

Solent University

MSc Applied AI and Data Science

Dissertation COM726

Bug Detection Using Artificial Intelligence and Cyber Security

Project Presented

by

Khushbu Kalgude

September 2022

Abstract

Every year, more vulnerabilities are found in software and websites, whether they are found internally in proprietary code or reported officially. These vulnerabilities have a strong probability of being used against users, which could damage systems and lead to data leakage. Injection attacks using structured query language and header vulnerabilities are among the risks. When speaking of web-based applications, SQL injection attacks are among the most common vulnerabilities. An organization or corporation may suffer harm as a result of this type of attack since the attacker can steal sensitive and important information. Nowadays, using ML to many situations shows to be greatly advantageous. It uses to construct a variety of items that function like human and save time. In this research paper used an algorithm called Extreme Gradient Boosting algorithm from ensemble machine learning methods to classify and detect SQL injection. And created a chatbot with it; by using it the user will be able to know about the identified vulnerabilities, the risk, and the mitigation technique.

Keywords: Vulnerabilities, SQL Injection, Machine Learning, XGBoost Algorithm, Chatbot

ACKNOWLEDGEMENT

I feel fortunate to get continuous guidance, support, and encouragement from my supervisor Mr. Hamidreza Soltani. His experience in Artificial Intelligence and Machine Learning helped me to develop and implement my project and overcome the technical challenges that I faced during implementation. I would like to thank him for his constant guidance that helped me in the research and implementation related to my project. I appreciate the support provided by the Solent University Southampton lecturers, Dr. Olufemi Isiaq, and my course leader Dr. Shakeel Ahmed, during my studies.

Contents

1) Introduction and Background	1
1.1) Research Questions	2
1.2) Aim and Objectives.....	2
2) Literature Review	4
3) Method and Methodology	5
3.1) Resources	5
3.2) Dataset.....	5
3.3) Exploratory Data Analysis.....	6
4) Machine Learning Algorithms	14
4.1) Logistic Regression	14
4.2) Random Forest	14
4.3) Decision Tree.....	15
4.4) Extreme Gradient Boosting	15
5) Web Application	16
5.1) Web Application using .NET Core.....	16
5.2) Web Application using Java.....	20
5.3) Main Web Application using Python.....	23
6) Experiment and Results.....	26
6.1) Compare ML Algorithms.....	27
6.2) Results	30
6.3) Final Backend Coding	31
7) Chatbot	37
8) Conclusion	39
9) Limitation and Future Work	40
References.....	41

Figures

Figure 1: Imbalance Dataset

Figure 2: Random Undersampling

Figure 3: Balanced SQL Query Class Distribution

Figure 4: Duplicate Data in Dataset

Figure 5: Remove Duplicate Rows

Figure 6: Log Query Length

Figure 7: TF-IDF

Figure 8: Load _Layout.cshtml in Variable

Figure 9: Faculty Index Page

Figure 10: Update Page

Figure 11: Database of .NET Web Application

Figure 12: Model File Coding

Figure 13: Using Method

Figure 14: SQL Configuration

Figure 15: SQL Configuration for Update

Figure 16: Login Page

Figure 17: Registration Page

Figure 18: Home Page

Figure 19: Result of Home Page

Figure 20: Database of Java Web Application

Figure 21: Style the Content

Figure 22: Default Page Coding of Vulnerability Detection

Figure 23: Front-end of Vulnerability Detection Website

Figure 24: Web Application Flow Chart

Figure 25: CountVectorizer Function

Figure 26: Output of matrix

Figure 27: Flask and HTTP Session

Figure 28: Fetch Forms Details

Figure 29: Test Vulnerability in Form without ML

Figure 30: Result 1 of Vulnerabilities Detection Without ML Algorithm

Figure 31: Result 2 of Vulnerabilities Detection Without ML Algorithm

Figure 32: XGBoost Algorithm Apply to Detect SQLi

Figure 33: Web Page of SQLi Detection with ML

Figure 34: Web Page Result of SQLi Detection with ML

Figure 35: Coding of intents.json File

Figure 36: Chatbot

Tables

Table 1: Resources

Table 2: Decision Tree vs Extreme Gradient Boosting

Acronym

SQL – Structure Query Language

OWASP – Open Web Application Security Project

AI – Artificial Intelligence

IDS – Intrusion Detection System

HTML – Hypertext Markup Language

CSS – Cascading Style Sheets

C# - C-sharp

SQLi – SQL injection

NLTK – Natural Language Toolkit

NLP – Natural Language Processing

TF-IDF - Term Frequency and Inverse Document Frequency

ML – Machine Learning

MVC – Model View Controller

1) Introduction and Background

Every year, development and uses of websites or software has been very rapid in every field, with that also increase the number of security challenges. In order to develop web applications, various types of code are used like jQuery, HTML, CSS, and many more. A database is also used to store user data. When a website is being developed, there are occasionally certain coding flaws that go undiscovered. Additionally, developers generally utilize open-source codes without verifying whether the entire code is secure, and security gaps allow attackers to access systems and apps. There are numerous types of vulnerabilities, including SQL injection, XSS, broken authentication, session management, etc. Injection attacks have been ranked as the most common and dangerous type of attack that web applications receive by OWASP for several years running. SQL and XSS injection are the most major injections. For example, when an attacker sends SQL commands to a SQL database, they gain control over the server and commit attacks known as SQL injections. By doing this, an attacker may be able to take complete control of the server. The attacker might gain access to incredibly sensitive data, and as a result, they can destroy and modify it. Using various tools, the cyber security analyst can identify these types of vulnerabilities in various websites or pieces of software, however the procedure takes time. Another method to identifying vulnerabilities which has static analysis of incoming online traffic, commonly referred to as signature detection, is a common method of classical web attack mitigation. This tactic requires the creation of a signature specific to the online attack, and when this signature is recognized, a firewall or other security appliance can block the suspect traffic (Ross, Spring 2018).

The use of machine learning algorithms to identify and stop different cyber security risks has recently generated a lot of discussion. Numerous machine learning methods have been the subject of much research on the detection of injection attacks. There are vulnerabilities detection using machine learning does not yet have a single ideal method or technique that can be used to solve every injection detection challenge. Before choosing a specific strategy, an issue must be examined against a variety of algorithms that fall within the classification or regression procedures, and the results must be compared for maximum accuracy (Mishra, Spring 5-23-2019).

Bug Detection Using Artificial Intelligence and Cyber Security

Random forest, decision trees, Logistic Regression, Support Vector Machine, Gradient Boosting, and neural networks are often used methods in this research. The fact that these methods can identify attacks is one of their main advantages. However, depending on the chosen method, there may be a risk of longer processing times with these techniques (Ross, Spring 2018).

In this research begins with an introduction to various cyber security vulnerabilities, different websites created using various programming languages, and the requirement and reason for creating a better injection detection system. After that, learn about SQL injection attacks and how related work has been done in the injection detection field so far. There is enough literature available from all the important implementations and research work that has been done thus far to understand the issue and make improvements. An introduction to machine learning methods, which is the general strategy utilizing to fix this problem, it is given in the section of implementation code and artefact explanation. It is a web application for detecting vulnerabilities, but since everyone is not aware of every vulnerability, a chatbot developed to provide information about work of vulnerabilities, their risks, and how to prevent it.

1.1) Research Questions

Does Machine Learning have the potential to improve to vulnerabilities detection technique?

Does ML serve as a bridge between research and reality?

How might technologies help to make work easy and time saving?

1.2) Aim and Objectives

Aim: Detect cyber security bugs or vulnerabilities in a web application using cyber security and AI concept.

Objectives:

Developing one web application using Machine learning algorithm which detect vulnerabilities in given URL and it is the main goal of this research. Machine learning always work with large amount of dataset for training and testing. So for that need to data collection, data cleaning, select different ML algorithms then choose best ML algorithm, use different

Bug Detection Using Artificial Intelligence and Cyber Security

libraries, flask, and create some models. For testing creates vulnerable dummy websites to check vulnerability detection web application.

2) Literature Review

A study of the literature has been done to look at the current research in the area of web application vulnerability analysis using machine learning, as well as its limits and potential future directions. The next section provides a summary of the identified possible uses for current techniques and methodologies (Asra Kalim, February 2020). Utilizing various technologies like Acunetix, Nessus, BurpSuite, and other cyber threats are identified. For instance, Acunetix online generates a vulnerability testing report and many execute dynamic and interactive application security testing scans. However, in order to save both time and memory, research is now working to identify the injection using machine learning methods. In terms of financial loss and the exposing of private information, SQL injection remains one of the most harmful security flaws. Traditional defense tactics frequently use IDS rules that are static, signature-based, and mainly successful against known attacks but ineffective against unknown or zero-day attacks. In that study, data collection for SQL injection research is typically done using one of two ways: either by recording actual web traffic entering a business or honeypot, or by creating realistic-looking simulated traffic (Ross, Spring 2018). Use plain-text and SQL injection queries to detect vulnerabilities in other research paper. Also utilized the tokenization method and found accuracy to compare algorithms. The regular expression object is generated using numerous SQL queries and SQL reserved words. Regular expressions are frequently used in pattern matching. Lexical analysis along with regular expression in python is used to implement tokenization. The final model that was put into action can distinguish between plain-text data and SQL injection (Mishra, Spring 5-23-2019).

In this research makes use of earlier research approaches and methodologies, including tokenization, comparison algorithms, and various machine learning algorithms. Additionally developed a malicious website using various programming languages. Based on comparisons between the accuracy of various algorithms, the best algorithm is found.

3) Method and Methodology

3.1) Resources

Platform	Jupyter Visual Studio 2022 Eclipse Visual Studio Code Github
Database Platform	MySQL workbench 8.0 CE Microsoft SQL Server Management Studio 18
Languages:	
Front-end	HTML CSS jQuery C#
Back-end	.net Java Python

Table 1: Resources

3.2) Dataset

In this research, a variety of vulnerabilities were discovered, but SQLi was one of the most significant. There are numerous datasets available for it from various sources. SQLi is a technique where the attacker creates or modifies SQL queries, meaning the attacker or hacker who wants to gain access to the organization's backend database and information creates malicious SQL queries that can quickly obtain the information that is essential to the organization or company. The dataset was gathered through the Kaggle website for the initial phase, which had as its primary goal (Sajid, 2021). Query and label are the two columns in the dataset. SQLi, authentic SQL, and plain-text are all combined in the query column (Farooq, 2021). The values in the label column are 1 and 0. Where 1 denotes a specific query that may

Bug Detection Using Artificial Intelligence and Cyber Security

access a database, such as a SQLi query, and 0 denotes a query that cannot access a database, such as a genuine SQL query or plain-text. There are 30919 rows in the dataset.

Data preprocessing is the second stage. Unnecessary spaces and escape sequences were removed, and all the queries were changed to lowercase in order to reduce data noise and increase precision. The SQL queries in the data are syntactically incorrect, which could cause problems.

