REPUBLIC OF TÜRKİYE

ALTINBAŞ UNIVERSITY

Institute of Graduate Studies

Information Technologies

DETECTION OF SQL INJECTION ATTACKS

Rana B. Hadi ALRUBAE

Master's Thesis

Supervisor

Asst. Prof. Dr. Ayça TÜRKBEN

Istanbul, 2023

DETECTION OF SQL INJECTION ATTACKS

Rana B. Hadi ALRUBAE

Information Technologies

Master’s Thesis

ALTINBAŞ UNIVERSITY

2023

The thesis titled DETECTION OF SQL INJECTION ATTACKS prepared by RANA B. HADI ALRUBAE and submitted on 05/04/2023 has been **accepted unanimously** for the degree of Master of Science in Information Technology.

Asst. Prof. Dr. Ayça Kurnaz TÜRKBEN
Supervisor

Thesis Defense Committee Members:

Asst. Prof. Dr. Ayça Kurnaz TÜRKBEN	Department of software Engineering, Altınbaş University	_____
Asst. prof. Dr. Abudullahi abdu IBRAHIM	Department of computer Engineering, Altınbaş University	_____
Asst.Prof. Dr. Zeynep ALTAN	Department of Software Engineering, Beykent University	_____

I hereby declare that this thesis meets all format and submission requirements of a Master's thesis.

Submission date of the thesis to Institute of Graduate Studies: ___/___/___

I hereby declare that all information/data presented in this graduation project has been obtained in full accordance with academic rules and ethical conduct. I also declare all unoriginal materials and conclusions have been cited in the text and all references mentioned in the Reference List have been cited in the text, and vice versa as required by the abovementioned rules and conduct.

Rana B. Hadi AL-RUBAE

Signature

DEDICATION

I dedicate this work to my supervisor in the first place who was the light that illuminates my path. And to my father's soul my God have mercy on him. And to my family who helped me in all my difficult times my mother, my husband, my daughter. Also, I will not forget my friends who support and encouraged me.

PREFACE

I would like to thank my supervisor professor Ayça Kurnaz TÜRKBEN for all support during my study. It’s great pleasure to express my deepest gratitude to my friends who have shared with me best moments during my study for the master’s degree.

ABSTRACT

DETECTION OF SQL INJECTION ATTACKS

AL-RUBAE, Rana B. Hadi

M.Sc., Information Technologies, Altınbaş University,

Supervisor: Asst. Prof. Dr. Ayça Kurnaz TÜRKBEN

Date: April / 2023

Pages: 77

Web applications that employ SQL datasets are at serious risk from SQL injection attacks. Using two separate datasets, this study examined the utility of ML classification approaches for identifying SQL injection threats. On two datasets—one taken from the actual world and the other created artificially—random forest (RF), descion tree (DT), k-nearest neighbours (KNN), gradient boost (GB), xgb classifier, linear SVM (Support Vector Machines), and RBF (Radial Basis Function) SVM were trained. The outcomes demonstrated that both algorithms could precisely and precisely detect SQL injection attacks. The model that was trained using the synthetic dataset outperformed the model that was trained using the real-world dataset. These findings highlight the potential of ML classification for identifying SQL injection attacks as well as the value of employing a variety of datasets to increase model accuracy (ACC).

Keywords: SQL, ML, Injection, Attacks, Classification, Web.

TABLE OF CONTENTS

Pages

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

1. INTRODUCTION AND PURPOSE

1.1 INTRODUCTION

SQL injection is a sort of cyberattack that takes advantage of holes in online applications that use SQL databases. In a SQL injection attack (SQLIA), the attacker can insert malicious code into a SQL query, which the database can subsequently use to execute. This might provide the attacker access to, the ability to change or delete important data, or even the ability to obtain unauthorized access to the whole system [1].

Various strategies, such as signature-based detection, anomaly-based detection, and ML-based detection, have been suggested to identify and stop SQLIAs. The detection methods based on ML have shown to be the most reliable and efficient [2].

One of the most often utilized methods for SQLID (SQLID) is ML classification [3]. In general, ML methods are used to monitor online traffic and find trends that point to probable SQLIAs. In this method, the ML model is trained using instances of both legitimate and malicious online traffic from a labelled dataset of web traffic. The model may then be used to categorize fresh web traffic as either benign or malicious after being trained.

For the purpose of detecting SQL injection, a variety of ML methods [4], including as DT, RF, SVM, and NN, can be utilized. Using supervised learning algorithms to categorize online traffic as legitimate or malicious and unsupervised learning algorithms to find patterns in web traffic that could point to an attack are two prevalent strategies.

The dataset that is used for training and assessing the ML model is also very important. Typically, the dataset is created by gathering online traffic from actual web apps, which is then classified as benign or harmful. The fundamental issue in this subject is the scarcity of labelled dataset, albeit it may be overcome by simulating SQLIAs to create synthetic dataset [4].

Overall, using ML classification to detect and stop SQL injection attempts is an effective strategy [5]. When training ML models on smaller datasets, it's vital to keep in mind that overfitting is a possibility and that the quantity and quality of the dataset used to train the models can have a major influence on their performance. To stay up with new and developing SQL injection tactics, it is also crucial to routinely update and perfect the models.

Building a high-quality labelled dataset is one of the major hurdles in applying ML for SQLID [6]. A representative sample of typical online traffic should also be included in the dataset, as well as a range of possible SQLIAs. Due to the rarity and frequent unreported nature of SQLIAs, getting such a dataset might be challenging. In order toth replicate SQL injection threats, several research papers employ datasets that were intentionally created.

Several well-known ML methods, including as DT, RF, SVMs, and NN, have been used to the detection of SQL injection. Each of these methods has advantages and disadvantages of its own, and the selection of algorithm will rely on the details of the dataset and the application.

Examples of straightforward yet effective algorithms for datasets with plenty of characteristics are DTs and RFs. They do this by dividing the data into branches, each of which represents a distinct classification choice that the algorithm must make. This results in a tree-like structure that divides the data into sections. An extension of DTs, RFs use a large number of trained DTs together to enhance the performance of the model.

The SVM method, on the other hand, is a strong algorithm that performs well with datasets that contain a lot of noise and outliers. SVMs divide the data into two groups using the idea of a "hyperplane," and the algorithm selects the hyperplane that optimizes the distance between the two classes of data.

SQLID has also been done using neural networks, particularly deep neural networks. These algorithms are meant to replicate the workings of the human brain, and they can learn complicated patterns and correlations in data. However, enormous quantities of data are required to train them, and they can be computationally expensive to run.

As you can see, the algorithm of choice is determined on the unique properties of the dataset and application. To acquire the best outcomes, it is necessary to experiment with various algorithms and assess their performance using proper evaluation measures.

It's also worth mentioning that ML classification isn't the only technique to SQLID; it can be paired with other approaches like signature-based detection and anomaly-based detection to construct a multi-layered defensive mechanism that improves overall system security.

1.2 OBJECTIVES

The key objective of employing various ML classification techniques for SQLID is to increase the precision (PREC) and resilience of the detection model.

Here are some particular objectives that researchers may strive for while working on SQLID using various ML classification methods:

a. Comparing the performance of several ML methods, such as DTs, RFs, SVMs, and neural networks, on the problem of SQLID to discover which technique or combination of algorithms is most successful.

b. Because SQLIAs may take various forms, employing a broad dataset and evaluating the model's effectiveness in detecting SQL injection helps increase the model's robustness.

c. To increase overall performance, predictions from many models trained with various ML algorithms are combined.

The study hypothesis is that alternative ML methods and feature representations may be more adapted against different forms of SQL injection assaults. Researchers can obtain insight into which algorithms are most suited for different forms of SQLIAs and which feature representations give the most information for detecting these types of assaults by comparing the performance of numerous algorithms on the same job.

1.3 CONTRIBUTION

The proposed study will contribute to the development of improved techniques for identifying and thwarting SQLIAs. The study's findings will provide light on the best ML techniques for developing and testing these algorithms for real-time SQLID. The results will be helpful to businesses looking to strengthen the security of their online applications and safeguard against possible dangers from SQL injection.

1.4 METHODOLOGY

The following techniques will be applied to answer the research questions:

a. Literature review: To identify the most successful methods and methodologies for SQLID, a review of the current literature on ML for SQLID will be done.

b. Data collection and preparation: A dataset of common and exceptional SQLID will be collected in order to train and evaluate ML methods.

c. Algorithm selection and evaluation: Several ML algorithms will be chosen and trained on the dataset. The algorithms' performance in accurately recognizing intrusions in test data will be examined.

d. Limitations and challenges: Using ML approaches for SQLID may have limitations and issues, which will be discovered and presented.

1.5 EXPECTED RESULTS

It is anticipated that the proposed research's findings would provide important light on the best ML approaches and strategies for SQLID. The outcomes will also point out any drawbacks and difficulties of employing ML for this purpose.

1.6 REPORT ORGANIZATION

The remainder of this study is divided into the following chapters:

a. The second chapter: covers SQLID, ML techniques, and the most recent research in this area.

b. The third chapter: a detailed explanation of the technique, the system-building phases, and the procedures that will be included in the dataset, including the pre-processing stage, the models, the cooperative study of the models, and the last stage, the training stage.

c. The fourth chapter: provides the final training results for the model. Apply assessment tools, then compare the results.

d. The fifth chapter, which is the conclusion, provides an overview of all the work that will be done in our thesis, including the selection of the models, the phases, and the outcomes.

1.7 CONCLUSION

To sum up, chapter 1 of "SQLID" has covered the importance of preventing SQLIAs in the current digital world. Organizations must put strong detection and prevention procedures in place to protect their networks from unwanted access and possible threats. By enabling the accurate identification of SQLIAs in big and complicated databases, ML methods, especially classification algorithms, play a significant role in SQLID. These categorization algorithms will be improved through the proposed study, which will also offer useful information to companies wishing to strengthen their network security.

2. GENERAL INFORMATION

2.1 INTRODUCTION

The purpose of this literature review is to investigate the current status of research on SQLID using various ML classification algorithms. It will summarize and analyze various studies on this topic, with a focus on the use of algorithms such as KNN, DTs, RF, Linear SVM, RBF SVM, XGBoost, and Gradient Boosting (GB). The evaluation will compare the performance of these algorithms to standard approaches for SQLID in terms of ACC, PREC, recall (REC), and other parameters. It will also look at the difficulties and limits of applying ML techniques in real-world contexts. The overall goal of this literature review is to shed light on the viability and potential of applying ML classification methods for SQLID, as well as the practical implications of the research.

2.2 SQLID

SQL injection is a sort of cyber-attack in which a malicious SQL code is inserted into a web form input or URL in order to obtain unauthorized access to a database [8]. This gives the attacker the ability to access, alter, or remove sensitive data contained in the database. The method of detecting and blocking SQLIAs is known as SQLID. SQLIAs may be detected using Intrusion Detection Systems (IDS), which monitor network traffic for unusual activities and notify administrators of any possible security breaches[7]. IDS may be used for SQLID in a variety of ways, including host-based, network-based, and application-based IDS [9]. Network-based IDS scans network traffic for certain patterns or abnormalities in the data to identify SQLIAs[10]. In order to identify SQLIAs, host-based IDS examines system logs [11] and events on a single host or server. SQLIAs within a particular program or online form are especially targeted by application-based IDS.

ML techniques may also be used to identify SQL injection threats [12]. These techniques train a model on a dataset of known attacks and valid_querry behaviour, which may subsequently be used to detect new, previously undetected SQL injection attempts. Popular SQLID ML methods include, but are not limited to, RF, k-nearest neighbour, SVM, and GB.

Overall, SQLID is an important part of network security that needs a combination of proactive and reactive procedures to be effective, figure 1 shows an example of this attack.

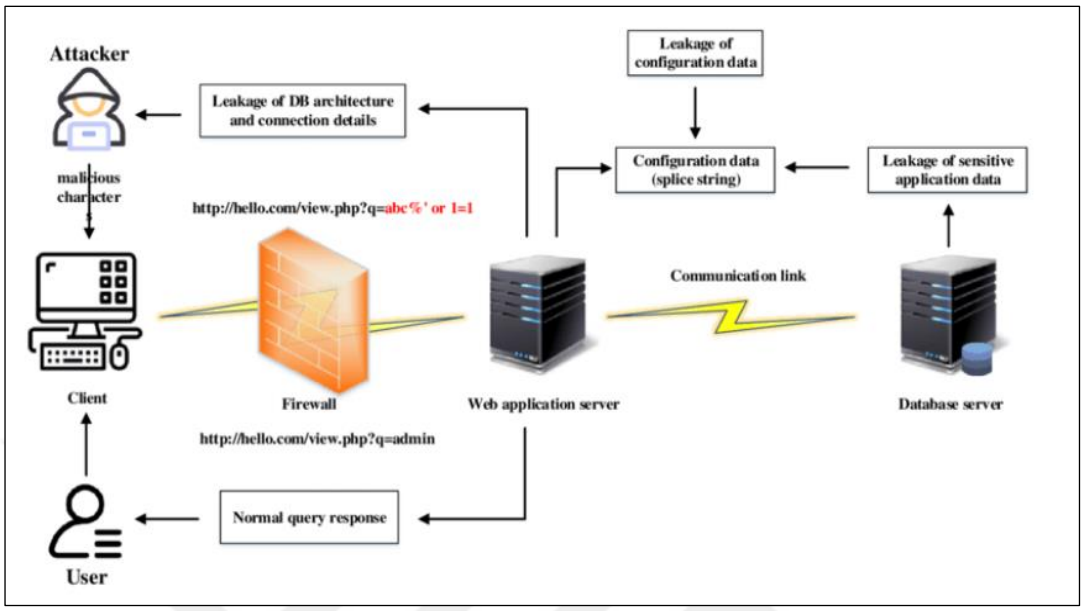


Figure 2.1: SQLIA Schematic Diagram [13].

SQLID may be used in a variety of ways to detect and prevent these sorts of assaults. These are covered in the next section.

2.3 SIGNATURE-BASED DETECTION

In order to recognize SQLIAs, a technique called as signature-based detection [14] develops a set of guidelines or patterns that specify how a known assault appears. Examples of this include specific SQL terms or statements, as well as data patterns that point to an assault. Once the rules or patterns have been established, they are compared against network traffic or system logs in an effort to spot well-known attack patterns. An alarm is set out when a match is discovered, alerting the system administrator, who may then take the necessary steps to stop the attack and stop any unauthorized access to the database.

Because the rules or patterns are pre-defined, this sort of detection is very straightforward to deploy, but it has its own limitations:

a. Because it can only identify known assaults, new or unknown attacks may go undiscovered.

b. It may yield many false positives since not all traffic that fits a certain pattern is malicious.

c. avoiding detection by switching assault patterns or approaches

For these reasons, signature-based detection should be combined with other forms of detection, such as anomaly-based detection or Web Application Firewall-based detection [15].

2.3.1 Anomaly-Based Detection

Anomaly-based detection [16] is a technique for detecting SQLIAs that involves establishing a baseline of usual activity and then monitoring for departures from that baseline. This method uses ML models to understand the system's typical behavior and detect any new and unknown SQL injection attempts [17]. The model is often built by gathering a dataset of both valid_querry and malicious traffic and then training a ML model on that data. Once trained, the model may be used to scan fresh traffic for trends or abnormalities that may suggest a SQLIA.

Anomaly detection offers various advantages over signature detection, including [18]:

a. Because it looks for abnormalities from typical behavior rather than particular attack patterns, it can identify both known and undiscovered assaults.

b. It can adapt to system changes since it continually checks for abnormalities, allowing it to identify new types of SQLIAs.

However, this approach has its own drawbacks, some of which are as follows [19]:

a. It can be more complex to install and maintain than signature-based detection.

b. It may produce a large number of false positives since valid deviations from typical behavior may exist.

c. It's difficult to establish the parameters of typical behavior, which might make the model less realistic.

Anomaly-based detection, like signature-based detection, is most successful when combined with other types of detection methods.

2.3.2 Network-Based Detection

Network-based detection detects SQLIAs by examining network data for certain patterns or abnormalities that may suggest an attack is taking place [21]. This sort of surveillance employs Intrusion Detection Systems (IDS) strategically positioned throughout the network to monitor all incoming and outgoing data. This might include searching for specific SQL phrases or instructions, as well as strange patterns in the data that could suggest a SQLIA [20]. An odd number of database connections, for example, or an unusual volume of data sent, might suggest a SQLIA.

This strategy is useful for detecting SQLIAs that occur outside of a given application or host. It is also appropriate for big networks with several hosts and applications. However, network-based detection has its own set of restrictions, including:

a. It may be unable to identify encoded or encrypted SQL injection attempts.

b. It is feasible to avoid detection by masquerading as typical network traffic.

c. It may yield a large number of false positives since not all traffic that fits a certain pattern is malicious.

For these reasons, it is critical to integrate network-based detection with other forms of detection, such as host-based detection or application-based detection. Additionally, deploying network-based IDS in conjunction with other security measures, such as firewalls, can provide additional levels of protection against SQLIAs and boost the overall efficacy of the security system.

