completed. At this stage, detailed data preprocessing steps are undertaken to ensure the quality and suitability of the dataset for machine learning models. These steps include dataset verification, identifying and removing redundant data, and ensuring consistent data formatting. For verification, we check for missing values and drop any rows with null entries. We also split the dataset into balanced subsets for both training and testing, ensuring that the distribution of labels is consistent. Using the prepared SQL Injection dataset, we train and test the model using three different machine learning techniques, namely SVM, which is known for its optimal decision margin, KNN with a distance-based approach for classification; and LR, which is effective in predicting the probability of an event occurring. Each of these models will be tested to determine which provides the best performance in the context of SQL Injection attack detection, hoping to achieve the highest accuracy and reliability for real-world applications.

2.3 Predict Accuracy, Precision, Recall/Sensitivity

This stage is where the machine learning model can be measured and analyzed regarding the accuracy, precision, and recall/sensitivity of the SQL injection attack detection system. The prediction performance produced by various machine learning models such as SVM, KNN, and LR can be assessed at this stage.

SVM classifies data by defining a collection of support vectors, which are members of the named training data samples, which are based on statistical learning theory. An SVM’s primary goal is to find the best hyperplane for classifying new data points. For the classification of non-linear data samples, SVM classifiers may use a variety of kernel functions such as Linear, polynomial, Gaussian radial basis function (RBF), and sigmoid [14].

K-Nearest Neighbor (KNN) classification is a method used in clustering or grouping a variable into a certain group. It will use their similarity of a variable to classify them in a group. To use this method, k value has to be specified and the result of the calculation will group the number into the nearest k value. Sensor technology is most beneficial when using the KNN [15].

Logistic regression is used to estimate the association of one or more independent (predictor) variables with a binary dependent (outcome) variable.2 A binary (or dichotomous) variable is a categorical variable that can only take 2 different values or levels, such as “positive for hypoxemia versus negative for hypoxemia” or “dead versus alive.” [16] .

This stage is carried out to determine the system's effectiveness in recognizing and distinguishing SQL Injection attacks with a high success rate. By comparing the prediction results produced by each model, we can determine which model is the most accurate, has the highest precision, and has optimal recall/sensitivity in detecting SQL Injection attacks.

2.4 Evaluation

This evaluation stage compares the predicted results and the actual labels in the test data. The evaluation metrics such as accuracy, precision, and recall are calculated using specific formulas. Accuracy is calculated as (TP + TN) / (TP + TN + FP + FN), precision is calculated as TP / (TP + FP), and recall is calculated as TP / (TP + FN), where TP, TN, FP, and FN stand for true positives, true negatives, false positives, and false negatives, respectively [17]. The accuracy, precision, and recall/sensitivity level obtained in this system are calculated to measure the performance of SQL Injection attack detection. The hope at this stage is that the SQL Injection attack detection system can correctly detect that the input string matches the labels in the dataset. If the results differ from expected, then this research will be carried out again using data processing.

At this evaluation stage, several diagrams will be presented to make visualization easier when comparing the three models (SVM, KNN, LR) that will be tested. Below are some diagrams that will be displayed:

1. Line diagram: A line diagram is a type of data visualization used to show the relationship or trend between two different variables along the x and y axes [18]. In a line diagram, data points are connected by straight lines, which form a line pattern that depicts changes in variable values as other variables change [19].
2. Confusion Matrix: A confusion matrix is a powerful tool for evaluating performance by quantifying classification overlap. The confusion matrix results are compared with statistics from current techniques to demonstrate their effectiveness in providing a concise and clear understanding of multi-label classification behavior [20].
3. Receiver Operating Characteristic (ROC): ROC helps measure the sensitivity and specificity of predictors, which is a critical component in considering accuracy. ROC is generally used to evaluate predictor accuracy in medicine, radiology, engineering, data mining, and machine learning [21].

