Journal of Artificial Intelligence and Engineering Applications

Website: https://ioinformatic.org/

15th October 2024. Vol. 4. No. 1; e-ISSN: 2808-4519

Enhancing SQL Injection Attack Detection Using Naïve Bayes and SMOTE Method on Imbalanced Datasets

Adam Arnap[1]*, Kusrini[2]

[1, 2]Program Magister Teknik Informatika, Universitas Amikom Yogyakarta
adam.arnap.s2@students.amikom.ac.id[1], kusrini@amikom.ac.id[2]*

Abstract

SQL injection attack detection is a crucial aspect of cybersecurity, considering the potential damage that such attacks can cause. This study aims to evaluate the effectiveness of the Naive Bayes model in detecting SQL injection attacks on an imbalanced dataset. To address the data imbalance issue, the SMOTE (Synthetic Minority Over-sampling Technique) method was applied. The study consists of two phases: first, training and testing the Naive Bayes model on the original dataset without SMOTE, and second, training and testing on the dataset with SMOTE applied. The results indicate that the Naive Bayes model on the dataset without SMOTE achieved an accuracy of 0.9948, F1 Score of 0.9885, Precision of 0.9906, and Recall of 0.9946. After applying SMOTE, the model's performance improved significantly, with an accuracy of 0.9950, F1 Score of 0.9950, Precision of 0.9950, and Recall of 0.9950. This improvement suggests that SMOTE effectively enhanced class balance in the dataset, improving the model's ability to detect both malicious and safe queries.

Keywords: *SQL Injection Detection, Naive Bayes Model, Data Imbalance, SMOTE (Synthetic Minority Over-sampling Technique), Machine Learning*

1. Introduction

SQL injection is a cybercrime that targets a website. This issue remains a significant challenge in the field of cybersecurity that needs to be addressed [1]. SQL Injection is an attack with relatively simple techniques and is easily understood with many resources available on the Internet [2]. SQL Injection attacks can originate from input forms on websites where the form validation is inadequate. Attackers achieve their goals by inserting modified SQL Queries. SQL statements also allow attackers to gain administrator rights over the database, enabling them to add, edit, or delete data without any hindrance [2]. Code Injection is a highly popular and impactful attack, ranking third in the OWASP Top 10 Security Risks [3].

Methods have been developed to detect SQL Injection attacks in order to prevent such attacks on applications or websites. These methods include the use of machine learning and deep learning models. Machine Learning algorithms create models based on example data, known as "training data," to make predictions. Various Machine Learning methods for intrusion detection have recently developed, including Naïve Bayes Classifier and Support Vector Machine (SVM) [4]. In the study titled "Optimization of Naïve Bayes With Particle Swarm Optimization for Sentiment Analysis of Jakarta E-Prix," the Naïve Bayes machine learning model was used to analyze sentiment (text data) of the Formula E-Jakarta with results showing an accuracy of 89.16%, precision of 91.10%, and recall of 86.81% [5]. Machine learning models require datasets for training. This training process allows the model to recognize whether a query falls into the category of harmful or not. Once the model has recognized harmful query types through training with the training data, it can be used to detect SQL Injection attacks on previously unseen data.

The role of the dataset in the machine learning model training process is crucial, as the quality of the dataset directly impacts the performance evaluation results of the model used. In this study, the dataset consists of 4,200 rows of SQL queries. In Figure 1, out of the 4,200 rows of SQL queries in the dataset, there are 1,128 rows with label 1 and 3,072 rows with label 0. Label 1 indicates rows of SQL queries that are Malicious Queries, while label 0 indicates rows of SQL queries that are Harmless Queries. The dataset used in this research has an imbalanced ratio of label 0 to label 1. This label imbalance (class imbalance) is detrimental to machine learning models because they struggle to classify minority classes (labels with fewer instances compared to others) [6].

