SQL INJECTION PATTERN RECOGNITION BASED ON NAÏVE BAYES MODEL

HSU WAI TUN

M.C.Sc. **DECEMBER 2022**

SQL INJECTION PATTERN RECOGNITION BASED ON NAÏVE BAYES MODEL

By

Hsu Wai Tun

B.C.Sc.

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Computer Science

(M.C.Sc.)

University of Computer Studies, Yangon

DECEMBER 2022

ACKNOWLEDGEMENTS

First and foremost, I would like to express all my teachers who advised, taught and helped me to complete my thesis successfully during the study at the University of Computer Studies, Yangon.

Secondly, I would like to special thanks to my principal Dr. Mie Mie Khin, the Rector of the University of Computer Studies, Yangon who gave me the opportunity to develop this thesis for her general guidance during the period of study.

I would like to express my respectful gratitude to Dr. Si Si Mar Win and Dr. Tin Zar Thaw, Deans of the Master 24th batch, University of Computer Studies, Yangon, for her excellent guidance.

My sincere thanks and regards go to my supervisor, Dr. Khaing Khaing Wai, Professor, Head of Department of Information Technology Support and Maintenance, University of Computer Studies, Yangon, for her support, guidance, supervision, patience and encouragement during the period of study towards completion of this thesis.

In addition, I would like to thank all of my thesis's board examiners who gave the precious comments and corrections to my work for getting good end result. I would like to thank Daw Mya Hnin Mon, Associate Professor, Department of English, University of Computer Studies, Yangon, for her valuable supports and editing my thesis from the language point of view.

Last but not at least, I am extremely thankful to my parents and my family for supporting, inspiring and encouragement to me from the childhood to the present time. Finally, I am extremely grateful to my all of teachers, my colleagues and all of my friends for their invaluable and precious help and general guidance.

STATEMENT OF ORIGINALITY

I hereby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.

----------------------------------	-------------------------------
Date	Hsu Wai Tun

ABSTRACT

In recent years, sharing information through the Internet across various platforms and web-applications has grown increasingly widespread. Users' critical information is stored in databases by the web-based applications that receive it. Due to its availability over the Internet, these apps and the databases that are connected to be vulnerable to numerous cybersecurity incidents. Therefore**,** cyber-security is critical for securing user’s critical data and information in this technology era. The attacker can steal critical and confidential information by using various threats. The threats include attacks such as Cross Side Scripting (CSS), Denial of Service Attack (DoS0, and Structured Query Language (SQL) Injection attacks. One of the 10 most popular risks and weaknesses to web applications with backend databases is SQL injection. It utilizes malicious SQL queries to modify internal data and to retrieve information from the back-end database that was not intended to be displayed. Since there are countless cyberattacks every day and have really been needing on developing a more secure system that can predict them and prevent them from happening. In this thesis, proposed system can be detected SQL Injection Attack successfully by applying machine learning algorithm based on Naïve Bayes Method. The proposed model was trained and evaluated with 21,523 instances of dataset which comprises SQL Injection and no Injection. The user interface is created for a test case that anticipates either a malicious or benign question from the user. Finally, this system is displayed the result of detecting the query that is SQL Injection or not and is evaluated how accurate the results based on having false negative and false positive rate.

Contents

LIST OF FIGURES

Page

LIST OF TABLES

Page

LIST OF EQUATIONS

Page

CHAPTER 1
INTRODUCTION

Web-based program are mainly used on daily life. Web application accept user's crucial information and store this information in database [7]. Attackers try to get easy access to the underlying database of web applications, making them susceptible. The various threats such as Cross Side Scripting (CSS), Denial of Service Attack (DoS), and Structured Query Language (SQL) Injection attacks are used to attack. Among these, one of the most common cyber-attacks is SQL Injection Attacks.

SQL injection is a popular attack method that allows access to data that was not intended to be revealed by altering the backend database utilizing malicious SQL code. Attackers can be able to access the databases that underlie Web applications using SQL Injection Attacks because it gives them the opportunity to leak, edit, or even destroy information that is retained in these databases. SQL Injection Attack has also been identified by the Open WEB Application Security Project (OWASP) as the top risks to web-based applications [21].

One of the most common and harmful types of hacker assault are SQL Injection Assault. Prevention of SQL injector attacks is crucial and complex topics to learn in information system security. A SQL injection attack (SQLIA) in Web applications supported by a database is the main security risk. Through this vulnerability, attackers can quickly access the application's underlying database and any potentially sensitive data that can be present there. By carefully crafting input, a hacker can access database content that would otherwise be impossible. Typically, this is accomplished by altering the SQL statements employed by online programs. Due to the safety of web applications, the researchers have extensively researched SQLIA detection and prevention and developed a variety of strategies.

However various techniques have been developed to counteract such attacks, fraudsters continue to find ways to circumvent the different protections put in place to prevent SQL Injection attacks. Currently, there has been a significant amount of debate concerning by using machine learning algorithms to detect and prevent certain cyber security risks. The effectiveness of using supervised and unsupervised learning techniques to identify security threats cannot be challenged but the computing power and processing time needed to run such complicated algorithms continue to be a major concern for the community working on cyber security, that is constantly evolving. For the purpose of recognizing SQL Injection attacks, a significant amount of study has been done utilizing various machine learning methods. This system uses Naïve Bayes algorithm to detect SQL Injection Attack.

1.1 Objectives of the Thesis

The main objectives of the thesis are as follows:

- To find serious threats that are embedded in query
- To protect the critical information that stored in the web-app’s backend database
- To help security officer or security analyst to terminate the attack early
- To help the organization/company to be better secure their backend database and customer information

1.2 Motivation of the System

Attackers try to use various security flaws to hack web-based apps that store sensitive information in databases. These security flaws are SQL Injection, Cross Site Scripting, Broken Authentication and Session Management, Insecure Direct Object References, Cross Site Request Forgery and Security Misconfiguration. SQL Injection is one of top ten vulnerabilities that is described Open Web Security Project (OWASP). If the injection is successful, the attackers would be able to access the database and retrieve the sensitive information as well as the credentials of other users. Through this injection, the attacker might change the data in the existing database. Many online consumers' intellectual property could be damaged by this kind of attack. Bangladesh's economy suffered a severe setback in February 2016 as a result of a hacking attack on the Bangladesh Bank, during which this company lost more than 81 million USD [8]. According to Statistica, the average cyber loss for mid-sized businesses in 2019 was 1.56 million dollars. In a survey of global companies conducted in May2019, the average damage across all company sizes was estimated to be 4.7 million US dollars [19]. SQL injection attacks are used by hackers in 51% of instances, according to another significant statistic about them.

1.3 Related Work

In this paper [16], the ResNet model is constructed using data collected from a variety of devices and the internet. The ResNet method is used to train on processed samples. The user interface is created for an experiment that predicts either a malicious or innocent question from the user. If a malicious or legitimate input request is made, the trained ResNet model can detect with accuracy which one it is. This system shows how ResNet can successfully identify several types of SQLIA.

In this paper [9], machine learning is employed, and the SQL Injection detector is trained using training data before being checked against testing data. The access log is extracted and divided into a test set and a training set. The detector learns from the training set and develops a Knowledge Base (KB). Finally, the detector checks the test set depending on KB. The outcome of the detection demonstrates that the proposed technique achieves excellent accuracy in differentiating between malicious and legitimate web requests.

In this paper [10], they suggested employing the Aho-Corasick pattern matching algorithm as a detection and prevention method to prevent SQL Injection Attacks (SQLIA). This system is evaluated by using sample of well-known attack patterns. The proposed scheme has the following two modules, 1) Static Phase and 2) Dynamic Phase. In the Static Pattern List, they keep a list of known Anomaly Pattern. In Dynamic Phase, an alarm will occur and a new anomaly pattern will be established whenever a new anomaly of any kind unexpectedly emerges. The Static Pattern List will be updated with the new anomalous pattern. The initial evaluation suggests that the proposed approach is effective against the SQL Injection Attack by taking in consideration a sample of common attack patterns.

In this paper [13], an approach to stop complex SQL injection attacks has been proposed in this system and was based on the adaptive deep forest algorithm. Here, the raw feature vector and an average of the prior outputs were combined to create the input for each layer. As a result, the deep forest structure in this study is improved. Later, they devised a method called the AdaBoost based deep forest model for using the error rates to update the weights in each layer. In this study, it was discovered that the proposed model performed better than the traditional machine learning techniques. The outcomes of the experiment have proven this. The proposed model for detecting the SQL injection had two stages. The off-line training phase and the online test phase were these. The collection of 10,000 SQL injection samples. Here, characteristics including the UNION query, SQL command executor, error-based injection, and blind injection were retrieved from several datasets.

In this paper [10], it has many SQL injection attack types and classifications represented. In addition, they bring forth a model that functions in three stages, checking ASCII characters first, tokens next, and threshold values last. This model is similar to a pattern lock. In addition, they bring forth a model that functions in three stages, checking ASCII characters first, tokens next, and threshold values last. This model is similar to a pattern lock. The model makes sure that only the right queries are sent to the database server in this way. In this paper, various sorts of attacks have been discussed, including client-side assaults, information-based attacks, command execution-based attacks, and attacks based on authentication. The paper claims that there are six different categories of SQL injection attacks, including tautologies, union queries, stored procedures, logically flawed queries, inference, piggy-backed queries, etc. The user interface under the suggested model would accept input from the user. The query length and pattern values of all the inputs would then be compared. If both values are the same, the anomaly value will be calculated. If the score for anomaly exceeds the threshold previously established, the model will reject the current query. The query won't be denied if the threshold wasn't exceeded. Additionally, some methods for preventing SQLI are examined, such as using the OWASP-provided SQL Cheat Sheet, avoiding detailed error messages, using parameterized queries, outlining automatic and dynamic access control lists, utilizing functions that would block quotes, granting users only the necessary permissions, etc. In the end, the authors admitted that their program was not running quickly enough and that the security of their suggested system has certain flaws. They suggested that the model can be made faster and safer by including a cross-site script protection technique.

1.4 Outline of the Thesis

There are five chapters in this thesis.

The objectives and organizational structure of the thesis are described in Chapter 1 and it is explanation of the section that serves as an introduction to SQL Injection Attack and related works.

Web architecture, web application vulnerabilities, SQL Injection Attacks, and machine learning methods are just a few of the foundational theories covered in length in Chapter 2 that are connected to this thesis.

The design of the suggested system is described in Chapter 3 along with the system flow, a detailed analysis of the algorithms and an assessment of the output results.

The detailed implementation of the suggested system and the outcomes of the experiments are discussed in Chapter 4.

This thesis is concluded in Chapter5 which also discusses the advantages, limitations, and possible future extensions of the suggested system.

CHAPTER 2
BACKGROUND THEORY

This chapter represents the underlying background theory for the proposed system. In the first section, web architecture, about web app and vulnerability, types of threat and SQL Injection are described. The next section describes about machine learning and types of machine learning. In the last section, Naïve Bayes algorithm and types of model are presented in details.

2.1 Web Architecture

A web app architecture displays the architecture of all the software elements, including middleware, databases and apps, as well as how they communicate with one another. It guarantees that both the client-side server and the backend server can understand the HTTP data delivery protocol by providing specifics about it. A web-based application architecture typically consists of 3 main components:

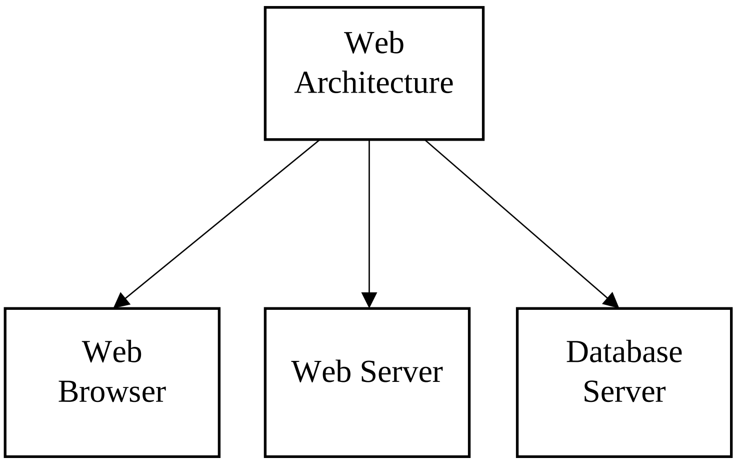


Figure 2.1 Components of Web Architecture

Every web application architecture is designed using a layered architecture. Multi-or Three-Tier Architecture refers to the organization of architectural patterns for web applications into various layers or tiers.

Presentation Layer: This layer which the client can access through a browser, contains user interface and UI process components.

Business/Application Layer: It receives the user's request from the browser, processes it and controls the paths taken to retrieve the contents.

Persistence Layer: It is also known as a data access or storage layer. This layer aggregates all data calls and grants access to an application's persistent storage.

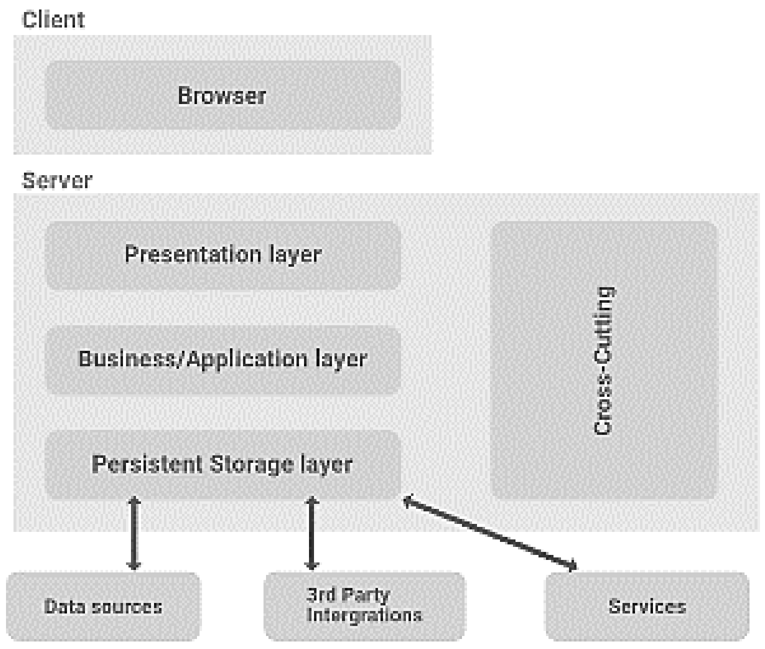


Figure 2.2 Web App Architecture

The web application architecture one chooses to work with concerns several elements of web applications, including resilience, security, scalability, reliability and responsiveness.