Using a combination of line charts, confusion matrix, and ROC, this researcher can comprehensively understand the performance of SVM, LR, and KNN models in predicting data. These methods provide helpful information for comparing the accuracy, precision, and recall (sensitivity) of these models, thus enabling the selection of the most appropriate model for the data analysis [10]. In particular, this research compares the performance of SVM, LR, and KNN models in detecting SQL Injection attacks.

III. RESULTS AND DISCUSSION

This research was carried out using the system design in part 1. This research includes stages such as dataset search, data preprocessing, data separation, data processing, predictive accuracy, Precision, Recall/Sensitivity, and evaluation or testing.

3.1 Dataset

This dataset process includes checking the entire dataset to ensure each SQL Injection script is correctly identified according to its label. This step is crucial to ensure the accuracy of the labels in each SQL Injection script and to prevent errors during data processing. By checking manually, you can ensure that all SQL Injection scripts have been classified correctly according to their characteristics.

This manual process involves carefully analyzing each entity of the dataset and comparing it against predefined criteria for the SQL Injection script. This helps ensure that the datasets used in analysis and testing are safe from SQL Injection attacks and minimizes security risks for applications to be developed. After the data preprocessing stage, the dataset carried out in this research is implemented into Google Drive storage to make it easier to import data into the system that is being built.

3.2 Splitting Data

Training and test data are separated into the SQL injection dataset at this stage. The training data used in this research was 70%, while the test data used was 30% of the entire dataset. In this process, scripts that do not have the potential for SQL Injection attacks, or scripts with a label of 0, are separated into 70% training data and 30% test data. Similarly to scripts that have a label of 0, scripts with a label of 1, which can be called scripts with the potential to cause SQL Injection attacks, are separated into 70% training data and 30% test data.

3.3 Data Preprocessing

To obtain accuracy, precision, and recall from models such as Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Logistic Regression (LR), a series of steps are carried out. First, a model object corresponding to the desired algorithm is created by setting the relevant parameters, such as the kernel type or the number of nearest neighbors. Next, the model is trained using the training data and appropriate labels. After training the model, predictions are made on the test data, and the results are compared with the actual labels to calculate accuracy. Accuracy measures how often the model predictions match the actual labels. In addition to accuracy, precision is calculated as the proportion of correctly predicted positives out of all positively predicted examples. Recall, also known as sensitivity, measures the proportion of correctly predicted positives out of all positive examples. These steps are applied to each model, and the results are printed on the screen for evaluation.

3.3.1 Predict Accuracy, Precision, Recall/Sensitivity

This stage is for predicting accuracy, precision, and recall (sensitivity). In the previous stage, data processing was carried out to obtain levels of accuracy, precision, and recall (sensitivity). We configure and train different machine learning models to detect SQL Injection attacks. The models used in this research are SVM, KNN, and Logistic Regression (LR). Each model has specific parameters that were tuned to achieve optimal performance:

1. Support Vector Machine (SVM): We used a linear kernel (kernel='linear') with a regularization parameter C=1.0. The linear kernel is chosen for its simplicity and effectiveness in high-dimensional spaces. The gamma parameter is set to 'scale', which means it uses $1/(n_{\text{features}} \times \mathrm{X.var()})$ as value, where X is the input data.
2. K-Nearest Neighbors (KNN): We configured the model with n neighbors=5, which means the algorithm considers the 5 nearest neighbors to make a classification decision. The distance metric used is the Euclidean distance, which is the default setting in KNN.
3. Logistic Regression (LR): For LR, we used the solver='lbfgs', which is an optimizer in the family of quasi-Newton methods. The max_iter parameter is set to 100 to ensure convergence.

The following are the results obtained.

TABLE I
PERFORMANCE COMPARISON OF SVM, KNN, LR

	SVM	KNN	LR
Akurasi	99.82%	79.28%	98.99%
Presisi	99.88%	98.38%	99.94%
Recall(Sensitivity)	99.34%	73.31%	98.70%

Table 1 shows that the level of accuracy obtained by the SVM model for detecting SQL Injection attacks reached 99.82%. The precision level obtained by the SVM model for detecting SQL Injection attacks reached 99.88%, and the Recall Rate (Sensitivity) obtained by the SVM model for detecting SQL Injection attacks reached 99.34%.