3.3) Exploratory Data Analysis

Data can be read using the panda function `data = pd.read_csv('Modified SQL Dataset.csv')` and the number of rows can be obtained using `.format(data.shape[0])`. Only the data fields Query and Label from the dataset were used. The query field is of the object type and contains SQLi queries, SQL queries or plain-text. Labels field has two labels, 0 and 1, label 0 for SQL or plain-text query, and label 1 for SQLi query. SQLi query means malicious code and SQL query means non-malicious code. Next, determine how the data points are allocated among the labels. For example, determine how many data points belong to label 1 and how many to label 0. There is a category imbalance between data points since most data points fall under label 0, meaning that there is more plain text or normal SQL queries in the data. In this situation, the dataset needs to be balanced first because it is imbalanced.

```
Total data points that belongs to label 1 is 11382 and percent is 36.81
Total data points that belongs to label 0 is 19537 and percent is 63.19
```

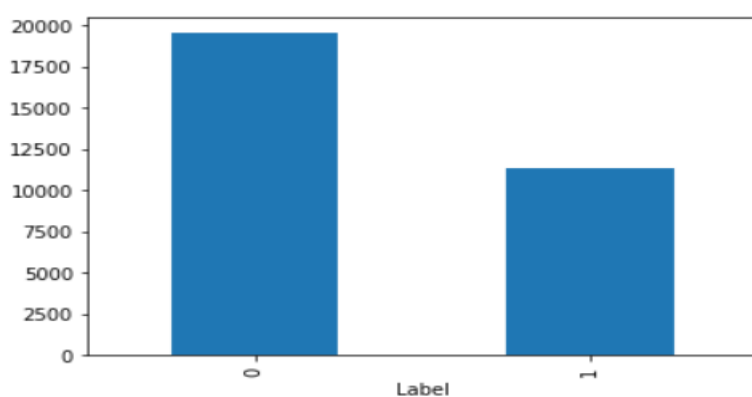


Figure 1: Imbalance Dataset

First read, described, and check dataset. Then give the label “non-malicious” and “malicious” to check imbalanced dataset. Here used undersampling method to balance selected dataset. In this method, removes the data from the majority class. It randomly selecting examples

Bug Detection Using Artificial Intelligence and Cyber Security

from the majority class to delete from the training dataset. In the transformed version of the training dataset, this results in a decrease in the number of cases in the majority class. Until the desired class distribution, such as an equal number of samples for each class, is reached, this process can be repeated. This method may be more appropriate for datasets with a class imbalance, while a useful model can still be fitted if there are enough examples in the minority class. A drawback of undersampling is that instances from the majority class that might be relevant, or even critical to fitting a strong decision boundary are eliminated. Because samples are eliminated at random, there is no method to identify or save “excellent” or more informative examples from the majority class. Before using this method need to import “RandomUnderSampler” after that defined dataset and undersample strategy. It is possible to define a class that supports the sampling strategy argument, which can be set to “majority” to automatically balance the majority class and minority class (Brownlee, 2021). The untransformed dataset can then be passed as inputs when running the fit_resample() method, which will fit the transform and apply it to the dataset in one step.

```
# example of random undersampling to balance the class distribution
from collections import Counter
from imblearn.under_sampling import RandomUnderSampler
# define dataset
# summarize class distribution
print('Before Sampling',Counter(yy))
# define undersample strategy
undersample = RandomUnderSampler(sampling_strategy='majority')
# fit and apply the transform
X_over, y_over = undersample.fit_resample(X_neww, yy)
# summarize class distribution
print('After Sampling',Counter(y_over))

Before Sampling Counter({0: 19537, 1: 11382})
After Sampling Counter({0: 11382, 1: 11382})
```

Figure 2: Random Undersampling

After applying undersampling method got the balanced data.

Bug Detection Using Artificial Intelligence and Cyber Security

```
new_data = pd.DataFrame({'Label':y_over.values})

LABELS = ['non-malicious','malicious']
total_classes = pd.value_counts(new_data['Label'], sort=True)

total_classes.plot(kind = 'bar', rot=0)
plt.title('SQL Query Class Distribution')
plt.xticks(range(2), LABELS)
plt.xlabel('Class')
plt.ylabel('Frequency')
Text(0, 0.5, 'Frequency')
```

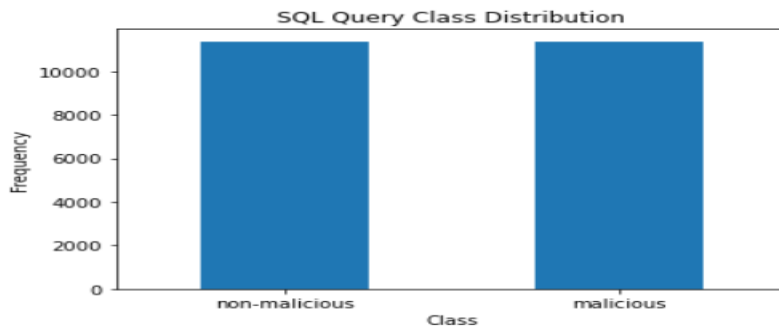


Figure 3: Balanced SQL Query Class Distribution

When there are many similar pieces of data, the duplicate function must be used to find it, and the drop function must be used to remove it from the dataset, which is done in the following step.

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

166

Figure 4: Duplicate Data in Dataset

There are 166 rows which are duplicates, so remove the duplicate rows using drop function.

```
#remove the duplicate rows
data.drop_duplicates(subset = ['Query','Label'],inplace = True)
```

Figure 5: Remove Duplicate Rows

Additionally, the “.isnull()” function was used to check whether a query was null, and the same query was verified with a different label. Unnecessary data was then eliminated using the same drop function. A plot of WordClouds was finished to show the most frequent terms across various labels. The queries for the various labels were first separated, and then WordCloud plots for label 1 and label 0 were performed separately. Null, chr, char union, and select count case are the words from label 1 that are used more frequently in SQL injection

Bug Detection Using Artificial Intelligence and Cyber Security

queries. For the purpose of feature engineering, these most common words are useful. As we can see, the standard SQL keywords that are used more frequently in SQL queries include select, top, order, count, average, join, etc. these are all the elements that differentiate between a regular SQL query and a SQL injected query, and they can all be seen in the queries for the chosen dataset that are composed of special characters, punctuation, etc. As a result, the text pre-processing procedure would not include the removal of punctuation, html tags, etc. Because it eliminates several special characters that are essential for feature engineering and differentiating between SQL and SQLi queries, word stemming and stop word removal are also not carried out. Text was only transformed to lowercase as part of the pre-processing using "lower()" function.

In the second stage, all the features from every query are extracted, and the best features are chosen. The stage after that deals with training the model (Farooq, 2021). Here used some are new features; that features store in a dataframe. Additionally, it included logistical operators, null values, real keywords, and more. The words select, top, order, fetch, join, average, count, sum, rows, and other real keywords are based on wordclouds. Then determine the quantity of actual keywords included in each query. Analysis of extracted features and it start to describe the length of each query. Now does different plotting of feature vs output plots and based on that check whether the feature helpful in predicting the output labels. Started with plot violin plot and distribution plots. Then checking distribution of query_length vs labels. If it is overlap then hard to distinguish between SQL and SQL injection queries. Checking the distribution of log transformed values to see they differentiate between labels.

Bug Detection Using Artificial Intelligence and Cyber Security

```
#checking the distribution of log transformed values to see if they are
data['log_query_len'] = data['query_len'].apply(lambda x : np.log(x))
sns.FacetGrid(data, hue="Label", height = 6) \
    .map(sns.distplot, "log_query_len") \
    .add_legend()

plt.show()
```

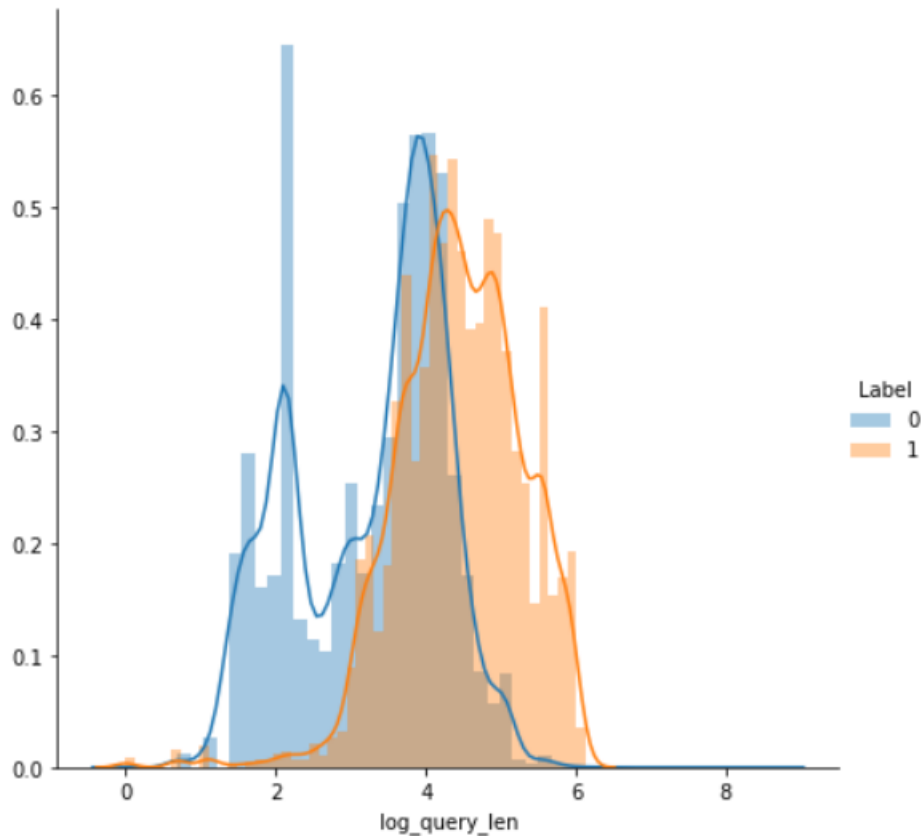


Figure 6: Log Query Length

The above distribution shows there is slight overlap between label classes so we avoid taking log transformed values for model building (Havannavar, 2021). After that find the number of words in query and based on that maximum and mean values are 216 and 12 respectively. Above mentioned all process done for every feature like number of double quotes in query, number of single quotes in query, number of punctuations in a query and all others.