2.2 Web Application

A database, an application server and a web server are essential for a Web application (Web app) to operate effectively [22]. It is an application program that is stored on a remote server and distributed over the Internet using a browser interface. Web browsers like Google Chrome, Microsoft Edge, Mozilla Firefox, Internet Explorer, Safari and Opera are used by users to access web applications.

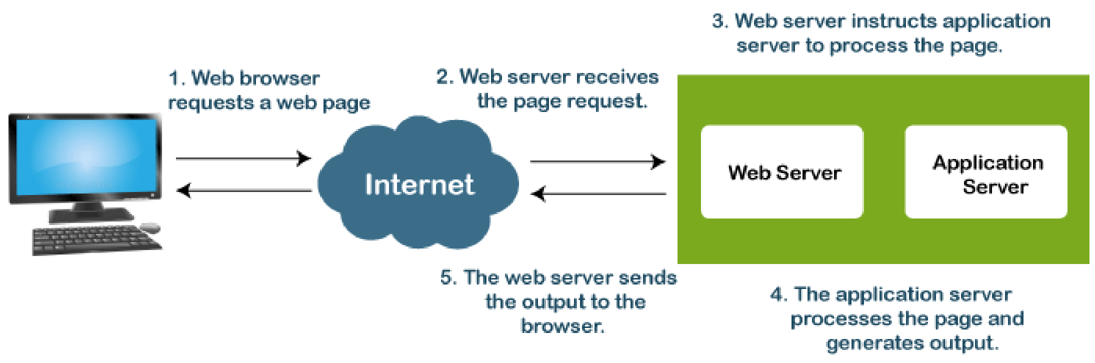

3. Web server instructs application server to process the page.
1. Web browser requests a web page
2. Web server receives the page request.
Internet
Web Server
Application Server
5. The web server sends the output to the browser.
4. The application server processes the page and generates output.

Figure 2.3 Flow of Web Applications

The requests made by clients are handled by web servers and the desired task is executed through the application server. In order to create a front-end application, the client-side programming language frequently makes use of JavaScript, HTML and CSS. Users can view information and communicate with the web server according to this. The server-side programming languages used most frequently are Python, Ruby, PHP, ASP and Java. The web server must be managed a web application, together with information retrieval and storage [12]. Web applications typically include ecommerce websites, webmail, calculators, social media platforms, etc.

2.2.1 Web Application Vulnerability

Any system weakness that a hacker can use to compromise a web application is known as a web application vulnerability. When using a web application with a web browser, website users can upload and retrieve data to and from a database over the internet. Most importantly, many of these databases contain valuable data, including private financial and personal information, sensitive client data, and other information that makes them valuable targets for attackers. If they have major weaknesses or vulnerabilities, the attackers have a variety of ways to obtain these crucial data. Attackers use these flaws as opportunities to damage individuals.

Web applications require the capacity to communicate and interact with various users from different networks, making them apart from other typical weaknesses like asset flaws or network vulnerabilities. A web application is a hacker's favorite target because it is easily accessible.

2.2.2 Common Types of Web Application Vulnerability

Vulnerabilities that are frequently found in web applications.

- **SQL Injection**

  Structured Query Language (SQL) is widely used programming language for managing database communications. Attackers can enter malicious SQL commands to steal information, edit, or delete data using SQL flaws [18]. Some cybercriminals utilize SQL to take control of the target system. Attacks using SQL injection target servers that host critical information utilized by web services or applications. When vital or sensitive data, such as user passwords and personal information is exposed they become extremely risky.

- **Cross-Site Scripting (XSS)**

  The injection of malicious scripts into websites or web applications is component of XSS attacks, which are similarly to SQL injection attacks. The main distinction is that the malicious code only executes in the browser when a user accesses a hacked website or app. By inserting code into input fields, attackers typically launch XSS attacks that the target page executes when users view the page. An XSS attack can reveal user data without revealing a compromise, which could eventually harm a company's reputation. Users might not be aware that attackers are stealing any private information they send to the compromised app.

- **Broken Authentication and Session Management**

  For each valid session, websites often generate a session cookie and session ID, and these cookies hold sensitive information like usernames and passwords. These cookies should be invalidated when the session ends, either by logout or a sudden browser close, meaning that a new cookie should be created for every session. Sensitive information will be present in the system if the cookies are not expired. An attacker can use this flaw to hijack a session, obtain unauthorized access to the system and allow the exposure and modification of unauthorized information. Utilizing stolen cookies or sessions using XSS, the sessions can be hijacked.

- **Insecure Direct Object References (IDOR)**

  Access control problems in web applications frequently lead to widespread, extremely dangerous vulnerabilities called insecure direct object references. By altering a "direct object reference," such as a database key, query parameter, or filename, IDOR flaws enable an attacker to interact maliciously with a web application. The attacker can construct a future attack to gain access to the unauthorized data and use this information to gain access to other objects. An attacker might compromise the application, change data, or get access to unapproved internal objects using this vulnerability.

- **Cross Site Request Forgery**

  When a victim is forced into using the web application in an unauthorized manner, this is known as a CSRF attack. The victim first logs into the web app, which has already verified the legitimacy of the user and browser. The program will therefore execute any malicious operations after the attacker fakes the victim into sending a request to the web app. Sensitive data will be stolen if the user clicks the malicious request after logging into the original website, and unauthenticated action will be carried out on the user's behalf.

- **Security Misconfiguration**

  Some of the most critical web application vulnerabilities are caused by security misconfigurations because they make it simple for attacks to access the application. Numerous security configuration flaws could be exploited by attackers. Ad hoc or incomplete configurations, data maintained in the cloud, plaintext error messages containing sensitive information, unmodified default configurations, and misconfigured HTTP headers are a few examples. Any operating system, library, framework, or application might contain security misconfigurations.

- **Insecure Cryptographic Storage**

  When sensitive data is not saved securely, there is a common vulnerability known as insecure cryptographic storage. The term "Insecure Cryptographic Storage" refers to a group of vulnerabilities rather than a single one. Sensitive data information on a website includes items like user login passwords, profile information, health information, credit card information, etc. This information will be kept in the application database. If this data is wrongly stored without encryption or hashing, it will be susceptible to attackers. An attacker can use this weakness to perform crimes like identity theft and credit card fraud by stealing and altering such weak security data.

- **Failure to restrict URL Access**

  Forced browsing is a method that can risk security if web application doesn't properly limit URL access. When an attacker uses a web browser to request specific pages or data files in an attempt to obtain sensitive information, forced browsing can be a very significant issue. An attacker can bypass website security using this method by directly accessing files instead of by clicking links. This enables the attacker to bypass the web application and access data source files directly. The attacker can then identify and access source code or other information left on the server, locate backup files that hold sensitive information, and get around the "order" of web pages. Web programs check the URL access credentials before generating restricted links and buttons. The most of apps do not expose privileged users who have access to privileged sites locations, or resources. By making a precise assumption, an attacker can access restricted pages. An intruder has access to features, sensitive pages and personal data. An attacker can use this vulnerability to access and exploit unauthorized URLs without having to enter the program. A hacker has access to sensitive pages, can use functionalities, and read confidential information.

- **Unvalidated Redirects and Forwards**

  An attacker can use this vulnerability to access and exploit unauthorized URLs without having to enter the program. A hacker has accessed to sensitive pages, can use functionalities and read confidential information. The web application accomplishes its objectives by sending users to other pages using a finite number of techniques. If there is insufficient validation while redirecting to other pages, attackers can take advantage of this and exploit forwarding to drive users to malicious or phishing websites. An attacker may join a valid URL with a malicious URL that has been encoded before sending it to the user. A person can navigate the URL provided by the attacker and perhaps become a victim just by accessing the legitimate part of it.

- **Directory Traversal**

  Backtracking or directory traversal attacks take advantage of the way a web application gathers data from a web server. Access Control Lists (ACLs) are frequently used in web programs to limit user permissions to specific files located in the root directory. A malicious actor is capable of determining the URL structure the target program uses when requesting files.

- **Insufficient Transport Layer Protection**

  Deals with information transfer between the user (client) and the server (application). Applications often send sensitive data over a network, including session tokens, authentication information, and credit card details. A web application can be compromised and/or sensitive data can be stolen if weak algorithms, expired or incorrect certificates, or the lack of SSL are used in the communication. An attacker can steal legitimate user credentials using this web security flaw and get access to the application.

In this thesis, the proposed system detects that the attackers exploit SQL Injection Attack.

2.2.3 SQL Injection Attack

SQL injection, sometimes referred to as SQLI, is a popular attack method that utilizes malicious SQL code to manipulate the backend database and access data that was not intended to be exposed [4]. Given the right conditions, a hacker can use a SQL Injection vulnerability in order to bypass an internet application's security and authorization measures and retrieve the entire database's contents.

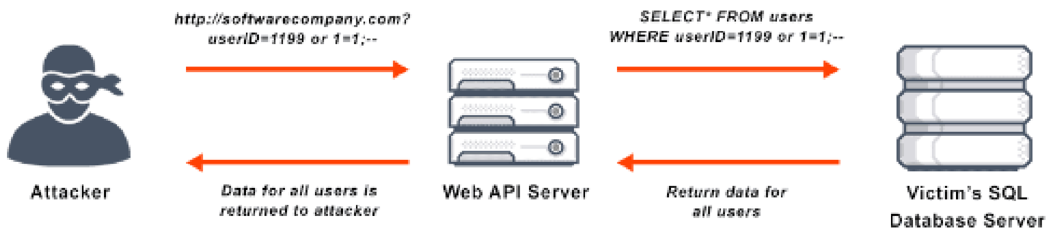

http://softwarecompany.com?
userID=1199 or 1=1;--
SELECT* FROM users
WHERE userID=1199 or 1=1;--
Attacker
Data for all users is returned to attacker
Web API Server
Return data for all users
Victim's SQL Database Server

Figure 2.4 SQL Injection Attack

SQL Injection is a server-side code injection technique that uses a predefined SQL statement to take advantage of a web application's vulnerability to attack the system, inserting the query into the URL or input fields. This query is sent by the web application to the database which then processes it and sends data back to the web application [5]. By using SQL Injection, the attackers successfully get access to the database in this manner. Therefore, attackers get access to sensitive data, the opportunity to modify database information, to run database administrator commands, and the ability to recover system files.

The following are the impact of SQL Injection when it enters the application

- **Steal credentials**— attackers can impersonate users and use relevant privileges after obtaining credentials using SQLI.
- **Access databases**— attackers can access the private information on database servers.
- **Alter data**— attackers have access to the accessed database and can edit or add new data.
- **Delete data**— attackers are able to delete entire tables or erase database records.
- **Lateral movement**— attackers who have operating system privileges can log into database servers and utilize these rights to break into other sensitive systems.

Sample of SQLI Attacks

1. Using SQLI to Authenticate as Administrator

This sample shows how a hacker can bypass authentication in an application and achieve administrative rights by using SQL injection [15]. Assume of a basic username-and-password database table-based authentication solution. The variables user and pass are obtained via a user's POST request, which is added to the following SQL statement:

SQL = "SELECT id FROM users WHERE username='" + user + "' AND password='" + pass + "'"

The attacker uses this SQL statement by concatenating a string like this instead of the `pass` variable:

password' OR 5=5

As a result:

SELECT id FROM users WHERE username='user' AND password='pass' OR 5=5'

In this statement, 5=5 is always true. Consequently, the WHERE clause will return the first ID from the users table, which is usually the administrator. This shows that the attacker is able to access the program without logging in since they have administrator privileges. A more effective variation of this attack involves inserting a code comment symbol at the conclusion of the SQL statement, which allows the attacker to further modify the SQL query.

2. Using SQLI to Access Sensitive Data

In this sample, the code gets the current username before checking for items with a certain item name whose owner is the current user.

```
string userName = ctx.getAuthenticatedUserName();
string query = "SELECT * FROM items WHERE owner = "'"
        + userName + "' AND itemname = '"
            + ItemName.Text + "'";
```

The code generates the following query when the username and item name are combined:

```
SELECT * FROM items WHERE owner =AND

itemname = ;
```

If the attacker inserts the following string for itemname:

```
Widget' OR 5=5
```

SQL statement become:

```
SELECT * FROM items WHERE owner = 'John'

AND itemname = 'Widget' OR 5=5';
```

This is similar to

```
SELECT * FROM items;
```

As a result, the query will return all of the table's data, giving the attacker unauthorized access to confidential information.

3. Injecting Malicious Statements into Form Field

This is a simple user-input-based SQL injection attack. The attacker utilizes a form that asks for the user's first and last names.

The SQL statement that receives the form inputs has the following format:

```
SELECT id, firstname, lastname FROM authors
```

When the attacker inserts a fraudulent expression to the first name, SQL statement become:

SELECT id, firstname, lastname FROM authors WHERE firstname = 'malicious'ex' and lastname ='newman'

Due to the single punctuation, the database recognizes invalid syntax and attempts to execute the malicious statement.

2.3 Machine Learning

Machine learning is a field of technology that allows systems to carry out a range of activities, such as forecasts, suggestions, estimations, etc., based on prior knowledge or historical data [2]. It encourages computers to behave like people by using projected data and prior experience. It is a technique for computer algorithms that improve mechanically with time.