Apart from that, it can be seen that the level of accuracy obtained by the KNN model for detecting SQL Injection attacks reached 79.28%. The KNN model's precision for detecting SQL Injection attacks reached 98.38%, and the Recall Rate (Sensitivity) obtained by the KNN model for detecting SQL Injection attacks reached 73.31%. Likewise, the accuracy level obtained in the LR model for detecting SQL Injection attacks reached 98.99%. The precision level obtained by the LR model for detecting SQL Injection attacks reached 99.94%, and the Recall Rate (Sensitivity) obtained by the LR model for detecting SQL Injection attacks reached 98.70%.

This evaluation stage is the final stage to compare the performance of the SVM, KNN, and LR models. After completing the entire series of stages, the following is the testing stage carried out to detect SQL Injection attacks. The main idea of this testing stage is that a string is entered into the is_sql_injection(model, vectorizer, text) function, and the function must be able to produce appropriate results. In text1, a string has been entered for testing, which could potentially cause an SQL Injection attack, and in text2, a regular string has been entered or one that does not have the potential to cause an SQL Injection attack. When the function is called using the parameters of the three models that have been trained, the function produces output that meets expectations. The hope in question is that this function can recognize that text1 is a string that has the potential for SQL Injection attacks and text2 is a string that does not have the potential for SQL Injection attacks.

This evaluation stage was carried out to compare the performance of the three SVM, KNN, and LR models, in addition to testing a string. The following are the performance comparison results obtained.

3.4 Evaluation

This evaluation stage is the final stage to compare the performance of the SVM, KNN, and LR models. After completing the entire series of stages, the following is the testing stage carried out to detect SQL Injection attacks. The main idea of this testing stage is that a string is entered into the is_sql_injection (model, vectorizer, text) function, and the function must be able to produce appropriate results. In text1, a string has been entered for testing, which could potentially cause an SQL Injection attack, and in text2, a regular string has been entered or one that does not have the potential to cause an SQL Injection attack. When the function is called using the parameters of the three models that have been trained, the function produces output that meets expectations. The hope in question is that this function can recognize that text1 is a string that has the potential for SQL Injection attacks and text2 is a string that does not have the potential for SQL Injection attacks. Apart from testing a String, this evaluation stage was carried out to compare the performance of the three SVM, KNN, and LR models. Analysis of these results reveals that SVM performs better than KNN and LR for several reasons. SVM excels due to its ability to maximize the margin between classes, which enhances its generalizability and effectiveness on non-linear data through the kernel trick. KNN struggles with large datasets due to computational inefficiencies and sensitivity to data distribution, while LR's linear assumption limits its performance on non-linear patterns.

For the KNN model, false negatives occur when it fails to identify positive cases. This can be attributed to several factors such as the choice of the number of neighbors (K), the distance metric used, or dataset characteristics unsuitable for distance-based classification approaches. Additionally, KNN's performance can suffer due to computational inefficiencies with large datasets and sensitivity to data distribution.

On the other hand, Logistic Regression (LR) faces challenges due to its linear assumption, which limits its performance on non-linear patterns. LR may also exhibit a higher number of false negatives, especially if the decision threshold is not appropriately set or if the data contains multicollinearity among predictor variables. Furthermore, outliers in the data can significantly affect the performance of Logistic Regression. Regularization techniques such as Ridge or Lasso Regression can help mitigate some of these issues by reducing the impact of multicollinearity and improving the model's generalizability. Adjusting the decision threshold and using techniques like polynomial features or interaction terms can enhance the model's ability to capture non-linear relationships. The following are the performance comparison results obtained.