The training and testing sets are randomly chosen from the dataset using a normal ratio of 70:30 (70% for training and 30% for testing) using the train test split function included in the Sklearn package:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.3,stratify = y)
```

Bug Detection Using Artificial Intelligence and Cyber Security

The model runs into a common issue while processing the data: although classic machine learning models expressly take in organised tabular numeric data, the datasets obtained are entirely unstructured textual data. In this case, a text parser is helpful. Text parsing is the process of dividing a given textual string into more manageable pieces in accordance with pre-established rules. The two most widely used techniques for parsing texts are regular expression separation and tokenization. The former processes the targeted text to parse the desired regular expressions, such as “[a-z]” and “[t]”. In the latter, each token – which could be a character, word, or phrase – division the text into smaller pieces called tokens. For this approach, tokenization works well because regular expressions cannot determine whether a query is malicious in the context of SQL injection attacks. Word tokens are used to split queries.

For example: Parsing “or 1=1 -- 1” into “or”, “1=1”, “--”, “1”.

CountVectorizer is used to enhance text data queries for bags of words. A collection of text documents is transformed into a vector of token counts using CountVectorizer. Additionally, it makes it possible to pre-process text data before creating the vector representation. It is a very flexible feature representation module for text because of this functionality (Kumar, 2021). Use the bigram and unigram NLTK methods. Pass the value of n=1 to the NLTK’s ngrams function to produce 1-grams. However, you must first tokenize the sentence and then provide these tokens to the ngrams function. Bigram is similar to a unigram but differs slightly in that the NLTK’s ngrams function generates 2-grams when n=2 (Fardeen, 2021).

Although word level TF-IDF vectors are one of the featured engineering techniques for NLP, they turn out to be the most successful for this SQLi attacks detecting model. Word Level TF-IDF vectors, also known as Term Frequency and Inverse Document Frequency, are an important index for term searching and figuring out the relevance of specific terms in a document. Term Frequency tracks how frequently a word appears in a specific document, as opposed to Document Frequency, which counts the number of times a word appears across all documents. With the help of this widely used method, text may be transformed into useful numerical representations that can be used to fit machine prediction algorithms. The following equation can be used to determine a word’s relevance:

Term-Frequency / Document-Frequency **or** Term-Frequency * Inverse-of-Document-Frequency

Bug Detection Using Artificial Intelligence and Cyber Security

The main advantage of TF-IDF is that it will presuppose that the documents are just a list of words with no real connection between it. Since SQL does not have tense or grammar restrictions like human languages do, this is a simple yet effective solution for the use situation. NLP read queries word by word, to put it simply. In order to check for harmful code, NLP split down malicious searches into single words. The number of times a word appears in a document relative to the total number of documents it appears in is used to calculate a word's TF-IDF score.

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

Term frequency: number of times term t appears in a document d

Idf is computed as

$$\text{Idf}(t) = \log[n / \text{df}(t)] + 1,$$

Where n is the total number of documents in the document set and $\text{df}(t)$ is the document frequency of t , the document frequency is the number of documents in the document set that contain the term t (developers, 2007 - 2022).

Bug Detection Using Artificial Intelligence and Cyber Security

```
#considering only unigrams
tfidf_bow = TfidfVectorizer(ngram_range = (1,1))
tfidf_train_bow = tfidf_bow.fit(x_train['Query'].values)

print("After Encoding text data")
x_train_tfidf_unigram = tfidf_train_bow.transform(x_train['Query'].values)
x_test_tfidf_unigram = tfidf_train_bow.transform(x_test['Query'].values)

print("the shape of train dataset unigram is {}".format(x_train_tfidf_unigram.shape))
print("the shape of test dataset unigram is {}".format(x_test_tfidf_unigram.shape))
```

After Encoding text data
the shape of train dataset unigram is (21523, 18629)
the shape of test dataset unigram is (9225, 18629)

```
#considering only bigrams
tfidf = TfidfVectorizer(ngram_range = (2,2))
tfidf_train_bigram = tfidf.fit(x_train['Query'].values)

print("After Encoding text data")
x_train_tfidf_bigram = tfidf_train_bigram.transform(x_train['Query'].values)
x_test_tfidf_bigram = tfidf_train_bigram.transform(x_test['Query'].values)

print("the shape of train dataset unigram is {}".format(x_train_tfidf_bigram.shape))
print("the shape of test dataset unigram is {}".format(x_test_tfidf_bigram.shape))
```

After Encoding text data
the shape of train dataset unigram is (21523, 51979)
the shape of test dataset unigram is (9225, 51979)

Figure 7: TF-IDF

After TF-IDF section, featurizing text data or SQL queries using average word2vec. Compute average word2vec for each train data query. And these same things do it using TF-IDF weighted word2vec. Then encoding numerical feature and at last concatenating encoded features with TF-IDF word2vec.

4) Machine Learning Algorithms

The proposed model used the following machine learning algorithms:

4.1) Logistic Regression

Using the supervised learning classification process known as logistic regression, the probability of a target variable is predicted. There are only two potential classes because the dependent variable is bidirectional.

The dependent variable is, to describe it simply, a binary variable, with data recorded as either 1 (which represents success/yes) or 0 (which represents failure/no). A logistic regression model predicts the value of $P(Y=1)$ as a function of X mathematically. It is one of the most fundamental machine learning algorithms and may be used to categories a variety of problems, such as the detection of spam, the prediction of diabetes, the detection of cancer, etc.

The use of logistic regression and linear regression differs significantly. In contrast to linear regression, which is used to solve classification issues, logistic regression handles regression issues. Logistic regression is an important machine learning technique because it can categorise new data using both continuous and discrete datasets. Using several data formats, logistic regression may be used to categorise observations and quickly find the variable that will be most useful for the classification (Anon., 2011 - 2021).

4.2) Random Forest

The supervised learning method includes the well-known machine learning algorithm Random Forest. It can be applied to ML classification and Regression issues. Its foundation is the idea of ensemble learning, which is the technique of combining various classifiers to solve a challenging problem and improve the performance of the model.

Random Forest is a classifier that, as the title indicates, “contains a number of decision trees on various subsets of the provided dataset and takes the average to enhance the predictive accuracy of that dataset.” Instead of depending on a single decision tree, the random forest uses predictions from all the trees to anticipate the outcome based on which predictions received the most points (Anon., 2011 - 2021).

4.3) Decision Tree

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. In a decision tree, the algorithm begins at the root node and uses that information to predict the class of the dataset that is provided. This algorithm follows the branch and jumps to the next node by comparing the root attribute's values to those of the record attribute (Anon., 2011 - 2021). The splitting, pruning, and tree selection processes are used to operate it. It allows for the construction of the decision tree using both numerical and categorical data (Pedamkar, 2022).

4.4) Extreme Gradient Boosting

To eliminate training errors, an ensemble learning strategy known as “boosting” transforms a collection of weak learners into strong learners. A model is fitted to a random sample of data before being successively trained in boosting. In other words, each way to approach to overcome the flaws of the one that came before it. The weak rules from each classifier are joined on each iteration to produce a single strong prediction rule.

Boosting is a powerful algorithm that converts a weak learner into a strong learner. They make use of the idea of the weak learner and strong learner conversation by using the weighted average values and higher votes values for prediction. These algorithms handle data using margin maximizing categorization and decision making.

5) Web Application

A web application was created to detect injections from websites based on the research, and machine learning applies for the injection detection process. That web application created in the python programming language. Created two dummy websites as well, one of which done using the java programming language and the other using the .net core framework. Additionally, the frontend makes use of HTML, C#, CSS, and jQuery. First started with an explanation of two dummy websites and then started the main vulnerability detection website.

5.1) Web Application using .NET Core

.Net is software framework. It is free, open-source developer platform for developing different kinds of web applications. In this case the website uses .net core for development. The most recent edition of Microsoft's general-purpose programming platform, .NET Framework, is called .Net Core. It is free and open-source. Working with windows, Mac OS X, and Linux, it is a cross-platform framework. Many other types of apps, including those for mobile, desktop, web, cloud, IoT, machine learning, microservices, games, and more, may be made using the .NET Core Framework (Anon., 2022). Modern libraries and languages are compatible with .NET Core, which quicker, scalable, and made for them (Singh, 2019).

A website built in .NET Core stores the data of the school. It has three parts faculty, student, and subjects. In each of the three, data can be added, removed, or changed. A search bar on the website that allows users to quickly look up any information which has stored in web application.

Code Explanation

The type of the model provided to a view is defined by the @model directive. The following model definition is seen in the Views/Faculty/Create.cshtml view of an ASP.NET Core MVC application built with individual user accounts (Edward, 2019):

```
@model Faculty
```

Then load _Layout.cshtml into a single variable. The Views/Shared folder automatically contains a _Layout.cshtml file when you create an ASP.NET MVC Web application in Visual

Bug Detection Using Artificial Intelligence and Cyber Security

Studio 2015 or later versions. The layout that can be applied to the application's views is represented by this file. The fundamental framework of a view can be found in this `_Layout.cshtml` file (Adil, 2021).