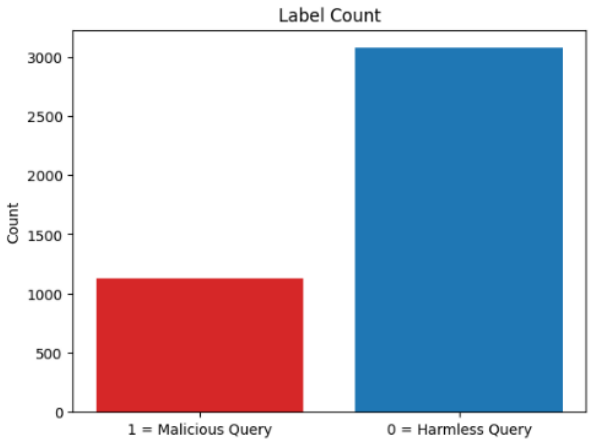

Fig. 1: Number of dataset labels used

Data imbalance is a common challenge in solving classification problems [7]. This imbalance occurs when the number of observations in the training data for each class label is uneven, resulting in one class having significantly more data compared to others [8]. In the use of machine learning models or other learning models, it is crucial to pay attention to the class or label proportions in the dataset. Several previous studies have focused on discussing and proposing methods to address issues related to dataset imbalance for training machine learning models. Data imbalance handling methods include comparing undersampling (reducing the majority class), oversampling (increasing the number of samples from the minority class), and Synthetic Minority Over-sampling Technique (SMOTE) to address data imbalance in data mining. Results indicate that random oversampling and hybrid resampling work best [9]. The SMOTE method for addressing data imbalance was visually inspected, with the proposed data imbalance handling methods showing that it not only surpasses recent pixel-based oversampling and GAN algorithms but also offers exceptional resilience to various imbalance ratios with high model stability and produces high-quality synthetic images [10]. The SMOTE method used to address the KEEL, breast cancer, and Z-Alizadeh sani datasets, which were imbalanced for use in Convolutional Neural Networks, achieved results indicating that the technique of handling imbalanced datasets with SMOTE and CNN normalization combined outperformed various other methods, achieving 99.08% accuracy on 24 imbalanced datasets [8].

The application of Feature Forward Selection and SMOTE to predict undergraduate students' performance using the Naïve Bayes model concluded that implementing feature forward selection and SMOTE improved the Naïve Bayes model, with results showing Accuracy of 87.13%, Recall of 83.82%, and Precision of 89.76% [11]. Combining the Naïve Bayes model with SMOTE to address data imbalance yielded an accuracy of 94%, compared to 36% with the Naïve Bayes model without SMOTE [12]. In previous studies, the SMOTE method was compared with the ADASYN method for handling data imbalance. Results showed that the SMOTE method, when tested with classification methods, effectively managed the number of majority (negative) and minority (positive) classes in imbalanced data, achieving MCC of 0.64 and Gmean of 0.74, indicating better prediction performance compared to using classification methods alone or using the ADASYN method with MCC of 0.63 and Gmean of 0.73 [13]. The SMOTE method has also been used in previous research to handle imbalanced image datasets, concluding that SMOTE can balance training data from: 300 genuine shoe images and 150 fake shoe images, to: 300 genuine shoe images and 300 fake shoe images [14]. The application of a CNN model combined with SMOTE to predict the authenticity of Converse shoes from images achieved an accuracy of 84% [14]. In previous research, the C4.5 model combined with the SMOTE method demonstrated that SMOTE could improve accuracy results to 86% [7].

In this study, the researcher will focus on SQL injection attack detection. This research will use a SQL Injection dataset with an imbalanced label or class ratio. Based on the background and findings from previous literature discussing methods for addressing imbalance cases in datasets and machine learning models used for data prediction, the researcher will conduct experiments using the Naïve Bayes model to predict whether queries are malicious or not. Given the imbalanced label or class proportions in the SQL Injection dataset for research purposes, the researcher will apply the SMOTE method with the hope of improving the accuracy of the Naïve Bayes model. To measure whether the SMOTE method effectively improves the accuracy of the Naïve Bayes model, the researcher will compare the accuracy of the Naïve Bayes model without SMOTE to the accuracy of the Naïve Bayes model combined with the SMOTE method. This research is expected to provide a deeper understanding of applying the Naïve Bayes model and SMOTE techniques in detecting SQL injection attacks, as well as enhance expertise in handling imbalanced datasets. For future researchers, this study is expected to serve as a methodological reference and basis for further development, as well as a comparative study to evaluate the effectiveness of other methods.

2. Research Method

In this study, the author outlines the methodological steps used to achieve the research objectives. The research process begins with a literature review and data collection focused on the theme of SQL injection attack detection. Subsequent stages are designed to ensure favorable results and alignment with the research objectives. To facilitate understanding of the applied methodology flow, please refer to Figure 2 below.