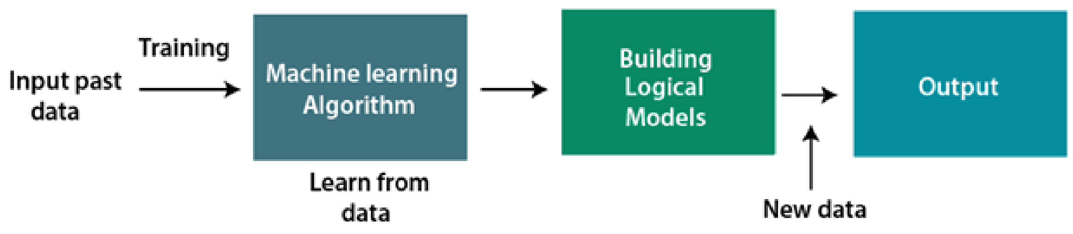


Figure 2.5 Working of Machine Learning

2.3.1 Machine Learning Techniques

Machine Learning techniques are divided mainly into the following 4 categories:

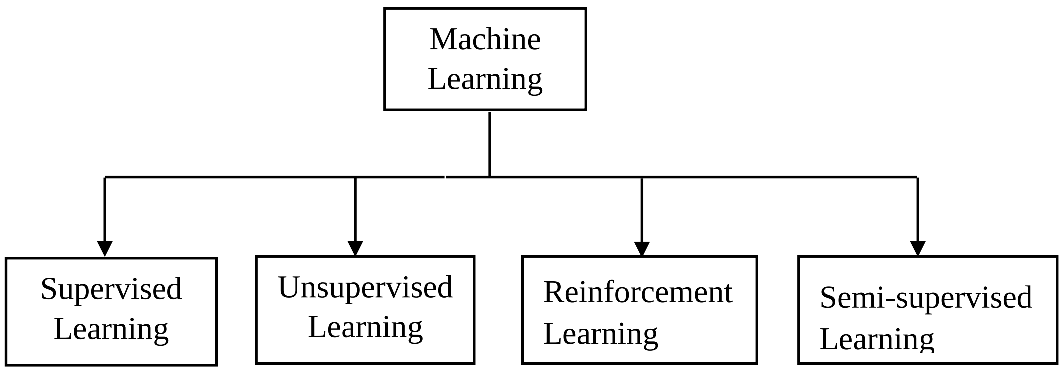


Figure 2.6 Types of Machine Learning

2.3.1.1 Unsupervised Learning

Unsupervised learning is a type of learning where a computer needs to pick up information without any human intervention [23]. The machine is trained by using a set of unlabeled, unclassified, or uncategorized data and the algorithm is required to respond independently on that data. Unsupervised learning's objective is to reorganize the incoming data into new features or a collection of objects with related patterns. There is no predefined outcome in unsupervised learning. The machine searches through the massive volume of data for helpful insights. It can also be divided into two different types of algorithms:

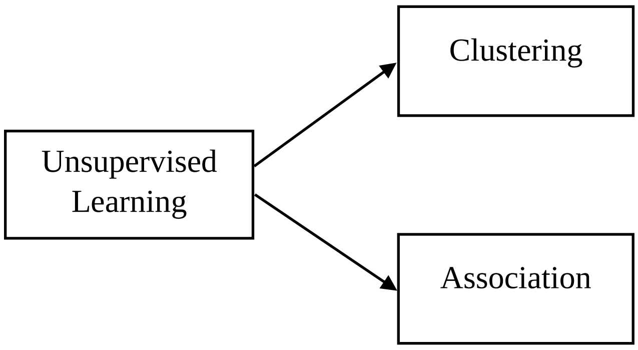


Figure 2.7 Unsupervised Learning

2.3.1.2 Supervised Learning

In supervised learning, sample labeled data is given to the machine learning system as training material and then it uses that information to predict the outcome [3]. The system builds a model using labeled data to analyze the datasets and learn about each one. After training and processing, the model is tested by utilizing sample data to determine whether it accurately predicts the desired outcome. Two categories of algorithms can be used to further categorize supervised learning:

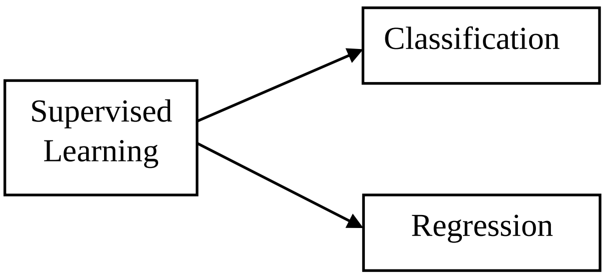


Figure 2.8 Types of Supervised Learning

Reinforcement Learning

Reinforcement Learning is a feedback-based machine learning technique [11]. Compared to supervised learning, reinforcement learning does not require the explicit correction of undesirable behavior or the presentation of labelled input/output pairs. There are no labeled data, therefore the agent can only learn through experience. Agents (computer programs) in this type of learning are required to investigate their surroundings, take actions and then receive rewards as feedback for their actions. They receive positive reinforcement for all actions and negative reinforcement for all actions. A reinforcement learning agent's objective is to maximize the good outcomes.

Semi-supervised Learning

Semi-supervised learning is a method that stands between supervised and unsupervised learning [14]. It combines the benefits of both supervised and unsupervised learning without the challenges involved with finding a large amount of labeled data by using both enormous amounts of unlabeled data and limited amounts of labeled data. Its functions apply to datasets with a limited number of labels as well as datasets with unlabeled data. However, the data is typically unlabeled. As a result, it also reduces the cost of the machine learning model because labels are costly yet could not be required for business goals. Furthermore, it raises the efficiency and precision of the machine learning model. Data scientists can overcome the drawbacks of supervised and unsupervised learning with the help of semi-supervised learning.

2.3.2 Machine Learning Model

Machine learning contributes to building a model that can analyze more data to make predictions after being trained on a set of training data [1]. Machine learning systems have utilized that a variety of models are:

- Decision Trees
- Support Vector Machine (SVM)
- Linear Regression
- K Nearest Neighbour (KNN)
- Random Forest
- Logistic Regression
- Naïve Bayes

2.3.3 Decision Trees

Decision Tree is a supervised learning method that can be applied to classification and regression issues, however it is most frequently used to solve classification issues. It is a tree-structured classifier, where internal nodes stand in for a dataset's features, branches for the decision-making process and each leaf node for the classification result. The Decision Node and Leaf Node are the two nodes of a decision tree. While Leaf nodes are the results of decisions and do not have any more branches, Decision nodes are used to create decisions and have numerous branches. The decision tree's general structure is shown in the diagram below:

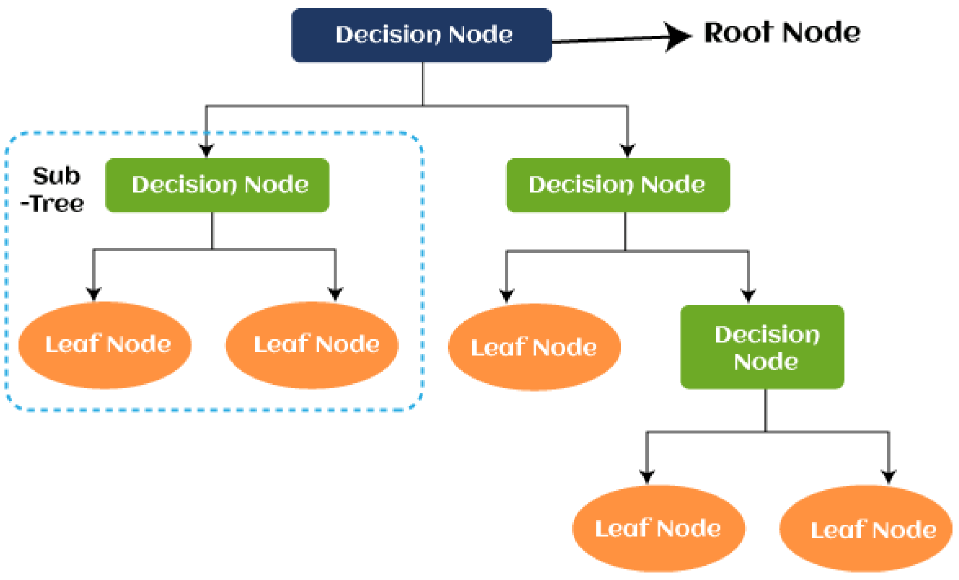


Figure 2.9 Decision Tree

2.3.4 Support Vector Machine (SVM)

One of the most widely used methods for Supervised Learning, Support Vector Machine (SVM), is used to solve Classification and Regression issues. However, it is mainly employed in Machine Learning Classification issues. The SVM algorithm's objective is to establish the best line or decision boundary that can divide n-dimensional space into classes, allowing us to quickly classify fresh data points in the future. The term "hyperplane" refers to this optimal decision boundary. SVM picks the extreme vectors and points that aid in the creation of the hyperplane. Support vectors, which are used to represent these extreme situations, form the basis for the SVM method.

Consider the diagram below, where a decision boundary or hyperplane is used to categorize two distinct categories:

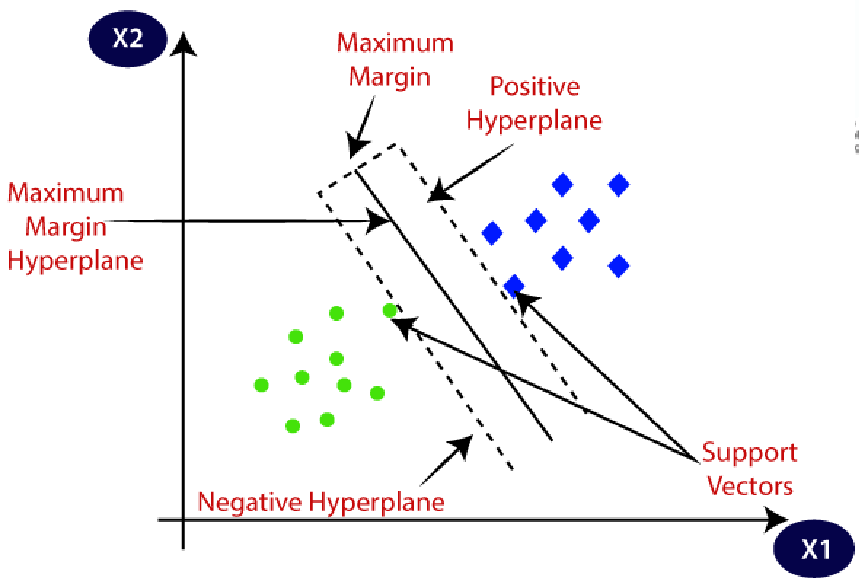


Figure 2.10 Support Vector Machine

2.3.5 Linear Regression

A variable's value can be predicted using linear regression analysis based on the value of another variable. The term "dependent variable" refers to the variable that want to forecast [17]. The independent variable is the one using to make a prediction about the value of the other variable. This type of analysis involves one or more independent variables that can most accurately predict the value of the dependent variable and estimates the coefficients of the linear equation. The differences between expected and actual output values are minimized by linear regression by fitting a straight line or surface. The best-fit line for a set of paired data can be found using straightforward linear regression calculators that employ the "least squares" technique.

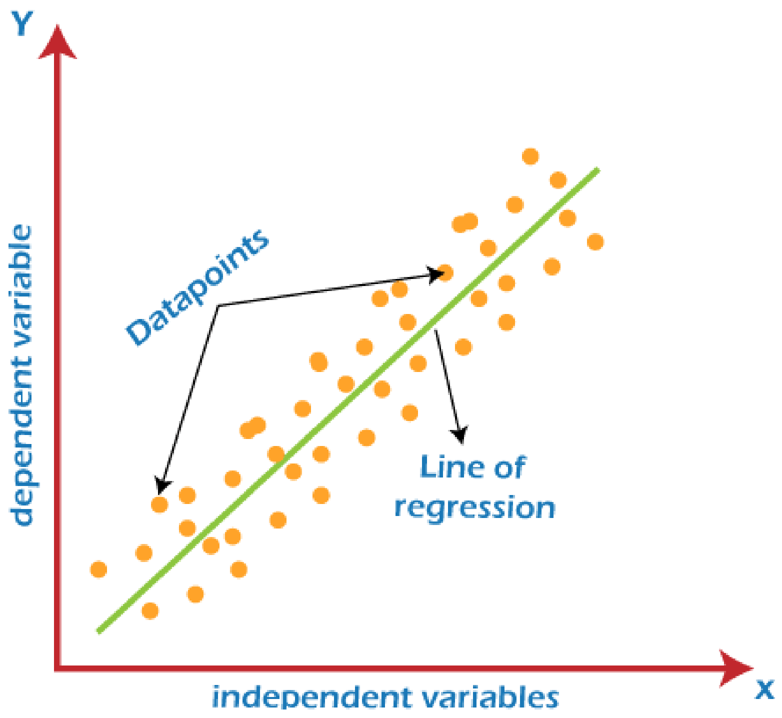


Figure 2.11 Linear Regression

2.3.6 K Nearest Neighbour (KNN)

One of the simplest machine learning algorithms, based on the supervised learning method, is K-Nearest Neighbour [11]. The K-NN algorithm assumes that the new case and the existing cases are comparable, and it places the new sample in the category that is most like the existing categories. The K-NN algorithm saves all the information that is available and categorizes new input based on similarity. This means that utilizing the K-NN method, fresh data can be quickly and accurately sorted into a suitable category.

Although the K-NN approach is most frequently employed for classification problems, it can also be utilized for regression. Since K-NN is a non-parametric technique, it makes no assumptions about the underlying data. It is also known as a lazy learner algorithm since it saves the training dataset rather than learning from it's immediately. Instead, it uses the dataset to perform an action when classifying data. The KNN method simply saves the information during the training phase, and when it receives new data, it categorizes it into a category that is quite similar to the new data.

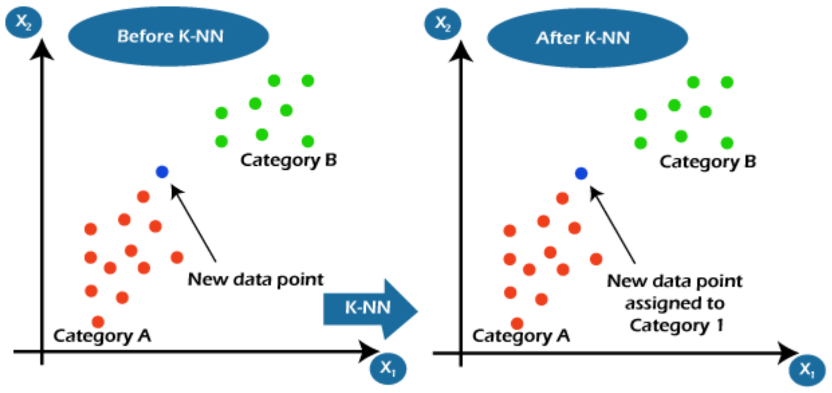


Figure 2.12 K Nearest Neighbour

2.3.7 Random Forest

Popular machine learning algorithm Random Forest is a part of the supervised learning methodology [2]. It can be applied to machine learning issues involving both classification and regression. It is built on the idea of ensemble learning, which is a method of integrating various classifiers to handle difficult issues and enhance model performance. Random Forest, as the title suggests, is a classifier that uses a number of decision trees on different subsets of the provided dataset and averages them to increase the dataset's predictive accuracy. Instead of depending on a single decision tree, the random forest uses forecasts from each tree and predicts the result based on the votes of the majority of predictions. Higher accuracy and overfitting are prevented by the larger number of trees in the forest. The following diagram illustrates Random Forest algorithm.