Some organizations use ML techniques for analyzing network traffic to improve the ACC of network-based detection by training the system to recognize valid_querry traffic patterns and then detecting any anomalies that deviate from these patterns; this approach is known as ML based Network IDS (ML-NIDS) [22]. This sort of system may identify new, undiscovered SQL injection threats and is more versatile and adaptive than standard signature-based solutions.

Overall, network-based detection is an important feature of guarding against SQLIAs and should be deployed as part of a multi-layered security approach.

2.3.3 Host-Based Detection

The monitoring of system logs and events on a single host or server is used to detect SQLIAs [23]. This sort of detection employs IDS placed on the host to monitor all system activity and events. This might include looking for specific problem messages or unusual system activity, such as an unusually high number of database requests or modifications to the database. The host-based IDS can be designed to identify and warn on known SQLIA patterns [24], or it can employ ML models to detect any anomalies or deviations from typical behavior.

This approach is useful for detecting SQLIAs within a given application or host. It may also be used to detect assaults that are undetected by network-based detection. However, host-based detection has its own set of restrictions, including [25]:

a. It necessitates the installation of an intrusion detection system (IDS), which might be resource intensive.

b. It is feasible to avoid detection by concealing the assault among authorized system activity.

c. SQLIAs that occur outside of the host, such as those targeting network services or other hosts, may go undetected.

2.3.4 Application-Based Detection

Application-based detection [26] is a technique for locating SQLIAs that is particularly made to find assaults inside a particular application or online form. By including security safeguards in the application's code to thwart SQLIAs, this form of detection may be accomplished at the application level. To prevent SQLIAs, you might use prepared statements or parameterized queries, validate user input to make sure it only contains expected characters and is free of SQL keywords or commands, and put other security measures in place like input sanitization, output encoding, and access control.

This strategy is more successful in dealing with application-level vulnerabilities and is more efficient in detecting SQLIAs that target a single application. However, this strategy has its own disadvantages, such as:

a. It is necessary to have access to the application's source code in order to install the security measures.

b. It is necessary that the application be created with security in mind and that security measures be adequately applied.

c. SQLIAs that occur outside of the application, such as those that target the underlying database or network services, may go undetected.

2.3.5 Web Application Firewall (WAF) Based Detection

A technique for recognizing and stopping SQLIAs that are especially directed at web applications is Web Application Firewall (WAF) [27] based detection. An intrusion detection system called a WAF stands in front of a web application and keeps track of all incoming HTTP and HTTPS traffic [28]. It can be set up to recognize and prevent well-known SQLIA patterns, or it can employ ML models to find any abnormalities or departures from the norm. This may entail searching for certain SQL phrases or instructions or searching for odd patterns in the data that might point to a SQLIA. The WAF can block a suspicious request when one is found, preventing the attack from reaching the application.

A WAF is an important security feature for online applications since it is effective at detecting SQLIAs that target web applications [29]. However, this approach has certain drawbacks, including [30]:

a. It can be difficult to configure and maintain.

b. It can be circumvented if the attacker discovers a means to get around the WAF.

c. If the WAF is not configured appropriately, it may create a large number of false positives.

ML (ML) can help with SQLID for a number of reasons [31]:

a. Increased PREC: ML models may be taught to identify typical behavior before spotting any abnormalities or departures from it. As a result, there may be fewer false positives and a greater rate of SQLIA detection.

b. Adaptability: ML models are capable of ongoing learning and adaptation to system and attack technique changes. This implies that they are able to identify fresh SQL injection assaults that more established signature-based techniques could miss.

c. Scalability: ML models may be used to network traffic and large-scale data analysis. Because manual analysis would be impossible in big and complicated systems, this makes them a good fit.

d. Automation: By automating the process of identifying SQL injection threats, ML models may lighten the load on security staff and increase the security system's overall effectiveness.

e. Cost-effective: By automating the detection process, a company may save money on labor expenses and time by minimizing the need for manual inspections. False positives are also decreased.

Although ML may considerably increase the PREC and effectiveness of SQLID, it's crucial to utilize it in conjunction with other types of detection techniques as part of a multi-layered security approach. In order to assure the models' dependability and robustness, it's crucial to train and test them using the right methodologies.

2.4 TYPE OF SQL INJECTION

According to the Open Web Application Security Project, SQLI assaults are the #1 vulnerability and one of the most destructive forms of attacks on web-based systems (OWASP). According to the National Vulnerability Database (NVD), SQLIAs accounted for 7% of all occurrences in 2011 and are tried an average of 71 times per hour. When a young guy was able to acquire over 200,000 names and credit card details from the Guess.com database in February 2002, the first known SQLI was discovered [76]. This emphasizes how serious the problem is since a successful SQLI attack might provide a hacker access to private database data.

An attacker or bad individual can enter a database without authorization using the SQL injection technique in order to get sensitive data. The attacker can modify standard SQL statements like SELECT, INSERT, UPDATE, and MODIFY to insert malicious code into web applications as an input string. This enables the attacker to get around the web application's authentication and permission procedures and access the database without authorization.

Different techniques may be used to accomplish SQL injection [77, 78]. These are the several kinds of SQLIAs:

a. Tautologies: In this kind of SQL injection, the WHERE statement is followed by statements that always return true. Bypassing authentication, this is used to retrieve data from the database.

b. Illegal or illogical queries: Error reports produced by a web server include crucial information for troubleshooting. Attackers purposefully run bad queries to find weak parameters from the error message.

c. Union Query: To extract information, the attacker uses the UNION operator to insert an erroneous query inside a legitimate query. The union query is conducted if both the erroneous and original queries have the same structure (data type and number of columns).

d. Piggy-backed queries: The attacker uses a semicolon to tack on another query to the original one (;). The database server consequently receives several queries to process.

e. Stored procedures: Because they execute more quickly and provide stronger validations, stored procedures are frequently employed in databases. Every time they are called, the same execution strategy is used. Attackers take advantage of this by creating a query and parameter concatenation dynamically before delivering it to a stored procedure for execution.

f. Blind Injection: Also referred to as inferential injection, this kind of SQL injection is carried out by posing true-or-false queries to the database. Blind injection happens when a program is designed to display general faults rather than the specifics of any database mistake.

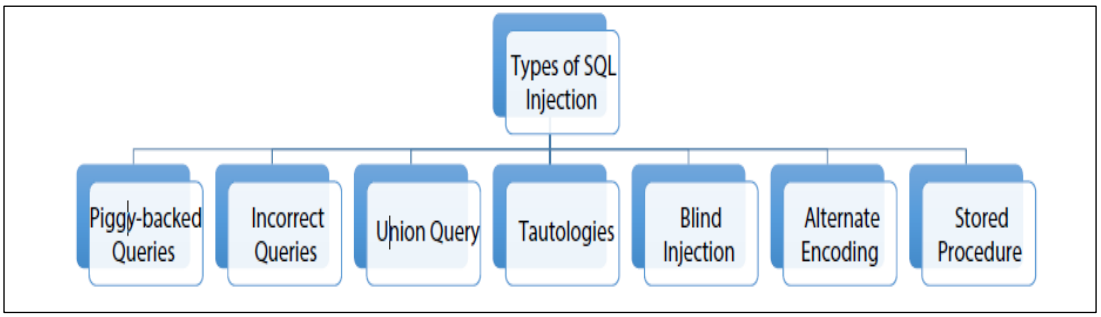


Figure 2.2: Types of SQL Attack [11].

Understanding the effects of these attacks is just as crucial as knowing the many forms of SQLIAs. The businesses and people whose data is stored in the impacted database may suffer significant repercussions as a result of SQLIAs. Common repercussions include the following:

a. Exposure of confidential data: Attackers have the ability to obtain sensitive data that can be utilized for bad intentions, such as login passwords, financial information, and personal data.

b. Data tampering: Attackers have the ability to change or remove data from the database, which may lead to data loss and inaccurate information being stored there.

c. Reputational harm: If it is discovered that a database has been hacked, organizations may experience reputational harm. Customers, partners, and stakeholders may stop trusting you as a result.

d. Financial loss: Attackers may utilize the database information to their advantage, for as by utilizing credit card data that has been taken to make unauthorized transactions.

It is crucial to put in place suitable security measures to stop SQLIAs in order to reduce these risks. Among the typical actions are:

a. Input validation is a process that verifies user-generated input for ACC and eliminates any input that does not adhere to the established standards. By ensuring that user-generated input is correctly sanitized, this can aid in the prevention of SQLIAs.

b. Dynamic queries are riskier than parameterized queries because they decouple the data from the structure of the query. Attackers will find it challenging to insert harmful code into the query as a result of this.

c. Database firewall: A database firewall is a layer of protection that lies between a database and a web application and keeps an eye on and blocks any nefarious activity.

d. Regular security audits: Regular security audits are a good way to find and fix any possible system flaws, including SQL injection flaws.

e. Employee education: Staff members may get instruction on how to identify and report any possible security problems, such as SQLIAs.

SQLIAs may be grouped into numerous traditional forms and are carried out in various ways. Tautologies are statements that are always true, and the attacker tries to insert harmful code into the main code that always returns true. This kind of attack aims to get around authentication and take data. Another form of attack is a piggybacked query, in which the attacker inserts new statements into the original query; the database then executes the original query first, followed by the new statement. This attack aims to extract, edit, or add data from the database or to carry out remote operations on it.

The attacker makes use of error messages supplied by the database to carry out the illogical attack. The information from these messages is used by the attacker to identify application or database settings that are weak. Union queries include utilizing the union command, or the apostrophe ";," to tack on an additional statement to the original query. Both queries' return results will be combined to provide the attack's intended data extraction and authentication bypass. A stored procedure attack is carried out when the attacker tries to access the database procedure, modifies it with malicious code, and then launches it. This kind of attack seeks to escalate privileges, launch a DOS attack, or execute a remote command.

Timing attack and blind injection are two well-known inference attack techniques that are used to alter the behavior of a database or application. In an alternate encoding attack, the attacker alters the injection query to use different encodings, such as hexadecimal, ASCII,

and Unicode, to get over any application-specific filters and extract data while evading authentication.

Table 2.1: SQLI Attack Classical Types.

Name of the attack	Definition	Goals
Tautology	A simple assertion that is consistently accurate in this type of attack, the attacker tries to insert malicious code that returns true in every state into the main code. [79]	Remove data while avoiding authentication.
Piggybacked Query	Add new statements to the original query by injecting them; in this scenario, the database will first perform the basic (main) query before executing the new ones. [80]	Use a remote command to extract, change, or add data from a database.
Logically Incorrect	The error message that the database returned is advantageous to the attacker. Utilized this data to compile database or application vulnerable parameters. [79]	Search for injectable parameters using database fingerprinting.
Union query	simply adding an additional statement to the original query using the union command or the character ";" Both queries' results will be included in the output. [81]	Remove data while avoiding authentication.
Stored Procedure	The attacker attempted to access the database-stored procedure, modify it with malicious code, and execute it directly. [82, 81]	Perform privilege escalation, Perform DOS, run a remote command.
Inference	An attacker can alter a database's or application's behavior by employing an inference attack. There are two widely used strategies that fit under this category of attack: Timing and blind injection attacks [79]	Adapt database behaviour.
Alternate Encoding	By employing different encodings, such as hexadecimal, ASCII, and Unicode, the attacker alters the injection query. By doing this, the attacker tries to get past any application developer filters. [81]	Remove data while avoiding authentication

2.5 MACHINE LEARNING

ML [32] is a potent technique that is being utilized more frequently to promote automation and innovation. It may be applied in several ways to resolve challenging issues and open up fresh possibilities. We'll talk about the many varieties of ML and why they're significant in this blog article.

Supervised learning [33] is the first category of ML. This method of learning trains a computer to recognize specific patterns using labeled data. For instance, tagged photographs might be used to train a computer vision system to distinguish things if one wished to develop one. This kind of ML is useful for a variety of tasks, including reading handwritten writing, forecasting stock prices, and spotting possible fraud.

Unsupervised learning [34] is the second kind of ML. Unlabeled data is used in this form of learning to find patterns in the data. Unsupervised ML, for instance, might be used to divide a set of clients into several categories. This kind of learning is beneficial in a variety of situations, including data analysis, customer segmentation, and product recommendation.

Reinforcement learning [35] is the third category of ML. To learn from its experiences, this sort of learning makes use of feedback. For instance, reinforcement learning may be used to teach a machine how to drive if you want to create an autonomous automobile. This kind of learning works well for autonomous systems, video games, and robots.

Whatever kind of ML you employ, it's critical to comprehend its potential and the consequences of doing so. Although ML may be used to address challenging issues and open up new possibilities, it is crucial to comprehend its ethical implications before utilizing it. ML may be a potent tool for fostering innovation and automation with the correct skills and expertise.

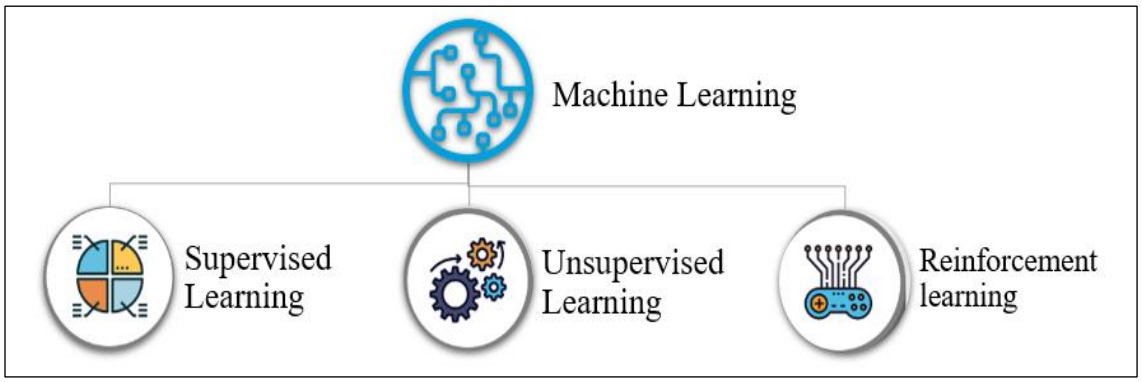


Figure 2.3: Types of ML Algorithms.

To effectively defend against these assaults, combine ML with SQLID. Organizations can identify and stop SQL injection threats by utilizing the power of AI-based solutions, protecting their networks and data.

2.5.1 K-Nearest Neighbours (KNN)

A supervised learning approach for classification and regression is called KNN. When a new observation is made, the method first determines the k-number of training examples that are closest to it. Then, it places the observation in the class that has the most members of that KNN. Both categorical and continuous target variables can be used with KNN.

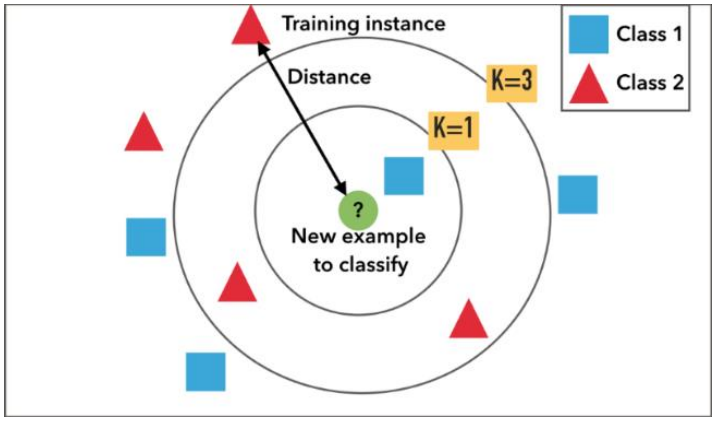


Figure 2.4: Example KNN [37].

An approach that may be utilized to stop SQLIAs is KNN [37]. A KNN-based intrusion detection system (IDS) specifically makes use of the KNN algorithm to categorize network traffic as either valid_querry or abnormal based on patterns of network traffic that have previously been observed. A KNN-based IDS could classify a new packet of network traffic

as a SQLIA if it resembles other packets of network traffic that have previously been categorized as such.

2.5.2 Random Forest (RF)

Both classification and regression are performed using RF [37], an ensemble ML method. It is made up of many DTs, each of which was built using a unique random subset of the data. To obtain the final projection, the predictions of each DT in the forest are averaged. The robust algorithm known as the RF can tolerate missing data and find non-linear correlations in the data.

RF is a technique that, similar KNN, may be employed in IDS to identify SQL injection attempts [38]. An IDS that uses the RF algorithm to identify network traffic as valid_querry or abnormal would do so based on patterns of network traffic that have previously been observed. For instance, an IDS based on RF may categorize a new packet of network traffic as a SQLIA if it resembles other packets that have previously been classified as such.

The algorithm RF is suitable for anomaly identification in intrusion detection [39]. RF can distinguish between regular and abnormal network traffic because it can spot non-linear relationships in the data. Because of its great ACC and resilience, the RF method is frequently employed as an intrusion detection system.