```
@{
    Layout = "_Layout";
    var title = "CREATE Faculty";
    ViewData["Title"] = title;
}
```

Figure 8: Load `_Layout.cshtml` in Variable

Then labels and input fields are arranged to enter the relevant data and finally the “submit” button.

The next file is “`index.cshtml`”, which is default page shown when start website. Here first “create” button to insert any new data entry and beside that given one search box to find any data. After that given list of stored data information which is faculty id, name, skills, total number of students and update and delete button in every rows. The “index” page's front view is shown below, and other fields are similar to it.

Faculty | Student | Subjects

READ Faculty

Create Search Faculty: Search

Id	Name	Skills	Total Students	Salary	Added On	Update	Delete
2	Raj	ASP.NET Core	20	\$6,000.00	Thursday, 04 August 2022	Update	<input type="button" value="Delete"/>
3	Parthiv	ASP.NET MVC	15	\$8,000.00	Thursday, 11 August 2022	Update	<input type="button" value="Delete"/>
4	Devid	ASP.NET Web Forms	25	\$7,000.00	Monday, 15 August 2022	Update	<input type="button" value="Delete"/>
5	rahul	ASP.NET Core	20	\$8,000.00	Monday, 15 August 2022	Update	<input type="button" value="Delete"/>
6	Aisha	ASP.NET Web Forms	15	\$6,000.00	Saturday, 27 August 2022	Update	<input type="button" value="Delete"/>
7	Fadi	ASP.NET Core	25	\$10,000.00	Saturday, 27 August 2022	Update	<input type="button" value="Delete"/>
8	Ali	ASP.NET MVC	45	\$20,000.00	Saturday, 27 August 2022	Update	<input type="button" value="Delete"/>
9	Vaghmi	ASP.NET Web Forms	18	\$9,000.00	Saturday, 27 August 2022	Update	<input type="button" value="Delete"/>
10	Ankita	ASP.NET MVC	29	\$6,000.00	Saturday, 27 August 2022	Update	<input type="button" value="Delete"/>
11	Hetvi	ASP.NET Core	9	\$4,000.00	Saturday, 27 August 2022	Update	<input type="button" value="Delete"/>
12	Shehnaz	ASP.NET Web Forms	28	\$5,555.00	Saturday, 27 August 2022	Update	<input type="button" value="Delete"/>
13	Atif	ASP.NET MVC	14	\$4,200.00	Saturday, 27 August 2022	Update	<input type="button" value="Delete"/>
14	Rubby	ASP.NET Web Forms	16	\$9,510.00	Saturday, 27 August 2022	Update	<input type="button" value="Delete"/>

Figure 9: Faculty Index Page

Let's discuss the “`update.cshtml`” file now. The view structure of the update data is set in that file. The labels and input boxes all are set to update the data of the selected row. Then lastly set up the “submit” button to update the dataset with the updated information.

Bug Detection Using Artificial Intelligence and Cyber Security

Faculty | Student | Subjects

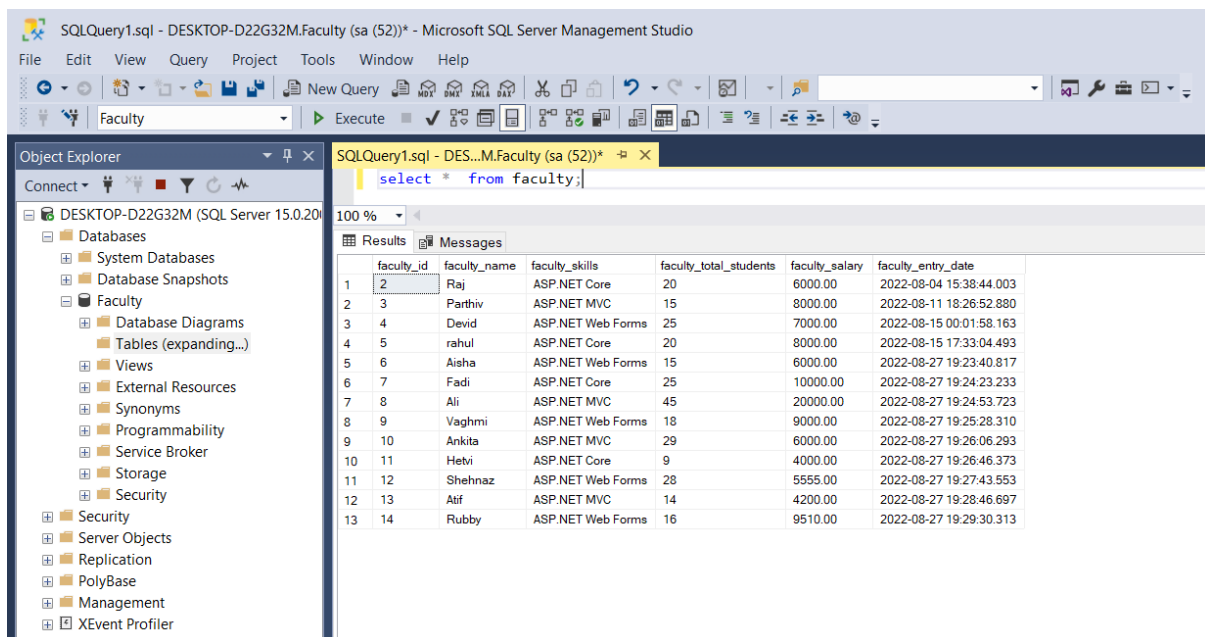
UPDATE Faculty

faculty_id	2
faculty_name	Raj
faculty_skills	ASP.NET Core
faculty_total_students	20
faculty_salary	6000.00
faculty_entry_date	04-08-2022
<input type="button" value="Submit"/>	

Figure 10: Update Page

In other fields, this kind of coding has been completed for subjects and students, and it is kept in the views folder.

The data of each website is stored in the database. “Microsoft SQL Server Management Studio” platform is used to store the database of .NET web application.



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'DESKTOP-D22G32M (SQL Server 15.0.20...)', including 'Databases', 'System Databases', 'Database Snapshots', 'Faculty', 'Database Diagrams', 'Views', 'External Resources', 'Synonyms', 'Programmability', 'Service Broker', 'Storage', 'Security', 'Server Objects', 'Replication', 'PolyBase', 'Management', and 'XEvent Profiler'. The main window shows a query window with the SQL query `select * from faculty;` and a results grid displaying the following data:

faculty_id	faculty_name	faculty_skills	faculty_total_students	faculty_salary	faculty_entry_date	
1	2	Raj	ASP.NET Core	20	6000.00	2022-08-04 15:38:44.003
2	3	Parthiv	ASP.NET MVC	15	8000.00	2022-08-11 18:26:52.880
3	4	Devid	ASP.NET Web Forms	25	7000.00	2022-08-15 00:01:58.163
4	5	raahul	ASP.NET Core	20	8000.00	2022-08-15 17:33:04.493
5	6	Aisha	ASP.NET Web Forms	15	6000.00	2022-08-27 19:23:40.817
6	7	Fadi	ASP.NET Core	25	10000.00	2022-08-27 19:24:23.233
7	8	Ali	ASP.NET MVC	45	20000.00	2022-08-27 19:24:53.723
8	9	Vaghmi	ASP.NET Web Forms	18	9000.00	2022-08-27 19:25:28.310
9	10	Ankita	ASP.NET MVC	29	6000.00	2022-08-27 19:26:06.293
10	11	Hetvi	ASP.NET Core	9	4000.00	2022-08-27 19:26:46.373
11	12	Shehnaz	ASP.NET Web Forms	28	5555.00	2022-08-27 19:27:43.553
12	13	Atif	ASP.NET MVC	14	4200.00	2022-08-27 19:28:46.697
13	14	Rubby	ASP.NET Web Forms	16	9510.00	2022-08-27 19:29:30.313

Figure 11: Database of .NET Web Application

In the MVC framework, the model represents domain-specific data and business logic. It keeps the application’s data up to date. Model objects use the persistence store as a database to retrieve and store model state (Anon., 2022). In the model part stored all “.cs” file. Here,

Bug Detection Using Artificial Intelligence and Cyber Security

the data is obtained from the users and sets the data. A similar coding task has performed for students and subjects.

```
26 references
public class Faculty
{
    13 references
    public int faculty_id { get; set; }

    [Required]
    18 references
    public string faculty_name { get; set; }

    [Required]
    [SkillsValidate(Allowed = new string[] { "ASP.NET Core", "ASP.NET MVC", "ASP.NET Web Forms" }, ErrorMessage = "You skills are
invalid")]
    18 references
    public string faculty_skills { get; set; }

    [Range(5, 50)]
    18 references
    public int faculty_total_students { get; set; }

    [Required]
    18 references
    public decimal faculty_salary { get; set; }

    8 references
    public DateTime faculty_entry_date { get; set; }
}
```

Figure 12: Model File Coding

Now talk about the final subject, “Controllers,” which establishes the control strategy. First start with “using” method. Two main applications exist for the keyword “using” : a scope that will be the final point for an object’s elimination is defined by the using statement. “Using” a namespace’s alias or importing types from other namespaces are both possible with the using directive.

Using a namespace’s declared types without using the namespace’s fully qualified name allows you to use those kinds. The using directive imports every type from a single namespace in its simplest form, as demonstrated in the following (Microsoft, 2022):

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using CRUDADO.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
```

Figure 13: Using Method

Created home controller class and inside the class created one more class for configuration and one class for action. A code is written in the action to provide configuration with a SQL connection. After that, SQL query “Select * from faculty” stored in one variable and that data type is string. Then used a while loop to read the entire set of data using a SqlDataReader. Next close the SQL connection.

Bug Detection Using Artificial Intelligence and Cyber Security

```
public ActionResult Index()
{
    List<Faculty> facultyList = new List<Faculty>();

    string connectionString = Configuration["ConnectionStrings:DefaultConnection"];
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        //SqlDataReader
        connection.Open();

        string sql = "Select * From Faculty";
        SqlCommand command = new SqlCommand(sql, connection);

        using (SqlDataReader dataReader = command.ExecuteReader())
        {
            while (dataReader.Read())
            {
                Faculty faculty = new Faculty();
                faculty.faculty_id = Convert.ToInt32(dataReader["faculty_id"]);
                faculty.faculty_name = Convert.ToString(dataReader["faculty_name"]);
                faculty.faculty_skills = Convert.ToString(dataReader["faculty_skills"]);
                faculty.faculty_total_students = Convert.ToInt32(dataReader["faculty_total_students"]);
                faculty.faculty_salary = Convert.ToDecimal(dataReader["faculty_salary"]);
                faculty.faculty_entry_date = Convert.ToDateTime(dataReader["faculty_entry_date"]);

                facultyList.Add(faculty);
            }
        }
    }
}
```

Figure 14: SQL Configuration

This website uses three separate functions to “create”, “update”, and “delete” data and has appropriate SQL configuration for each.

```
if (ModelState.IsValid)
{
    string connectionString = Configuration["ConnectionStrings:DefaultConnection"];
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        string sql = $"Insert Into Faculty (faculty_name, faculty_skills, faculty_total_students, faculty_salary) Values ('{faculty.faculty_name}', '{faculty.faculty_skills}', '{faculty.faculty_total_students}', '{faculty.faculty_salary}')";

        using (SqlCommand command = new SqlCommand(sql, connection))
        {
            command.CommandType = CommandType.Text;

            connection.Open();
            command.ExecuteNonQuery();
            connection.Close();
        }
    }
    return RedirectToAction("Index");
}
else
    return View();
```

Figure 15: SQL Configuration for Update

All this process is written for only one field faculty and same process and coding is done for student and subject part.

5.2) Web Application using Java

A java web application is made up of both static and dynamic resources, such as Java classes and jars, Servlets, and JavaServer Pages (HTML pages and pictures). A WAR (Web ARchive) file can be used to deploy a java web application (Vogel, 2022). This is second dummy website

Bug Detection Using Artificial Intelligence and Cyber Security

which created in java language using eclipse platform and for the database used “MYSQL Workbench 8.0 CE.” For that need configuration of Apache tomcat server. This is one login website and, in this web, application has three different pages; login page, registration page, and home page. Login is default page; in that page given registration page link and after login user can see home page.

Code Explanation

This web application coding start with three “.jsp” files; index.jsp, login.jsp, and registration.jsp. Login is the default page so let’s start the code with the login.jsp file; this file contains the coding of the front view of the login page. Using the <div> and <section> method arranged the username, password field and login button on the right-hand side. And the opposite side organized create login account with one image. If user enter correct username and password then it redirects on home page otherwise show error message and that related coding written at last between <script> tag </script>.

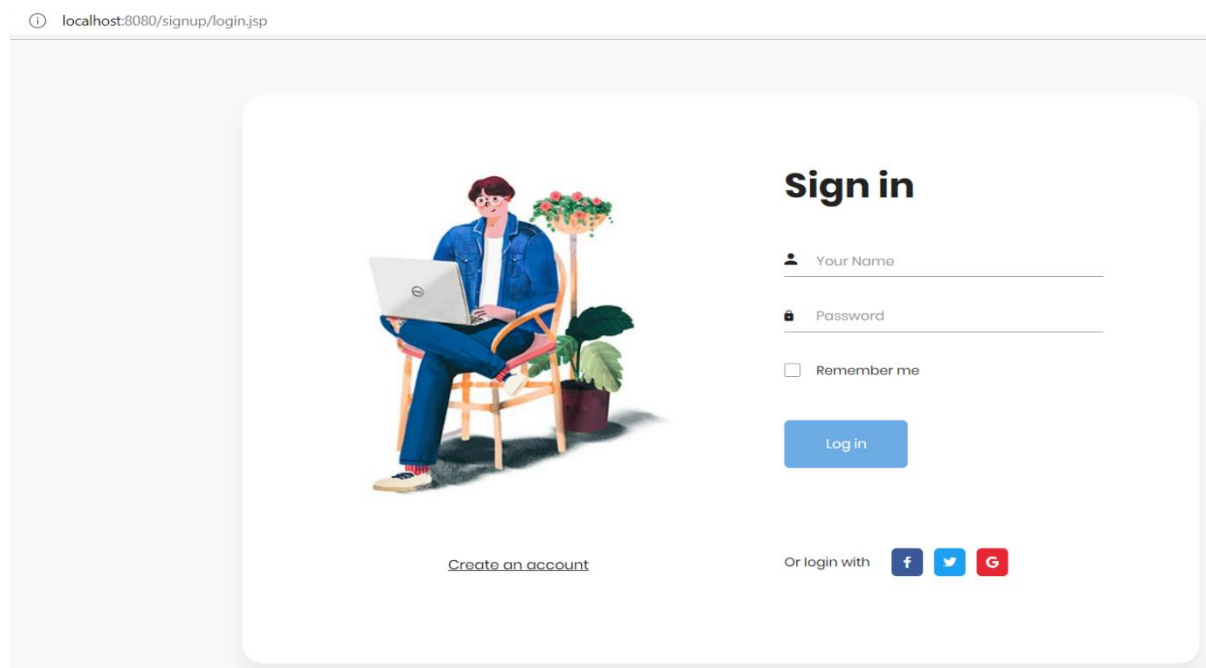


Figure 16: Login Page

Utilized the same formatting techniques to create the registration page and set up five text fields for users to enter their information. When a user clicks the submit button, the data is stored, but if a necessary field is left empty or contains incorrect information, an error notice is displayed.