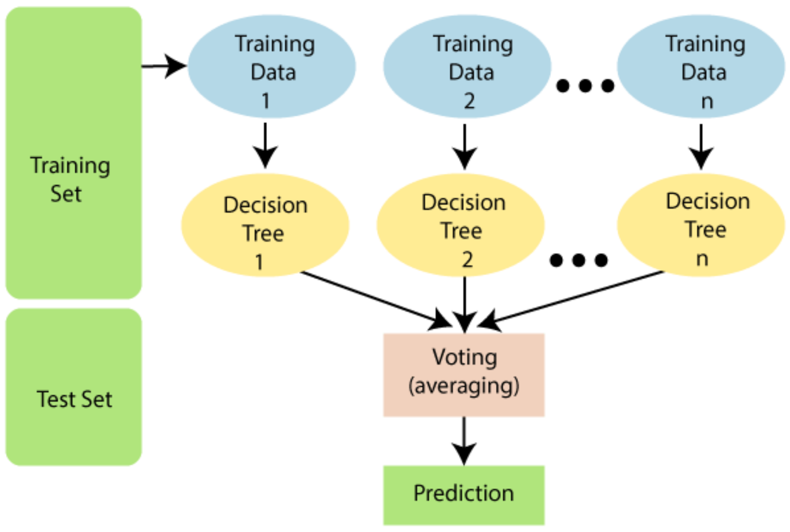


Figure 2.13 Random Forest

2.3.8 Naïve Bayes

The Naïve Bayes algorithm is a supervised learning method for classification issues that is based on the Bayes theorem. It is mainly employed in text classification with a large training set. The Naïve Bayes Classifier is one of the most simple and effective classification algorithms available today. It aids in the development of quick machine learning models capable of making accurate predictions. Being a probabilistic classifier, it makes predictions based on the likelihood that an object will occur. Spam filtration, Sentimental analysis and article classification are a few examples of Naïve Bayes algorithms that are frequently used. There are three different kinds of Naïve Bayes models and they are as follows:

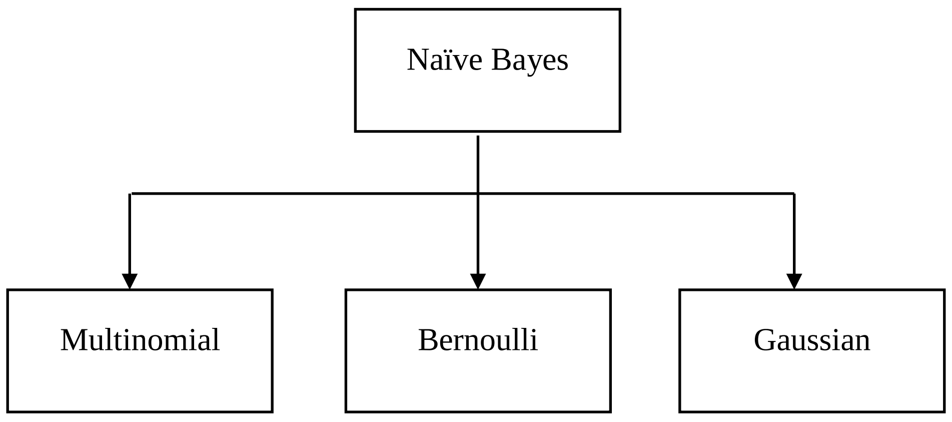


Figure 2.14 Types of Naïve Bayes Model

Multinomial Naïve Bayes

One of the variations of the Naive Bayes algorithm used in machine learning is the Multinomial Naïve Bayes. When the data is multinomially distributed, the Multinomial Naive Bayes classifier is employed. It indicates the category a specific document falls under, such as Sports, Politics, Education, etc., and is largely used to solve document classification issues. In classification tasks based on natural language processing, this algorithm is particularly preferred. Word frequency is used by the classifier as a predictor. When modeling feature vectors with each value indicating, for instance, the frequency or number of occurrences of a keyword, a multinomial distribution is helpful. If each of the n elements in the feature vectors can take on one of k possible values with probability pk, then:

$$P(X_1 = x_1 \cap X_2 = x_2 \cap \ldots \cap X_k = x_k) = \frac{n!}{\prod_i x_i!} \prod_i p_i^{x_i}$$

Equation 2.1

This approach can be applied to both continuous and discrete data. It is easy to use and can be applied to predict real-time applications. Large datasets may be handled with ease and it is very scalable.

Bernoulli Naïve Bayes

A member of the Naive Bayes family is Bernoulli Naive Bayes. Similar to the Multinomial classifier, the Bernoulli classifier uses independent Boolean variables as predictor variables. Such as if a particular word is present or not in a document. This model is also famous for document classification tasks. When the dataset has a binary distribution and the output label is either present or absent, it can be used very effectively. This algorithm's main feature is that it only recognizes features as binary values, such as:

- True or False
- Spam or Ham
- Yes or No
- 0 or 1

Compared to other classification methods, it is very quick. This approach provides more accurate results when compared to other classification algorithms in the situation of a little dataset, which is opposite to certain machine learning algorithms that do not perform well when the dataset is small. It is quick and can easily manage irrelevant features.

Gaussian Naïve Bayes

In Gaussian Naive Bayes, it is assumed that each feature's continuous values are distributed according to a Gaussian distribution. The term "Normal distribution" can also be used to describe a Gaussian distribution. As illustrated below, when plotted, it produces a bell-shaped curve that is symmetric about the mean of the feature values:

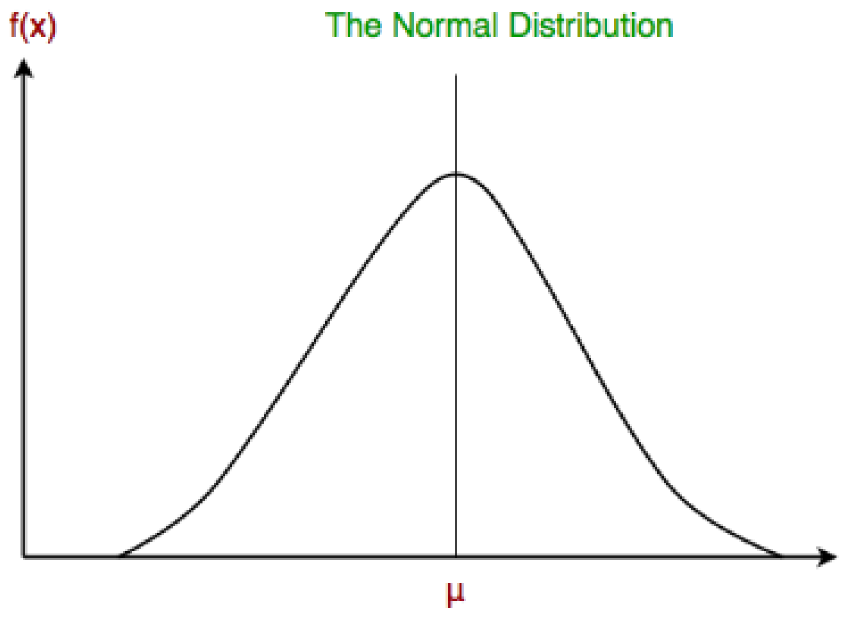


Figure 2.15 Gaussian Distribution

2.3.9 Logistic Regression

One of the most often used Machine Learning algorithms, within the category of Supervised Learning, is logistic regression. Using a predetermined set of independent factors, it is used to predict the categorical dependent variable. In a categorical dependent variable, the output is predicted via logistic regression. As a result, the result must be a discrete or categorical value. The logistic function's curve represents the probability of a certain outcome, such as whether cancerous cells are present or not, or whether a mouse is fat or not based on its weight, etc. Because it can classify new data using both continuous and discrete datasets, logistic regression is a key machine learning approach. When classifying observations using various sources of data, logistic regression can be used to quickly identify the factors that will work well. The logistic function is displayed in the graphic below:

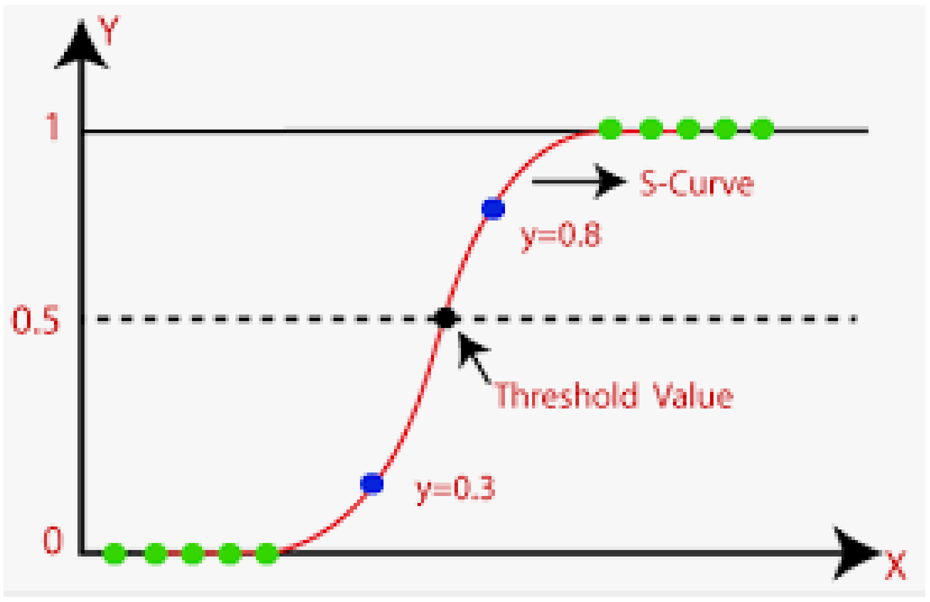


Figure 2.16 Logistic Regression

2.4 Chapter Summary

This chapter described about the concept of web architecture, web application and vulnerability. And then, the sample of SQL Injection Attack is also explained. Moreover, about machine learning and techniques are also presented. Finally, how machine learning model build is described as detail.

CHAPTER 3
DESIGN OF THE PROPOSED SYSTEM

The main goal of the thesis is to detect SQLI Attack. Firstly, the overview of the proposed system is described. And each of the algorithm that takes part in the main program is described in a detailed explanation. Moreover, software requirement to use the proposed system is described. Finally, it is described that the summary of this chapter.

3.2 Overview of the Proposed System

The following Figure 3.1 shows the block diagram of the proposed system. The dataset is taken from the Kaggle website. Data pre-processing was necessary because the collected data does not contain many useless pieces of information. The dataset is partitioned in this step using the 70/30 rule for machine learning. Data pre-processing entails turning raw data into clean data and maintaining the data that is most significant for training. In essence, it eliminated duplicate data and noisy data. In this system removes duplicate rows and same query but different label in the dataset. Moreover, the proposed system extracts the feature by using Regular Expression. The system then used the dataset to simultaneously train and test the model. After that, the performance of the system is evaluated. Finally, the system created a model that worked well with the dataset. These datasets were classified into two parts: training data for the model's training and testing data for determining whether the algorithm is producing the desired outcomes. A website was created to evaluate the model's usability in different areas. The model was then tested after being connected to the website.

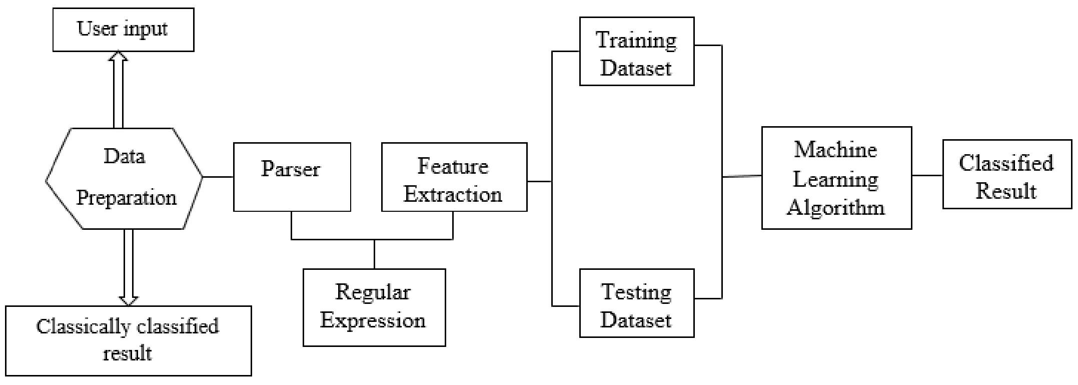


Figure 3.1 Block Diagram of the Proposed System

3.1.1 Software Requirement

Jupyter Notebook: An open-source IDE is called Jupyter Notebook is used to produce Jupyter documents, which can be shared with live codes and developed. Additionally, it is an interactive computing environment based on the web. The Jupyter notebook can handle a number of widely used data science languages, including Python, Julia, Scala, R, and others. A web-based interactive computing platform is Jupyter Notebook. Live code, mathematics, narrative text, infographics, interactive dashboards, and other media are all combined in the notebook. It uses blocks to provide in-line code execution and provides help for in-line graphing. It’s very flexible.

Python Programing Language: PyQt is a Python binding for Qt, a collection of C++ libraries and development tools that offer abstractions for graphical user interfaces that are agnostic of platform (GUIs). Along with many other delicate features, Qt offers tools for networking, threading, regular expressions, SQL databases, SVG, OpenGL, XML, and many more. The basis for a Qt class's operation is a slot mechanism that facilitates communication between objects to make the construction of easily reusable software components possible. Python is utilized in video games, operating systems, artificial intelligence, machine learning, and mobile application development. Python is regarded as one of the easiest programming languages to learn for a novice, but it can be challenging to master.

3.1.2 System Flow Diagram

In Figure 3.2 is shown system flow of the proposed system. At first, this system picks up the SQL Injection dataset from the database. And then, this system checks duplicate rows in the dataset. If it has duplicate rows, it will remove from this system. Additionally, this system checks same query with different label in the dataset. If it has same query with different label, the system drops rows that happen in this case. Finally, the system converts lower case of all statement in the dataset. And then this system extracts the feature by using regular expression and save the dataset. After that, this dataset splits into training dataset and testing dataset. Furthermore, this system builds a model with Naïve Bayes method by using training dataset and then checks model accuracy with testing dataset by using confusion matrix. Additionally, this system extracts the feature when user input is entered into the system. After that, this input is checked with training model whether this statement is malicious or not. And then, this system shows to the user that this statement is SQL Injection or not.

