IMPROVING DETECTION OF CROSS-SITE SCRIPTING (XSS) ATTACKS TOWARDS WEB APPLICATION THROUGH COMPREHENSIVE MACHINE LEARNING ALGORITHMS

BASYIRATUL ULFA BINTI BAHARUDDIN

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

IMPROVING DETECTION OF CROSS-SITE SCRIPTING (XSS) ATTACKS TOWARDS WEB APPLICATION THROUGH COMPREHENSIVE MACHINE LEARNING ALGORITHMS

BASYIRATUL ULFA BINTI BAHARUDDIN

This report is submitted in partial fulfillment of the requirements for the Bachelor of Computer Science (Computer Security) with Honours.

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2024

DECLARATION

I hereby declare that this project report entitled

IMPROVING DETECTION OF CROSS-SITE SCRIPTING (XSS) ATTACKS TOWARDS WEB APPLICATION THROUGH COMPREHENSIVE MACHINE LEARNING ALGORITHMS

is written by me and is my own effort and that no part has been plagiarized without citations.

Signature : [illegible]

Student : Basyiratul Ulfa binti Baharuddin

Date : 02/09/2024

APPROVAL

I hereby declare that I have read this project report and found this project report is sufficient in term of the scope and quality for the award of Bachelor of Computer Science (Computer Security) with Honours.

Signature	: [illegible]
Supervisor	: Profesor Madya Ts. Dr. Mohd Faizal Bin Abdollah
Date	: 10 SEPTEMBER 2024

DEDICATION

Dedicated to all my beloved and supporting family, friends and lecturers.

Thank you for your understanding and guidance.

May Allah bless.

ACKNOWLEDGEMENTS

Bismillahirrahmanirrahim

Alhamdulillah, I extend my heartfelt gratitude to Allah Subhanahu wa Ta'ala for His blessings of time, health, guidance and strength, which were essential during my journey in finishing this Final Year Project which titled Improving Detection of Cross-Site Scripting (XSS) Attacks towards Web Application through Comprehensive Machine Learning Classifier/ Algorithm. This final year project report was prepared for Faculty of Information and Communication Technology (FTMK), Universiti Teknikal Malaysia Melaka (UTeM), as part of the requirement for final year student to complete the undergraduate studies and earn a Bachelor of Computer Science degree.

I am deeply thankful to my parents, family members, friends, and those who have helped me whether directly or indirectly. Their understanding, guidance, and unwavering support have been invaluable during my journey in completing this final year project and report. Their presence has been the cornerstone of my success in this journey.

I also want to express my sincere appreciation for my supervisor, Assoc. Prof. Dr. Mohd. Faizal Abdollah for his belief in my abilities and determination. His constructive criticism, patience and guidance have encouraged me to persevere and grow, shaping a new version of me, to always continue to strive and move forward.

Above all, I wish everyone happiness, success and smooth sailing in their lives. May Allah bless our lives, guiding us always on the right path.

ABSTRACT

In today's rising digital world, Cross-Site Scripting (XSS) poses a significant security threat to web applications, enabling attackers to inject malicious code that compromise user data, hijack user sessions, and modify web content. Despite current detection methods, many systems have significant false positive rates and are not adaptable to new attack vectors. This study intends to improve XSS detection by using Machine Learning approaches, in conjunction with various feature selection methods. The study includes gathering and preprocessing a complete dataset, training and testing the machine learning model, and evaluating its performance against various XSS attack scenarios. The results indicate that the proposed method significantly improves detection accuracy and reduces false positives, providing a robust solution for safeguarding web applications. This study promotes online security by proposing a practical and adaptable strategy for detecting XSS vulnerabilities. The findings of this study show the potential for enhanced web application security and emphasize the need for future investigation of machine learning techniques in threat detection.

ABSTRAK

Dalam dunia digital yang semakin berkembang hari ini, Cross-Site Scripting (XSS) merupakan ancaman keselamatan yang signifikan kepada aplikasi web, membolehkan penyerang menyuntik kod berniat jahat yang boleh membahayakan data pengguna, merampas sesi pengguna, dan mengubah kandungan web. Walaupun terdapat kaedah pengesanan semasa, banyak sistem mempunyai kadar positif palsu yang tinggi dan tidak boleh menyesuaikan diri dengan vektor serangan baharu. Kajian ini bertujuan untuk meningkatkan pengesanan XSS dengan menggunakan pendekatan Pembelajaran Mesin, bersama dengan pelbagai kaedah pemilihan ciri. Kajian ini merangkumi pengumpulan dan prapemprosesan set data yang lengkap, melatih dan menguji model pembelajaran mesin, dan menilai prestasinya terhadap pelbagai senario serangan XSS. Keputusan menunjukkan bahawa kaedah yang dicadangkan meningkatkan ketepatan pengesanan dengan ketara dan mengurangkan positif palsu, memberikan penyelesaian yang kukuh untuk melindungi aplikasi web. Kajian ini mempromosikan keselamatan dalam talian dengan mencadangkan strategi praktikal dan boleh disesuaikan untuk mengesan kerentanan XSS. Penemuan kajian ini menunjukkan potensi untuk meningkatkan keselamatan aplikasi web dan menekankan keperluan untuk penyelidikan lanjut mengenai teknik pembelajaran mesin dalam pengesanan ancaman.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

All kinds of business web application vulnerability statistics are growing in recent and new forms of hacker attacks. Well secured enterprise web applications are threatened by smaller vulnerable webs misused by hackers in targeted attacks (Tamara Saad Mohamed, 2020). Many developers also did not put a heartful focus into the web development phase, which unluckily led to the rise of vulnerabilities inside one website, (Divyam Goyala, Pulkit Jainb, Bharat Bhushanc, 2020). This lack of good practice is considered a vulnerability. A hacker could easily take advantage of this flaw to execute some malicious code on the systems, (Germán E. Rodríguez, Jenny G. Torres, Pamela Flores, Diego E. Benavides, 2020). One of the most common cyber-attacks typically found in web applications is Cross-site scripting (XSS), (Jalen Mack, Yen-Hung (Frank) Hu, and Mary Ann Hoppa, 2019).