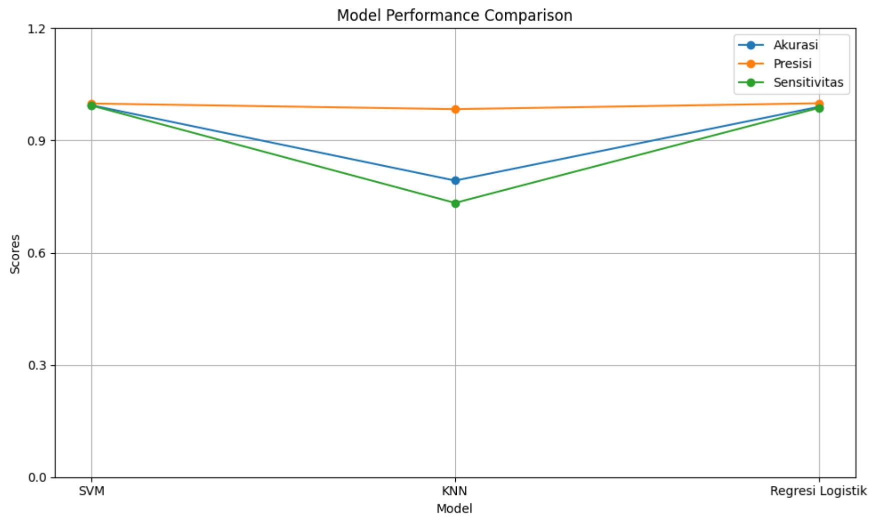


Fig. 3. Performance Charts

Figure 3 displays a line graph comparing the performance of three classification models, namely Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Logistic Regression, in three main evaluation metrics: accuracy, precision, and sensitivity (recall). Analysis of these graphs reveals several significant findings. First, SVM stands out with consistent performance across all three metrics. This shows that SVM achieved a good balance between accuracy, precision, and sensitivity, indicating the model's ability to classify data thoroughly without excessive tendencies on precision or recall alone. On the other hand, KNN shows lower values for all metrics compared to SVM and Logistic Regression. These results indicate that KNN may need help classifying data effectively, which could be caused by various factors such as the choice of number of neighbors (K), the distance metric used, or dataset characteristics unsuitable for distance-based classification approaches. Finally, logistic Regression shows satisfactory performance, especially regarding high precision. This indicates that the Logistic Regression model has a low false positive rate, although the sensitivity is not as good as SVM and is still competitive. Thus, this analysis provides valuable insight into the strengths and weaknesses of each model in dealing with a given classification task.

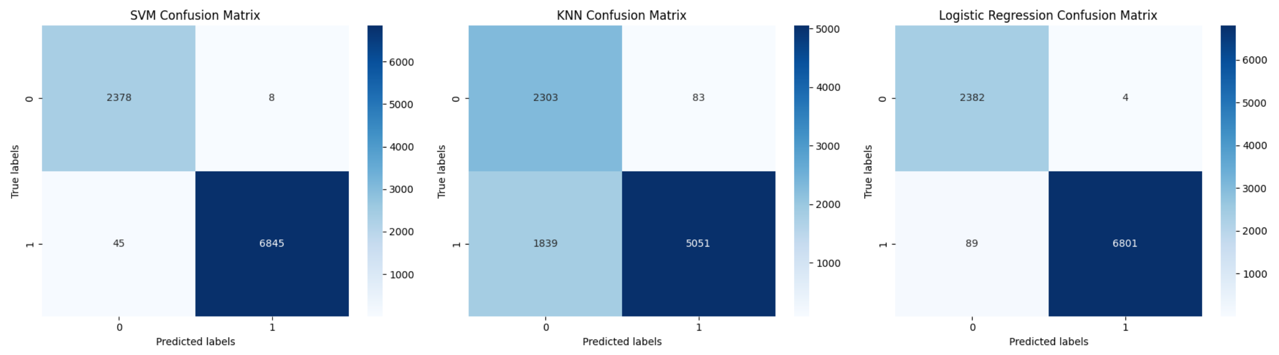


Fig. 4. Confusion matrix diagram

Figure 4 displays a Confusion Matrix that provides details of predicted versus actual labels, noting true positives, false positives, false negatives, and true negatives. This matrix provides a visual representation of the performance of the classification model, breaking down true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

For the SVM model, the Confusion Matrix shows:

a. True Negatives (TN): 2378
b. False Positive (FP): 8
c. False Negatives (FN): 45
d. True Positives (TP): 6845

SVM showed strong performance with a high number of true positives and true negatives, indicating high sensitivity and specificity. The deficient number of false positives and false negatives indicates good accuracy and recall of the model.