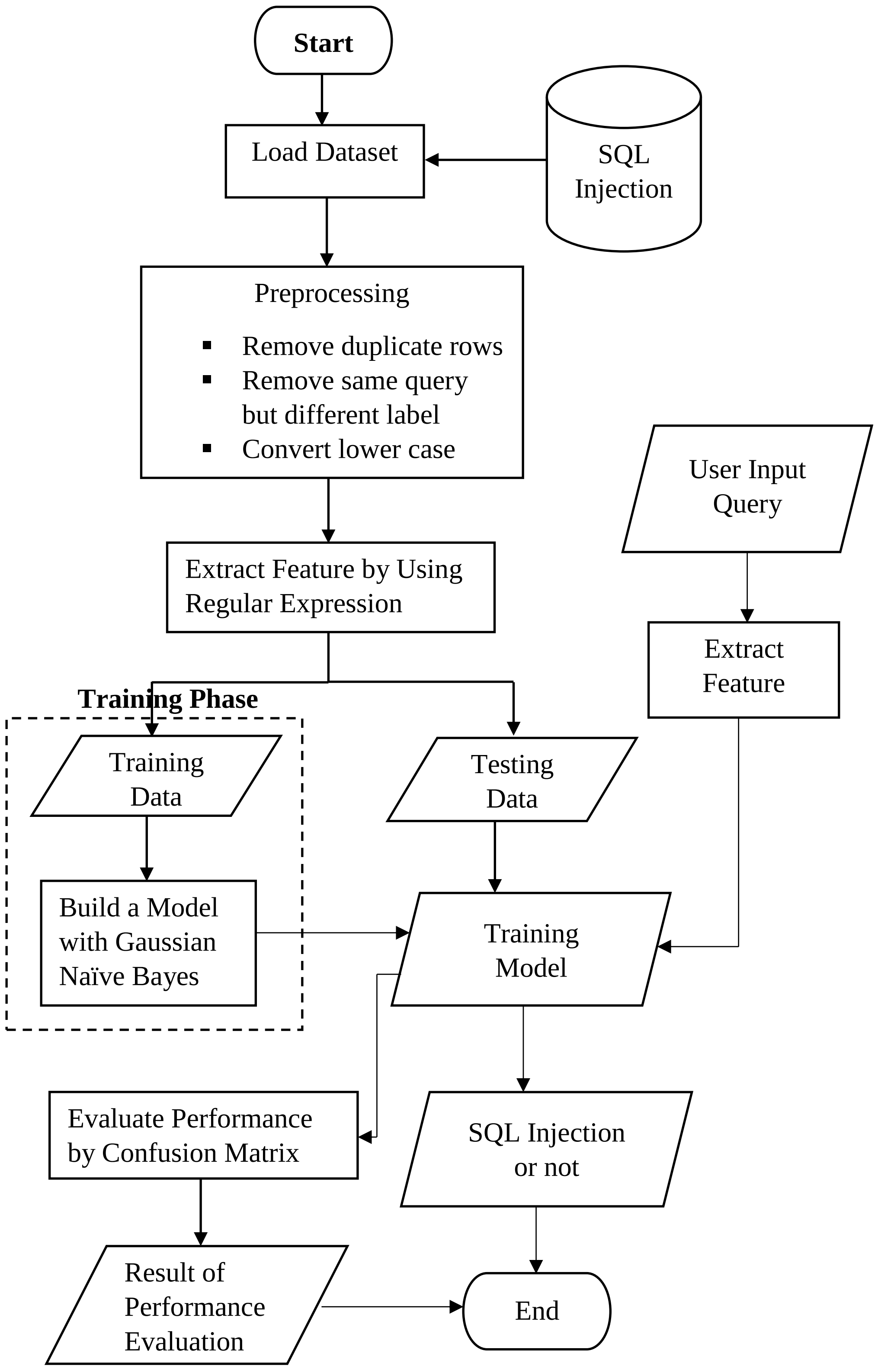
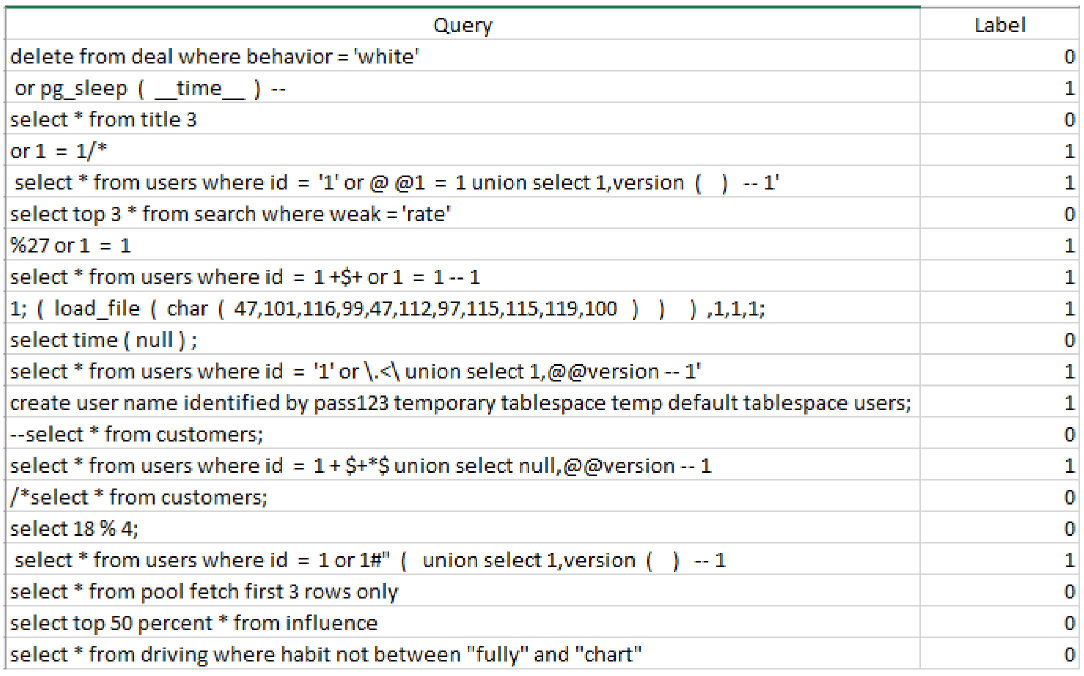

Start
Load Dataset
SQL Injection
Preprocessing
Remove duplicate rows
Remove same query but different label
Convert lower case
User Input Query
Extract Feature by Using Regular Expression
Extract Feature
Training Phase
Training Data
Testing Data
Build a Model with Gaussian Naïve Bayes
Training Model
Evaluate Performance by Confusion Matrix
SQL Injection or not
Result of Performance Evaluation
End

Figure 3.2 System Flow Diagram

3.3 Description of the Dataset

The experimental dataset for the SQL Injection Attack was obtained from the Kaggle website. The sample dataset has two columns of data and 30,919 different values. The first column represents a query that has to be identified as either a normal statement or a SQL Injection Attack, and the second column provides a numeric value that helps identify the type of statement it is. In this case, the sentence has been represented by the value 1 as a SQL Injection query and by the value0 as a regular statement. The sample dataset of the proposed system is described in table 3.1.

Table 3.1 Sample Dataset of the Proposed System



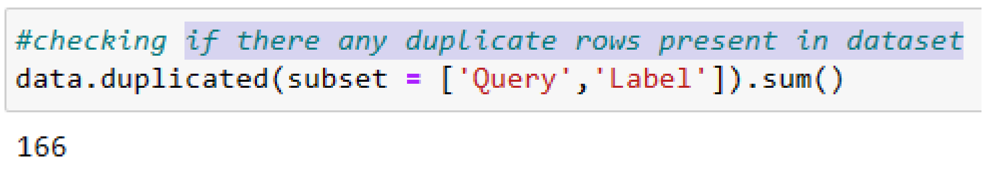
Query	Label
delete from deal where behavior = 'white'	0
or pg_sleep (__time__) --	1
select * from title 3	0
or 1 = 1/*	1
select * from users where id = '1' or @ @1 = 1 union select 1,version ( ) -- 1'	1
select top 3 * from search where weak = 'rate'	0
%27 or 1 = 1	1
select * from users where id = 1 +$+ or 1 = 1 -- 1	1
1; (load_file (char (47,101,116,99,47,112,97,115,115,119,100))) ,1,1,1;	1
select time (null) ;	0
select * from users where id = '1' or \.<\ union select 1,@@version -- 1'	1
create user name identified by pass123 temporary tablespace temp default tablespace users;	1
--select * from customers;	0
select * from users where id = 1 + $+*$ union select null,@@version -- 1	1
/*select * from customers;	0
select 18 % 4;	0
select * from users where id = 1 or 1#" (union select 1,version ( ) -- 1	1
select * from pool fetch first 3 rows only	0
select top 50 percent * from influence	0
select * from driving where habit not between "fully" and "chart"	0

3.4 Data Preprocessing

Data preprocessing is the process of transforming raw data into something that can be used by a machine learning model. It is the first and most crucial step in the process of creating a machine learning model. Since real-world data frequently contains noise, missing values, and can be in an unfavorable format, it cannot be used to directly train machine learning models. The accuracy and efficiency of a machine learning model are increased by data preprocessing, which is required to clean the data and prepared it for the model.

In this system, the queries are made up of punctuation, special letters, etc. All of these characteristics set the SQL Injected query apart from regular SQL queries. So, as a part of text preprocessing, punctuation, HTML elements is will not be deleted. Because it eliminates some special characters that are essential for feature engineering and distinguishing between SQL and SQLI queries, word stemming and stop word removal are also not carried out. As a result, text preprocessing will only convert the text as lowercase.

Firstly, the system checks that there have any duplicate rows present in dataset. If it has any duplicate rows present in dataset, this rows are deleted.

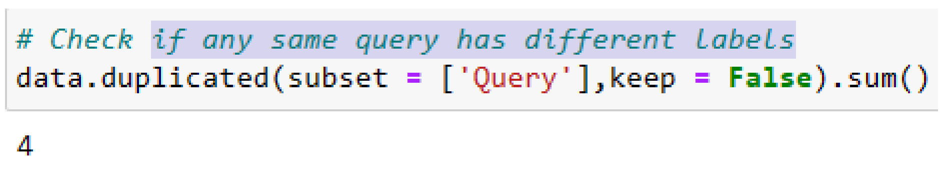


```
#checking if there any duplicate rows present in dataset
data.duplicated(subset = ['Query','Label']).sum()
```

```
166
```

Figure 3.3 Result of Duplicated Rows after checking

After that, the system checks that there has any same query but different labels. If it has any same query with different labels, this rows are deleted. Figure 3.4 shows that the output after checking the same query but different labels.



```
# Check if any same query has different labels
data.duplicated(subset = ['Query'],keep = False).sum()
```

```
4
```

Figure 3.4 Result of Same Query but Different Labels

The following Table 3.2 shows the outcome of the dataset after data preprocessing.

Table 3.2 Output of the Dataset after Data Preprocessing

SQL Injection or not	Each of Types before Data Preprocessing	Each of Types after Data Preprocessing
SQL Injection	11382	11375
No SQL Injection	19537	19373
Total	30,919	30748

3.5 Feature Extraction

Feature extraction is the process of breaking down the input data into a set of features so that the task at hand can be carried out using this condensed representation rather than the original, full-size input [1]. It is a method for reducing important features from a big input data collection. Any extraction technique is used to extract each feature from datasets. A machine learning model can benefit from feature extraction when being trained. Any extraction technique is used to extract every phrase from datasets. It results in:

- A Boost in training speed
- An improvement in model accuracy
- A reduction in risk of overfitting
- A rise in model explain ability
- Better data visualization

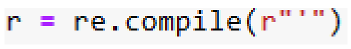
3.5.1 Regular Expression

Regular Expression is a set of symbols that defines a pattern of text that must match in order to make a filter more specific or general. RegEx, sometimes referred to as regular expression, is a generalized expression that is used to match patterns with different character sequences [17]. A string formatted sequence of characters is defined by regular expressions. It uses patterns to match strings. The following Figure 3.5 shows sample of Regular Expression.

Pattern	Description
--	Single line comments
/* */	Multiline comments
‘	Single quotation
‘’	Double quotation
[ ] ( ) , ;	Punctuations
&& || != ==	Logical operator
+ - * /	Arithmetic operators

Figure 3.5 Sample of Regular Expression

RegEx in python is used for tokenizing each entry in both the SQL Injection and normal query datasets. A sequence of characters is defined a string format. Regular expressions are popularly used in pattern matching. In this thesis, a regular expression sequence is converted into a regular expression object using the re.compile( ) method.



```
r = re.compile(r"'")
```

Figure 3.6 re.compile( ) for Pattern Matching

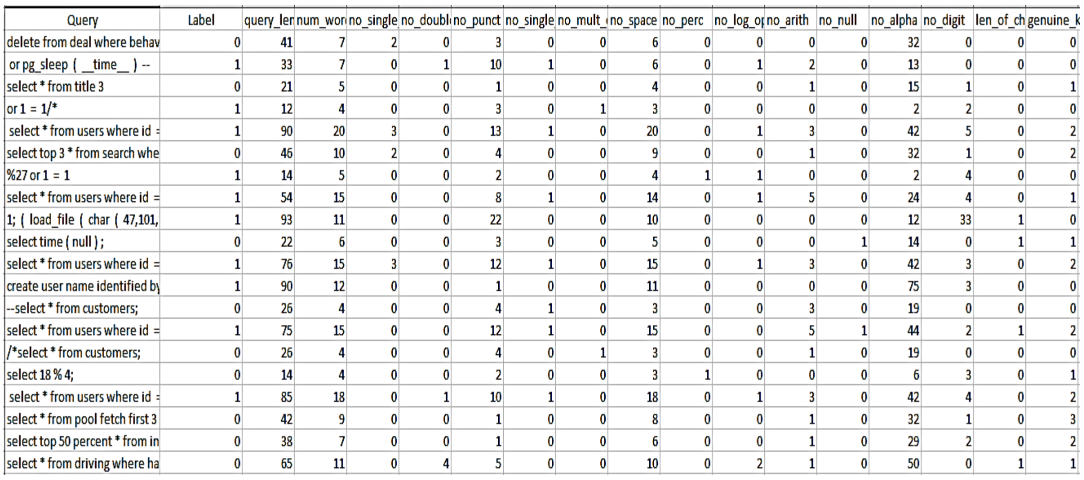
In order to generate the regular expression object, certain SQL queries and SQL reserved words are used. findall(string) method is used to match the string with the result of re.compile( ). The string is scanned left-to-right, and matches are returned in the order found. The outcome relies on how many capturing groups there are string in the pattern. A list of strings matching the entire pattern if there are no groups is returned. A list of strings that match that group is returned if there is exactly one group. A list of tuples of matched strings is returned if there are numerous groups. Non-capturing groupings have no impact on the shape of the outcome. Table 3.3 shows the extracted feature of the proposed system. The sample dataset after extracting the feature is shown in Table 3.4.