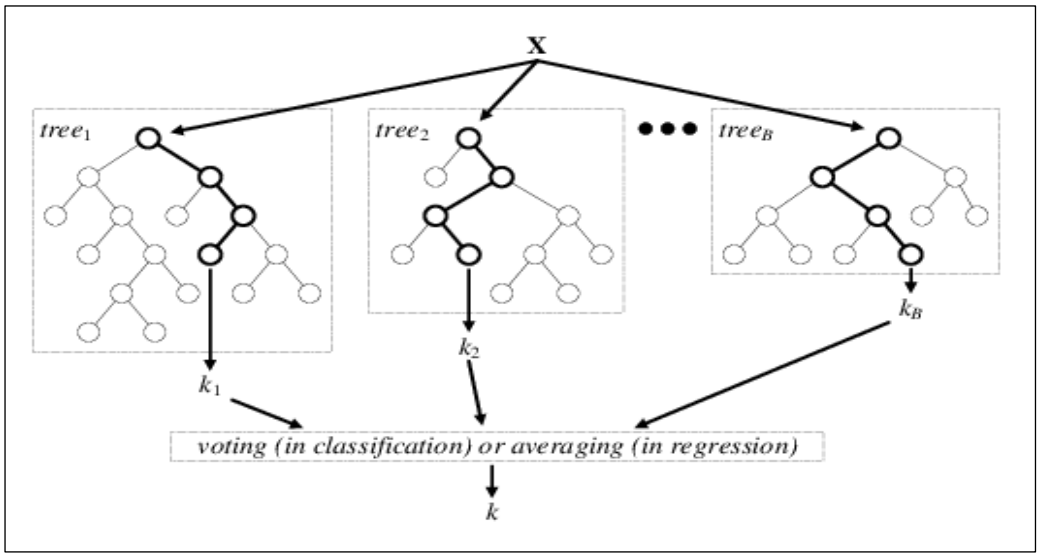


Figure 2.5: RF Example [39].

2.5.3 Linear Support Vector Machine (SVM)

An approach for supervised learning that may be applied to both classification and regression is the linear SVM [40]. A linear SVM's objective is to identify the optimal linear boundary for dividing the various classes of data. The nearest data points from each class are separated from the line by a distance called the margin, which is found by the linear SVM method to be the line (or hyperplane) that maximizes the margin. For data sets with many instances and high dimensionality, linear SVM can be effective.

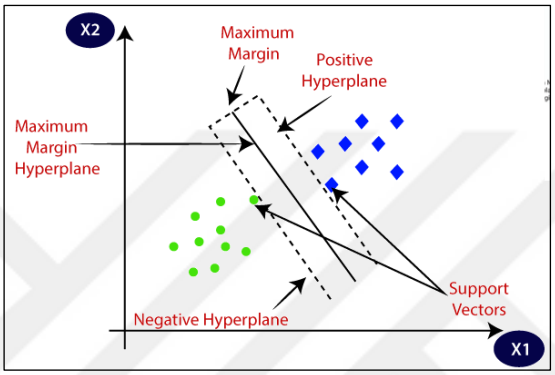


Figure 2.6: General SVM Example [40].

A linear SVM might be employed in an intrusion detection system to identify SQLIAs with relation to SQLID [41]. Based on earlier observed patterns of network traffic, a linear SVM-based IDS would employ the linear SVM algorithm to categorize network traffic as valid_querry or abnormal. For instance, a linear SVM-based IDS might categorize a new packet of network traffic as a SQLIA if it resembled previous packets of network traffic that had been marked as such in the past.

Although linear SVM may be used for anomaly identification [42], it is less successful than non-linear SVM, such as kernel SVM. It could have difficulties detecting specific SQLIAs, such as evasion attacks, but it is still useful for detecting well-known injection assaults.

2.5.4 Radial Basis Function SVM

A non-linear variation of the linear SVM technique is the radial basis function (RBF) SVM [43]. The RBF SVM technique transforms the input data into a higher-dimensional feature space using a radial basis function kernel, such as the Gaussian kernel, to discover a non-linear boundary rather than a linear border. The SVM method determines the boundary that

maximizes the margin in this higher-dimensional space by using the radial basis function kernel to estimate similarity between two points in the feature space.

A RBF SVM might be utilized in an intrusion detection system to find SQLIAs in terms of SQLID [44]. Based on earlier observed patterns of network traffic, an RBF SVM-based IDS would employ the RBF SVM algorithm to categorize network traffic as valid_querry or abnormal. For instance, an RBF SVM-based IDS might categorize a new packet of network traffic as a SQLIA if it resembled previous packets of network traffic that have been tagged as such in the past.

By modelling non-linear and complex relationships among features, RBF SVM is effective at identifying SQLIAs. This could help to detect novel, unknown, and sophisticated attacks that a simple linear model would miss, and it's also resistant to evasion attacks that some SQL injection uses.

2.5.5 XGB Classifier

Extreme GB, often known as XGBoost, is a robust and effective ML technique that is frequently applied to classification and regression issues [45]. It is a GB implementation, which creates a group of DTs. The XGBoost method has been successful in many ML contests and is especially helpful for managing huge and complicated datasets.

Regarding the detection of SQLIAs, XGBoost might be included into an intrusion detection system [46]. Based on patterns of network traffic that have previously been observed, an XGBoost-based IDS would employ the XGBoost algorithm to categorize network traffic as valid_querry or abnormal. An XGBoost-based IDS could classify a new packet of network traffic as a SQLIA, for instance, if it resembles other packets of network traffic that have previously been categorized as such.

By modeling non-linear and complicated correlations between features, the XGBoost technique may be used to identify known and novel SQL injection attempts. It is also reasonably efficient and resilient to overfitting. It is frequently employed in feature selection, which may assist to reduce the dimensionality of the data and raise the model's ACC.

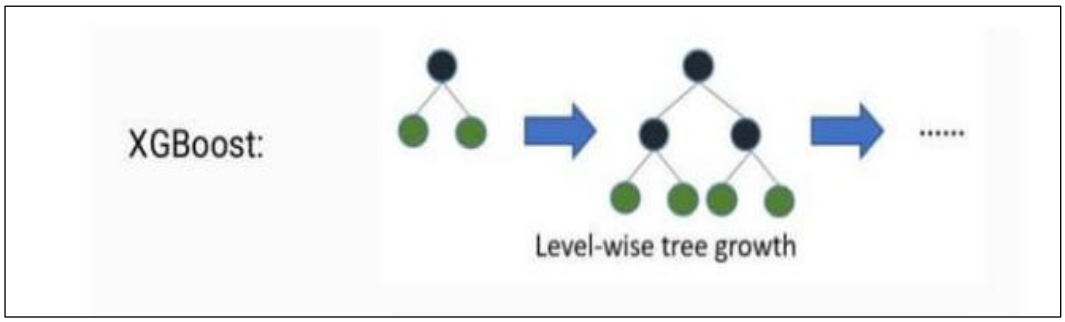


Figure 2.7: XG Boost Example.

2.5.6 Gradient Boosting (GB)

DTs serve as the basis learners in the ensemble learning technique known as GB (GB) [47]. It employs DTs that have been repeatedly trained and then combines them to produce a final model that is more accurate than any single DT. By sequentially adding DTs, each tree in the GB method is trained to reduce the residual errors of the preceding trees in an effort to minimize a certain loss function.

The method is trained using a dataset of typical and unusual network traffic patterns, such as well-known SQLIAs, when utilizing GB for intrusion detection [48]. On the basis of how closely fresh network traffic resembles the patterns it was trained on, the trained model is then used to categorize it as valid_querry or abnormal. By taking note of the intricate and irregular patterns present in network data, the GB method may be utilized to identify SQL injection attempts. This is significant because SQLIAs frequently employ evasion strategies to get past common detection measures. The performance of GB may be improved by adjusting a variety of factors, including learning rate, number of estimators, and DT depth. The generalization ability of GB may be enhanced by setting these parameters to the proper levels, which is crucial for identifying unexpected SQL injection attempts. Additionally, GB has a method known as feature significance that allows users to choose the data's most crucial details, therefore decreasing data dimensions and noise and enhancing model ACC.

In conclusion, GB is an effective technique for intrusion detection, notably for the detection of SQLIAs, since it can handle high-dimensional data, model complicated connections between characteristics, and is resilient to overfitting. GB, a well-known approach in ML and data science with many pre-built libraries that could be used easily, can be a powerful tool when used in conjunction with other intrusion detection methods and features. It also

provides a solid foundation for developing an effective and efficient intrusion detection system.

2.5.7 Decision Tree (DT)

An easy-to-use yet effective approach for classification and regression issues is the DT [49]. It has a tree-like structure, with each internal node standing in for a characteristic or an attribute, each branch for a choice or a rule, and each leaf for a category or value. Recursively dividing the data into subsets according to the characteristic or attribute that results in the greatest information gain or impurity reduction, it creates a model.

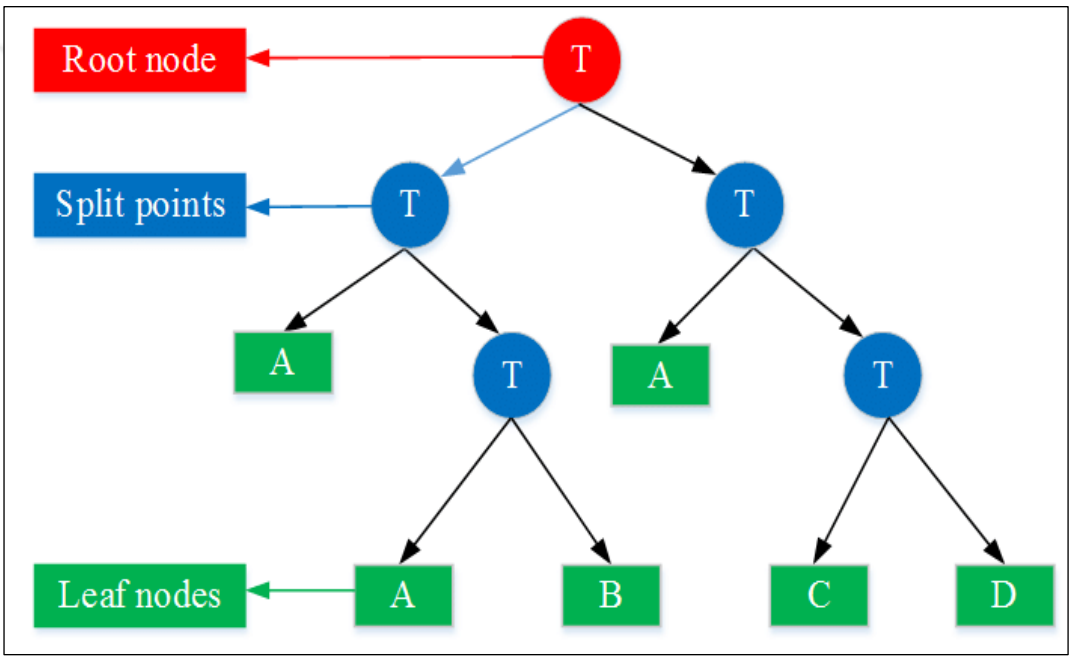


Figure 2.8: DT Example.

By examining patterns in network traffic, a DT method may be used in an intrusion detection system to find SQLIAs [50]. Recursively segmenting the data into subsets according to the characteristic that yields the greatest information gain is how the algorithm creates a model. This model would then be used by the system to categorize any new network traffic as either valid_querry or anomalous. For instance, the system might identify a new packet of network data as an attack if it closely resembled previously recognized SQLIA patterns.

The DT method has the advantages of being easy to develop, analyze, and understand, as well as being computationally efficient. It may be used in a feature selection process and is

excellent for managing high-dimensional data, but it may have overfitting issues and isn't as reliable as other more sophisticated models like RF or GB.

In conclusion, the DT algorithm is an effective method for intrusion detection, particularly for the detection of SQLIAs. It is straightforward to understand and interpret, simple to implement, and computationally efficient, but it may not be as reliable as other models. To address the overfitting issue, it is crucial to use cross-validation, pruning, and other techniques.

2.6 LITERATURE REVIEW

In a study [51], researchers developed their own dataset with unique SQL syntax and manually verified it to assess and compare the performance of 23 ML classifiers. They applied various algorithms, including "coarse k-NN, BT, linear SVM, fine k-NN, medium k-NN, RUS BT, SD, BT, W-k-NN, C-k-NN, LD, MT, S-k-NN, simple tree, Q-D, C-SVM, F-G-SVM, cosine k-NN, CT, LR, and EB. The five models that produced the highest levels of ACC were LD, BT, C-SVM, F-G-SVM, and EB.

By examining online access logs and extracting characteristics based on access behaviour mining and a grammatical pattern recognizer, researchers. [52] created a model to identify SQLIAs. The goal of the model was to find SQL injection statements in the training data that had never been observed before. They experimented with the NB, RF, SVM, IDthree, and k-means ML methods. According to the findings, the models based on RF and IDthree performed the best at spotting SQL injection attempts.

In a study [54], authors proposed a hybrid CNN-BiLSTM model for the purpose of detecting SQLIAs and evaluated its performance against existing ML techniques. Results showed that the CNN-BiLSTM method had an ACC rate of almost 98%

Seven ML models were trained using supervised learning in [53] using DT, Ada-Boost, RF, OL, TL, D-ANN, and a BT classifier. The RF classifier fared the best, reaching an ACC of 99.8%, when they compared the models' ACC and performance.

In a study [57], the authors trained an SVM to recognize malicious SQL queries by modeling a query's WHERE clause as an interaction network of tokens and calculating the centrality of the nodes. Three centrality metrics were used and were shown to be successful in

recognizing SQL injection attempts with no performance impact on a dataset gathered from five web apps using automated attack tools.

After reviewing the current SQLID techniques in an intelligent transportation system, an LSTM-based SID approach and a mechanism for creating SQL injection samples to augment the dataset were presented here [58]. Our method mimics SQLIAs and generates genuine positive samples to tackle the issue of overfitting brought on by a lack of positive data. The experimental results showed that the proposed method has ACC, PREC, and F1-S values > 92%.

A method for mitigating SQL injection using server-side scripting, employing ML and compiler platforms, was proposed in [59]. Four ML models—BDT, DT, SVM, and ANN—were trained using a dataset of enormous instances of SQL instructions. The findings showed that, among the evaluated models, the DT technique had the highest prediction ACC.

In [60] used the AdaBoost approach to discover SQL injection risks by converting the data into stumps and rating them as weak or strong depending on the weight they add to the final output. The results showed that the system recognized injection assaults properly and effectively.

Researchers in [61] proposed a technique for classifying dynamic SQL queries as either attacks or valid querry using a web profile built during the training phase. They used two datasets to classify the data and achieved detection rates of 91% and 90% using Naive Bayes, SVM, and parse trees.

Using DT techniques, author in [62] provided a technique for identifying various SQL injection levels. The suggested model successfully classified assaults as simple, unified, or lateral with an ACC of 92% and detected SQL injection attempts with 98% ACC.

In [63] developed a method for artificial neural network based SQLIA detection by analyzing a significant amount of SQL injection data to extract relevant features. MLP and LSTM were among the neural network models that were trained, and the findings revealed that MLP had a greater detection rate than LSTM.

In [64] objective was to use reinforcement learning agents to automate the process of exploiting SQLIAs by treating the problem as a Markov decision process. The findings

suggested that security assessment and penetration testing could be performed in the future using reinforcement learning agents.

In [65] developed a detection method by modelling SQL queries as a network of tokens, then training SVM classifiers on the token centrality metric. The findings showed that the recommended approach is successful at identifying fraudulent SQL queries. Different SVM classifiers were used to test the system utilizing both directed and undirected graphs.

In [66] published a TCSVM model to forecast binary labelled outcomes on the success or failure of a SQLIA in a web request. This method employed ML predictive analytics to block SQLIAs and intercept web requests at the proxy level.

This paper [67] introduced a unique approach for categorizing SQL queries, where the similarity metric between query strings is computed by using a gap-weighted string subsequence kernel algorithm. The legitimacy of the query strings was then determined using an SVM that was trained on these similarity measures. The proposed technique has an ACC of 92.48% after being tested on various datasets.

The outcomes revealed that the last two algorithms had a 97.6% ACC rate. The authors [68] also trained and assessed six classification algorithms: DT, SVM, RF, k-NN, NN, and multilayer perceptron. They also offered a novel approach for producing datasets utilizing a NoSQL query database.

The suggested technique has an ACC of 97.897% when used in [69] to train a progressive neural network model using a naïve Bayesian classifier to accurately identify SQLIAs utilizing parameters including error-based, time-based, SQL query, and union-based SQL injection assaults.

A tool that combines ML and natural language processing techniques to detect SID was introduced in [70], and the approach demonstrated excellent ACC and PREC in recognizing various types of SQLIAs.