For the KNN model, the Confusion Matrix shows:

a. True Negatives (TN): 2303
b. False Positive (FP): 83
c. False Negatives (FN): 1839
d. True Positives (TP): 5051

The KNN model had more false negatives and false positives than SVM, indicating a lack of accuracy in classifying negative and positive cases. This model encounters difficulties, especially in the positive class, as indicated by the high number of false negatives.

For the Logistic Regression model, the Confusion Matrix shows:

a. True Negatives (TN): 2382
b. False Positive (FP): 4
c. False Negatives (FN): 89
d. True Positives (TP): 6801

Logistic Regression shows strong performance, very similar to SVM, with a very low number of false positives and relatively low false negatives. low. This indicates a strong balance between precision and recall, almost comparable to the SVM model, with possibly a slight edge in terms of precision due to the lower number of false positives.

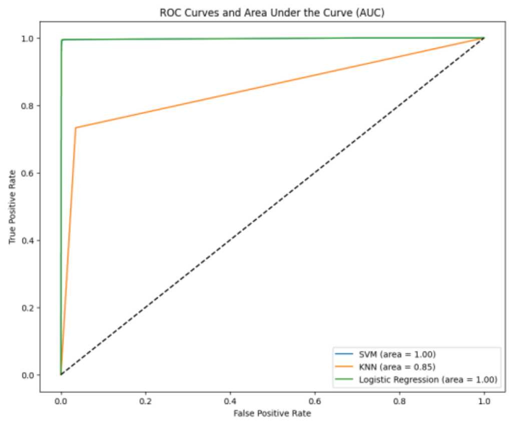


Fig. 5. ROC diagram

Logistic Regression shows strong performance, almost comparable to SVM, with a deficient number of false positives and a relatively low number of false negatives. This indicates a strong balance between precision and recall, with a possible slight edge in precision due to the lower number of false positives.

Figure 5 displays the Receiver Operating Characteristic (ROC) Curve and Area Under the ROC Curve (AUC), which are useful evaluation methods for comparing classifier performance. The ROC curve visualizes the true positive rate against the false positive rate at various thresholds, while the AUC presents the classifier's performance in a single number.

Three models, namely SVM, KNN, and Logistic Regression, were evaluated using ROC curve plots. The ROC curve provides a graphical representation of the capabilities of a binary classifier system when the discrimination threshold varies. From this evaluation, the results obtained are as follows:

a. The SVM model's area under the curve (AUC) was 1.00, indicating excellent performance with clear separation between positive and negative classes. The ROC curve follows the left limit and then the upper limit of the ROC space, indicating that the SVM has an excellent separability measure.

b. The KNN model showed an AUC of 0.85, indicating good performance distinguishing between positive and negative classes, although not as good as SVM. The ROC curve for KNN is lower than that of SVM, indicating a good but not perfect separation measure.

c. Logistic Regression also shows an AUC of 1.00, indicating it performs as well as SVM in terms of overall performance. The ROC Curve for Logistic Regression also follows the upper and left bounds, indicating high True Positive and low False Positive rates at various thresholds.

d. However, it should be noted that the blue curve that represents the SVM model is not visible in the plot. This happens because the AUC value is the same between the SVM and Logistic Regression models, so the blue curve belonging to SVM is overwritten by the green curve belonging to Logistic Regression. Nevertheless, the ROC and AUC evaluation still provide a clear picture of the relative performance of the three models.