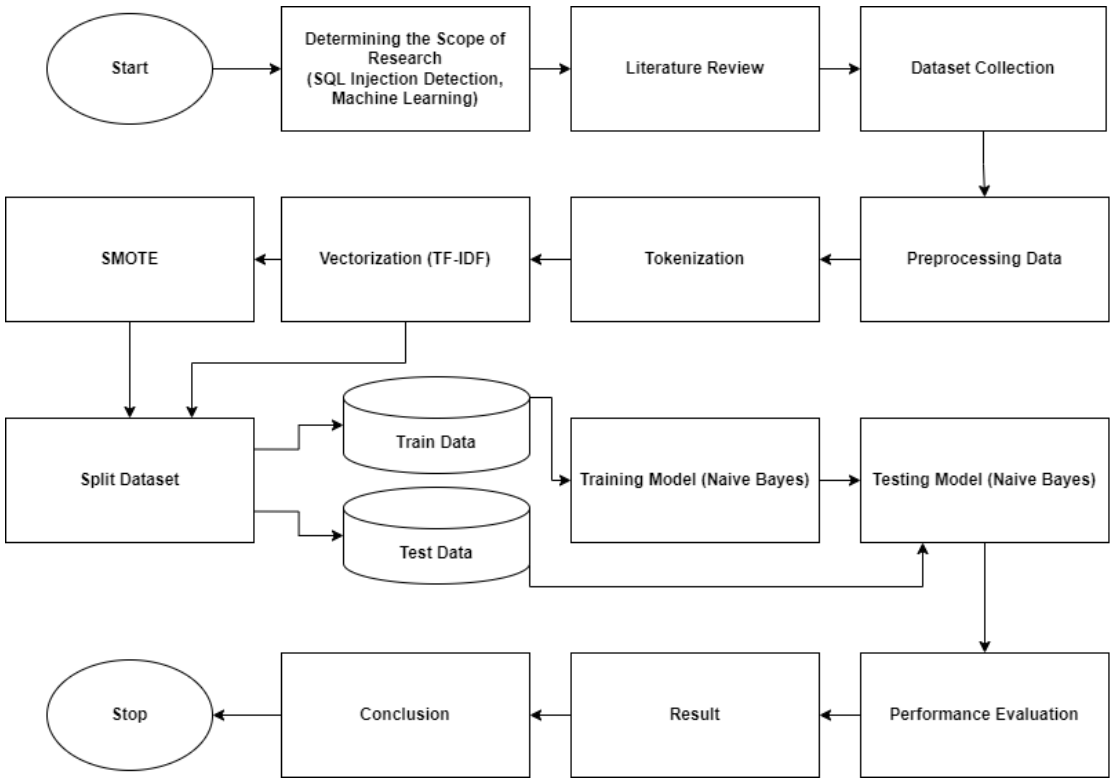

Fig. 2: Flow of research methodology

2.1. Determining the scope of research

Defining the scope of the research is a crucial first step. At this stage, the researcher determines the boundaries and focus of the study, which includes the identification of SQL injection attacks using machine learning. Specific research objectives, such as improving attack detection accuracy and model efficiency, are also established at this stage.

2.2. Literature Review

This stage involves a comprehensive literature review to understand the background and recent developments in SQL injection attack detection and the application of machine learning in cybersecurity, specifically for detecting SQL injection attacks. The researcher reviews previous studies to identify methods that have been used, the strengths and weaknesses of existing approaches, and to pinpoint gaps that this research can address. After dataset collection, the researcher discovered that the dataset consisted of imbalanced data. Therefore, in this literature review, the author not only focuses on searching for literature related to the application of machine learning in detecting SQL injection attacks but also conducts a literature review on previous research related to handling imbalanced datasets.

2.3. Dataset Collection

In this study, the researcher obtained the dataset from Kaggle. The dataset used in this research consists of SQL queries, including both malicious and harmless queries. The dataset contains 4,200 rows of SQL queries. Among these, 1,128 rows are labeled as 1, indicating that these queries are Malicious Queries, which potentially contain SQL injection attacks. Meanwhile, the remaining 3,072 rows are labeled as 0, indicating that these queries are Harmless Queries, which do not pose any security threats. The comparison of the number of label 0 and label 1 in the dataset used in this study indicates that the dataset is imbalanced. Therefore, this study will apply a process to handle imbalanced datasets. Before determining the method to handle the imbalanced dataset, the researcher revisited the literature review to identify the most suitable methods for addressing the issue of dataset imbalance.