Table 3.3 Result of the Dataset after extracting feature

Column	Description
query_len	Query length
num_word_query	Num words query
no_single_qts	Num of single line comment
no_double_qts	Num of double line comment
no_punctn	Num of single quotation
no_single_cmnt	Num of double quotation
no_double_cmnt	Num of punctuation
no_white_space	Num of white space in a query
no_percent	Num of percentage symbols
no_log_optr	Total number of logical operator in a query
no_arith_oprtr	Total number of arithmetic operator
no_null_val	Total number of null values in a query
no_alphabet	Total number of alphabets in a query
no_digits	Total number of digits

len_of_chr_char_null	length of chr + char + null keywords
genuine_keywords	genuine_keywords

Table 3.4 The Sample of Feature Extraction



Query	Label	query_ler	num_wor	no_single	no_doubl	no_punct	no_single	no_mult_	no_space	no_perc	no_log_op	no_arith	no_null	no_alpha	no_digit	len_of_ch	genuine_k
delete from deal where behav	0	41	7	2	0	3	0	0	6	0	0	0	0	32	0	0	0
or pg_sleep (__time__) --	1	33	7	0	1	10	1	0	6	0	1	2	0	13	0	0	0
select * from title 3	0	21	5	0	0	1	0	0	4	0	0	1	0	15	1	0	1
or 1 = 1/*	1	12	4	0	0	3	0	1	3	0	0	0	0	2	2	0	0
select * from users where id =	1	90	20	3	0	13	1	0	20	0	1	3	0	42	5	0	2
select top 3 * from search whe	0	46	10	2	0	4	0	0	9	0	0	1	0	32	1	0	2
%27 or 1 = 1	1	14	5	0	0	2	0	0	4	1	1	0	0	2	4	0	0
select * from users where id =	1	54	15	0	0	8	1	0	14	0	1	5	0	24	4	0	1
1; (load_file (char (47,101,	1	93	11	0	0	22	0	0	10	0	0	0	0	12	33	1	0
select time (null) ;	0	22	6	0	0	3	0	0	5	0	0	0	1	14	0	1	1
select * from users where id =	1	76	15	3	0	12	1	0	15	0	1	3	0	42	3	0	2
create user name identified by	1	90	12	0	0	1	0	0	11	0	0	0	0	75	3	0	0
--select * from customers;	0	26	4	0	0	4	1	0	3	0	0	3	0	19	0	0	0
select * from users where id =	1	75	15	0	0	12	1	0	15	0	0	5	1	44	2	1	2
/*select * from customers;	0	26	4	0	0	4	0	1	3	0	0	1	0	19	0	0	0
select 18 % 4;	0	14	4	0	0	2	0	0	3	1	0	0	0	6	3	0	1
select * from users where id =	1	85	18	0	1	10	1	0	18	0	1	3	0	42	4	0	2
select * from pool fetch first 3	0	42	9	0	0	1	0	0	8	0	0	1	0	32	1	0	3
select top 50 percent * from in	0	38	7	0	0	1	0	0	6	0	0	1	0	29	2	0	2
select * from driving where ha	0	65	11	0	4	5	0	0	10	0	2	1	0	50	0	1	1

3.6 Data Splitting

Data splitting is the segment of a set of data into two or more subgroups. Testing or evaluating the data in one part of a two-part split and training the model in the other are common practices. Data splitting is an essential part of data science, particularly when developing models from data. This technique helps to guarantee the correctness of data model generation and the processes, such machine learning, that employ data models. In a conventional two-part data split, the training data set is utilized to train and develop models. It is common practice to estimate different parameters or evaluate the effect of multiple models using training sets. The testing data set is used after the training. The training and test sets of data are compared to confirm that the final model performs as expected.

In order to avoid overfitting, data splitting is widely utilized in machine learning. In that case, a machine learning model is unable to continuously fit new data because it fits existing training data too well. A machine learning model often divides the original data into three or four sets. The training set, the development set and the testing set are the three sets that are frequently used:

- **Training Set:** The training set refers to the portion of data used to train the model. The model must observe and learn from the training set in order to improve any one of its parameters.

- **Development Set/Validation Set:** The development set is a data set of examples used to alter the settings for the learning process. It is also known as the model validation set or cross-validation set. The objective of this data set is to evaluate the model's accuracy which can be aided in model selection.
- **Testing Set:** The data component tested in the final model is referred to as the "testing set," and it is compared with the data from the earlier sets. The testing set serves as an assessment of the chosen algorithm and method.

In this system, data is split into as following:

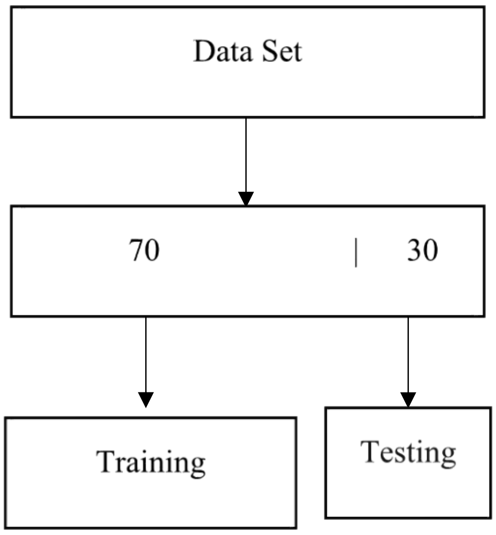


Figure 3.7 Splitting Dataset for Classification

3.7 Methodology of the Proposed System

Machine Learning is the powerful for classification and regression. It has typically four techniques such as supervised learning, un-supervised learning, reinforcement and semi-supervised learning. Supervised learning uses labeled datasets, in order to train algorithms that effectively identify data or predict outcomes. Among various of supervised learning method, the proposed system detects SQL Injection or not by using Naïve Bayes method because it is simplest to understand when the technique is explained using binary or categorical input values.

3.7.1 Naïve Bayes

The Naive Bayes classification algorithm is appropriate for both binary and multiclass classification. Compared to numerical input variables, naive Bayes performs better in cases of categorical input variables. It is helpful for analyzing data and making predictions based on past situations. Naive Bayes algorithm is a probabilistic classifier. It is based on probability models that make extensive assumptions about independence. The term "Naive Bayes classifiers" refers to a set of classification methods built on the Bayes theorem.

3.7.2 Bayes Theorem

The likelihood of an event occurring given the likelihood of an earlier event occurring is determined by the Bayes Theorem. The mathematical formulation of Bayes' theorem is given by the equation:

$$\boldsymbol{P(H|X) = \frac{P(X|H)P(H)}{P(X)}}$$

Equation 3.1

where, X=data sample

H=hypothesis

P(H|X)=posterior probability of hypothesis

P(H)=prior probability

P(X|H)=likelihood

P(X)=probability that sample data is observed

3.7.3 Gaussian Naïve Bayes

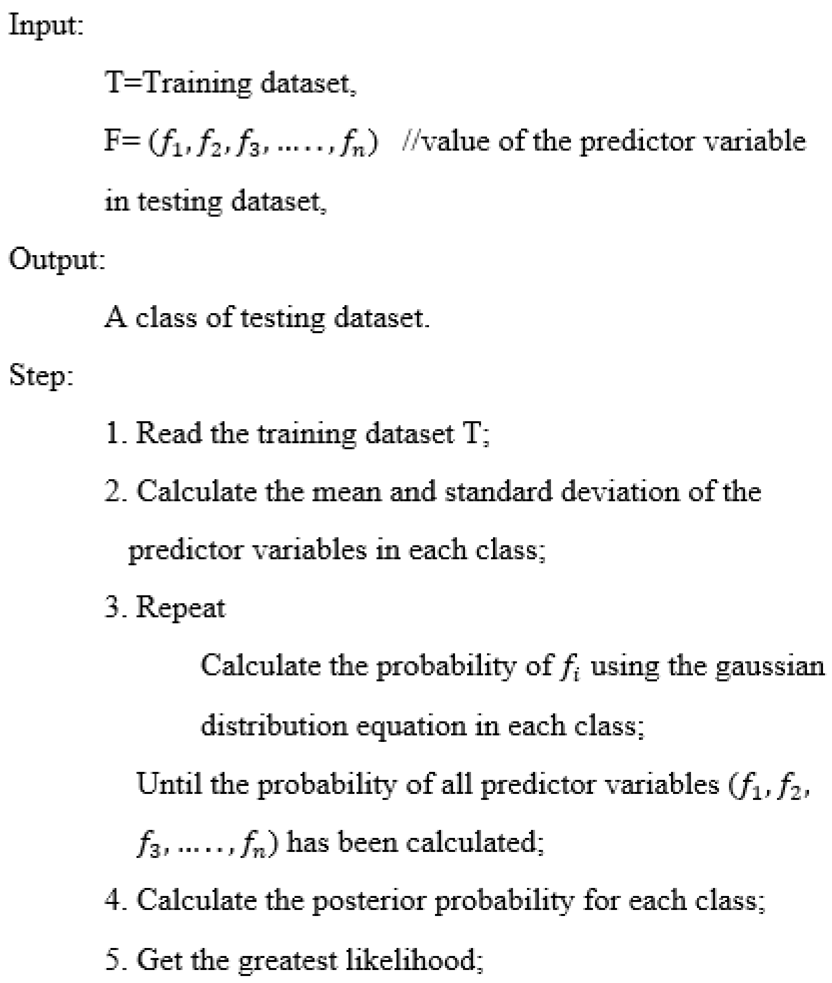
The extension of naive Bayes is called Gaussian Naïve Bayes. The central limit theorem makes the normal distribution (or Gaussian distribution), commonly known as the bell curve, extremely helpful. When there are several random variables, the states of the normal distribution that are averages of the random variables converge to the normal distribution and are normally distributed. Gaussian Naïve Bayes makes the assumption that every class has a Gaussian distribution. All the continuous variables associated with each feature distributed according to Gaussian Distribution,

$$\mathbf{f(x)} = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\overline{x})^2}{2\sigma^2}}$$

Equation 3.2

The Gaussian (or Normal) distribution is the simplest to deal with because it calculates to estimate the mean and standard deviation from given training data. Other functions can be used to estimate the distribution of the data. To build the Gaussian Naïve Bayes model, mean and standard deviation is calculated for the training data.

3.7.2.1 Algorithm of Gaussian Naïve Bayes



```
Input:
        T=Training dataset,
        F= $(f_1, f_2, f_3, \dots\dots, f_n)$   //value of the predictor variable
        in testing dataset,
Output:
        A class of testing dataset.
Step:
        1. Read the training dataset T;
        2. Calculate the mean and standard deviation of the
           predictor variables in each class;
        3. Repeat
                Calculate the probability of $f_i$ using the gaussian
                distribution equation in each class;
           Until the probability of all predictor variables $(f_1, f_2,$
           $f_3, \dots\dots, f_n)$ has been calculated;
        4. Calculate the posterior probability for each class;
        5. Get the greatest likelihood;
```

3.8 Chapter Summary

In this chapter, overview of the proposed system is presented. Moreover, the explanation about step-by-step of system's implementation is described. And then explains about the dataset and the detail of the method which are used in this system are described in this chapter.

CHAPTER 4
IMPLEMENTATION AND EXPERIMENTAL RESULT OF THE PROPOSED SYSTEM

In this chapter, the output classification results of the system have been shown step by step. And then, the performance evaluation of the system is explained and result of the system is shown in the following figures.

4.1 Implementation of the System

As soon as the system launches, the user can see its main form., as shown in Figure 4.1. A menu bar makes up the main form. When user clicks file menu, emerges start menu.

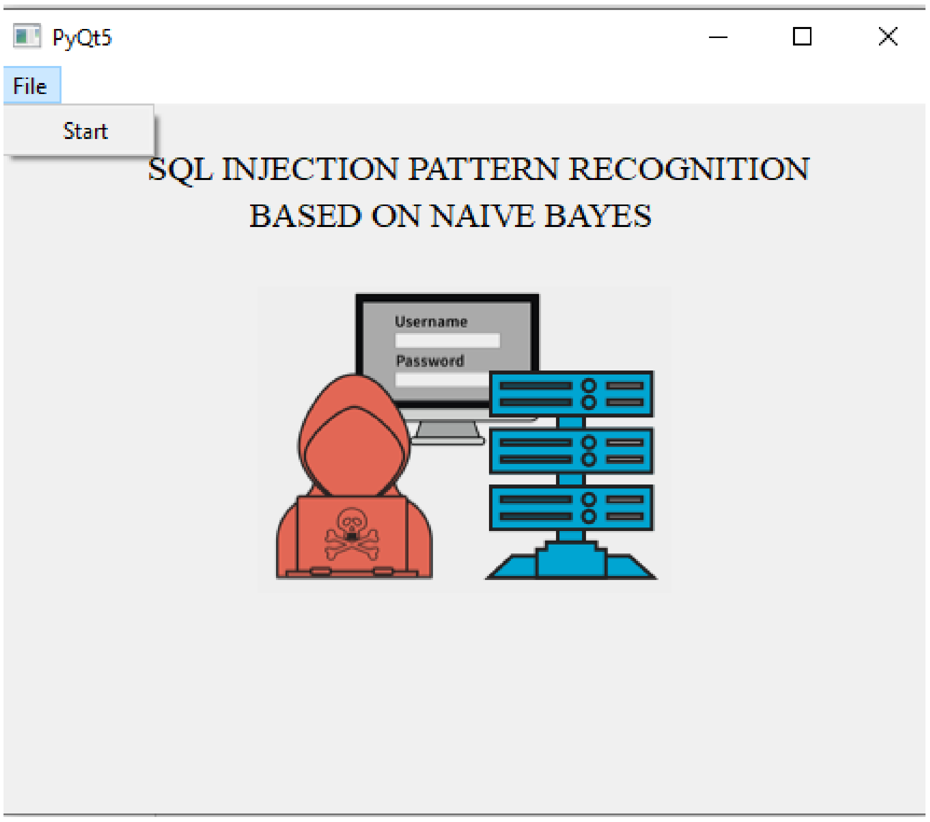


Figure 4.1 Main Form of the Proposed System

If user clicks start menu, user can see detection of SQL Injection form as shown in Figure 4.2. This form corporates label, text box, button and menu bar.