The findings of this research show that SVM and Logistic Regression models have excellent performance in detecting SQL Injection attacks, with perfect area under the curve (AUC). Integration of these models into security systems can significantly improve detection capabilities. For example, these models can be implemented as part of a web application firewall that filters and checks each user input before it is sent to the database. In addition, the KNN model, although not as good as SVM and LR, can still be used in scenarios where computing resources are limited and speed of execution is preferred. With proper implementation, these three models can significantly contribute towards strengthening the security layer in detecting and preventing SQL Injection attacks.

IV. CONCLUSION

This research compares accuracy, precision, and recall (sensitivity). After conducting research, the results obtained were that the SVM model had a superior level of performance compared to LR and KNN. Compared with research conducted by Triloka and his colleagues in 2022, the accuracy level obtained was 0.05% higher, although the SVM model was still superior to the LR and KNN models. In previous research conducted by Triloka and his colleagues, no research was carried out regarding the level of precision and recall (sensitivity). The test carried out in this research was the creation of a function that had parameters for the machine learning model that had been trained. So, the model must determine whether the input entered has the potential for an SQL injection attack to occur.

In research that has been carried out to compare the performance of the SVM, KNN, and LR models for detecting SQL Injection attacks, results have been obtained for the SVM model, which has an accuracy level of 99.82%, precision of 99.88% and recall (Sensitivity) of 99.34%. KNN obtained an accuracy rate of 79.28%, a precision of 98.38%, and a recall (Sensitivity) of 73.31%. LR obtained an accuracy rate of 98.99%, precision of 99.94%, and recall (Sensitivity) of 98.70%. KNN may show lower performance compared to SVM and Logistic Regression due to its sensitivity to high-dimensional data, the presence of noise, and the need for feature scaling, which may impact its ability to accurately classify examples when the features in the dataset do not contribute equally to distance metric.

The superior performance of SVM can be attributed to its ability to handle high-dimensional spaces effectively, its use of kernel tricks to manage non-linear data splitting, and its focus on maximizing the margin between classes, which helps achieve better generalization. This makes SVMs more adept at finding firm decision boundaries, especially in complex data sets, leading to higher accuracy, precision, and sensitivity scores in scenarios where the characteristics of the data match the strengths of the SVM.

Implementation of the prevention strategy to detect SQL Injection attacks has been carried out by creating a function with parameters for the machine learning model that has been trained. It is just that the case study carried out in this research could only detect SQL Injection attacks based on the input column of an application or website.