2.4. Preprocessing Data

Raw data often contains significant amounts of noise and inconsistency that can impact model performance. Therefore, the researchers performed data preprocessing, which includes cleaning the data from irrelevant entries, handling missing values, and normalizing the data to ensure it is on a consistent scale. This process ensures that the data is ready for further analysis. The flow of the data preprocessing can be seen in figure 3.

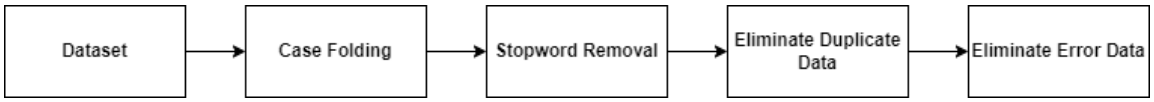

Fig. 3: Alur proses preprocessing data

a. Case folding
Case folding is a text normalization process that involves converting all letter characters to lowercase. This is important to ensure data consistency, as SQL queries are case-insensitive. For example, the words "SELECT" and "select" are considered the same after case folding. This process helps in reducing irrelevant text variations in analysis.
b. Stop Word Removal
At this stage, only numbers are removed from the SQL query. Numbers often do not have significant meaning in the context of text analysis for SQL injection attack detection. Removing numbers helps in simplifying the text and focusing on more meaningful and relevant words for pattern detection, without losing other important information present in the query.
c. Eliminate Duplicate Data
Duplication in the dataset can lead to bias and inflation of certain unwanted information. Therefore, removing duplicate data is an essential step to ensure that each SQL query appears only once in the dataset. This helps in maintaining data quality and enhancing model accuracy by avoiding the excessive influence of repeated data.
d. Eliminate Error Data
Removing error data involves identifying and removing invalid or corrupted SQL queries. Error data can arise from input errors, formatting issues, or inappropriate characters. Cleaning error data ensures that only high-quality and relevant data is used in training and testing the model. This is crucial to avoid disruptions in analysis and to improve the performance of the detection model.

2.5. Tokenization

Tokenization is a fundamental process in text analysis, where text is broken down into smaller units such as words or phrases. In this context, the researcher performs tokenization on the input data to identify patterns and structures of SQL attacks. This process involves using tools and techniques from natural language processing (NLP) to ensure accurate and meaningful tokenization. In this study, the researcher will use tools and libraries from the Natural Language Toolkit (NLTK) to perform the tokenization process on the data that has undergone preprocessing.

2.6. Vectorization

After tokenization, the next stage is vectorization, where tokens are transformed into numerical representations that can be utilized by machine learning algorithms. The researchers employ the Term Frequency-Inverse Document Frequency (TF-IDF) technique to convert text into vectors. This representation enables the model to understand and analyze patterns in the data. The stages of the vectorization process using TF-IDF in this study are depicted in figure 4.

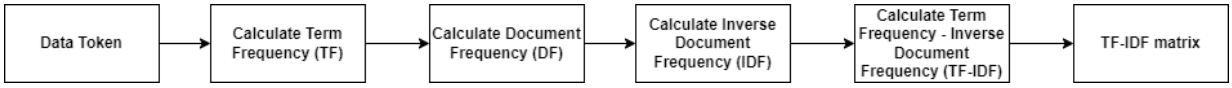

Fig. 4: Flow of the token data vectorization process with TF-IDF

a. Calculate Term Frequency (TF)
Term Frequency (TF) measures how often a term appears in a document compared to the total number of terms in that document. The formula for calculating TF for a term *t* in a document *d* is:

$$\mathrm{TF}(t,d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d} \quad (1)$$

Definition 1:
- *t* = Term whose frequency is being calculated.
- *d* = Document in which the term frequency is being calculated.
- TF(*t*,*d*) = Term frequency of term *t* in document *d*.

b. Calculate Document Frequency (DF)
Document Frequency (DF) is the number of documents in the corpus that contain a specific term. The formula for calculating DF is:

$$\mathrm{DF}(t,d) = \text{Number of documents that contain term } t \quad (2)$$

Definition 2:
- *t* = Term whose frequency is being calculated.
- DF(*t*) = Document frequency of term *t*.

c. Calculate Inverse Document Frequency (IDF)
Inverse Document Frequency (IDF) measures how important a term is within the entire corpus. Terms that appear in many documents have a low IDF, while terms that appear in fewer documents have a high IDF. The formula for calculating IDF is:

$$\text{IDF}(t) = log(\frac{\text{Total number of documents}}{\text{Number of documetns that contain term } t}) \quad (3)$$