Bug Detection Using Artificial Intelligence and Cyber Security

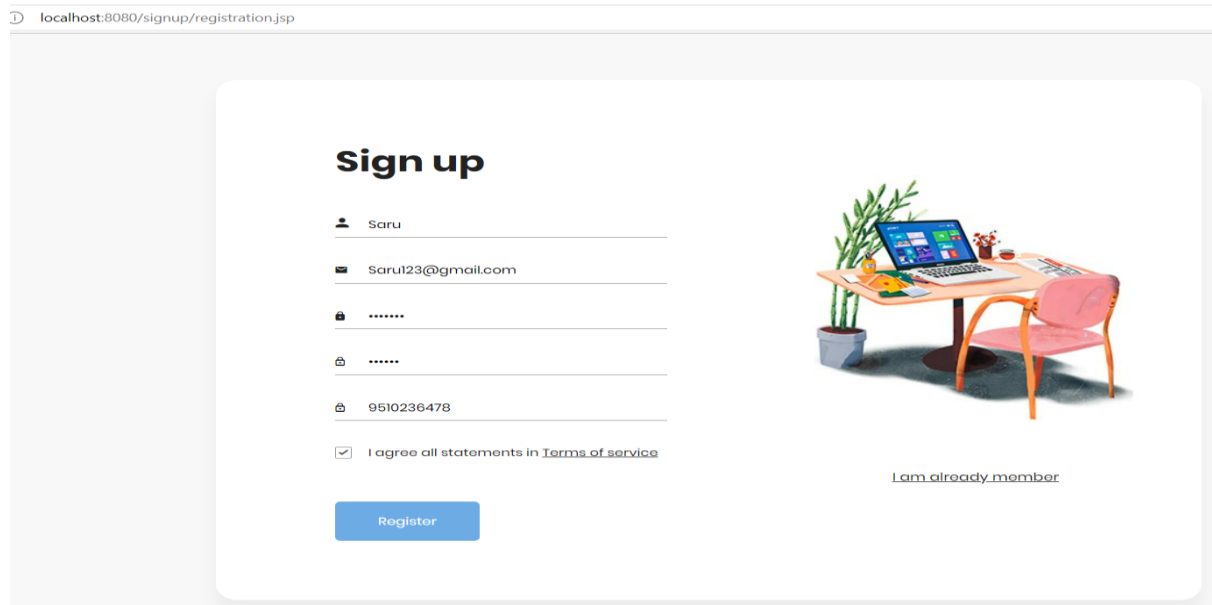


Figure 17: Registration Page

Using the same method added some static content on home page; when user login then they can see that content.



Figure 18: Home Page

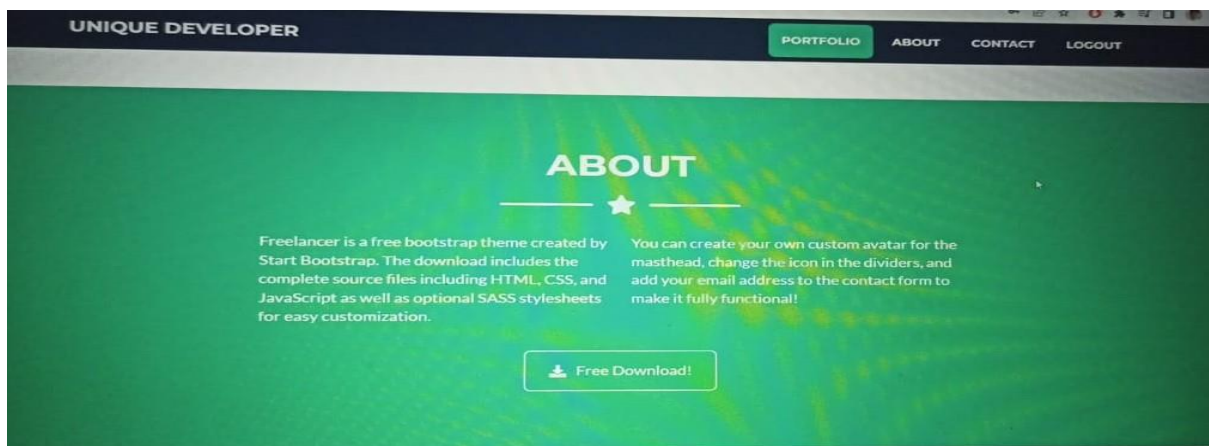


Figure 19: Result of Home Page

Bug Detection Using Artificial Intelligence and Cyber Security

The coding of SQL connectivity is given inside the “.java” files. First import package in login.java file and created public class login. Take two variable and get the username and password value. Uses try a catch method. Inside try method give connection with mysql and set the select query. Then used if...else... condition for login. If user name and password is right then it redirect on “index.jsap” page otherwise it shows the failed status. In registrationservlet.java file, it has four parameters so it get values of all four parameters and using insert query that four parameters data store in database and it is used when user login on web application.

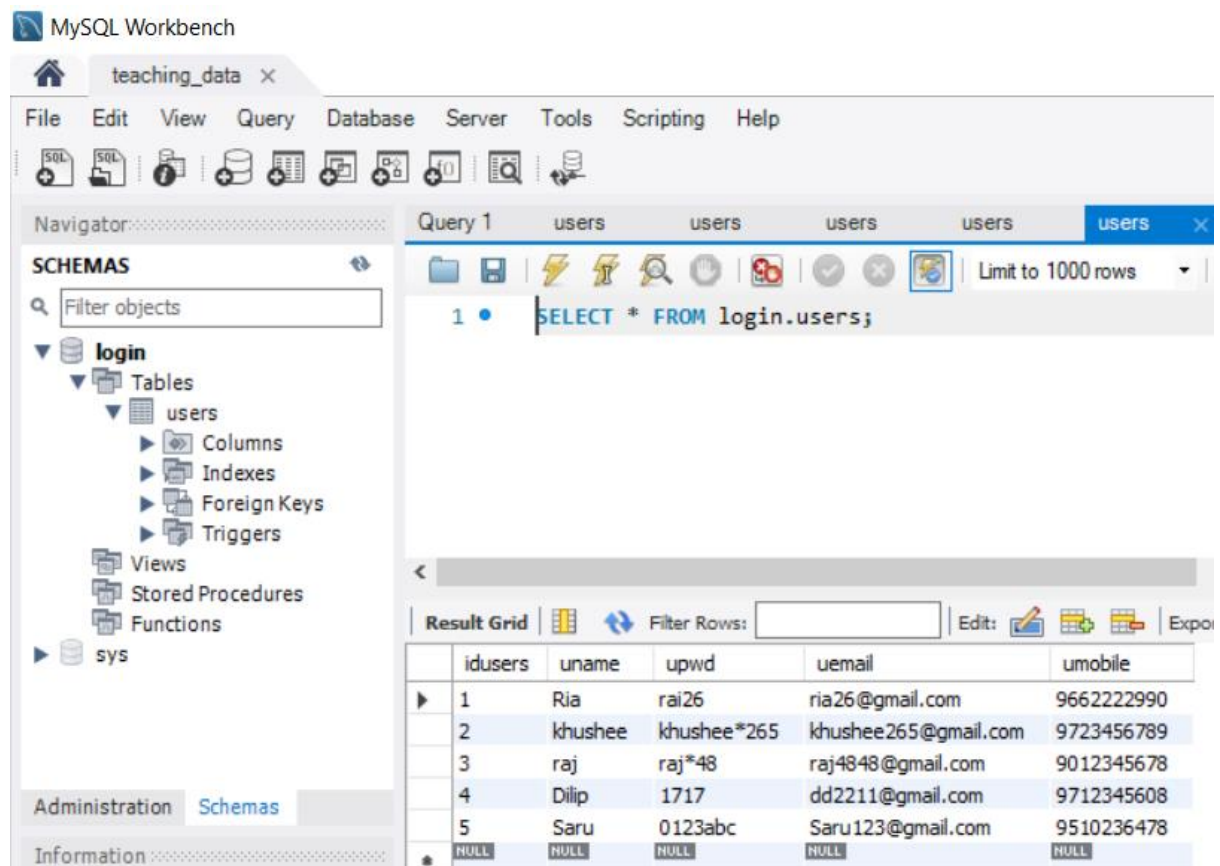


Figure 20: Database of Java Web Application

5.3) Main Web Application using Python

The main Vulnerability Detection website is built in Python language. In this part, the front end of the website will be discussed and the backend will be described briefly in the machine learning part. The whole front-end coding for the website is included in the “index.html” file. Additionally, declared formatting for the content in the same file, such as background colour, font-family, font size, height, etc.


```
13 h1 {
14
15     float: center;
16     display: block;
17     background-color: #333;
18     color: #f2f2f2;
19     text-align: center;
20     vertical-align: middle;
21     line-height: 50px;
22     height: 50px;
23 }
24
25 /* Style the content */
26
27
28 .review {
29     background-color: #ddd;
30     padding: 10px;
31     height: 50px; /* Should be removed. Only for demonstration */
32     text-align: center;
33     margin: auto
34 }
35
36 textarea {
37
38     width: 80%;
39     padding: 12px 20px;
40     margin: 8px 0;
41     display: inline-block;
42     border: 1px solid #ccc;
43     border-radius: 4px;
44     box-sizing: border-box;
45 }
46
47 input[type=submit] {
48     width: 20%;
49     background-color: #4CAF50;
```

Figure 21: Style the Content

Here, two menus are defined: the home menu and the SQLi-Detection-with-Machine-Learning menu. Home menu is the default page, and it has been used to find all five defined vulnerabilities on the same page. Another page, SQLi-Detection-with-Machine-Learning, is used to find only the SQLi vulnerability; on that page uses machine learning algorithm to find it and that part is briefly describe in machine learning part.

```
<div>
  <ul style="margin-top: 10px;">
    <li><a href="/">Home</a></li>
    <li><a href="/sqliML">SQLi-Detection-with-MachineLearning</a></li>
  </ul>
</div>

<h1>VULNERABILITY DETECTION</h1>

<div class = "review" style = "">
  <form action="/predict" method="POST">
<textarea name="query" placeholder="Enter Url here..." rows="1" cols="6"></textarea>
  <input type="submit" value="Check Vulnerability">
</form>
</div>
```

Figure 22: Default Page Coding of Vulnerability Detection

Bug Detection Using Artificial Intelligence and Cyber Security

There are two menus, one text area, and a submit button on the default page. Here used post method. The text area and submit button on the second page are also the same as on the default page but only the backend coding is different.

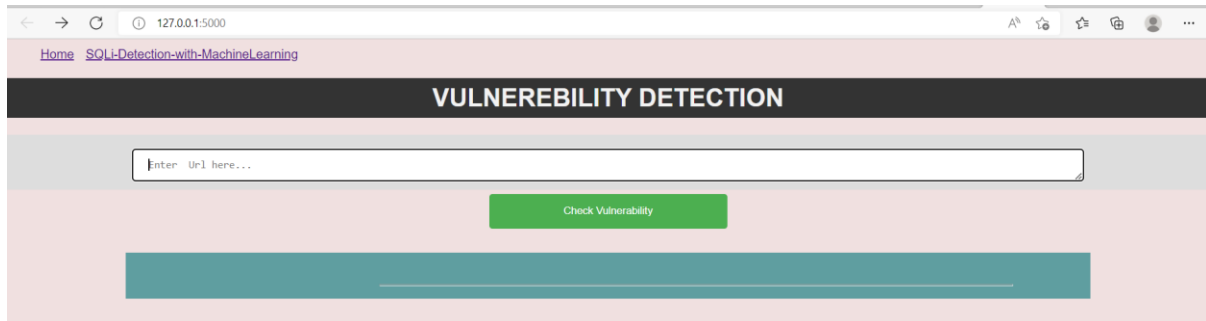


Figure 23: Front-end of Vulnerability Detection Website

The URL of the website from which the vulnerability is to be found is entered in the text area. After entering URL in text area if any vulnerabilities detected then it shows on the same page below of vulnerability detection button, it is only for default page but for other menu that detected vulnerability statement show on new tab.