Through the use of a deep belief network (DBN) model to pick just required characteristics, authors in [71] suggested a deep learning strategy to identify SQL injection threats in network data. They evaluated the effectiveness of several models, LSTM, CNN, and MLP, and found that the DBN model's ACC rate was 96%.

For the purpose of identifying numerous online application vulnerabilities, including SQL Injection, XSS, CSRF, and buffer overflow, the authors of the study [72] suggest a hybrid string matching technique called KMPS. To find dangerous patterns in URLs, the method combines the shifting step from Sunday Search Algorithm and the matching logic from KMP Algorithm. The Boyer Moore Algorithm and the suggested approach are compared by the authors, and it is discovered that KMPS outperforms it in terms of search time, throughput, and ACC. Testing the KMPS algorithm on 120 URLs allows for an evaluation of its performance. The authors stress in their conclusion that while vulnerabilities cannot entirely be eradicated, multiple techniques may be used to identify them at various phases of web application development.

The authors of the study [73] describe a method for authentication called SQL Injection Protector (SQLIPA). This method focuses on guarding against tautology SQLIAs. The method creates hash values for the username and password input fields using a hashing process. When a new user establishes an account, hash values are created for the username and password fields, and these values, together with the username and password columns, are kept in distinct columns in the database. The login credentials are verified against the database's stored values whenever a user tries to log in. The resulting hash values are then compared to the username and password's stored hash values. This additional security measure guards against tautology injection attacks.

The authors of the study [74] offer AMNESIA, a tool for identifying and thwarting SQLIAs (SQLIA). To find and stop SQLIA, the method combines static analysis with runtime monitoring. The program employs static analysis to locate problem areas and create a model of valid SQL queries. Runtime monitoring then compares incoming queries to this model. The method is automated and simply needs a web application as input; no other runtime environment configuration is necessary. The empirical analysis demonstrates the tool's effectiveness in identifying and preventing SQLIA. A disadvantage of the suggested approach is that it depends on Java libraries. In some circumstances, the method may also result in false positives and false negatives.

The authors of the publication [75] offer a method for identifying and stopping SQLIA that makes use of encryption using stored procedures. The method entails creating a secret key for the user-entered data in the intermediate layer and storing the encrypted data in the database. The authors suggest creating dynamic hashes using AES ENCRYPT (). Due to the secret key's unknown and dynamic generation, the attacker cannot infect the web application using this approach. This strategy helps to stop SQLIA by adding an extra layer of protection to the online application.

The authors of the work [76] compare and contrast several string-matching methods. Naive string matching, Brute force, Rabin-Karp, Boyer-Moore, Knuth-Morris-Pratt, Aho-Corasick, and Commentz-Walter algorithms are among the methods researched. Performance, temporal complexity, and spatial complexity are among the measures used to assess the algorithms. In order to choose the best method for a given use case based on the necessary performance characteristics, this research aims to offer a thorough examination of several string-matching algorithms.

The study in [77] suggests new guidelines for Snort, a signature-based intrusion detection system, in an effort to enhance its capacity to recognize SQL injection assaults. The new regulations are evaluated on a dataset of SQL-injected and regular websites, and they address concerns such attackers' use of hexadecimal numbers, white spaces, and comments in SQL injection assaults. According to the study, the suggested criteria have a high REC rate, proving their efficacy in identifying SQLIAs.

The study in [78] suggests new guidelines for Snort, a signature-based intrusion detection system, in an effort to enhance its capacity to recognize SQL injection assaults. The new regulations are evaluated on a dataset of SQL-injected and regular websites, and they address concerns such attackers' use of hexadecimal numbers, white spaces, and comments in SQL injection assaults. According to the study, the suggested criteria have a high REC rate, proving their efficacy in identifying SQLIAs. The study also assesses the effect of the suggested strategy on query processing time and discovers that it utilizes a lower query processing time than dynamic SQL without producing significant overhead that may cause a high response time. According to the study, more enhancements to database security are required, especially in network systems where many elements need to be taken into account, such as the servers being utilized, network traffic, and assaults from multiple sources.

The study [79] evaluates the weaknesses brought on by SQLIAs and offers protection strategies that take into account all weaknesses and recognize SQLIAs. The article also offers reasonably priced and simple-to-configure methods for preventing SQLIAs. Experimental testing has shown that the preventive strategies effectively detect and guard against SQLIAs. The study claims that the preventative strategies under discussion have been successfully evaluated and put into practice, with positive outcomes.

Several studies have examined various ML techniques for detecting SQLIAs, including AdaBoost, DT, NN, RF, SVM, LSTM and Hybrid approaches. These methods utilize a variety of techniques such as pattern matching, web profile analysis, token modeling, graph analysis, feature extraction, and natural language processing to extract key information from SQL queries. By training these algorithms using labeled datasets of both malicious and valid_querry SQL queries, these studies have found that those methods have demonstrated high levels of ACC in detecting SQLIAs and may be suitable for use in IDS.

3. METHODOLOGY

3.1 INTRODUCTION

This chapter focuses on the methodology and process used for detecting SQLIAs. The project consists of two major steps: pre-processing and algorithm implementation. In the pre-processing step, we use two different datasets, the necessary data is cleaned and prepared for analysis. The second step involves the implementation of seven ML algorithms including KNN, DTs, RF, Linear SVM, RBF SVM, XGB Classifier, and Gradient Boosting to classify and detect SQLIAs.

3.2 THESIS APPROACH

The proposed SID approach involves the following steps as depicted in the figure:

a. Collecting a dataset of SQL injection attempts

b. Pre-processing the data to ensure high ACC.

c. Training the models using a subset of the collected data.

d. Using the trained models to predict and detect SQL injection attempts using a testing subset of the collected data.

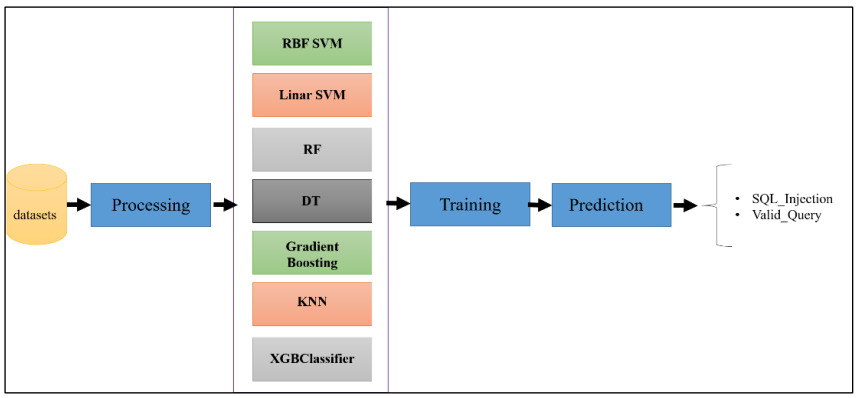


Figure 3.1 :Proposed SQLID Classification Approach.

3.3 DATASET COLLECTION

A SQLID dataset with only two columns, "sentence" and "label," would most likely be used to train a ML model to recognize SQL injection assaults. The "Sentence" column would contain different SQL query or web request characteristics needed to identify a possible SQLIA. These characteristics may include the duration of the query, the inclusion of certain keywords or characters, and other characteristics that might indicate the presence of a malicious query.

The "label" field indicates if the query or request is valid_querry (labeled as "0") or a SQLIA (labeled as "1"). Using the features to discover patterns and correlations between the query and its label, the ML model would be trained using this dataset, and then utilize that knowledge to predict the label of future incoming data.

Using two datasets with the same structure but different sizes for SQL injection can be a beneficial strategy for testing a model's performance in a more robust and generalizable fashion. By testing the model on several data distributions and data sizes, any biases or flaws in the model that may not be obvious when using only one dataset may be identified. This can aid in determining the model's scalability and performance on a bigger dataset, as well as identifying any overfitting to a given dataset. Furthermore, testing the model on two separate datasets can aid in determining if the findings gained from one dataset are consistent with the results obtained from the second dataset, allowing for more reliable conclusions to be drawn. Always consult with your thesis supervisor to ensure that utilizing several datasets is suitable for your study.

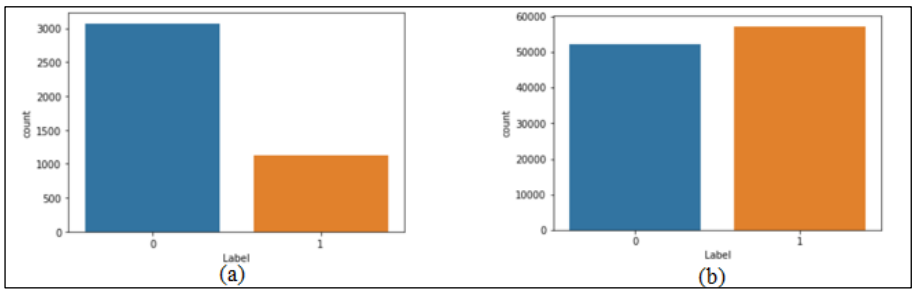


Figure 3.2: Dataset Class Labels Distribution; (a). First Dataset, (b). Second Dataset. Dataset Pre-Processing.

Data pre-processing is a vital step in preparing real-world data for use in ML and deep learning techniques. Raw data often contains inconsistencies, incompleteness, and errors, making it challenging for algorithms to process. Pre-processing helps to organize the data in a way that is more suitable for these techniques, and can also decrease the computational resources and training time required to achieve accurate and reliable results.

In this specific scenario, the data is arranged in a tabular format and stored in a Data Frame. In order to prepare the data for use in ML and deep learning techniques, several pre-processing steps are applied. These steps include identifying and handling null or missing values and duplicate entries, extracting the labels of the data samples to identify the class categories (such as SQL injection or valid query), applying techniques such as label encoding and data normalization to standardize the dataset, and using the RF approach to extract relevant features as shown in Figure 5.

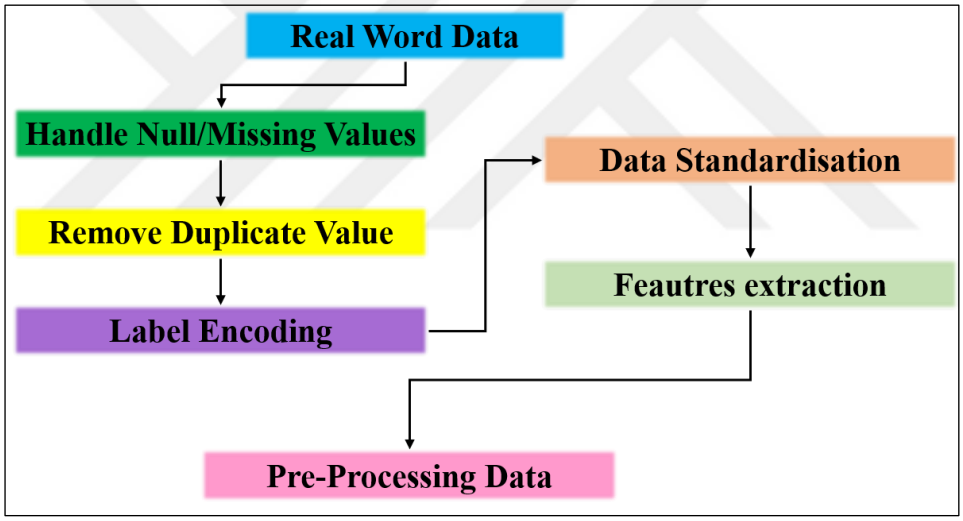


Figure 3.3: Pre-Processing Steps.

a. Handle Null/Missing Values

Handling missing values is an important step in data pre-processing. This step involves identifying and filling in missing data in the dataset. There are several ways to handle missing values, such as imputation, deletion, or interpolation. The choice of method will depend on the specific dataset and problem being addressed. In SQLID, it is important to handle missing values as they could affect the ACC of the detection model.

b. Remove Duplicate Value

Removing duplicate values is another important step in data pre-processing. Duplicate values can be caused by data entry errors, data integration issues, or other factors. Removing duplicate values can improve the performance of the detection model by reducing the amount of irrelevant data it has to process. This step can be done by using the. drop duplicates () method on the data frame.

c. Label Encoding

By using label encoding, categorical variables are converted into numerical values. Categorical variables are those that can only take on one of a limited set of values. This step is important in SQLID because many features in the dataset may be categorical, and most ML algorithms require numerical input. Label encoding can be done using the sklearn pre-processing Label Encoder class.

d. Data Standardisation

Data standardization is the process of transforming data into a common format. This step is important in SQLID because it can help to make the data more consistent and easier to work with. This can be done by using the sklearn. Pre-processing. Standard Scaler class.

e. Features Extraction

Feature extraction is the process of creating new features from existing data. This step is important in SQLID because it can help to identify patterns and relationships in the data that are not immediately obvious. This can be done by using different types of feature selection method, such as mutual information, correlation-based feature selection or recursive feature elimination.

It is worth mentioning that the order of these steps is not fixed, it could vary depending on the specific dataset and problem being addressed.

3.4 BUILD METHODS

This study involved the construction of seven different models using pre-processed data: K-KNN, DTs, RF, Linear SVM, RBF SVM, XGBoost Classifier and GB. These models were created after the data was cleaned, transformed and prepared for modelling.

```
from sklearn.linear_model import LogisticRegression# For LogisticRegression method
from sklearn.ensemble import RandomForestClassifier # For  Random forest method
from sklearn.svm import SVC# For both  RBF and linear SVM method
from sklearn.neighbors import KNeighborsClassifier#For  KNN method
from sklearn.ensemble import GradientBoostingClassifier#For Gradient Boosting method
from xgboost import XGBClassifier#For xgboost method
```

Figure 3.4: The Proposed Models.

3.4.1 Libraries

In this project, we made use of several powerful libraries in Python for data handling and ML. These include NumPy, Matplotlib, Seaborn, Keras, and Pandas.

NumPy is a library that provides powerful numerical calculations, including multi-dimensional arrays and mathematical operations. This library is essential for processing large for implementing ML algorithms in Python.

Matplotlib is a data visualization library that allows us to create various types of plots and charts to help understand and interpret the data. This library is useful for visualizing trends, patterns and relationships in the data and for creating publication-quality figures.

A library for statistical graphics charting is called Seaborn. It is based on Matplotlib and offers a more advanced interface for designing attractive and instructive statistical visuals. For investigating connections between various variables and for visualizing statistical models, Seaborn is extremely helpful.

A high-level NN library created in Python is called Kera's. It allows us to easily build and train deep learning models, which are particularly useful for tasks such as image and speech recognition, natural language processing and time series forecasting.

Pandas is a strong library for manipulating and analysing data. It offers data structures and tools for data analysis that make working with structured data in Python simple. Pandas is particularly useful for data cleaning, preparation and Pre-processing, allowing us to easily handle missing values, outliers, and other issues.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from tensorflow.keras import models, layers
```

Figure 3.5: The Used Libraries.

In summary, these libraries provide valuable resources for handling data and implementing ML algorithms in Python, and are widely used in data science and ML projects.

3.4.2 Training

In this project, the train test split tool from scikit-learn was used to divide the dataset into training and testing sets for SQLID. The function takes in the features and labels for the dataset (X_train and Y_train, respectively) and a train size parameter to specify the proportion of the data that should be used for training. The test set is automatically determined based on the train size. In this case, 80% of the data is used for training and the remaining 20% is used for testing. The random_state parameter specifies the random seed to use for the data splitting, ensuring that the same split is obtained each time the code is run.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)# For the test_size equal to 0.2 equivalent of 20%
```

Figure 3.6: Test and Train Data Identification.

The predict step involves passing the new data through the model and using the trained model to generate predictions about the malicious or benign nature of SQL injection.

3.5 CHAPTER SUMMARY

The methodology utilized in this study has been thoroughly described in this chapter, including a comprehensive discussion of the flowchart and the step-by-step process of implementing the Python programming. We have also outlined the different ML algorithms that were proposed and used in this work. In the next chapter, we will present a detailed analysis of the performance metrics, including the equations used to calculate them, and the results obtained from the simulation.

4. RESULTS

4.1 INTRODUCTION

In this chapter, we evaluate the proposed algorithms for SID on the dataset using the training and test sets. We trained, tested, and compared various ML models in terms of ACC, PREC, REC (REC), and F1-score(F1-S). The dataset was divided into training and testing sets using the train test split method, with 80% used for training and 20% used for testing. We used Python to assess the performance of the algorithms for detecting SQLIAs.

4.2 CLASSIFICATION METRICS

The confusion matrix is a method for assessing a classification model's performance by showing how many predictions were correct and incorrect. True positives, true negatives, false positives, and false negatives are four of its essential phrases. True positives refer to instances where the actual label is positive and the model also predicted it as positive. True negatives refer to instances where the actual label is negative and the model also predicted it as negative. False positives refer to instances where the model predicted a positive label, but the actual label was negative. False negatives refer to instances where the model predicted a negative label but the actual label was positive.