Definition 3:

- t = Term whose frequency is being calculated.
- IDF(t) = Inverse document frequency of term t.

d. Calculate Term Frequency – Inverse Document Frequency (TF-IDF)

$$\text{TF} - \text{IDF}(t, d) = \text{TF}(t, d) * \text{IDF}(t) \quad (4)$$

Definition 4:

- t = Term whose frequency is being calculated.
- d = Document in which the term frequency is being calculated.
- TF(t,d) = Term frequency of term t in document d.
- IDF(t) = Inverse document frequency of term t.
- TF-IDF(t,d) = TF-IDF weight of term t in document frequency.

e. TF-IDF Matrix
The TF-IDF Matrix is a representation of all documents in the corpus using TF-IDF weights for each term. Each row in the matrix represents a document, and each column represents a term in the corpus. This matrix is used as input for machine learning algorithms during the model training and testing process. The TF-IDF matrix provides a compact and informative numerical representation of the text, enabling algorithms to analyze and detect patterns more effectively.

2.7. Handling imbalanced datasets

Dataset imbalance is a common issue in anomaly detection, including SQL injection attacks. At this stage, researchers apply techniques to address data imbalance, such as oversampling (adding to minority data), undersampling (reducing majority data), or using weighting algorithms that adjust the importance of each class. This is done to ensure that the model is not biased towards the majority class. In this study, the researchers encountered dataset imbalance where the dataset used contained fewer instances of label 1 (malicious SQL queries) compared to label 0 (benign SQL queries).

From the literature review conducted by the researchers, this study will implement the Synthetic Minority Over-sampling Technique (SMOTE). SMOTE aims to balance class distribution by augmenting and duplicating samples from the minority class randomly [7]. This method creates new instances of the minority class by mixing existing samples. SMOTE uses linear interpolation to generate virtual training records for the minority class. These synthetic training records are formed by randomly selecting one or more of the k-nearest neighbors for each example in the minority class [15].

2.8. Split Dataset

In the dataset splitting process, there are two datasets to be divided: the dataset that has undergone imbalanced dataset handling with SMOTE, and the dataset that has not had imbalanced dataset handling with SMOTE. The division of each dataset (the dataset that does not apply the SMOTE method and the dataset that applies the SMOTE method) into training and testing subsets is a crucial step for objectively evaluating model performance. The researcher divides the data into a training set to train the model and a testing set to evaluate the model's performance. In this study, the researcher splits the dataset into two parts: data for training the model and data for testing the model. The proportion of dataset splitting in this study is done with an 80% ratio for training data and 20% for testing data.

2.9. Training Model

Based on the literature review focusing on SQL injection attack detection using machine learning, this research will utilize the Gaussian Naïve Bayes model to process a SQL Injection dataset that has undergone vectorization in the previous stage. At this stage, the Naïve Bayes model will be trained using the processed training data. The researcher applies the Naïve Bayes algorithm to learn patterns and characteristics of SQL injection attacks from the training data. The goal of this training process is for the Naïve Bayes model to recognize patterns of SQL queries that are harmful and those that are not. Thus, when the Naïve Bayes model encounters new SQL query data, it is expected to identify and detect whether the SQL query is harmful or not.

In this study, the researcher will compare the Naïve Bayes model with a Naïve Bayes model combined with SMOTE. Therefore, the model training will be conducted in two stages. In the first stage, the Naïve Bayes model will be trained with a dataset without the SMOTE method applied. In the second stage, the Naïve Bayes model will be trained with a dataset to which the SMOTE method has been applied.

2.10. Testing Model

After the model is trained, the next step is testing it with data that the model has not seen before. The researchers conduct tests to evaluate how well the model can detect SQL injection attacks on new data. These test results provide insights into the model's performance in real-world scenarios and help identify areas that need improvement. During the testing phase, the model will be evaluated in two stages. In the

first stage, the Naïve Bayes model without SMOTE will be tested with data that has not been processed with SMOTE. In the second stage, the Naïve Bayes model combined with SMOTE will be tested with data that has been processed with SMOTE. The comparison of results from these two training stages will provide valuable insights into the impact of applying SMOTE on the Naïve Bayes model's ability to detect SQL injection attacks.