6) Experiment and Results

This section discusses the coding process, including how it began, the challenges it faced, and the major focus of the discussion – a comparison of various algorithms. And last machine learning algorithm that has been employed in this research or coding. Below given one flow chart of vulnerability detection web application means how it work.

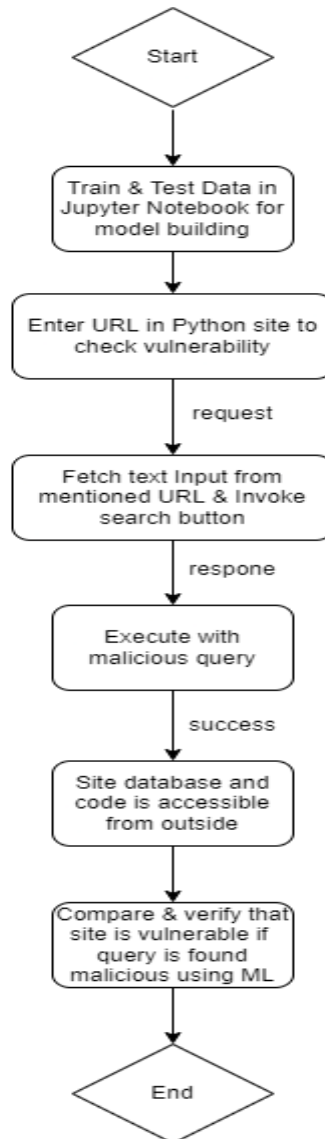


Figure 24: Web Application Flow Chart

Above given web application flow chart. First collect dataset; that dataset train and test in jupyter notebook for model building, based on that finalize ML algorithm and using that algorithm created vulnerability detection website. In third section, enter URL in python site to check vulnerabilities. That URL get as request and fetch() functions fetch that structure and

Bug Detection Using Artificial Intelligence and Cyber Security

fetch text input from mentioned URL and invoke search button. That give as a response and execute with malicious query. There site database and code are accessible form outside. It compares and verify that site is vulnerable or not if query is found malicious using machine learning, then it shows message and end the process.

6.1) Compare ML Algorithms

In starting of this research document already described various machine learning algorithms which has used to develop vulnerability detection web application. Before choosing one, all algorithms must be compared, and to do this, accuracy, f1-score, precision, and recall must be determined.

NLTK should be imported first because it is an important library for python tasks like tokenization, stemming, tagging, and classification (Adamjee, 2020). Using NLTK download stopwords. Stopwords are common terms like “the”, “and”, “I”, etc. that appear frequently in text but don’t provide context for the document’s main idea. To clean up the data and find terms that are more uncommon and perhaps more relevant to the research, we can exclude these stop words from the text in a specific sample. Stopwords like “the”, “is”, and “are” may appear in text. Stopwords in the text that must be processed can be filtered out. Although the NLTK module includes a list of stopwords, there is no single list of stop words used in all NLP research (Anon., 2015 - 2022).

After that, import the required libraries, and then use the read_csv function to read the data from the dataset which has two columns query and label. Utilised countvectorizer technique with stopwords, min_df, and max_df. Min_df = 2 means ignore terms that appear in less than 2 documents and max_df = 0.8 means ignore terms that appear in more than 8% of the documents (Markham, 2021). Now use “.fit_tranform()” for X values. By using a fit_transform() on the training data, the scaling parameters of the training data are determined. In this instance, the model will determine the mean and variance of the features in the training set. The fit approach is used to determine the mean and variation of each characteristic reported in the data. All features are transformed using the associated means and variances by the transform method (JavaTpoint, 2011-2021).

Bug Detection Using Artificial Intelligence and Cyber Security

```
vectorizer = CountVectorizer(min_df = 2, max_df = 0.8, stop_words = stopwords.words('english'))
X = vectorizer.fit_transform(X.values.astype('U')).toarray()

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=1)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)

(3360, 4717)
(3360,)
(840, 4717)
(840,)
```

Figure 25: CountVectorizer Function

The first starts with logistic regression and for that logistic regression is imported from the Sklearn library. Then use “.fit()” for x_train and y_train. The mean and standard deviation of the specific feature will be calculated using the .fit() method, and these values will be used for analysis in the future (JavaTpoint, 2011-2021). Using the score method get the accuracy of model and using classification_report measured the quality of predictions from a classification algorithm means measuring precision, recall, f1-score, and support. Metrics like recall and precision are frequently used to determine how well machine learning models or overall, AI solutions are working. It aids in understanding how accurately models predict the future. The percentage of accurate positive predictions is known as precision. The fraction of actual positive labels that the model properly detected is measured by recall (Ganesan, 2021). Precision and recall are combined into the F1-score regarding a particular positive class. The f1-score can be seen of as a weighted average of precision and recall, with its best value being 1 and its lowest being 0 (Ng., 2022).

$$F1 = 2 * (precision * recall) / (precision + recall)$$

There is a 0.92 accuracy score for logistic regression. Accuracy score of random forest algorithm is same as logistic regression. The Extreme Gradient Boosting technique, Decision Tree, and Random Forests all use the same procedure. First importing necessary algorithm (i.e. random forest classifier) from assemble module, creating a RF classifier, fit function is used to train the model using the training sets as parameters, performing predictions on the test dataset, metrics are used to find accuracy or error, and using metrics module for accuracy calculation. Accuracy score of random forest algorithm is same as logistic regression. Decision tree has an accuracy score of 0.87 and Extreme Gradient Boosting is 0.89.

The first step is to import the relevant libraries, such as sklearn, scipy, pickle, joblib, pandas, numpy, matplotlib, and seaborn. After that load the countvectorizer of unigram_bow which has done previous and using pickle function load it. If need persistency in data, pickle is helpful

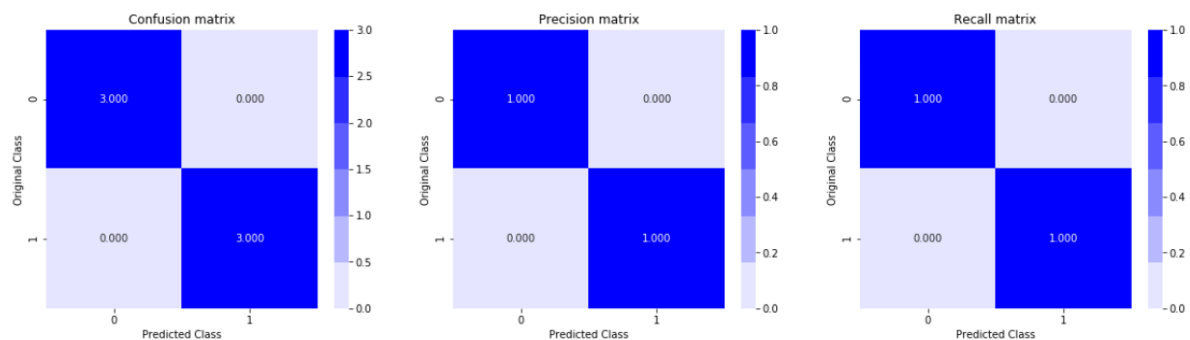
Bug Detection Using Artificial Intelligence and Cyber Security

in apps. Using pickle is the best option if need to save the program's state data on a disc for later usage. Alternatively, use pickle to store the data in our database or communicate it through a TCP or socket connection. Then written code to print the classification metrics in that first representing A in heatmap format and then representing B in heatmap format. But this coding section represent at last. But before that, output the prediction whether it belongs to class 1 or class 0 for a given SQL Query.

Here finding most occurring words and that are null, chr, char, select. These words are used to create new features. For example, count of null values in text, count of chr values in text, count of char values in text etc. these features will help the model to differentiate between SQL and SQLi queries. The most occurring words are select, fetch, count, order, first, etc. which are normal SQL keywords. There are creating query by appending keywords, characters numbers and special symbols as well by using regular expressions.

The next step was to produce the classification metrics for the given list of query and labels after receiving the predicted results from the loaded model. Defined a single function, divided the query and labels into separate lists, and then called function_1 on the query lists above to retrieve the predicted labels. In function_1, that will create entire pipeline to predict the output i.e., for any given query will predict the output of it. In function_2, that will create entire pipeline to output the performance metric. Created list of tuples of queries and labels and calling the function_2 method to print the classification metrics. Below given precision matrix, confusion matrix, and recall matrix.

```
the 0 query passed belongs to class 1 i.e the query is SQL injection query
the 1 query passed belongs to class 1 i.e the query is SQL injection query
the 2 query passed belongs to class 0 i.e the query is not a SQL injection query
the 3 query passed belongs to class 1 i.e the query is SQL injection query
the 4 query passed belongs to class 0 i.e the query is not a SQL injection query
the 5 query passed belongs to class 0 i.e the query is not a SQL injection query
```



f1-score on the data is 1.0

Figure 26: Output of matrix

6.2) Results

Research shows that boosting approaches in machine learning can help reduce bias in models. The results of experiments show that Gradient Boosting approach does perform better in terms of prediction accuracy (Mishra, Spring 5-23-2019).

Parameters	Decision Tree	Extreme Gradient Boosting
Accuracy	87.02	89.64

Table 2: Decision Tree vs Extreme Gradient Boosting

Gradient Boosted decision trees are represented in XGBoost. Many Kaggle competitions have a pronounced dominance of XGBoost models. This technique generates decision trees in a sequential manner. In XGBoost, weights are significant. All independent variables are given weights, which are subsequently used to feed information into the decision tree that predicts outcomes. The second decision tree is then fed these variables after increasing the weight of variables that the tree incorrectly predicted. To create a robust and accurate model, these independent classifiers and predictors are then combined. Regression, classification, ranking, and user-defined prediction issues can all be solved by it (XGBoost, 2022).

According to research, machine learning boosting techniques can aid in the reduction of model bias. The outcomes of the experiments demonstrate that the Extreme Gradient Boosting technique does perform better in terms of prediction accuracy.

As ensemble learning techniques are considered to perform better than basic classifiers, a gradient boosting classifier model has been constructed for this issue because it looked to be the best fit given the importance of being unable to detect even a SQL Injection. Because SQL injection involves a huge quantity of query datasets and when working with machine learning needs large datasets for training and testing, just the machine learning section is used in this study to discover SQL vulnerabilities. The website has two menus, the first default page utilizes regular code to detect vulnerabilities, and the other page uses machine learning part to detect SQL injection. For other (XSS, header vulnerabilities) injection detection, there is no usage of machine learning part.