There are different performance measures that can be calculated using the information from the confusion matrix. ACC, PREC, REC, and F1-S are a few examples. ACC is calculated as the ratio of correct predictions to the total number of predictions. PREC refers to the percentage of positive predictions that were correct and is calculated as the ratio of true positives to the total number of true positives and false positives. REC is a measure of how well the model predicted actual positive cases, calculated as the ratio of true positives to the total number of true positives and false negatives. F1-S is a measure of the balance between ACC and REC, calculated as the harmonic mean of PREC and REC. A higher score indicates better balance between the two.

These metrics can be calculated using the following equations:

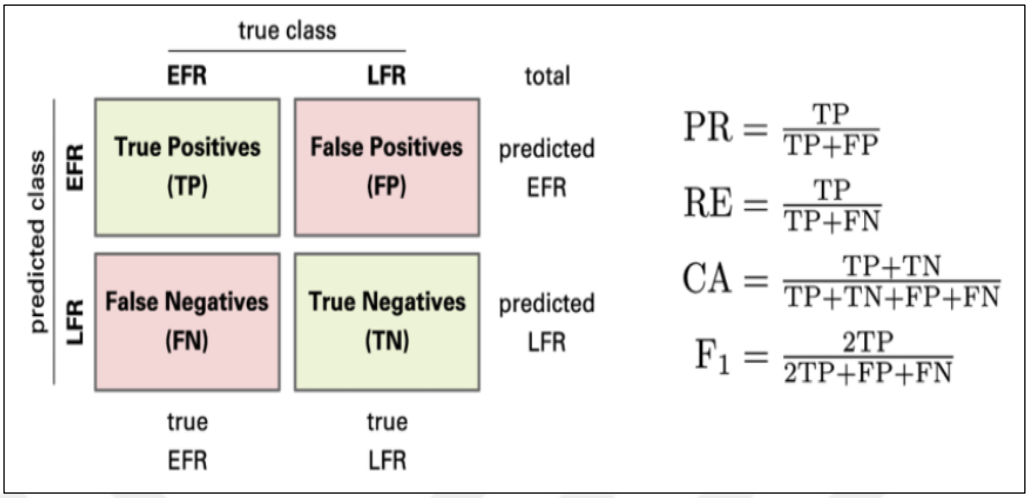


Figure 4.1: Confusion Matrix. with the Formulas of Precision (PR), Recall (RE), Accuracy (CA), and F 1-Measure [75].

Two crucial assessment criteria for classification models are sensitivity and specificity, especially in security or medical diagnostic procedures.

The fraction of real positive cases that the model properly detected is measured by sensitivity, often known as the true positive rate. It is determined as the proportion of real positive instances to all genuine positives.

a. Sensitivity = (True Positives) / (True Positives + False Negatives)

The percentage of genuine negative situations that the model accurately recognized is known as specificity or the real negative rate. The ratio of real negatives to all cases of actual negative results is used to compute it.

b. Specificity = (True Negatives) / (True Negatives + False Positives)

We may assess a model's sensitivity and specificity by looking at how effectively it can distinguish between positive and negative examples in a binary classification task. High specificity means that the model can identify most negative situations, while high sensitivity suggests that it can identify most positive cases.

In the simulation results, these metrics will be used to evaluate the performance of the proposed models for SQLID.

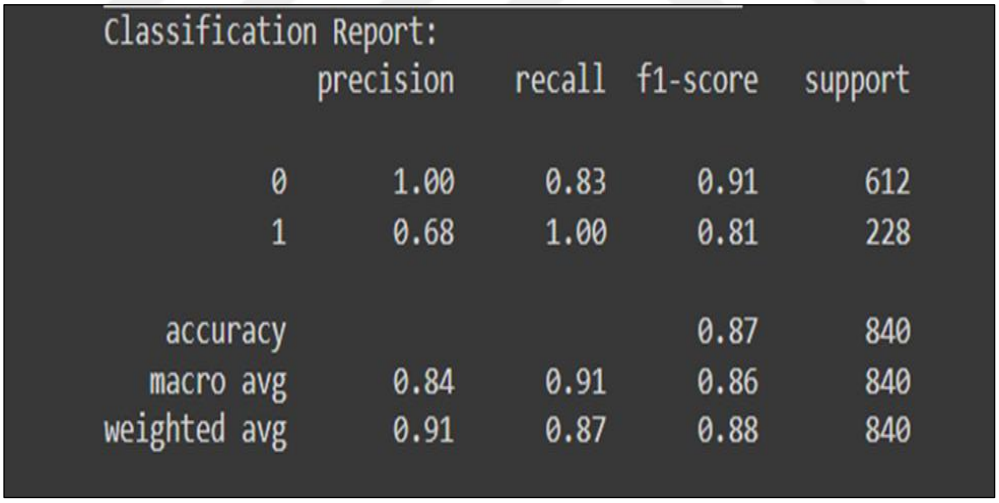
4.3 EVALUATION RESULTS

Using evaluation metrics like ACC, PREC, REC, and F1-S, we will provide the outcomes of each ML approach utilized for SQLID with two distinct datasets in this section. These metrics are commonly used to assess the performance of a classifier or predictor and provide valuable insight into the strengths and weaknesses of each method. The results of each method will be compared and analyzed to determine the most effective approach for SQLID.

4.4 DT RESULTS

4.4.1 First Dataset

The classification report for the DT classifier in Figure 9 shows that the model has excellent performance in terms of PREC, REC, and f1-S for both the anomaly class and the valid_querry label, with a 68%, 100%, and 81% respectively score. Additionally, the model reaches an ACC of 87%. This indicates which the model is capable to correctly identify both injection and valid query cases in the dataset.



Classification Report:

	precision	recall	f1-score	support
0	1.00	0.83	0.91	612
1	0.68	1.00	0.81	228
accuracy			0.87	840
macro avg	0.84	0.91	0.86	840
weighted avg	0.91	0.87	0.88	840

Figure 4.2: DT Outperforms_1.

The confusion matrix in Figure 10 also provides valuable information. It shows that the model has a high ability to correctly predict valid_querry instances with 506, and zero for the true negative injection attack.

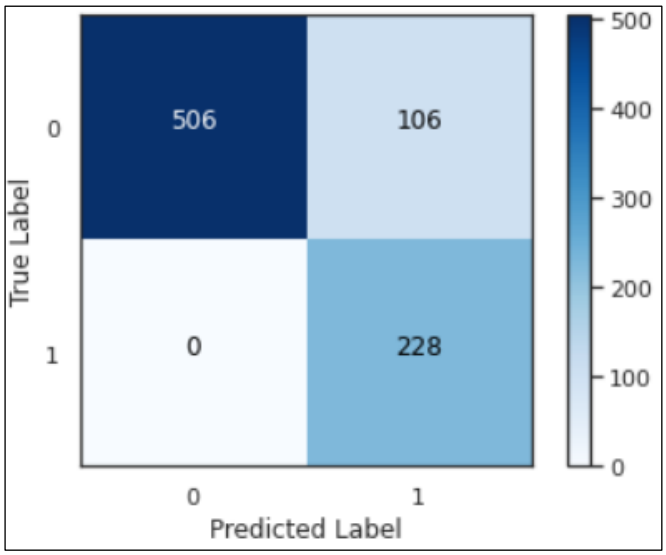


Figure 4.3: DT Confusion Matrix_1.

In the case of the SQLID, a high sensitivity and specificity would indicate which the model is capable to accurately identify a large proportion of both actual positive (SQLIAs) with 100% and actual negative cases with 82%. This would suggest which the model is capable to effectively detect SQLIAs while minimizing the number of false positives (normal cases incorrectly identified as SQLIAs). Based on the results provided, it can be inferred that the model has a good sensitivity and specificity for SQLID.

```
Sensitivity: 1.0
Specificity: 0.826797385620915
```

Figure 4.4: DT Sensitivity and Specificity_1.

Overall, the results demonstrate that the DT classifier is a strong performer for SQLID.

4.4.2 Second Dataset

Figure 12 shows that the DT classifier performs well in identifying SQLIAs and valid queries. The PREC is 93%, REC is 92%, F1-S is 93% and ACC is 87%. These results indicate that the classifier is efficient in detecting SQLIAs while maintaining a good balance of PREC and REC.

```
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.93      0.92      5216
           1       0.93      0.92      0.93      5736

    accuracy                           0.92     10952
   macro avg       0.92      0.92      0.92     10952
weighted avg       0.92      0.92      0.92     10952
```

Figure 4.5: DT Outperforms_2.

The confusion matrix in Figure 13 also offers valuable information. It illustrates that the model has a high ability to correctly predict valid_querry instances, with a count of 5270 for true negatives and 384 for false positives (normal instances incorrectly identified as SQLIAs).

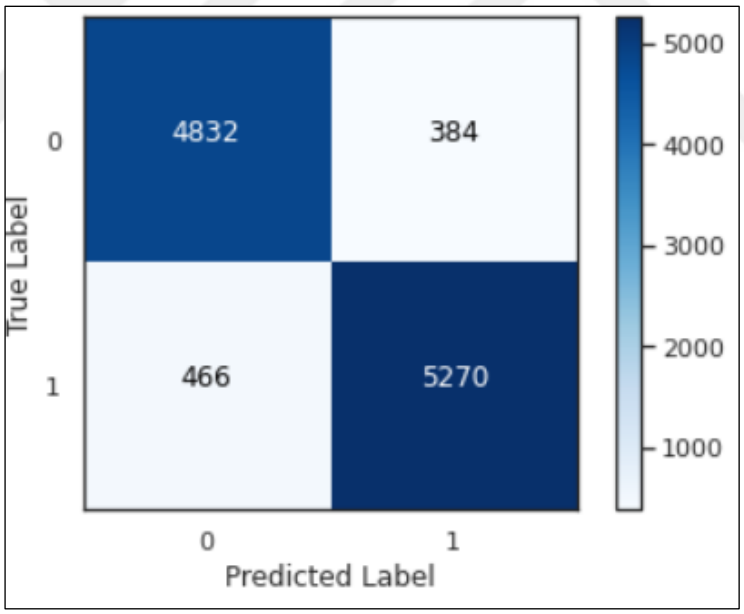


Figure 4.6: DT Confusion Matrix_2.

A high sensitivity and specificity would mean which the model is capable to reliably identify a significant number of both genuine positive instances (SQLIAs) with 91% and actual negative cases with 92% in the case of SQLID. This would indicate that the model can efficiently identify SQLIAs while reducing the number of false positives (normal cases

incorrectly identified as SQLIAs). The model has a decent sensitivity and specificity for SQLID, according to the supplied findings.

```
Sensitivity: 0.9187587168758717
Specificity: 0.9263803680981595
```

Figure 4.7: DT Sensitivity and Specificity_2.

4.5 KNN RESULTS

4.5.1 First Dataset

The classification report for the KNN in Figure 15 shows that the model has outstanding performance in terms of PREC, REC, and F1-S for both the SQl_injection class and the valid_querry label, with 35%, 97%, and 51% respectively. Additionally, the model reaches an ACC of 50%.

Classification Report:

	precision	recall	f1-score	support
0	0.97	0.32	0.49	612
1	0.35	0.97	0.51	228
accuracy			0.50	840
macro avg	0.66	0.65	0.50	840
weighted avg	0.80	0.50	0.49	840

Figure 4.8: KNN Outperforms_1.

The confusion matrix in Figure 10 also offers valuable information. It illustrates that the model has a high ability to correctly predict valid_querry instances, with a count of 222 for true negatives and 414 for false positives (normal instances incorrectly identified as SQLIAs).

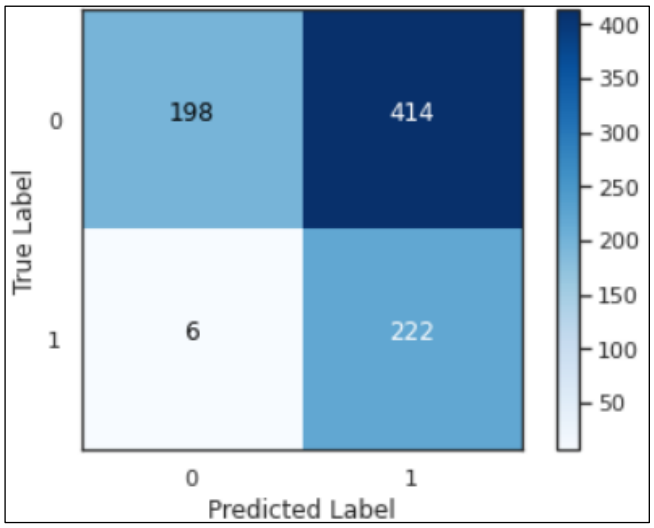


Figure 4.9: KNN Confusion Matrix_1.

A high sensitivity and specificity would mean which the model is capable to reliably identify a significant number of both genuine positive instances (SQLIAs) with 97% and actual negative cases with 32% in the case of SQLID. This would indicate that the model can efficiently identify SQLIAs while reducing the number of false positives (normal cases incorrectly identified as SQLIAs). The model has a decent sensitivity and specificity for SQLID, according to the supplied findings.

```
Sensitivity: 0.9736842105263158
Specificity: 0.3235294117647059
```

Figure 4.10: KNN Sensitivity and Specificity_1.

4.5.2 Second Dataset

The classification report for the KNN in Figure 18 shows that the model has outstanding performance in terms of PREC, REC, and F1-S for both the SQL injection class and the valid_querry label, with 75%, 65%, and 70% respectively. Additionally, the model reaches an ACC of 70%.

```
Classification Report:
              precision    recall  f1-score   support

           0       0.66      0.76      0.71      5216
           1       0.75      0.65      0.70      5736

    accuracy                           0.70     10952
   macro avg       0.71      0.71      0.70     10952
weighted avg       0.71      0.70      0.70     10952
```

Figure 4.11: KNN Outperforms_2.

The confusion matrix in Figure 19 also offers valuable information. It illustrates that the model has a high ability to correctly predict valid_querry instances, with a count of 3728 for true negatives and 1235 for false positives (normal instances incorrectly identified as SQLIAs).

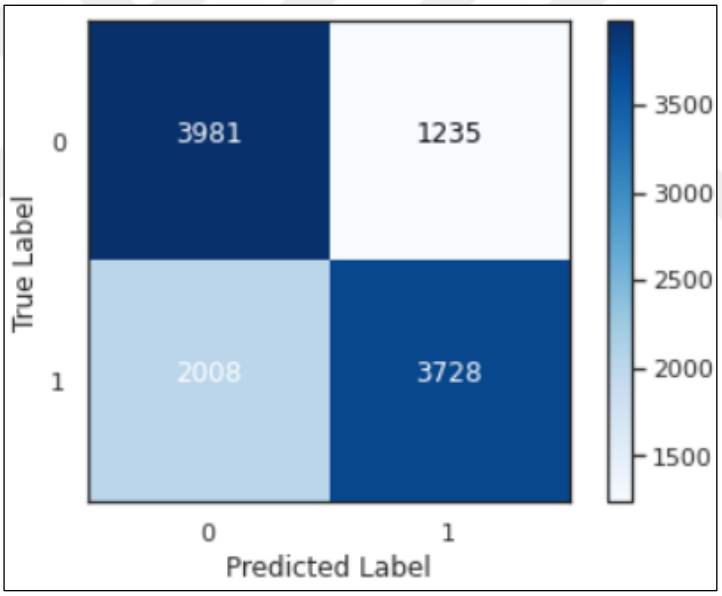


Figure 4.12: KNN Confusion Matrix_2.

A high sensitivity and specificity would mean which the model is capable to reliably identify a significant number of both genuine positive instances (SQLIAs) with 97% and actual negative cases with 32% in the case of SQLID. This would indicate that the model can efficiently identify SQLIAs while reducing the number of false positives (normal cases incorrectly identified as SQLIAs). The model has a decent sensitivity and specificity for SQLID, according to the supplied findings.

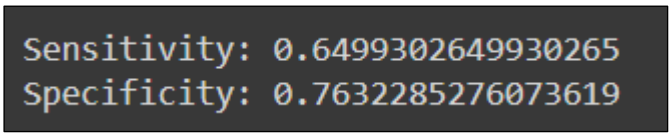
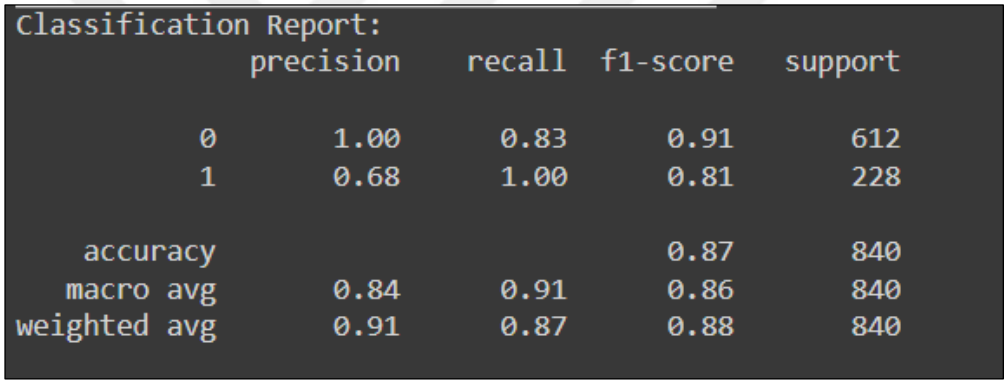


Figure 4.13: KNN Sensitivity and Specificity_2.