2.11. Performance Evaluation

Model performance evaluation is the final step in the research methodology. The researchers use evaluation metrics such as accuracy, precision, recall, and F1-score to assess the overall performance of the model. This analysis helps in understanding the strengths and weaknesses of the model and provides a basis for further improvement. The evaluation results are also used to compare the Naïve Bayes model with the Naïve Bayes model combined with SMOTE to determine how much the SMOTE model can enhance the performance metrics of a model. Additionally, the performance evaluation results can serve as a basis for comparing the models used in this research with other approaches that may be employed in future studies.

3. Result and Discussion

Figure 5 shows a visualization comparing the number of dataset entries before and after the application of the SMOTE method to address label or class imbalance in the dataset used in this study. Before the application of SMOTE, the dataset consisted of 4,200 SQL query rows, with 1,128 rows labeled as 1 and 3,072 rows labeled as 0. The comparison between the number of label 1 and label 0 indicates a significant imbalance in the dataset. After the application of the SMOTE method, this imbalance was successfully addressed, resulting in an equal number of SQL query rows for each label. With the application of SMOTE, the dataset now has 2,978 SQL query rows for label 1 and 2,978 SQL query rows for label 0. This visualization clarifies how the SMOTE method effectively balances the class distribution in the dataset, allowing for a fairer and more accurate analysis of SQL injection attacks.

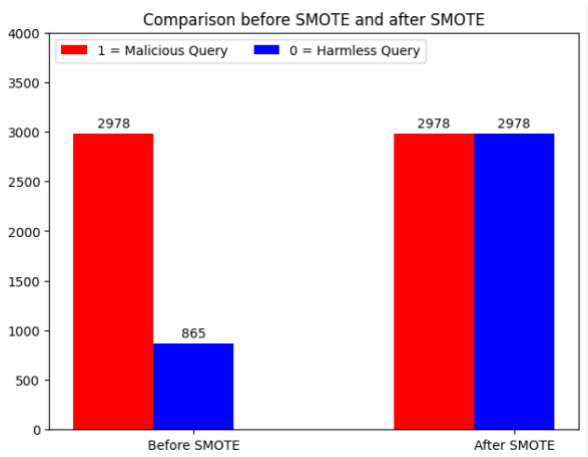

Fig. 5: Comparison before SMOTE and after SMOTE on SQL Injection Dataset

This study focuses on detecting SQL injection attacks using a Naive Bayes model and addressing dataset imbalance using the SMOTE method. To evaluate the effectiveness of the SMOTE method in improving model performance, two stages of training and testing were conducted. The first stage involved training and testing the Naive Bayes model on a dataset without applying the SMOTE method. The results of this stage showed an accuracy of 0.9948, an F1 Score of 0.9885, Precision of 0.9906, and Recall of 0.9946. Although these results indicate very good model performance, data imbalance still potentially affects the model's ability to detect the minority class, namely SQL injection attacks. Figure 6 presents the detailed results of the model testing, and figure 8 shows the visualization of the confusion matrix table for the Gaussian Naïve Bayes model without SMOTE applied to handle the imbalanced dataset.

In the second stage, the dataset with the SMOTE method applied was used for training and testing the Naive Bayes model. The application of SMOTE successfully produced an evaluation of model performance with the Naive Bayes model accuracy reaching 0.9950, the F1 Score increasing to 0.9950, Precision reaching 0.9950, and Recall also increasing to 0.9950. This improvement indicates that the SMOTE method is effective in addressing data imbalance issues, which in turn enhances the model's ability to detect SQL injection attacks more effectively. Figure 7 provides the detailed results of the model testing, and figure 9 presents the visualization of the confusion matrix table for the Gaussian Naïve Bayes model combined with SMOTE to handle the imbalanced dataset.

```
Accuracy of Naive Bayes on test set: 0.9947984395318595
F1 Score of Naive Bayes on test set: 0.9885057471264367
Precision of Naive Bayes on test set: 0.9905868205868206
Recall of Naive Bayes on test set: 0.9945930480661054

Classification Report (Before SMOTE):
              precision    recall  f1-score   support

           0       1.00      0.99      1.00       596
           1       0.98      0.99      0.99       173

    accuracy                           0.99       769
   macro avg       0.99      0.99      0.99       769
weighted avg       0.99      0.99      0.99       769
```

Fig. 6: Detailed performance evaluation of Naive Bayes without SMOTE

```
Accuracy of Naive Bayes + SMOTE on test set: 0.9949664429530202
F1 Score of Naive Bayes + SMOTE on test set: 0.9949748743718593
Precision of Naive Bayes + SMOTE on test set: 0.9949720167111471
Recall of Naive Bayes + SMOTE on test set: 0.9949664429530201

Classification Report (After SMOTE):
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       596
           1       0.99      1.00      0.99       596

    accuracy                           0.99      1192
   macro avg       0.99      0.99      0.99      1192
weighted avg       0.99      0.99      0.99      1192
```