REFERENCES

[1] Jumanto *et al.*, "Optimizing Support Vector Machine Performance for Parkinson's Disease Diagnosis Using GridSearchCV and PCA-Based Feature Extraction," *Journal of Information Systems Engineering and Business Intelligence*, vol. 10, no. 1, pp. 38–50, Feb. 2024, doi: 10.20473/jisebi.10.1.38-50.
[2] M. Hasan, Z. Balbahaith, and M. Tariqu, "2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA).," *International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, pp. 1–6, 2019.
[3] A. Waldman, "MoveIT Transfer Attacks Highlight SQL Injection Risks," Tech Target. Accessed: May 26, 2024. [Online]. Available: https://www.techtarget.com/searchsecurity/news/366541006/MoveIT-Transfer-attacks-highlight-SQL-injection-risks, 2023.
[4] M. Shachi, N. Siddiqui Shourav, A. Syeed, S. Ahmed, A. A. Brishty, and N. Sakib, "A Survey on Detection and Prevention of SQL and NoSQL Injection Attack on Server-side Applications," *Int J Comput Appl*, vol. 183, no. 10, pp. 975–8887, 2021.
[5] J. Triloka, H. Hartono, and S. Sutedi, "Detection of SQL Injection Attack Using Machine Learning Based On Natural Language Processing," *International Journal of Artificial Intelligence Research*, vol. 6, no. 2, Aug. 2022, doi: 10.29099/ijair.v6i2.355.
[6] Z. C. Su, S. Hlaing, and M. Khaing, "A Detection and Prevention Technique on SQL Injection Attacks," *A Detection and Prevention Technique on SQL Injection Attacks*, 2020.
[7] M. A. Kausar, M. Nasar, and A. Moyaid, "SQL injection detection and prevention techniques in ASP.NET web application," *International Journal of Recent Technology and Engineering*, vol. 8, no. 3, pp. 7759–7766, Sep. 2019, doi: 10.35940/ijrte.C6319.098319.
[8] S. S. A. Krishnan, A. N. Sabu, ; Priya, P. Sajan, and ; A L Sreedeep, "SQL Injection Detection Using Machine Learning," 2021.
[9] J. M. Helm *et al.*, "Machine Learning and Artificial Intelligence: Definitions, Applications, and Future Directions," *Curr Rev Musculoskelet Med*, vol. 13, no. 1, pp. 69–76, Feb. 2020, doi: 10.1007/s12178-020-09600-8.
[10] W. Li and Q. Guo, "Plotting receiver operating characteristic and precision–recall curves from presence and background data," *Ecol Evol*, vol. 11, no. 15, pp. 10192–10206, Aug. 2021, doi: 10.1002/ece3.7826.
[11] A. P. Derek and M. S. David, "Support vector machine," *Machine Learning: Methods and Applications to Brain Disorders*, pp. 101–121, Jan. 2019, doi: 10.1016/B978-0-12-815739-8.00006-7.
[12] S. S. H. Shah, "Kaggle." Accessed: Nov. 24, 2023. [Online]. Available: https://www.kaggle.com/
[13] A. Rahman, "Based Data Preprocessing Methods and Machine Learning Algorithms for Big Data Analysis," 2019.
[14] M. A. Almaiah *et al.*, "Performance Investigation of Principal Component Analysis for Intrusion Detection System Using Different Support Vector Machine Kernels," *Electronics (Switzerland)*, vol. 11, no. 21, Nov. 2022, doi: 10.3390/electronics11213571.
[15] H. H. M. Zuraini, W. Ismail, R. Hendradi, and A. Justitia, "Students Activity Recognition by Heart Rate Monitoring in Classroom using K-Means Classification," *Journal of Information Systems Engineering and Business Intelligence*, vol. 6, no. 1, p. 46, Apr. 2020, doi: 10.20473/jisebi.6.1.46-54.
[16] P. Schober and T. R. Vetter, "Statistical Minute Logistic Regression in Medical Research," 2021. [Online]. Available: www.anesthesia-analgesia.org365
[17] T. J. Lawrence *et al.*, "AmPEPpy 1.0: A portable and accurate antimicrobial peptide prediction tool," *Bioinformatics*, vol. 37, no. 14, pp. 2058–2060, Jul. 2021, doi: 10.1093/bioinformatics/btaa917.
[18] S. R. Midway, "Principles of Effective Data Visualization," *Patterns*, vol. 1, no. 9, Dec. 2020, doi: 10.1016/j.patter.2020.100141.
[19] C. Hayat and I. A. Soenandi, "Hybrid Architecture Model of Genetic Algorithm and Learning Vector Quantization Neural Network for Early Identification of Ear, Nose, and Throat Diseases," *Journal of Information Systems Engineering and Business Intelligence*, vol. 10, no. 1, pp. 1–12, 2024, doi: 10.20473/jisebi.10.1.1-12.
[20] M. Heydarian, T. E. Doyle, and R. Samavi, "MLCM: Multi-Label Confusion Matrix," *IEEE Access*, vol. 10, pp. 19083–19095, 2022, doi: 10.1109/ACCESS.2022.3151048.
[21] A. J. Bowers and X. Zhou, "Receiver Operating Characteristic (ROC) Area Under the Curve (AUC): A Diagnostic Measure for Evaluating the Accuracy of Predictors of Education Outcomes," *J Educ Stud Placed Risk*, vol. 24, no. 1, pp. 20–46, Jan. 2019, doi: 10.1080/10824669.2018.1523734.