4.6 RF RESULTS

4.6.1 First Dataset

The classification report for the KNN in Figure 21 shows that the model has outstanding performance in terms of PREC, REC, and F1-S for both the SQL injection class and the valid_querry label, with 68%, 100%, and 81% respectively. Additionally, the model reaches an ACC of 87%.



Classification Report:

	precision	recall	f1-score	support
0	1.00	0.83	0.91	612
1	0.68	1.00	0.81	228
accuracy			0.87	840
macro avg	0.84	0.91	0.86	840
weighted avg	0.91	0.87	0.88	840

Figure 4.14: RF Outperforms_1.

The confusion matrix in Figure 22 also offers valuable information. It illustrates that the model has a high ability to correctly predict valid_querry instances, with a count of 228 for true negatives and 106 for false positives (normal instances incorrectly identified as SQLIAs).

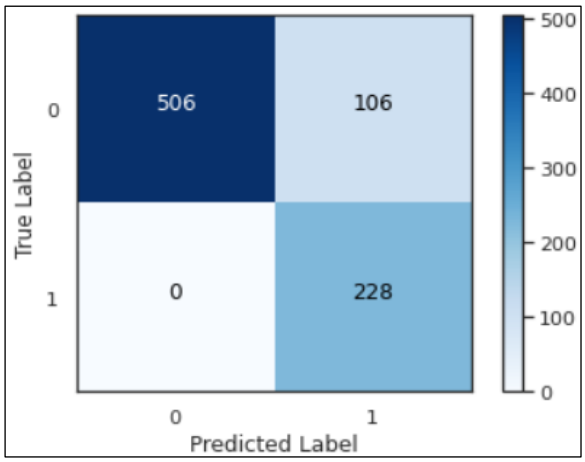


Figure 4.15: RF Confusion Matrix_1.

A high sensitivity and specificity would mean which the model is capable to reliably identify a significant number of both genuine positive instances (SQLIAs) with 100% and actual negative cases with 82% in the case of SQLID. This would indicate that the model can efficiently identify SQLIAs while reducing the number of false positives (normal cases incorrectly identified as SQLIAs). The model has a decent sensitivity and specificity for SQLID, according to the supplied findings.

```
Sensitivity: 1.0
Specificity: 0.826797385620915
```

Figure 4.16: RF Sensitivity and Specificity_1.

4.6.2 Second Dataset

The classification report for the KNN in Figure 24 shows that the model has outstanding performance in terms of PREC, REC, and F1-S for both the SQL injection class and the valid_querry label, with 93%, 92%, and 93% respectively. Additionally, the model reaches an ACC of 92%.

```
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.93      0.92      5216
           1       0.93      0.92      0.93      5736

    accuracy                           0.92     10952
   macro avg       0.92      0.92      0.92     10952
weighted avg       0.92      0.92      0.92     10952
```

Figure 4.17: RF Outperforms_2.

The confusion matrix in Figure 25 also offers valuable information. It illustrates that the model has a high ability to correctly predict valid_querry instances, with a count of 5270 for true negatives and 384 for false positives (normal instances incorrectly identified as SQLIAs).

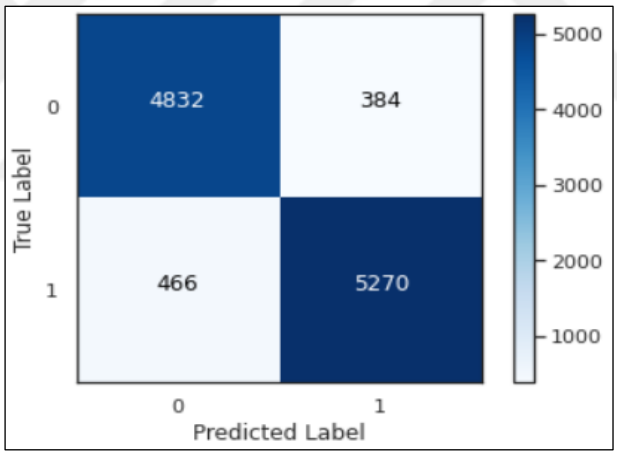


Figure 4.18: RF Confusion Matrix_2.

A high sensitivity and specificity would mean which the model is capable to reliably identify a significant number of both genuine positive instances (SQLIAs) with 97% and actual negative cases with 32% in the case of SQLID. This would indicate that the model can efficiently identify SQLIAs while reducing the number of false positives (normal cases incorrectly identified as SQLIAs). The model has a decent sensitivity and specificity for SQLID, according to the supplied findings.

```
Sensitivity: 0.6499302649930265
Specificity: 0.7632285276073619
```

Figure 4.19: RF Sensitivity and Specificity_2.

4.7 GRADIENT BOOSTING RESULTS

4.7.1 First Dataset

The classification report for the Gradient Boosting in Figure 27 shows that the model has outstanding performance in terms of PREC, REC, and F1-S for both the SQL injection class and the valid query label, with 100%, 66%, and 80% respectively. Additionally, the model reaches an ACC of 91%.

```
Classification Report:
              precision    recall  f1-score   support

           0       0.89      1.00      0.94       612
           1       1.00      0.66      0.80       228

    accuracy                           0.91       840
   macro avg       0.94      0.83      0.87       840
weighted avg       0.92      0.91      0.90       840
```

Figure 4.20: Gradient Boosting: Outperforms_1.

The confusion matrix in Figure 28 also offers valuable information. It illustrates that the model has a high ability to correctly predict valid quarry instances, with a count of 151 for true negatives and 0 for false positives (normal instances incorrectly identified as SQLIAs).

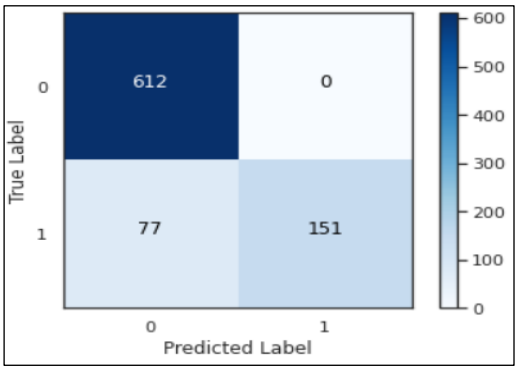


Figure 4.21: Gradient Boosting: Confusion Matrix_1.

A high sensitivity and specificity would mean which the model is capable to reliably identify a significant number of both genuine positive instances (SQLIAs) with 66% and actual negative cases with 100% in the case of SQLID. This would indicate that the model can efficiently identify SQLIAs while reducing the number of false positives (normal cases incorrectly identified as SQLIAs). The model has a decent sensitivity and specificity for SQLID, according to the supplied findings.

```
Sensitivity: 0.6622807017543859
Specificity: 1.0
```

Figure 4.22: Gradient Boosting: Sensitivity and Specificity_1.

4.7.2 Second Dataset

The classification report for the KNN in Figure 30 shows that the model has outstanding performance in terms of PREC, REC, and F1-S for both the SQL injection class and the valid_querry label, with 96%, 91% and 93% respectively. Additionally, the model reaches an ACC of 93%.

Classification Report:

	precision	recall	f1-score	support
0	0.90	0.96	0.93	5216
1	0.96	0.91	0.93	5736
accuracy			0.93	10952
macro avg	0.93	0.93	0.93	10952
weighted avg	0.93	0.93	0.93	10952

Figure 4.23: Gradient Boosting: Outperforms_2.

The confusion matrix in Figure 31 also offers valuable information. It illustrates that the model has a high ability to correctly predict valid_querry instances, with a count of 5202 for true negatives and 234 for false positives (normal instances incorrectly identified as SQLIAs).

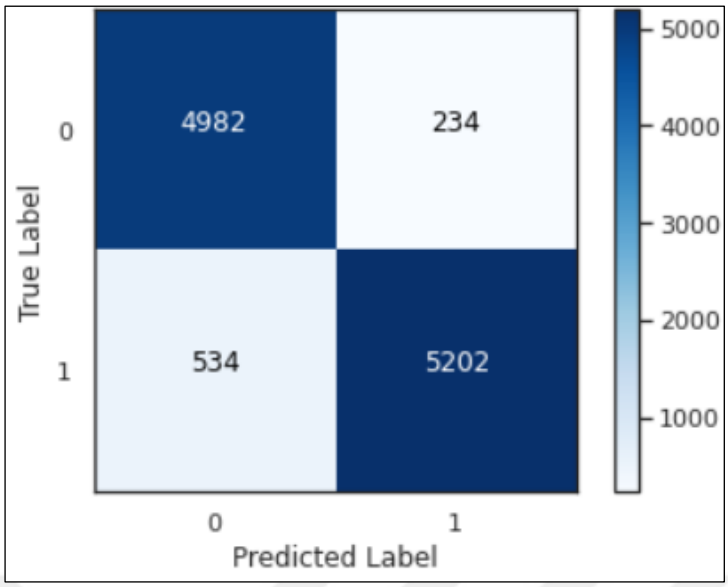


Figure 4.24: Gradient Boosting: Confusion Matrix_2.

A high sensitivity and specificity would mean which the model is capable to reliably identify a significant number of both genuine positive instances (SQLIAs) with 90% and actual negative cases with 95% in the case of SQLID. This would indicate that the model can efficiently identify SQLIAs while reducing the number of false positives (normal cases incorrectly identified as SQLIAs). The model has a decent sensitivity and specificity for SQLID, according to the supplied findings.

```
Sensitivity: 0.9069037656903766
Specificity: 0.9551380368098159
```

Figure 4.25: Gradient Boosting: Sensitivity and Specificity_2.

4.8 XGB CLASSIFIER

4.8.1 First Dataset

The classification report for the XGB Classifier in Figure 33 shows that the model has outstanding performance in terms of PREC, REC, and F1-S for both the SQL injection class and the valid_querry label, with 100%, 66%, and 80% respectively. Additionally, the model reaches an ACC of 91%.

```
Classification Report:
              precision    recall  f1-score   support

           0       0.88      1.00      0.94       612
           1       1.00      0.65      0.79       228

    accuracy                           0.90       840
   macro avg       0.94      0.82      0.86       840
weighted avg       0.92      0.90      0.90       840
```

Figure 4.26: XGB Classifier: Outperforms_1.

The confusion matrix in Figure 34 also offers valuable information. It illustrates that the model has a high ability to correctly predict valid_querry instances, with a count of 0 for true negatives and 148 for false positives (normal instances incorrectly identified as SQLIAs).

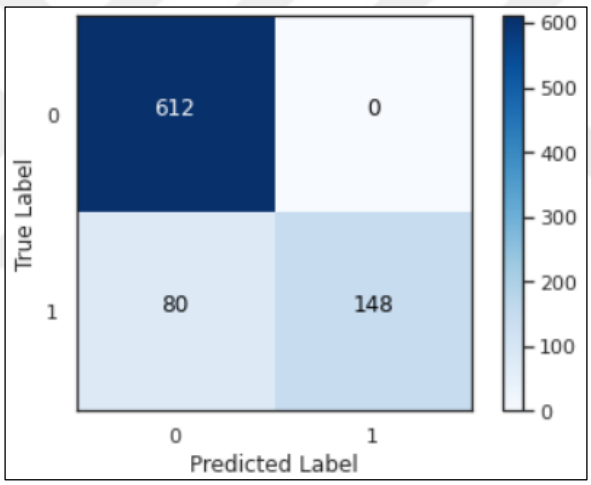


Figure 4.27: XGB Classifier: Confusion Matrix_1.

A high sensitivity and specificity would mean which the model is capable to reliably identify a significant number of both genuine positive instances (SQLIAs) with 64% and actual negative cases with 100% in the case of SQLID. This would indicate that the model can efficiently identify SQLIAs while reducing the number of false positives (normal cases incorrectly identified as SQLIAs). The model has a decent sensitivity and specificity for SQLID, according to the supplied findings.

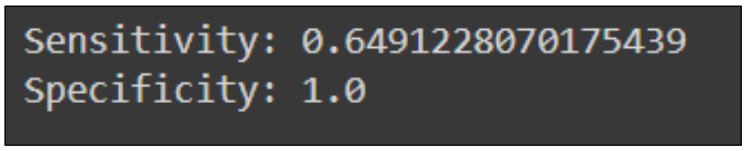
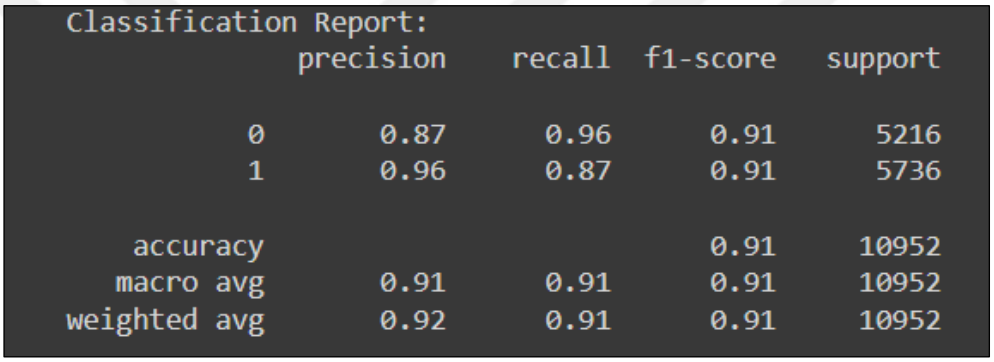


Figure 4.28: XGB Classifier: Sensitivity and Specificity_1.

4.8.2 Second Dataset

The classification report for the XGB Classifier in Figure 36 shows that the model has outstanding performance in terms of PREC, REC, and F1-S for both the SQL injection class and the valid_querry label, with 96%, 87% and 91% respectively. Additionally, the model reaches an ACC of 91%.



Classification Report:

	precision	recall	f1-score	support
0	0.87	0.96	0.91	5216
1	0.96	0.87	0.91	5736
accuracy			0.91	10952
macro avg	0.91	0.91	0.91	10952
weighted avg	0.92	0.91	0.91	10952

Figure 4.29: XGB Classifier: Outperforms_2.

The confusion matrix in Figure 37 also offers valuable information. It illustrates that the model has a high ability to correctly predict valid_querry instances, with a count of 4997 for true negatives and 224 for false positives (normal instances incorrectly identified as SQLIAs).

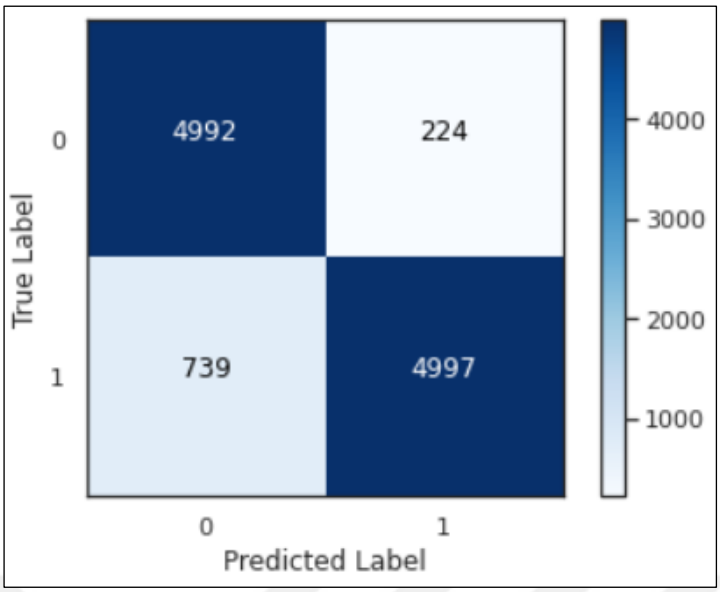


Figure 4.30: XGB Classifier: Confusion Matrix_2.

A high sensitivity and specificity would mean which the model is capable to reliably identify a significant number of both genuine positive instances (SQLIAs) with 87% and actual negative cases with 95% in the case of SQLID. This would indicate that the model can efficiently identify SQLIAs while reducing the number of false positives (normal cases incorrectly identified as SQLIAs). The model has a decent sensitivity and specificity for SQLID, according to the supplied findings.

```
Sensitivity: 0.8711645746164575
Specificity: 0.9570552147239264
```

Figure 4.31: XGB Classifier: Sensitivity and Specificity_2.

4.9 SVM (RBF)

4.9.1 First Dataset

The classification report for the SVM (RBF) in Figure 39 shows that the model has outstanding performance in terms of PREC, REC, and F1-S for both the SQl_injection class and the valid_querry label, with 86%, 95%, and 91% respectively. Additionally, the model reaches an ACC of 95%.

```
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.94      0.96       612
           1       0.86      0.95      0.91       228

    accuracy                           0.95       840
   macro avg       0.92      0.95      0.93       840
weighted avg       0.95      0.95      0.95       840
```

Figure 4.32: SVM (RBF): Outperforms_1.