6.3) Final Backend Coding

In starting of this coding did without machine learning detect vulnerabilities, so first will starting explanation with normal coding (without use of machine learning algorithm) part and after that explain with machine learning part. All required python and machine learning libraries are imported at the beginning of coding like bs4, BeautifulSoup, flask, requests, pandas, numpy, sklearn, joblib, pickle, scipy, hstack, os, sys, mechanize, and all. Flask is a well-known, server-side, lightweight, modular, and popular python framework that is quite like the Django framework. There can build the backend systems for own web apps using the library of modules and functions provided by frameworks like flask. Flask features a few benefits, including support for secure cookies, restful request dispatching, jinja templates, and integrated unit testing (Khanna, 2020). Creates an instance of Flask. The name of the current python module is `__name__`. `__name__` is a convenient way to provide the app with the location information it needs to set up some paths. Routes in flask are translated in to python methods. Already created one route, the `'/'` route. Here is used the route `/sqliML`, route `/clickjacking`, and route `/checkout` so it needs to bind it to that all function. The output of the function all that is shown in browser based on that use or need. After that initialize an HTTP session and set the browser (TUtorial, 2021).

```
from sklearn import linear_model
import joblib
from bs4 import BeautifulSoup
import re
import pickle
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.preprocessing import StandardScaler
from scipy.sparse import hstack
import os
import mechanize

app = Flask(__name__)

@app.route("/sqliML")
def SQLiML():
    return flask.render_template('ml_sqli.html' )

@app.route("/")
def home():
    return flask.render_template('index.html' )

@app.route("/clickjacking")
def clickjackingg():
    return flask.render_template('checkclickjacking.html')

@app.route("/checkout")
def checkoutform():
    return flask.render_template('checkoutform.html')

# initialize an HTTP session & set the browser
s = requests.Session()
s.headers["User-Agent"] = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.106 Safari/537.3"
```

Ln 64, Col 22 Spaces: 4 UTF-8 CRLF Python 3.9.7 (test: virtualenv)

Figure 27: Flask and HTTP Session

Bug Detection Using Artificial Intelligence and Cyber Security

Defined one `fetch_forms()` function and given URL parameter in it. This function created to fetch all forms from the HTML content from given URL. Then defined another `fetchformdetails()` function and pass form parameter in function, using this function extracts all possible useful information about an HTML form. In that first get the form action, get the form method like post, get, etc., then get all the input details such as type, name, value and all; and at last, put everything to the resulting dictionary.

```
def fetch_forms(url):
    """Given a `url`, it returns all forms from the HTML content"""
    soup = bs(s.get(url).content, "html.parser")
    return soup.find_all("form")

def fetchformdetails(form):
    """
    This function extracts all possible useful information about an HTML `form`
    """
    details = {}
    # get the form action (target url)
    try:
        action = form.attrs.get("action").lower()
    except:
        action = None
    # get the form method (POST, GET, etc.)
    method = form.attrs.get("method", "get").lower()
    # get all the input details such as type and name
    inputs = []
    for input_tag in form.find_all("input"):
        input_type = input_tag.attrs.get("type", "text")
        input_name = input_tag.attrs.get("name")
        input_value = input_tag.attrs.get("value", "")
        inputs.append({"type": input_type, "name": input_name, "value": input_value})
    # put everything to the resulting dictionary
    details["action"] = action
    details["method"] = method
    details["inputs"] = inputs
    return details
```

Figure 28: Fetch Forms Details

Defined `is_vulnerable()` function with response parameter. A simple Boolean function that determines whether a page is SQL injection vulnerable from its 'response.' In this function created one errors dictionary and using for loop try to found error if find one of these errors then it returns true. That means this function is determining whether a page is vulnerable or not. This function is checking if there is any wrong SQL syntax or error in SQL syntax. It is checking if there are enclosed quotations in string. It is also checking if quoted string is properly terminated or not.

After that defined `scanning_sqli_injection()` function withurl parameter, then declared two empty variable. Add quote/double quote character to the URL and used predefined function

Bug Detection Using Artificial Intelligence and Cyber Security

is_vulnerable() with if...else.... Condition. Here used predefined empty variables one is use to print message and one is use for checking and based on detection it print that related message. Now test the form which has fetched using previous defined fetch_form() function and fetchformdetails() function and then data body want to submit. Any input form that is hidden or has some value, just used it in the form body and for that used "try" and "except" method. All others except submit and with that use some junk data with special character after that join the URL with the action (form request URL) so it check which method is used like post or get. Using is_vulnerable() test whether the resulting page is vulnerable or not.

```
# test on HTML forms
forms = fetch_forms(url)
print(f"[+] Detected {len(forms)} forms on {url}.")
for form in forms:
    form_details = fetchformdetails(form)
    for c in "\'":
        # the data body we want to submit
        data = {}
        for input_tag in form_details["inputs"]:
            if input_tag["type"] == "hidden" or input_tag["value"]:
                # any input form that is hidden or has some value,
                # just use it in the form body
                try:
                    data[input_tag["name"]] = input_tag["value"] + c
                except:
                    pass
            elif input_tag["type"] != "submit":
                # all others except submit, use some junk data with special character
                data[input_tag["name"]] = f"test{c}"
        # join the url with the action (form request URL)
        url = urljoin(url, form_details["action"])
        if form_details["method"] == "post":
            res = s.post(url, data=data)
        elif form_details["method"] == "get":
            res = s.get(url, params=data)
        # test whether the resulting page is vulnerable
        if is_vulnerable(res):
            print("SQL Injection vulnerability detected, link:", url)
            print("Form:")
            pprint(form_details)
            # break
return message
```

Figure 29: Test Vulnerability in Form without ML

Bug Detection Using Artificial Intelligence and Cyber Security

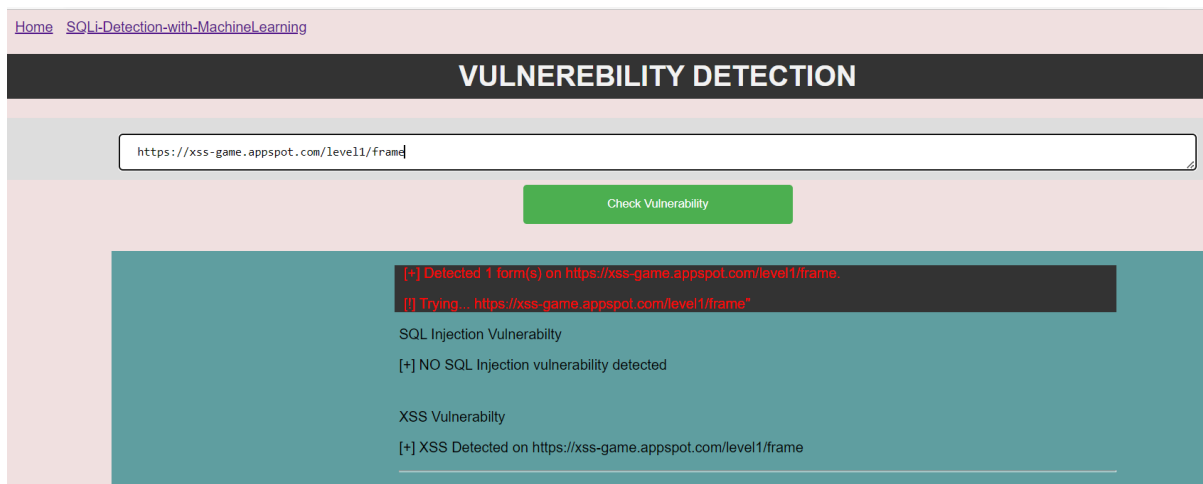


Figure 30: Result 1 of Vulnerabilities Detection Without ML Algorithm

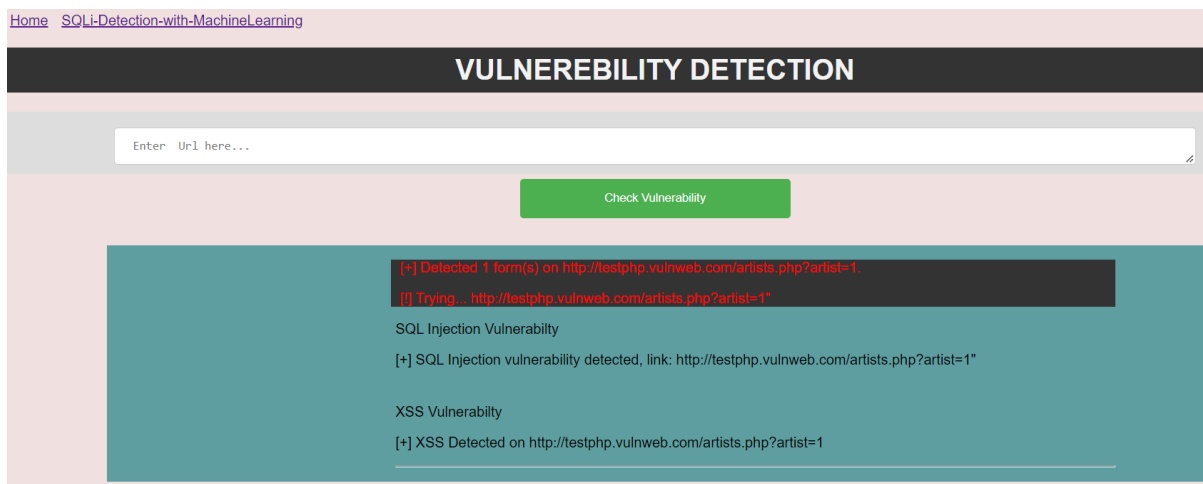


Figure 31: Result 2 of Vulnerabilities Detection Without ML Algorithm

That kinds of same coding done for XSS vulnerability detection; and for XSS defined submit_form() function and pass three parameters. Submits a form given form_details and its data type are list which is a dictionary that contain form information. There is other two parameter URL and value; and the data type of it is string. URL parameter will original URL that contain that form; and value parameter will be replaced to all text and search inputs and at the end returns the HTTP response after form submission. Same as SQL scanning part, defined scanning_XSS function with one parameter for XSS vulnerability. Given a URL, it prints all XSS vulnerable forms and returns true if any is vulnerable, false otherwise. End of the function does not used break because need to print available vulnerable forms. The other four vulnerabilities are header related which has X-Frame-Options, Strict-Transport-Security,

Bug Detection Using Artificial Intelligence and Cyber Security

X-Content-Type-Options, Content-Security-Policy. To test for a header vulnerability, you must determine if you can change the Host header and still send your request to the intended application. POST to /predict, means that add an element. In home1() function return the flask and show which vulnerabilities detected from the given URL.

Now starting machine learning part for SQL injection detection in website. First, defined preprocess() function with query parameter and inside that defined other three function process(), combined_keywords(), and genuine() with different parameters. That related brief explanation given at the end of “Compare ML Algorithms” part. Then, POST to /predictsqli, means that add an element. Given query parameter is a string given to predict that it is SQL injection or not. Here, also used fetch_forms() and fetchfordetails_url() functions to fetch form. Then used “machezize.Browser()” which gives a browser like object to interact with web pages. Before applying any query, it converting query to lowercase using “.lower()” When did feature extraction part that time generated train_bow file and that file save it and that file now use in this code. It is one of the trained files which is used at here for SQLi detection. First open and load train_bow; then stacks the arrays in a sequence horizontally and for that uses unigram_bow pre-generated file. Here, used XGBoost algorithm. First loading the “saved_model.pkl” model using “.load()” function, then prediction the output from the loaded model.

```
query = query.lower()#converting query to lowercase
arr = preprocess(query)#preprocessing the query
li.append(query)

with open('train_bow','rb') as f:
    train_bow = pickle.load(f)
unigram_bow = train_bow.transform(li)

combine = hstack((unigram_bow,arr))

#loading the model
xgboost_model = joblib.load('saved_model.pkl')

#predicting the output from the loaded model
predict = xgboost_model.predict(combine)

for i in predict:
    if(i == 1):
        prediction = "Positive"
    else:
        prediction = "Negative"

return flask.render_template('predict.html', prediction = prediction)
except:
return flask.render_template('predict.html')
```

Figure 32: XGBoost Algorithm Apply to Detect SQLi

Bug Detection Using Artificial Intelligence and Cyber Security

As a result, it returns “predict.html” file; in that file written two statements related the SQL query if site is secure then it shows “THE GIVEN QUERY BELONGS TO CLASS 0 MEANS THIS IS NOT A SQL INJECTION QUERY” or site is vulnerable then it shows “THE GIVEN QUERY BELONGS TO CLASS 1 MEANS THIS IS A SQL INJECTION QUERY.” Below given a website view and result.