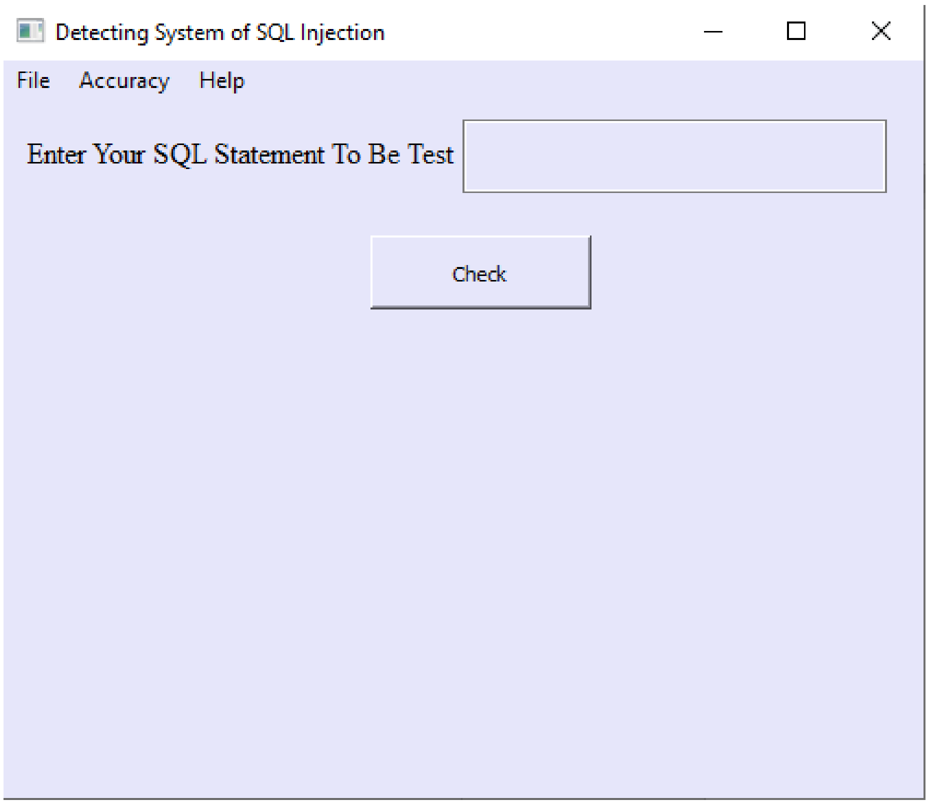
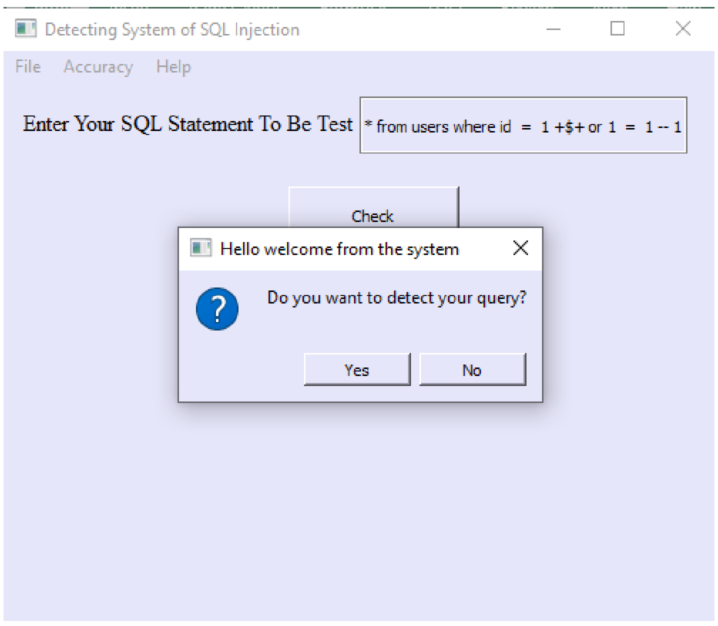

Detecting System of SQL Injection
File
Accuracy
Help
Enter Your SQL Statement To Be Test
Check

Figure 4.2 Home Page of the System

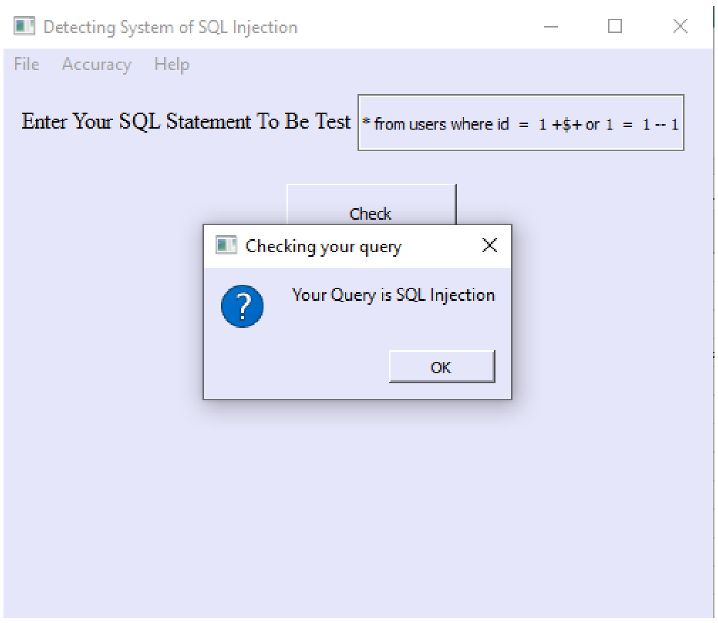
User must enter SQL statement in the textbox to test. And then, this system asks "Do you want to detect your query?" if the button is clicked. If user clicks the "OK" button, this system checks this query and displays the malicious or not as shown in Figure (4.3 and 4.4).


Detecting System of SQL Injection
File
Accuracy
Help
Enter Your SQL Statement To Be Test
Check
Hello welcome from the system
Yes
No

* from users where id = 1 +$+ or 1 = 1 -- 1

Do you want to detect your query?

(a)



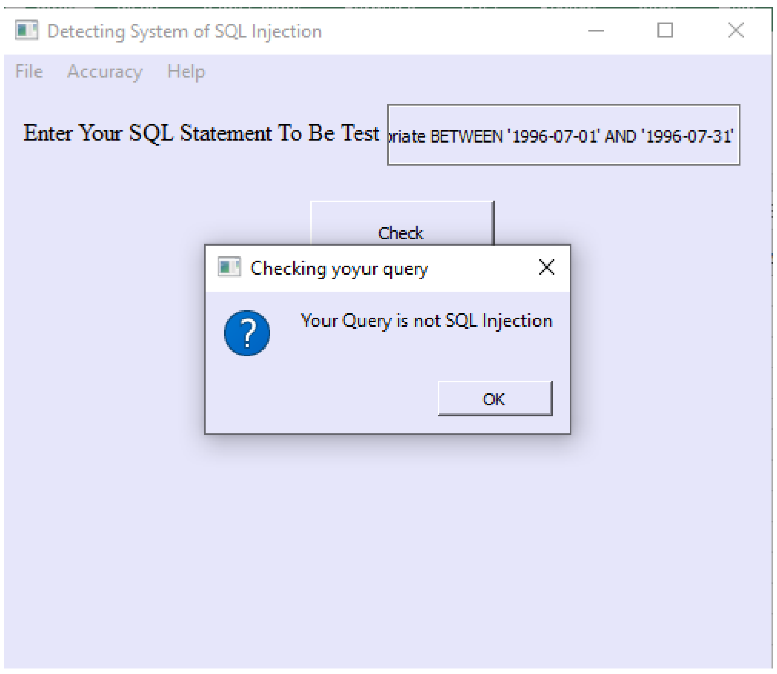
Enter Your SQL Statement To Be Test

```
* from users where id  =  1 +$+ or 1  =  1 -- 1
```

Your Query is SQL Injection

(b)

Figure 4.3 Display the result that SQL Injection Attack

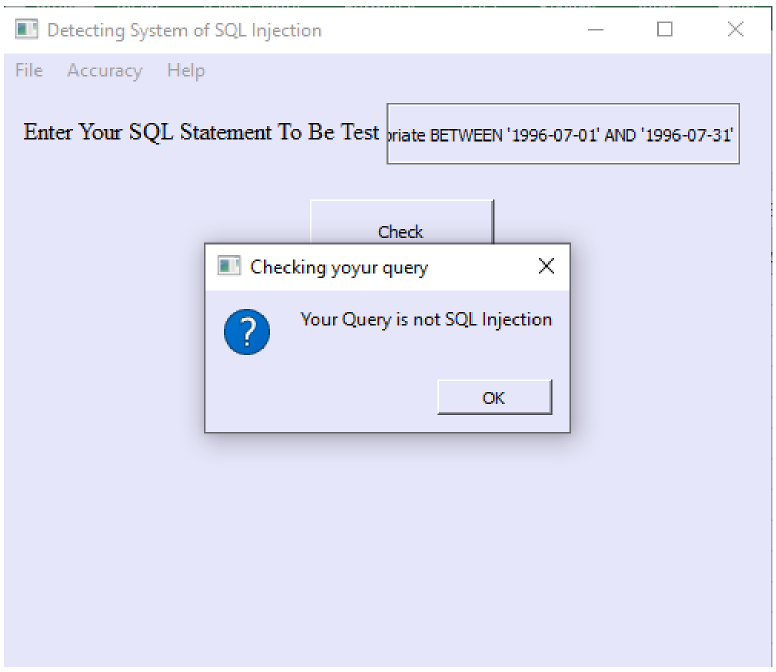


Enter Your SQL Statement To Be Test

```
priate BETWEEN '1996-07-01' AND '1996-07-31'
```

Your Query is not SQL Injection

(a)


Detecting System of SQL Injection
File Accuracy Help
Enter Your SQL Statement To Be Test
Check
OK

priate BETWEEN '1996-07-01' AND '1996-07-31'

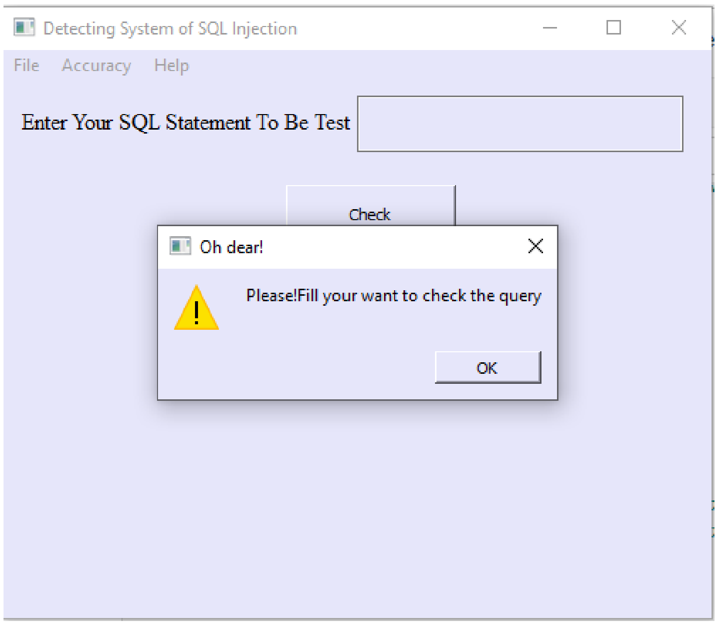
Checking yoyur query

Your Query is not SQL Injection

(b)

Figure 4.4 Display the result that no SQL Injection

If user clicks the "Check" button without entering the query, this system shows the notification for the user.


Detecting System of SQL Injection
File Accuracy Help
Enter Your SQL Statement To Be Test
Check
OK

Oh dear!

Please!Fill your want to check the query

Figure 4. 5 Notification of the Proposed System

4.2 Performance Evaluation of the System

The machine learning method includes performance evaluation as a crucial step. It is crucial to evaluate how machine learning models generalize on test data in order to confidently trust their predictions.

4.2.1 Model Evaluation Metric for Classification

Four types of results are produced by predictions for classification problems: true positives, true negatives, false positives, and false negatives. Types of results are produced by prediction shown in Table 4.1.

Table 4.1 Types of results are produced by prediction

Prediction Result	Explanation
TP (True Positive)	predicted SQL Injection and are actually SQL Injection
FP (False Positive)	predicted no SQL Injection and are actually SQL Injection
TN (True Negative)	predicted no SQL Injection and are actually no SQL Injection.
FN (False Negative)	predicted SQL Injection and are actually no SQL Injection

4.2.2 Confusion Matrix

Confusion Matrix is a diagram that illustrates the differences between actual and predicted values [6]. It measures how well a machine learning classification model is performing. Table 4.2 shows a confusion matrix that describes how well a classification system performs.

The visualization of crucial predictive metrics, such as accuracy, precision, recall and f1 score, is done using confusion matrices.