The confusion matrix in Figure 40 also offers valuable information. It illustrates that the model has a high ability to correctly predict valid_querry instances, with a count of 217 for true negatives and 34 for false positives (normal instances incorrectly identified as SQLIAs).

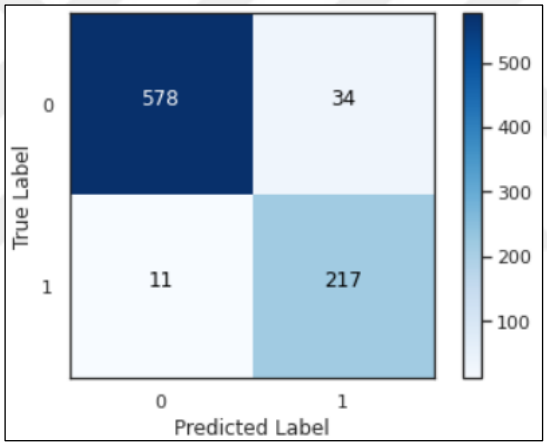


Figure 4.33: SVM (RBF): Confusion Matrix_1.

A high sensitivity and specificity would mean which the model is capable to reliably identify a significant number of both genuine positive instances (SQLIAs) with 95% and actual negative cases with 94% in the case of SQLID. This would indicate that the model can efficiently identify SQLIAs while reducing the number of false positives (normal cases incorrectly identified as SQLIAs). The model has a decent sensitivity and specificity for SQLID, according to the supplied findings.

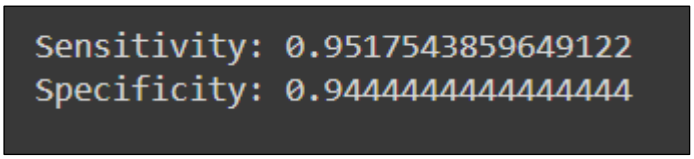
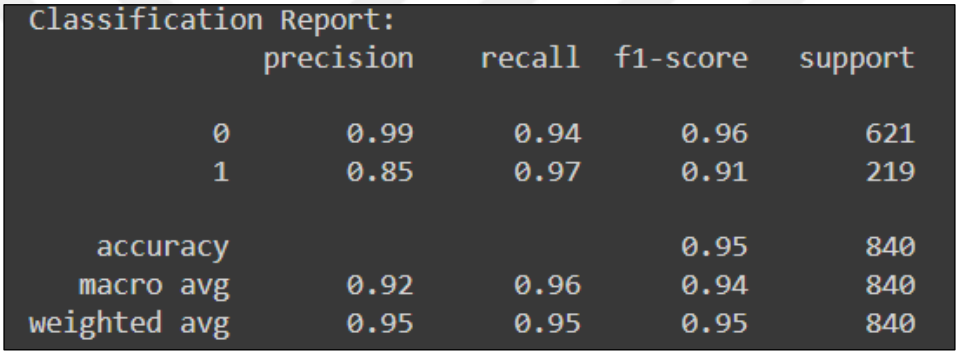


Figure 4.34: SVM (RBF): Sensitivity and Specificity_1.

4.9.2 Second Dataset

The classification report for the SVM (RBF): in Figure 43 shows that the model has outstanding performance in terms of PREC, REC, and F1-S for both the SQL injection class and the valid_querry label, with 85%, 97% and 91% respectively. Additionally, the model reaches an ACC of 95%.



Classification Report:

	precision	recall	f1-score	support
0	0.99	0.94	0.96	621
1	0.85	0.97	0.91	219
accuracy			0.95	840
macro avg	0.92	0.96	0.94	840
weighted avg	0.95	0.95	0.95	840

Figure 4.35: SVM (RBF): Outperforms_2.

The confusion matrix in Figure 44 also offers valuable information. It illustrates that the model has a high ability to correctly predict valid_querry instances, with a count of 212 for true negatives and 36 for false positives (normal instances incorrectly identified as SQLIAs).

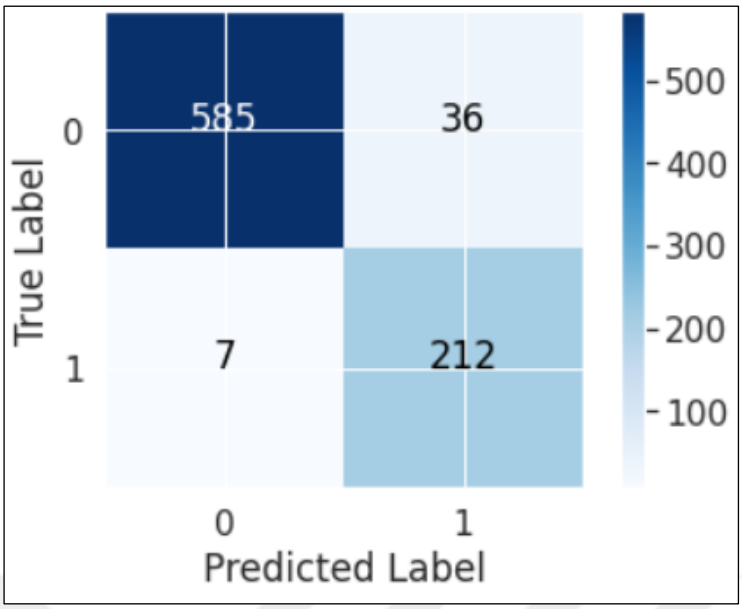


Figure 4.36: SVM (RBF): Confusion Matrix_2.

A high sensitivity and specificity would mean which the model is capable to reliably identify a significant number of both genuine positive instances (SQLIAs) with 96% and actual negative cases with 94% in the case of SQLID. This would indicate that the model can efficiently identify SQLIAs while reducing the number of false positives (normal cases incorrectly identified as SQLIAs). The model has a decent sensitivity and specificity for SQLID, according to the supplied findings.

```
Sensitivity: 0.9680365296803652
Specificity: 0.9420289855072463
```

Figure 4.37: SVM (RBF): Sensitivity and Specificity_2.

4.10 SVM (LINEAR)

4.10.1 First Dataset

The classification report for the SVM (linear) in Figure 39 shows that the model has outstanding performance in terms of PREC, REC, and F1-S for both the SQL injection class and the valid_querry label, with 86%, 100%, and 92% respectively. Additionally, the model reaches an ACC of 95%.

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.94      0.97       612
           1       0.86      1.00      0.92       228

    accuracy                           0.95       840
   macro avg       0.93      0.97      0.95       840
weighted avg       0.96      0.95      0.96       840
```

Figure 4.38: SVM (Linear) Outperforms_1.

The confusion matrix in Figure 40 also offers valuable information. It illustrates that the model has a high ability to correctly predict valid_querry instances, with a count of 277 for true negatives and 37 for false positives (normal instances incorrectly identified as SQLIAs).

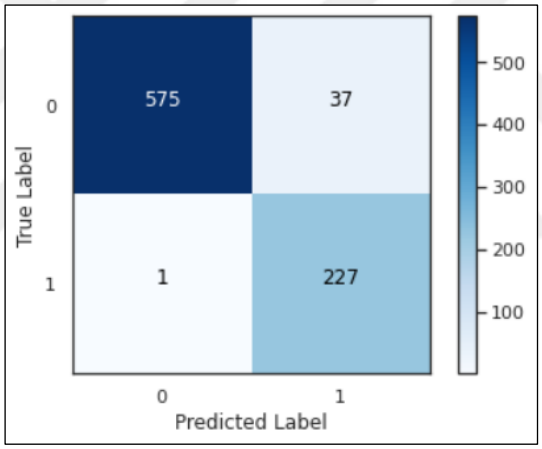


Figure 4.39: SVM (Linear): Confusion Matrix_1.

A high sensitivity and specificity would mean which the model is capable to reliably identify a significant number of both genuine positive instances (SQLIAs) with 99% and actual negative cases with 93% in the case of SQLID. This would indicate that the model can efficiently identify SQLIAs while reducing the number of false positives (normal cases incorrectly identified as SQLIAs). The model has a decent sensitivity and specificity for SQLID, according to the supplied findings.