Fig. 7: Detailed performance evaluation of Naive Bayes with SMOTE

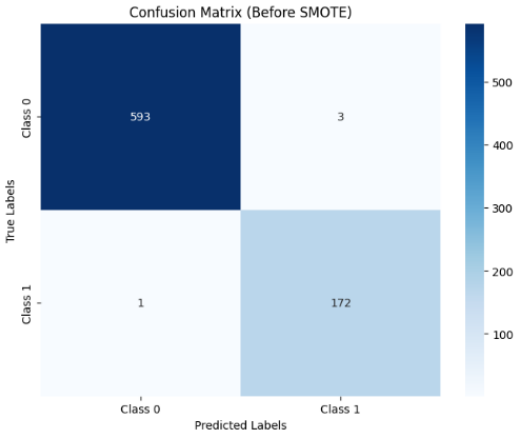

Fig. 8: Naive Bayes Confusion Matrix without SMOTE

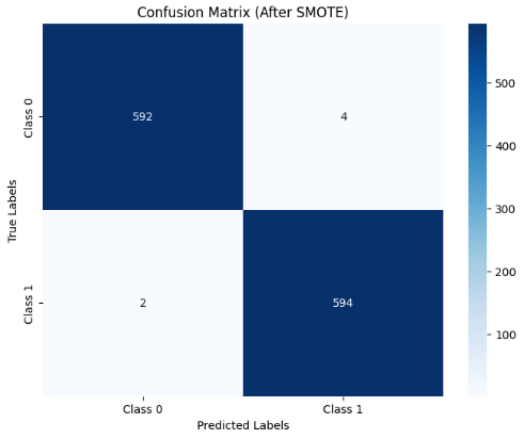

Fig. 9: Naive Bayes Confusion Matrix with SMOTE

Table 1. Comparison of evaluation performance results

Performance Evaluation	Naïve Bayes	Naïve Bayes + SMOTE
Accuracy	99.48%	99.50%
Recall	98.85%	99.50%
Precision	99.46%	99.50%
F1-Score	99.06%	99.50%

Table 1 presents a comparison of the performance evaluation results between the Gaussian Naïve Bayes model without SMOTE and the Gaussian Naïve Bayes model combined with SMOTE. These results indicate that although the Naïve Bayes model already performs very well on the imbalanced dataset, the application of SMOTE provides a significant performance improvement. The increase in the F1 Score, in particular, suggests that the model is better at balancing precision and recall, thus being more effective in identifying SQL injection attacks. In conclusion, the SMOTE method has proven to be beneficial in enhancing the performance of the Naïve Bayes model for SQL injection detection, especially in the context of imbalanced datasets.

4. Conclusion

Based on the results of this study, it can be concluded that the application of the SMOTE method in training the Naive Bayes model significantly improves the model's performance in detecting SQL injection attacks. During the training phase with the dataset without SMOTE, the Naive Bayes model achieved an accuracy of 0.9948; however, data imbalance could affect its effectiveness. After applying SMOTE, there was a significant improvement in all evaluation metrics: accuracy increased to 0.9950, F1 Score to 0.9950, Precision to 0.9950, and Recall to 0.9950. This indicates that SMOTE successfully balanced the classes within the dataset and enhanced the detection of SQL injection attacks. For future research, it is recommended to explore other resampling methods such as ADASYN, as well as additional feature engineering techniques. Testing other models, such as Random Forest or SVM, and applying the models in real-world environments could also provide further insights and improve detection effectiveness. Evaluating on more diverse datasets and more complex scenarios could help ensure better model generalization.

References

[1] D. Putra Purbawa, A. J. Ulhaq, G. Ikhsan, A. M. Shiddiqi, D. Ary, and M. Shiddiqi, "An Enhanced SQL Injection Detection using Ensemble Method," *JUTI: Jurnal Ilmiah Teknologi Informasi* , vol. 21, no. 1, pp. 1–9, 2023, doi: 10.12962/j24068535.v21i1.a1060.

[2] A. Sunyoto and E. Pramono, "Deteksi Serangan SQL Injection Menggunakan Hidden Markov Model," *TECNOSCIENZA*, vol. 5, no. 2, 2021.