Table 4.2 Confusion Matrix

Predict / Actual	**SQLInjection(Positive)**	**SQL(Negative)**
SQLInjection	TP	FN
SQL	FP	TN

Accuracy: The percentage of accurate predictions for the test cases is what is meant by accuracy. Accuracy is the most typical evaluation parameter for classification issues. It is calculated as the proportion of accurate predictions to all other predictions (or input samples). As previously mentioned, accuracy is used to assess a model, but it is not a reliable predictor of model performance.

$$\textbf{Accuracy} = \frac{\mathbf{TP + TN}}{\mathbf{TP + FP + TN + FN}}$$

Equation 4.1

Precision: The proportion of positive outcomes among all positively expected occurrences. The denominator is the model prediction that was determined to be correct using the whole dataset utilized in this study. Consider determining "how much the model is right when it says it is right" to be the goal. The aim of precision is to quantify the fraction of positive predictions that were in fact accurate. The dataset utilized in this analysis serves as the denominator.

$$\textbf{Precision} = \frac{\mathbf{TP}}{\mathbf{TP + FP}}$$

Equation 4.2

Recall: Percentage of instances that are positive out of all instances that are actually positive. The actual number of positive cases in the dataset is the denominator (TP + FN) in this equation. Consider it as an attempt to determine "how many additional right ones, the model missed when it presented the right ones." Recall aims to find the percentage of correct positive predictions that were actually made.

$$\textbf{Recall} = \frac{\mathbf{TP}}{\mathbf{TP + FN}}$$

Equation 4.3

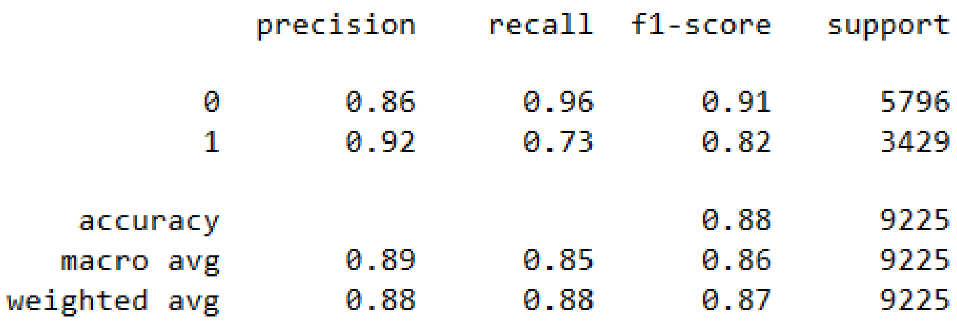
F1 score: also known as F-measure. The precision and recall of a test are both taken into account by the metric known as the F-score to generate the score. It is described as the harmonic mean of recall and precision in this post. An F-score have a maximum value of 1, which denotes perfect precision and recall, and a minimum value of 0, which occurs when either precision or recall are zero.

$$\text{F-measure} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

Equation 4.4

4.3 Experimental Results

The effective of supervised learning method as Naïve Bayes algorithm to predict SQL Injection Attack are used in this system. It has been discovered that the performance assessment of the suggested system performs well in terms of accuracy, precision, recall, and F1-score as evaluation metrics. This model is determined to be roughly 88% accurate. The classification report is displayed in Figure 4.6.



```
              precision    recall  f1-score   support

           0       0.86      0.96      0.91      5796
           1       0.92      0.73      0.82      3429

    accuracy                           0.88      9225
   macro avg       0.89      0.85      0.86      9225
weighted avg       0.88      0.88      0.87      9225
```

Figure 4.6 Result of Naïve Bayes Classifier

The proposed system's precision, recall, and f1-score results as a bar chart is shown in Figure 4.7,

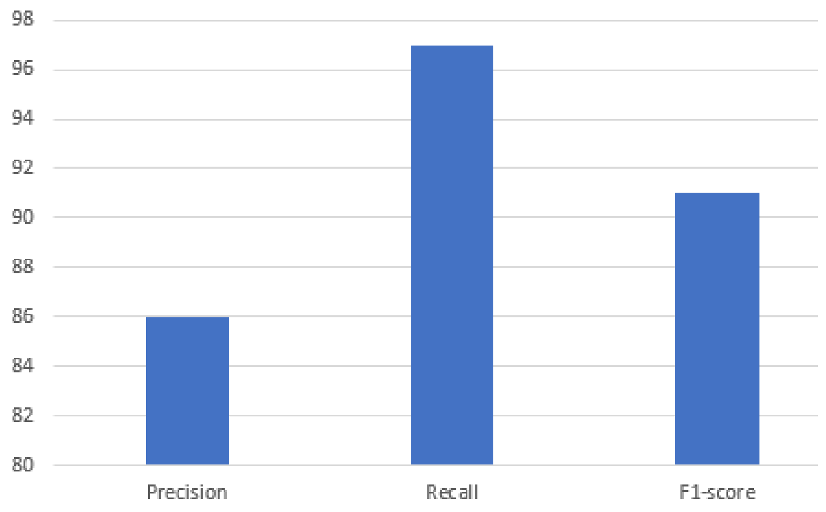


Figure 4.7 Precision, Recall and F1-score of the Proposed System

Model accuracy and error rate of the proposed system is shown in the following pie chart. Accuracy is 88% and error rate is 12% in this system.

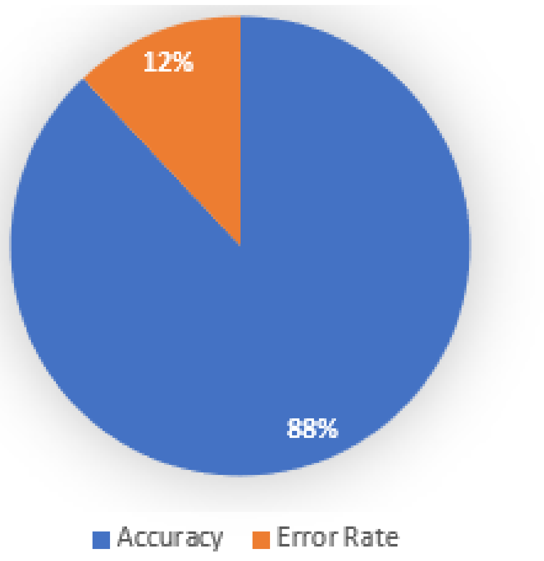


Figure 4.8 Accuracy and Error Rate of the Proposed System

The training data and testing data are constituted the majority of the dataset. The model is applied on 30,748 records, of which 30% (9,225) are used for testing and 70% (21,523) are used for model training. The confusion matrix, also known as the error matrix, for the suggested model is shown in Figure 4.9 below. It describes confusion matrix for Gaussian Naïve Bayes model. The model is tested with 4661 data in which 5592 are TP, 204 are FP and negligible data of FP and FN.

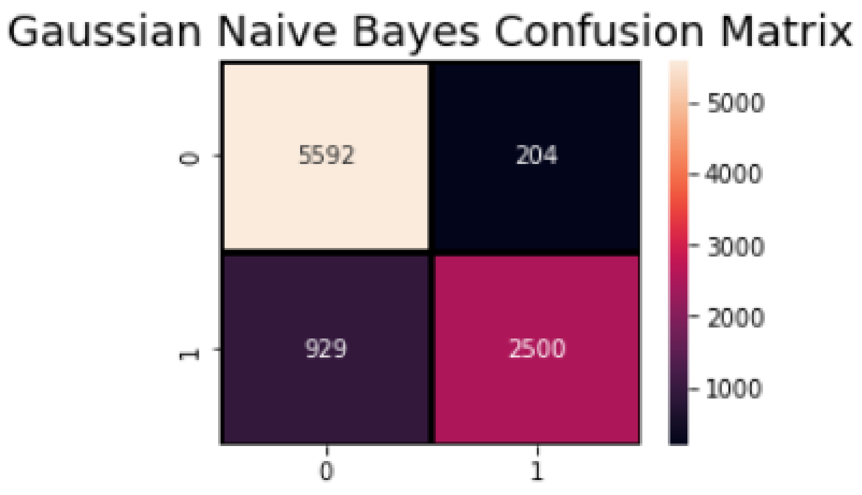


Figure 4.9 Confusion Matrix of the Proposed System

As illustrated in Table 4.3, the system's outcome is summarized. This model has an F1 score of 91% with accuracy, precision, and recall values of 88%, 86% and 97% respectively.

Tale 4. 3 Summarizing the Result of Naïve Bayes Method

	Precision	Recall	F1-score
Naïve Bayes Algorithm	86%	97%	91%

4.3 Chapter Summary

In this chapter, implementation of the proposed system is explained step-by-step. And then, the performance evaluation of the system is presented as detailed. Moreover, the experimental result of the proposed system is described as bar chart, pie chart and confusion matrix.

CHAPTER 5
CONCLUSION AND FURTHER EXTENSION

With so much data passing over the web every day, it is still crucial to identify SQL injection attacks that seriously compromise the security of any web-based application running on a server connected to the Internet or a cloud. Currently, one of the most common ways to attack a system's database is through a SQL injection attack. While many attack types come and go over time, SQL injection is a method that persists It keeps coming back to harm the security of websites from distinct viewpoints.

A web application could suffer serious issues if it was the target of a SQL injection attack [20]. It is imperative to come up with a feasible solution to this issue. For identifying and fixing this problem, researchers have developed a number of techniques. There is no method that can resist every kind of SQL injection attack. SQL Injection attacks continue to be a major source of concern for cyber security experts. Systems for detecting SQL Injection must be able to tell between new, entirely unknown attacks.

It takes on so many different forms that traditional defenses find it difficult to keep up. It can only be effectively managed by a defense mechanism that exist currently. The suggested system's model makes the use of such a machine learning method to enable future detection of new types of SQLI Attack attempts. It will be able to recognize any website from SQL injection attacks. By adding more datasets with SQL statements that will not threaten the website's security, this technology enhances the model. By include these kinds of datasets, the system will be able to train the model more successfully and reduce the model's false positive rate This will assist in raising the model's effectiveness.

In this proposed system, a Nave Bayes-based machine learning algorithm is used to recognize SQL injection attacks. This system uses a classifier to identify fraudulent queries. The Naïve Bayes Classifier is one of the most simple and effective classification algorithms available today. It aids in the development of quick machine learning models capable of making accurate predictions. A classification method is utilized to classify the SQL Injection or normal query. The proposed classifier classifies the test set with 88% accuracy. The suggested approach can be improved to detect various types of SQL injection attacks by appropriately extracting features. This is a huge aid in stopping a SQLI attack for a security officer or security analyst.

5.1 Further Extensions and Limitations of the System

This proposed system could be improved upon in terms of usability and effectiveness for future work. Static code analysis and web application firewalls, as well as the machine learning technique to identifying SQL Injections, may be utilized in conjunction with one another. Better feature extraction can also progress the machine learning model. In this system, features for the machine learning model are produced using regular expression. For feature extraction and more efficient model training, alternative methods can be applied.

REFERENCES

[1] Anamika Joshiand and Geetha V, " SQL Injection Detection using Machine Learning," International Conference on Control, Instrumentation, Communication and Computational Technologies, 2014.

[2] Basic Concepts in Machine Learning https://www.javatpoint.com/basic-concepts-in-machine-learning

[3] B. Hanmanthu, B. R. Ram, P. Niranjan, "SQL Injection Attack prevention based on decision tree classification," 2015 IEEE 9th International Conference on Intelligent Systems and Control (ISCO), Coimbatore, 2015, pp. 1-5.

[4] D. Kar, A.K. Sahoo, K. Agarwal, S. Panigrahi, M. Das, "Learning to detect SQLIA using node centrality with feature selection," 2016 International Conference on Computing, Analytics and Security Trends (CAST), Pune, 2016, pp. 18-23.

[5] D. Kar, S. Panigrahi, "Prevention of SQL Injection attack using query transformation and hashing," in Advance Computing Conference (IACC), 2013 IEEE 3rd International, vol., no., pp.1317-1323, 22-23 Feb. 2013

[6] Evaluating Machine Learning Model Performance, Collins Ayuya, November 26, 2020

[7] G.T.Buehrer, RW.Weide, and P.AG.Sivilotti, "Using Parse Tree Validation to Prevent SQL Injection Attacks," International Workshop on Software Engineering and Middleware (SEM), 2005.

[8] K. Zetter, “That insane, $81m bangladesh bank heist? here’s what we know,” Wired, 2016

[9] Muhammad Amirulluqman Azman, Mohd Fadzli Marhusin and Rossilawati Sulaiman, “Machine Learning-Based Technique to Detect SQL Injection Attack,” Journal-of-Computer-Science-1549-3636, 2021.

[10] M.KarthiKeyan, “An Efficient Technique For Preventing SQL Injection Attack Using Pattern Matching Algorithm,” IEEE International Conference on Emerging Trends in Computing, Communication and Nanotechnology (ICECCN), 2013.

[11] Musaab Hasan, Zayed Balbahaith, and Mohammed Tarique, “Detection of SQL Injection Attacks: A Machine Learning Approach,” International Conference on Electrical and Computing Technologies and Applications (ICECTA), 2019.

[12] P. N. Joshi, N. Ravishankar, M. Raju, and N. C. Ravi, “Encountering SQL Injection in web applications,” in 2018 Second International Conference on Computing Methodologies and Communication (ICCMC), IEEE, 2018, pp. 257– 261.
[13] Q. Li, W. Li, J. Wang, and M. Cheng, “A SQL Injection detection method based on adaptive deep forest,” IEEE Access, vol. 7, pp. 145 385–145 394, 2019.
[14] Ryohei Komiya, Incheon Paik, Masayuki Hisada, “Classification of Malicious Web Code by Machine Learning,” Awareness Science and Technology (iCAST), 2011 3rd International Conference on, vol., no., pp.406,411, 27-30 Sept. 2011.
[15] R. Komiya, I. Paik, M. Hisada, "Classification of malicious web code by machine learning," 2011 3rd International Conference on Awareness Science and Technology (iCAST), Dalian, 2011, pp. 406- 411.
[16] Sangeeta, S Nagasundari and PrasadB Honnavali, “SQL Injection Attack Detection using ResNet,” IEEE – 45670, Int Conf. 10th International Conference on Computing, Communication and Networking Technologies,2019
[17] Sonali Mishra “SQL Injection Detection Using Machine Learning,” 2019
[18] SQL Injection Attack: Real Life Attacks and Code Examples, Admir Dizdar, April 8, 2022
[19] Stasista, “Average cyber losses to global companies in the last fiscal year as of May 2019, by company size.” [online],” Wired, 2016.
[20] S.W.Boyd and AD.Keromytis, "SQLrand: Preventing SQL Injection Attacks," Proc. the 2nd Applied Cryptography and Network Security (ACNS) Conference, pp. 292-302, Jun 2004.
[21] The Open Web Application Security Project (OWASP). The Ten Most Critical Web Application Security Risks 2010.
[22] What is a Web Application? A beginner’s guide, January 11, 2021
[23] Z.Su and G.Wassermann " SQL Injection Detection Using Machine Learning," The 33rd Annual Symposium on Principles of Programming Language (POPL 2006), Jan 2006.

PUBLICATION

[1] Hsu Wai Tun, Khaing Khaing Wai, “SQL Injection Pattern Recognition Based on Naïve Bayes Model”, University of Computer Studies Yangon, Myanmar, 2022.