[Home](#) [SQLi-Detection-with-MachineLearning](#)

SQL INJECTION DETECTION USING MACHINE LEARNING

Figure 33: Web Page of SQLi Detection with ML

SITE IS VULNERABLE

THE GIVEN QUERY BELONGS TO CLASS 1 MEANS THIS IS A SQL INJECTION QUERY

Figure 34: Web Page Result of SQLi Detection with ML

7) Chatbot

Instead of offering direct contact with a real human agent, a chatbot or chatterbot is a software program that is used to conduct an online chat conversation using text or text-to-speech. By automating conversations and interacting with clients through messaging systems, chatbots are a type of software that may assist customers (Wikipedia, 2022). Here created vulnerability detection web application but every user does not know about injection, that related risk and prevention technique. This chatbot give that related information to users. The purpose of this chatbot, customer service representatives are not available 24 hours a day, 7 days a week to help customers with their problems; but chatbot give them answer 24/7. It analyses chats using natural language processing and extracts the user's purpose utilizing information comparable to the user's likes. First examine how to begin a chatbot discussion and how to progress the conversation by selecting or querying via several categories based on user needs. It is a project based on the "Neural Network" technique. It necessitates the use of natural language processing: tokenization, entity recognition, and intent extraction NLP approaches. In coding section, there are one "bse.html" file where written coding for chatbot design. Three main python file is "chat.py", "app.py", and "train.py". Neural Network, tokenization, and bag of words coding do in "chat.py" python file. Using "tag" and "pattern" train the chatbot; which is defined in "intents.json" file. And "app.py" is main file in that used flask; get response from user and based on that give answer; that related coding done in it.

Bug Detection Using Artificial Intelligence and Cyber Security

```
base.html 2, M # style.css 2, M app.py M intents.json M X train.py
intents.json > [ ] intents
31   "My pleasure"
32   },
33   {
34     "tag": "SQL Injection",
35     "patterns": [
36       "SQLinjection",
37       "SQLI",
38       "SQL Injection",
39       "What is SQL injection?",
40       "About SQL injection",
41       "What is concept of SQLI?"
42     ],
43     "responses": [
44       "SQLI is the most threat to Database system. It is common attack vector that uses malicious SQL commands for backend database manipu
45     ]
46   },
47   {
48     "tag": "Risk With SQLI",
49     "patterns": [
50       "What is risk with SQL injection?",
51       "risk of SQL injection."
52     ],
53     "responses": [
54       "There are different kinds of risk with SQLinjection : \n1)By Passing Authentication \n2)Identifying Injectable Parameters \n3)Execut
55     ]
56   },
57   {
58     "tag": "SQLI Prevention",
59     "patterns": [
60       "How to prevent SQLI?",
61       "How to protect against SQLI?",
62       "SQLI prevention"
63     ],
64     "responses": [
65       "1)Validate user inputs \n2)Sanitize data by limiting special characters \n3)Enforce prepared statments and parameterization \n4)Use
66     ]
67   }
}
```

Figure 35: Coding of intents.json File

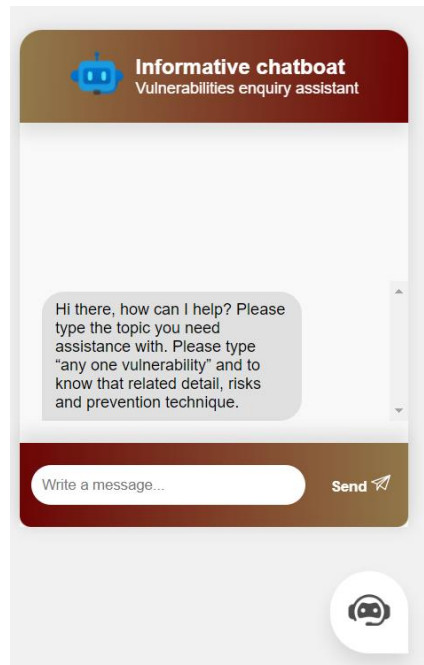


Figure 36: Chatbot

8) Conclusion

In this thesis, created one website to detect different vulnerabilities in various website using that URL. In that web application used two types of coding to detect vulnerabilities one without machine learning and other with machine learning; both parts working properly. Machine learning algorithms are used to solve the SQL injection detection problem. Compared different machine learning algorithms and decided to use XGBoost algorithm because Extreme Gradient Boost machine learning model provides results with an accuracy of is 89.64%. As a result, XGBoost from ensemble learning is selected to be implemented on the SQL injection problem. On the webpage has one chatbot; in that uses Neural Network technique. That chatbot gives information about different vulnerabilities information, that related risk and prevention technique.

9) Limitation and Future Work

In this research, created web application to detect vulnerability using machine learning algorithm but ML part only used to detect SQLi vulnerability. ML section did not use for other injection detection because that vulnerabilities have not large amount of data and ML only work with huge number of datasets. If in future collect large number of datasets for other injection then it is possible to detect other vulnerabilities using ML. But for some vulnerabilities only need single or few scripts and for that type of vulnerabilities is not possible to make large dataset.

References

- Adamjee, U., 2020. *Introduction to NLTK library in python*. [Online]
Available at: <https://python.plainenglish.io/introduction-to-nltk-library-in-python-6fa729b54ad>
- Adil, 2021. *Learning Never Ends*. [Online]
Available at: <https://learningneverendstech.com/2021/11/24/layout-cshtml-file-master-page-in-asp-net-mvc/#:~:text=Layout.cshtml%20File%20In%20ASP.Net%20MVC%20Application%20When%20you,can%20apply%20to%20the%20views%20of%20the%20application.>
- Anon., 2011 - 2021. *JavaTpoint*. [Online]
Available at: <https://www.javatpoint.com/machine-learning-random-forest-algorithm>
- Anon., 2011 - 2021. *JavaTpoint*. [Online]
Available at: <https://www.javatpoint.com/logistic-regression-in-machine-learning>
- Anon., 2011 - 2021. *JavaTpoint*. [Online]
Available at: <https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm>
- Anon., 2015 - 2022. *python tutorials*. [Online]
Available at: <https://pythonspot.com/nltk-stop-words/>
- Anon., 2022. *.NET Core vs .NET Framework*. [Online]
Available at: <https://www.interviewbit.com/blog/net-core-vs-net-framework/>
- Anon., 2022. *TutorialsTeacher*. [Online]
Available at: <https://www.tutorialsteacher.com/mvc/mvc-model>
- Asra Kalim, C. K. J. D. S. T. D. R. S., February 2020. A Framework for Web Application Vulnerability Detection. *International Journal of Engineering and Advanced Technology (IJEAT)*, 9(3), p. 544.
- Brownlee, J., 2021. *Machine Learning Mastery*. [Online]
Available at: <https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/>
- developers, s.-l., 2007 - 2022. *scikit learn*. [Online]
Available at: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html#:~:text=The%20formula%20that%20is%20used%20to%20compute%20the,the%20document%20set%20that%20contain%20the%20term%20t.
- Edward, 2019. *stack overflow*. [Online]
Available at: <https://stackoverflow.com/questions/57346982/what-does-model-exactly-do>
- Fardeen, A., 2021. *Generating Unigram, Bigram, Trigram and Ngrams in NLTK*. [Online]
Available at: <https://machinelearningknowledge.ai/generating-unigram-bigram-trigram-and-ngrams-in-nltk/>
- Farooq, U., 2021. Ensemble Machine Learning Approaches for Detection of SQL Injection Attack. *Technical Journal*, Volume 15, pp. 112-120.
- Ganesan, K., 2021. *Opinosis Analytics*. [Online]
Available at: <https://www.opinosis-analytics.com/ai-glossary/precision-and-recall-machine-learning>

Bug Detection Using Artificial Intelligence and Cyber Security

Havannavar, R., 2021. [Online]

Available at: <https://medium.com/@rohanhavannavar/predict-if-the-sql-injection-query-can-get-access-to-database-or-not-using-machine-learning-2c9303024da3>

JavaTpoint, 2011-2021. *JavaTpoint*. [Online]

Available at: https://www.javatpoint.com/fit-transform-and-fit_transform-methods-in-python

Khanna, M., 2020. *python Flask Tutorial - Getting Started with Flask*. [Online]

Available at: <https://scoutapm.com/blog/python-flask-tutorial-getting-started-with-flask>

Kumar, G. S., 2021. *CountVectorizer in Python*. [Online]

Available at: <https://suresh-analytics.medium.com/countvectorizer-in-python-2643b8c69453>

Markham, K., 2021. *stackoverflow*. [Online]

Available at: <https://stackoverflow.com/questions/27697766/understanding-min-df-and-max-df-in-scikit-countvectorizer>

Microsoft, 2022. *using directive*. [Online]

Available at: <https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/using-directive>

Mishra, S., Spring 5-23-2019. *SQL Injection Detection Using Machine Learning*, San Jose, CA, U.S.A.: San Jose State University.

Ng., R., 2022. *Ritchie Ng*. [Online]

Available at: <https://www.ritchieng.com/machinelearning-f1-score/>

Pedamkar, P., 2022. *EDUCBA*. [Online]

Available at: <https://www.educba.com/decision-tree-algorithm/>

Ross, K., Spring 2018. *SQL Injection Detection Using Machine Learning Techniques and Multiple Data Sources*, San Jose, CA, U.S.A.: San Jose State University.

Singh, J., 2019. *C#Corner*. [Online]

Available at: <https://www.c-sharpcorner.com/blogs/why-should-and-should-not-we-use-net-core#:~:text=Advantages%20of%20.NET%20Core%20is%20.NET%20Core%20is,Core%20is%20much%20faster%20compared%20to%20.NET%20Framework.>

TUtorial, P., 2021. *Flask Tutorial: Routes*. [Online]

Available at: <https://pythonbasics.org/flask-tutorial-routes/>

Vogel, L., 2022. *Introduction to Java Web Development - Tutorial*. [Online]

Available at:

<https://www.vogella.com/tutorials/JavaWebTerminology/article.html#:~:text=A%20%E%80%80Java%20web%20application%E%80%81%20is%20a%20collection%20of,%E%80%80Web%E%80%81%20Standards.%20Standard%20%E%80%80Java%E%80%81%20technologies%20are%20defined%20>

Wikipedia, 2022. *Chatbot Wikipedia*. [Online]

Available at: <https://en.wikipedia.org/wiki/Chatbot>

XGBoost, 2022. *GeeksforGeeks*. [Online]

Available at: <https://www.geeksforgeeks.org/xgboost/>