[3] OWASP, "Introduction: Welcome to the OWASP Top 10 - 2021." Accessed: Jul. 18, 2024. [Online]. Available: https://owasp.org/Top10/

[4] J. Friadi and S. Septian, "Aplikasi Machine Learning Untuk Deteksi Serangan Code Injection," pp. 443–451, 2021, doi: 10.37776/zt.vxix.xxx.

[5] H. B. Jatmiko, N. T. Kurniadi, and D. Maulana, "Optimasi Naïve Bayes Dengan Particle Swarm Optimization Untuk Analisis Sentimen Formula E-Jakarta," *JACIS : Journal Automation Computer Information System*, vol. 2, no. 1, pp. 22–30, 2022.

[6] V. Puspaning Ramadhan, F. Yulian Pamuji, and A. History, "Analisis Perbandingan Algoritma Forecasting dalam Prediksi Harga Saham LQ45 PT Bank Mandiri Sekuritas (BMRI)," *Jurnal Teknologi dan Manajemen Informatika*, vol. 8, pp. 39–45, 2022, [Online]. Available: http://http://jurnal.unmer.ac.id/index.php/jtmi

[7] W. Rahayu, D. Jollyta, A. Hajjah, Y. Nora Marlim, and Y. Desnelita, "Synthetic Minority Oversampling Technique (SMOTE) for Boosting the Accuracy of C4.5 Algorithm Model," *Journal of Artificial Intelligence and Engineering Applications*, vol. 3, no. 3, pp. 2808–4519, 2024, [Online]. Available: https://ioinformatic.org/

[8] J. H. Joloudari, A. Marefat, M. A. Nematollahi, S. S. Oyelere, and S. Hussain, "Effective Class-Imbalance Learning Based on SMOTE and Convolutional Neural Networks," *Applied Sciences (Switzerland)*, vol. 13, no. 6, Mar. 2023, doi: 10.3390/app13064006.

[9] T. Wongvorachan, S. He, and O. Bulut, "A Comparison of Undersampling, Oversampling, and SMOTE Methods for Dealing with Imbalanced Classification in Educational Data Mining," *Information (Switzerland)*, vol. 14, no. 1, Jan. 2023, doi: 10.3390/info14010054.

[10] D. Dablain, B. Krawczyk, and N. V. Chawla, "DeepSMOTE: Fusing Deep Learning and SMOTE for Imbalanced Data," *IEEE Trans Neural Netw Learn Syst*, vol. 34, no. 9, pp. 6390–6404, Sep. 2023, doi: 10.1109/TNNLS.2021.3136503.

[11] D. K. Fitri, N. Sri, M. Lestari, J. Ilmu, K. Jurusan, and I. Komputer, "Implementasi Algoritma Naïve Bayes Menggunakan Feature Forward Selection dan SMOTE Untuk Memprediksi Ketepatan Masa Studi Mahasiswa Sarjana," 2022.

[12] G. Gumelar, Q. Ain, R. Marsuciati, S. Agustanti Bambang, A. Sunyoto, and M. Syukri Mustafa, "Kombinasi Algoritma Sampling dengan Algoritma Klasifikasi untuk Meningkatkan Performa Klasifikasi Dataset Imbalance," *Prosiding Seminar Nasional Sistem Informasi dan Teknologi (SISFOTEK)*, vol. 5, 2021.

[13] F. Yulian Pamuji, S. Dwi Arma Putri, F. Teknologi Informasi, and U. Malang, "Komparasi Metode SMOTE dan ADASYN Untuk Penanganan Data Tidak Seimbang MultiClass," *JIP (Jurnal Informatika Polinema)* , vol. 9, no. 3, pp. 331–338, 2023.

[14] A. N. Hermana, M. Gustiana Husada, and O. Kurniawan, "Penerapan SMOTE Untuk Mengatasi Data Imbalance pada Identifikasi Originalitas Sepatu Converse Menggunakan CNN Arsitektur VGG-16," *Jurnal Pendidikan Tambusai*, vol. 8, no. 1, pp. 10710–10722, 2024, [Online]. Available: https://www.converse.id/

[15] A. Muneer, R. F. Ali, A. Alghamdi, S. M. Taib, A. Almaghthawi, and E. A. Abdullah Ghaleb, "Predicting customers churning in banking industry: A machine learning approach," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 26, no. 1, pp. 539–549, Apr. 2022, doi: 10.11591/ijeecs.v26.i1.pp539-549.