```
Sensitivity: 0.9956140350877193
Specificity: 0.9395424836601307
```

Figure 4.40: SVM (Linear): Sensitivity and Specificity_1.

4.10.2 Second Dataset

The classification report for the SVM (RBF): in Figure 43 shows that the model has outstanding performance in terms of PREC, REC, and F1-S for both the SQL injection class and the valid_querry label, with 82%, 100% and 90% respectively. Additionally, the model reaches an ACC of 94%.

Classification Report:	precision	recall	f1-score	support
0	1.00	0.92	0.96	621
1	0.82	1.00	0.90	219
accuracy			0.94	840
macro avg	0.91	0.96	0.93	840
weighted avg	0.95	0.94	0.94	840

Figure 4.41: SVM (Linear): Outperforms_2.

The confusion matrix in Figure 44 also offers valuable information. It illustrates that the model has a high ability to correctly predict valid_querry instances, with a count of 219 for true negatives and 49 for false positives (normal instances incorrectly identified as SQLIAs).

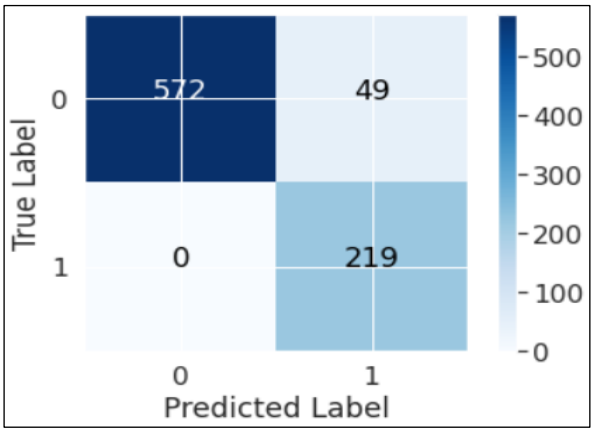


Figure 4.42: SVM (RBF): Confusion Matrix_2.

A high sensitivity and specificity would mean which the model is capable to reliably identify a significant number of both genuine positive instances (SQLIAs) with 100% and actual negative cases with 92% in the case of SQLID. This would indicate that the model can efficiently identify SQLIAs while reducing the number of false positives (normal cases incorrectly identified as SQLIAs). The model has a decent sensitivity and specificity for SQLID, according to the supplied findings.

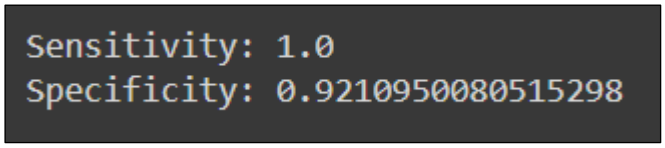


Figure 4.43: SVM (RBF): Sensitivity and Specificity_2.

4.11 COMPARISION RESULTS

4.11.1 First Dataset

The performance of seven ML classifiers on a dataset for detecting SQLIAs was evaluated. The classifiers were KNN, DT, RF, GB, XGB Classifier, RBF-SVM, and linear SVM. All of the classifiers demonstrated high ACC, with the highest being 50% for KNN and between 87% and 95% for DT, RF, GB, and XGB Classifier. PREC, which is the proportion of correctly identified positive cases to total positive predictions, was also high for all classifiers, with the highest being approximately to 100% for DT, RF, GB, and XGB Classifier, and 35% for KNN, 86% for RBF SVM, and 86% for Linear SVM. REC, or the proportion of correctly identified positive cases to total actual positive cases, was also high, with the highest being 100% for DT, RF, GB, and XGB Classifier, and 97% ,95%, 100% for KNN, RBF SVM, and Linear SVM respectively. The F1-S, a metric that combines PREC and REC, was also high for all classifiers, with the highest being between 87% and 91% for DT, RF, GB, and XGB Classifier, and 51% for KNN, 91% for RBF SVM, and 92% for Linear SVM. Overall, all classifiers performed well, but DT, RF, GB, and XGBClassifier had the highest scores across all metrics.

Table 4.1: Models Results Comparison for The First Dataset.

	ACC	**PREC**	**REC**	**F1-S**
KNN	50%	35%	97%	51%
DT	87%	68%	100%	81%
RF	87%	68%	100%	81%
GB	90%	100%	66%	80%
XGB	90%	100%	65%	79%
RBF SVM	94%	86%	95%	91%
Linear SVM	95%	86%	100%	92%

4.11.2 Second Dataset

The performance of KNN, DT, RF, GB, XGB, RBF SVM, and Linear SVM for detecting SQLIAs was evaluated on a dataset. All classifiers performed well, with high ACC, PREC, REC, and F1-S. The highest ACC was 70% for KNN, and between 91% and 92% for DT, RF, GB, and XGB. The highest PREC was between 91% and 96% for DT, RF, GB, and XGB, and 75% for KNN, 85% for RBF SVM, and 82% for Linear SVM. The highest REC was between 87% and 91% for DT, RF, GB, and XGB, and 65%, 97%, 91% for KNN, RBF SVM, and Linear SVM respectively. The highest F1-S was between 91% and 93% for DT, RF, GB, and XGB, and 70% for KNN, 91% for RBF SVM, and 90% for Linear SVM. Overall, all classifiers performed well, but DT, RF, GB, and XGB had the highest scores across all metrics.

Table 4.2: Models Results Comparison for The Second Dataset.

	ACC	**PREC**	**REC**	**F1-S**
KNN	70%	75%	65%	70%
DT	92%	93%	92%	93%
RF	92%	93%	92%	93%

Table 4.2: Models Results Comparison for The Second Dataset “Table Continued”.

GB	93%	96%	91%	93%
XGB	91%	96%	87%	91%
RBF SVM	95%	85%	97%	91%
Linear SVM	94%	82%	100%	90%

4.12 CHAPTER SUMMARY

This chapter presents an evaluation of the effectiveness of five ML, DL, and hybrid algorithms for detecting botnet attacks using the Network Intrusion Detection dataset. The seven algorithms evaluated are KNN, DTs, RF, Linear SVM, RBF SVM Machine, XGB Classifier, and Gradient Boosting.

5. DISCUSSION AND CONCLUSIONS

In this research, we applied seven ML classifiers to two datasets in order to assess their performance in detecting SQLIAs. The classifiers used were KNN, DT RF, GB, XGBoost (eXtreme GB), RBF SVM, and Linear SVM. The datasets used were a small dataset and a large dataset. The findings revealed that all of the classifiers performed well in terms of ACC, PREC, REC, and F1-S on both datasets.

However, DT, RF, GB, and XGBoost achieved the highest scores in all four metrics, indicating that they may be the most effective at detecting SIA. Overall, the results of this study suggest that ML techniques can be effective for detecting SIA, particularly those that are based on DTs, random forests, GB, and extreme GB. These methods can improve the ACC and efficiency of detecting SQLIAs by identifying patterns in network traffic that may indicate an intrusion or anomaly, this solution gives excellent results on both small and large datasets.

REFERENCES

[1] S.S., Anandha Krishnan. “SQLID Using ML.” *Revista Gestão Inovação E Tecnologias*, vol. 11, no. 3, 30 June 2021, pp. 300–310, 10.47059/revistageintec.v11i3.1939.

[2] A. Joshi, and V. Geetha. , “SQLID using ML”, In Proceedings of the International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), Kanyakumari, India, July 10-11, 2014, pp. 1111-1115.

[3] Chen, Zhuang, et al. “Research on SQLID Technology Based on SVM.” *MATEC Web of Conferences*, vol. 173, 2018, p. 01004, 10.1051/matecconf/201817301004. Accessed 2 Mar. 2020.

[4] Ross, Kevin, et al. “Multi-Source Data Analysis and Evaluation of ML Techniques for SQLID.” *Proceedings of the ACMSE 2018 Conference*, 29 Mar. 2018, 10.1145/3190645.3190670.

[5] Jide E. T., Akinsola, et al. “SQL Injection Attacks Predictive Analytics Using Supervised ML Techniques.” *International Journal of Computer Applications Technology and Research*, vol. 9, no. 4, 3 Apr. 2020, pp. 139–149, 10.7753/ijcatr0904.1004.

[6] Liu, Shigang, et al. “DeepBalance: Deep-Learning and Fuzzy Oversampling for Vulnerability Detection.” *IEEE Transactions on Fuzzy Systems*, 2019, pp. 1–1, 10.1109/tfuzz.2019.2958558. Accessed 11 May 2021.

[7] Mishra, Preeti, et al. “A Detailed Investigation and Analysis of Using ML Techniques for Intrusion Detection.” *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, 2019, pp. 686–728, 10.1109/comst.2018.2847722. Accessed 25 Oct. 2019.

[8] Muslihi, Muhammad Takdir, and Daniyal Alghazzawi. “Detecting SQL Injection on Web Application Using Deep Learning Techniques: A Systematic Literature Review.” *2020 Third International Conference on Vocational Education and Electrical Engineering (ICVEE)*, 3 Oct. 2020, 10.1109/icvee50212.2020.9243198. Accessed 17 Jan. 2023.

[9] OTHMAN, Suad Mohammed, ALSOHYBE, Nabeel T., BA-ALWI, Fadl Mutaher, *et al*. Survey on intrusion detection system types. *International Journal of Cyber-Security and Digital Forensics*, 2018, vol. 7, no 4, p. 444-463.

[10] Davis, Jonathan J., and Andrew J. Clark. “Data Preprocessing for Anomaly Based Network Intrusion Detection: A Review.” *Computers & Security*, vol. 30, no. 6-7, Sept. 2011, pp. 353–375, 10.1016/j.cose.2011.05.008. Accessed 9 Nov. 2020.

[11] Martins, Inês, et al. “Host-Based IDS: A Review and Open Issues of an Anomaly Detection System in IoT.” *Future Generation Computer Systems*, vol. 133, Aug. 2022, pp. 95–113, 10.1016/j.future.2022.03.001.

[12] PATTEWAR, Tareek, PATIL, Hitesh, PATIL, Harshada, *et al.* Detection of SQL injection using ML: a survey. *Int. Res. J. Eng. Technol.(IRJET)*, 2019, vol. 6, no 11, p. 239-246.

[13] Li, Qi, et al. "A SQLID Method Based on Adaptive Deep Forest." *IEEE Access*, vol. 7, 2019, pp. 145385–145394, 10.1109/access.2019.2944951. Accessed 15 Feb. 2021.

[14] LI, Jing, LI, Qinyuan, ZHOU, Sheng, *et al.* A review on signature-based detection for network threats. In: *2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN)*. IEEE, 2017. p. 1117-1121.

[15] Agarwal, Nancy, and Syed Zeeshan Hussain. "A Closer Look at Intrusion Detection System for Web Applications." *Security and Communication Networks*, vol. 2018, 14 Aug. 2018, pp. 1–27, www.hindawi.com/journals/scn/2018/9601357/, 10.1155/2018/9601357. Accessed 5 Aug. 2019.

[16] Min E, Long J, Liu Q, Cui J, Chen W. TR-IDS: Anomaly-based intrusion detection through text-convolutional neural network and random forest. Security and Communication Networks. 2018 Jul 5;2018.

[17] Gurina, Anastasia, and Vladimir Eliseev. "Anomaly-Based Method for Detecting Multiple Classes of Network Attacks." *Information*, vol. 10, no. 3, 26 Feb. 2019, p. 84, 10.3390/info10030084. Accessed 21 June 2020.

[18] Alsoufi, Muaadh A., et al. "Anomaly-Based Intrusion Detection Systems in IoT Using Deep Learning: A Systematic Literature Review." *Applied Sciences*, vol. 11, no. 18, 9 Sept. 2021, p. 8383, 10.3390/app11188383.

[19] HARBA, Hazem M., ELEYANB, Derar, et ELEYANC, Amna. SQL Injection Detection Tools Advantages and Drawbacks. *International Journal of Wireless and Microwave Technologies (IJWMT)*, 2021.

[20] Meidan, Yair, et al. "N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders." *IEEE Pervasive Computing*, vol. 17, no. 3, July 2018, pp. 12–22, arxiv.org/pdf/1805.03409.pdf, 10.1109/mprv.2018.03367731.

[21] Kaur, Sanmeet, and Maninder Singh. "Hybrid intrusion detection and signature generation using deep recurrent neural networks." *Neural Computing and Applications* 32.12 (2020): 7859-7877.

[22] LEE, Chie-Hong, SU, Yann-Yean, LIN, Yu-Chun, *et al.* ML based network intrusion detection. In : *2017 2nd IEEE International conference on computational intelligence and applications (ICCIA)*. IEEE, 2017. p. 79-83.

[23] JAYAPRAKASH, Souparnika et KANDASAMY, Kamalanathan. Database intrusion detection system using octraplet and ML. In: *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*. IEEE, 2018. p. 1413-1416.

[24] SAHASRABUDDHE, Atmaja, NAIKADE, Sonali, RAMASWAMY, Akshaya, *et al.* Survey on intrusion detection system using data mining techniques. *Int. Res. J. Eng. Technol*, 2017, vol. 4, no 5, p. 1780-1784.

[25] SAXENA, Aumreesh Ku, SINHA, Sitesh, et SHUKLA, Piyush. General study of intrusion detection system and survey of agent based intrusion detection system. In : *2017 International Conference on Computing, Communication and Automation (ICCCA)*. IEEE, 2017. p. 471-421.

[26] SINHA, Keshav et VERMA, Madhav. The Detection of SQL Injection on Blockchain-Based Database. In: *Revolutionary Applications of Blockchain-Enabled Privacy and Access Control*. IGI Global, 2021. p. 234-262.

[27] TEKEREK, A. D. E. M. et BAY, O. F. Design and implementation of an artificial intelligence-based web application firewall model. *Neural Network World*, 2019, vol. 29, no 4, p. 189-206.

[28] ESPINOSA REBOREDO, Hernan. Implementation of a web application firewall for a high availability front end. 2020.

[29] HAREFA, Jeklin, PRAJENA, Gredion, ALEXANDER, Abdillah Muhamad, *et al.* Sea waf: The prevention of sql injection attacks on web applications. *Advances in Science, Technology and Engineering Systems*, 2021, vol. 6, p. 405-411.

[30] GOGOI, Bronjon, AHMED, Tasiruddin, et SAIKIA, Hemanta Kumar. Detection of XSS attacks in web applications: A ML approach. *International Journal of Innovative Research in Computer Science & Technology (IJIRCST) ISSN*, 2021, p. 2347-5552.

[31] JEMAL, Ines, CHEIKHROUHOU, Omar, HAMAM, Habib, *et al.* Sql injection attack detection and prevention techniques using ML. *International Journal of Applied Engineering Research*, 2020, vol. 15, no 6, p. 569-580.

[32] Batta Mahesh. ML algorithms-a review. International Journal of Science and Research (IJSR).[Internet], 9:381–386, 2020

[33] Singh, Amanpreet, Narina Thakur, and Aakanksha Sharma. "A review of supervised ML algorithms." In 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), pp. 1310-1315. Ieee, 2016.

[34] Usama, Muhammad, Junaid Qadir, Aunn Raza, Hunain Arif, Kok-Lim Alvin Yau, Yehia Elkhatib, Amir Hussain, and Ala Al-Fuqaha. "Unsupervised ML for networking: Techniques, applications and research challenges." *IEEE access* 7 (2019): 65579-65615.

[35] Li, Y. (2017). Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*.

[36] Ross, Kevin. "SQLID using ML techniques and multiple data sources." (2018).

[37] Gholizadeh M, Jamei M, Ahmadianfar I, Pourrajab R. Prediction of nanofluids viscosity using random forest (RF) approach. Chemometrics and Intelligent Laboratory Systems. 2020 Jun 15;201:104010.

[38] Xie, Xin, Chunhui Ren, Yusheng Fu, Jie Xu, and Jinhong Guo. "SQLID for web applications based on elastic-pooling cnn." *IEEE Access* 7 (2019): 151475-151481.

[39] Iman, Alif Nur, and Tohari Ahmad. "Improving intrusion detection system by estimating parameters of random forest in Boruta." In *2020 International Conference on Smart Technology and Applications (ICoSTA)*, pp. 1-6. IEEE, 2020.

[40] Ghosh, S., Dasgupta, A. and Swetapadma, A., 2019, February. A study on support vector machine based linear and non-linear pattern classification. In *2019 International Conference on Intelligent Sustainable Systems (ICISS)* (pp. 24-28). IEEE.

[41] Chen, Zhuang, and Min Guo. "Research on SQLID technology based on SVM." In *MATEC web of conferences*, vol. 173, p. 01004. EDP Sciences, 2018.

[42] Erfani, S. M., Rajasegarar, S., Karunasekera, S., & Leckie, C. (2016). High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. *Pattern Recognition*, *58*, 121-134.

[43] Wang, Zheng, Cheng Yang, Sung-Kwun Oh, and Zunwei Fu. "Multi-radial basis function SVM classifier: Design and analysis." *Journal of Electrical Engineering and Technology* 13, no. 6 (2018): 2511-2520.

[44] Liu, Chao, Jing Yang, and Jinqiu Wu. "Web intrusion detection system combined with feature analysis and SVM optimization." *EURASIP Journal on Wireless Communications and Networking* 2020, no. 1 (2020): 1-9.

[45] Zhang, Wengang, Chongzhi Wu, Haiyi Zhong, Yongqin Li, and Lin Wang. "Prediction of undrained shear strength using extreme gradient boosting and random forest based on Bayesian optimization." *Geoscience Frontiers* 12, no. 1 (2021): 469-477.

[46] Roy, P., Kumar, R., & Rani, P. (2022, May). SQL Injection Attack Detection by ML Classifier. In *2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC)* (pp. 394-400). IEEE.

[47] Bentéjac, Candice, Anna Csörgő, and Gonzalo Martínez-Muñoz. "A comparative analysis of gradient boosting algorithms." *Artificial Intelligence Review* 54, no. 3 (2021): 1937-1967.

[48] Mishra, Sonali. "SQLID using ML." (2019).

[49] Charbuty, Bahzad, and Adnan Abdulazeez. "Classification based on decision tree algorithm for ML." *Journal of Applied Science and Technology Trends* 2, no. 01 (2021): 20-28.

[50] Hasan, Musaab, Zayed Balbahaith, and Mohammed Tarique. "Detection of SQL injection attacks: a ML approach." In *2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, pp. 1-6. IEEE, 2019.

[51] Hasan, M.; Tarique, M. Detection of SQL Injection Attacks: A ML Approach. In Proceedings of the 2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA), Ras Al Khaimah, United Arab Emirates, 19–21 November 2019

[52] Gao, H.; Zhu, J.; Liu, L.; Xu, J.; Wu, Y.; Liu, A. Detecting SQL Injection Attacks Using Grammar Pattern Recognition and Access Behavior Mining. In Proceedings of the 2019 IEEE International Conference on Energy Internet (ICEI), Nanjing, China, 27–31 May 2019

[53] Gandhi, N. A CNN-BiLSTM based Approach for Detection of SQL Injection Attacks. In Proceedings of the 2021 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE), Dubai, United Arab Emirates, 17–18 March 2021; pp. 378–383

[54] Tripathy, D.; Gohil, R.; Halabi, T. Detecting SQL Injection Attacks in Cloud SaaS using ML. In Proceedings of the 2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS), Baltimore, MD, USA, 25–27 May 2020; pp. 145–150.

[55] Kar, D.; Sahoo, A.K.; Agarwal, K.; Panigrahi, S.; Das, M. Learning to Detect SQLIA Using Node Centrality with Feature Selection. In Proceedings of the 2016 International Conference on Computing, Analytics and Security Trends (CAST), Pune, India, 19–21 December 2016; pp. 18–23.

[56] Li, Q.; Wang, F.; Wang, J.; Li, W. LSTM-Based SQLID Method for Intelligent Transportation System. *IEEE Trans. Veh. Technol.* **2019**, *68*, 4182–4191.

[57] Kamtuo, K.; Soomlek, C. ML for SQL Injection Prevention in Server-Side Scripting. In Proceedings of the 2016 International Computer Science and Engineering Conference (ICSEC), Chiang Mai, Thailand, 14–17 December 2016; pp. 1–6.

[58] Sivasangari, A. SQL Injection Attack Detection using ML Algorithm. In Proceedings of the 2021 5th International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 3–5 June 2021; pp. 1166–1169.

[59] Das, D.; Sharma, U.; Bhattacharyya, D.K. Defeating SQL injection attack in authentication security: An experimental study. *Int. J. Inf. Secur.* **2019**, *18*, 1–22.

[60] Kasim, Ö. An ensemble classification-based approach to detect the attack level of SQL injections. *J. Inf. Secur. Appl.* **2021**, *59*, 102852.

[61] Tang, P.; Qiu, W.; Huang, Z.; Lian, H.; Liu, G. Detection of SQL injection based on artificial neural network. *Knowl.-Based Syst.* **2020**, *190*, 105528.

[62] Erdődi, L.; Sommervoll, Å.Å.; Zennaro, F.M. SQL injection vulnerability exploitation using Q-learning reinforcement learning agents. *J. Inf. Secur. Appl. Simulating* **2021**, *61*, 102903.

[63] Kar, D.; Panigrahi, S.; Sundararajan, S. SQLiGoT: Detecting SQL injection attacks using the graph of tokens and SVM. *Comput. Secur.* **2016**, *60*, 206–225.

[64] Uwagbole, S.O.; Buchanan, W.J.; Fan, L. Applied ML Predictive Analytics to SQL Injection Attack Detection and Prevention. In Proceedings of the 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Lisbon, Portugal, 8–12 May 2017; pp. 1087–1090.

[65] Mcwhirter, P.R.; Kifayat, K.; Shi, Q.; Askwith, B. SQL Injection Attack classification through the feature extraction of SQL query strings using a Gap-Weighted String Subsequence Kernel. *J. Inf. Secur. Appl.* **2018**, *40*, 199–216.

[66] Mejia-Cabrera, H.I.; Paico-Chileno, D.; Valdera-Contreras, J.H.; Tuesta-Monteza, V.A.; Forero, M.G. *Automatic Detection of Injection Attacks by ML in NoSQL Databases*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 23–32.

[67] Pathak, R.K.; Yadav, V. *Handling SQL Injection Attack Using Progressive Neural Network*; Springer: Singapore, 2020; Volume 1170.

[68] Fang, Y.; Peng, J.; Liu, L.; Huang, C. WOVSQLI: Detection of SQL injection behaviors using word vector and LSTM. In Proceedings of the ICCSP 2018: Proceedings of the 2nd International Conference on Cryptography, Security and Privacy, Guiyang, China, 16–19 March 2018; pp. 170–174.

[69] Zhang, H.; Zhao, J.; Zhao, B.; Yan, X.; Yuan, H.; Li, F. SQLID based on deep belief network. In Proceedings of the CSAE 2019: Proceedings of the 3rd International Conference on Computer Science and Application Engineering, Sanya, China, 22–24 October 2019.

[70] Priyaa, B.D.; Student, P.G.; Devi, M.I. Hybrid SQLID System. In Proceedings of the 2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 22–23 January 2016.

[71] Bittrich, Sebastian, et al. "Application of an Interpretable Classification Model on Early Folding Residues during Protein Folding." *BioData Mining*, vol. 12, no. 1, 5 Jan. 2019, 10.1186/s13040-018-0188-2.

[72] :K.Poulsen,"securityfocus,"[Online].Available:https://www.securityfocus.com/news/5968. [Accessed 5 12 2020].

[73] KINI, Shreya, PATIL, Anushka Parvate, POOJA, M., *et al.* SQL Injection Detection and Prevention using Aho-Corasick Pattern Matching Algorithm. In: *2022 3rd International Conference for Emerging Technology (INCET)*. IEEE, 2022. p. 1-6.

[74] A. A. a. A. Khresiat, "New Strategy for Mitigating of SQL Injection Attack," International Journal of Computer Applications, vol. 154, pp. 1-10, 2016.

[75] B. A. a. E. O.-M. a. Z. Qin, "SQL injection attack detection using fingerprints and pattern matching technique," 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), pp. 583-587, 2017.

[76] M. F. Y. Zainab S. Alwan, "Detection and Prevention of SQL Injection Attack: A survey," International Journal of Computer Science and Mobile Computing, vol. 6, no. 8, August 2017, pp. 5-17, 2017.

[77] J. P. Singh, "Analysis of SQLID Techniques," Theoretical and Applied Informatics, vol. 28, pp. 37-55, 2016.

[78] M. N. R. A. Mamdouh Alenezi, "SQL injection attacks countermeasures assessments," Indonesian Journal of Electrical Engineering and Computer Science, vol. 21, no. 2, pp. 1121-1131, Feb 2021.

[79] Komal, S.Deswal, ―KMPS: A Hybrid Algorithm to Detect Web Application Vulnerabilities‖, International Journal of Computer Sciences and Engineering Vol.-6, Issue-6, June 20

[80] A. Pawar, D. Barthare, N. Rawat, M. Yadav and M. Shirole, "BlockAudit 2.0: PoA blockchain based solution for secure Audit logs," *2021 5th International Conference on Information Systems and Computer Networks (ISCON)*, Mathura, India, 2021, pp. 1-6, doi: 10.1109/ISCON52037.2021.9702378.

[81] Ahmad K, Karim M. A method to prevent SQL injection attack using an improved parameterized stored procedure. International Journal of Advanced Computer Science and Applications. 2021;12(6).

[82] Jesudoss A, Mercy TM, Christy A, Maheswari M, Selvi M, Ulagamuthalvi V. Analysis and implementation of SQL injection attack and countermeasures using SQL injection prevention techniques. International Journal of Engineering Systems Modelling and Simulation. 2022;13(4):262-7.