Cross-site scripting attack is a web application vulnerability, which allows attackers to inject malicious code into a web page that other users can view. This attack can steal sensitive info like session IDs and cookies, redirect users onto malicious sites, or perform other malicious actions. (Jaydeep R. Tadhani, Vipul Vekariya, Vishal Sorathiya, Samah Alshathri & Walid El-Shafai, 2024). With the help of injected code, an intruder or attacker can gain unauthorized access to user data, which could allow them to impersonate these users, perform illegal actions on the local computers of users and the network equipment of their companies or change the configurations of their network and software. (Mohammad Alsaffar, Saud Aljaloud, Badiea Abdulkarem Mohammed, Zeyad Ghaleb Al-Mekhlafi, Tariq S. Almurayziq, Gharbi Alshammari, and Abdullah Alshammari, 2022). Many network solutions only scan the headers of a user request while ignoring the payload. Many programmers and computer security specialists have tried to mitigate these attacks either through scanners, firewalls, encryption devices, (Jean Rosemond Dora, and Karol Nemoga, 2021). On top of that, the number of cross-site script (XSS) injection attacks has increased from 470 in 2011 to 22,000 in April 2022, (Rakesh Kumar Singh, and Amar Kumar Mohapatra, 2022).

Informally, the cause of XSS is a lack of input sanitization: user-chosen data “escapes” into a page’s template and makes its way into the JavaScript engine, or modifies the DOM, (José Carlos Pazos, Jean-Sébastien Légaré, Ivan Beschastnikh, and William Aiello, 2020). A thorough analysis of Cross-Site Scripting vulnerabilities has been presented in detail; the numerous forms of XSS assaults, how an attacker may exploit this weakness, the results of an XSS attack, and the protective strategies established. However, despite researchers’ efforts, XSS attacks can still disrupt web applications at a larger skill irrespective of the fact that various tactics and approaches for preventing vulnerabilities have been established, (Sonkarlay J. Y. Weamie, 2022).

1.2 PROBLEM STATEMENT

Cross-Site Scripting (XSS) attacks are a serious threat to web applications, as they exploit vulnerabilities to compromise user data and system integrity. Existing XSS detection approaches have limitations that can result in false positives or negatives, emphasizing the need for more advanced detection methods. This research attempts to improve XSS detection by experimenting various approach for feature selection and model training, as well as assessing its efficiency against various types of XSS attacks. Table 1 shows the related problem statement towards the project.

Table 1: Problem Statement

PS	Problem Statement
PS1	How can we enhance the detection of Cross-Site Scripting (XSS) attacks in web applications, by incorporating selected Feature Selection Methods and Model Training?

1.3 PROJECT QUESTION

Based on the problem statement, four projects (PQ) are constructed in the project as shown in Table 2 below.

Table 2: Project Question

PS	PQ	Project Question
PS1	PQ1	What are the existing XSS detection techniques and what limitations do they have?
	PQ2	How can these techniques be enhanced to address their limitations, specifically through the incorporation of selected feature selection methods and model training?
	PQ3	How effective is the enhanced system or tool in detecting and mitigating various forms of XSS attacks?

1.4 PROJECT OBJECTIVE

This project has four main objectives, which are to implement a robust and resilient approach towards cross-site scripting (XSS) attack within the web application.

Table 3: Summary of Project Objectives

PS	PQ	PO	Project Objective
PS1	PQ1	PO1	To analyze existing XSS detection techniques and identify their limitations.
	PQ2	PO2	To propose enhancements to the existed techniques that address their limitations, incorporating various selected feature selection methods and model training.
	PQ3	PO3	To evaluate the performance and effectiveness of the enhanced system or tool against various forms of XSS attacks.

1.5 PROJECT SCOPE

The scope of this project is:

1. Dataset collection

   : Collecting raw data from various sources of dataset. This list of datasets then will be compared to choose the best dataset that contains a wide list of features and attributes. During this procedure, we determine dataset size, feature richness, and representativeness. The purpose is to develop a list of prospective datasets for Cross-Site Scripting (XSS) detection.

2. Data preprocessing

   : A comprehensive preparation is required before putting data into any machine learning method. We deal with challenges including missing values, noisy data, and outliers. Imputation, data cleaning, and normalization techniques all help to prepare the dataset for model training.

3. Feature selection

   : Selecting the proper features has a big impact on model performance. For that, a thoroughly study has been done towards the properties and attributes of the selected dataset. Filter-based ranking, wrapper methods, and embedding strategies are all useful for identifying the most relevant features. The idea is to maintain informative elements while reducing noise.

4. Training and testing machine learning algorithms

   : With a preprocessed dataset, we train machine learning algorithms. The proposed algorithms learn from the labelled data. We divided the dataset into training and testing subsets to assess the model's performance. Rigorous testing ensures that the classifier can accurately identify XSS attacks.

5. Performance evaluation

: To validate the system's effectiveness, we evaluate its performance using appropriate measures. Metrics such as accuracy, precision, recall, F1-score, and ROC-AUC provide information about the model's behavior. We compare our findings to current literature to determine the system's competitiveness.

1.6 PROJECT CONTRIBUTION

In today's rising interconnected digital landscape along with the rising of XSS attacks, safeguarding web applications has become paramount. This project has made important contributions to several issues which are derived from the project objectives described in section 1.4, Table 3. The expected output of this project is to enhance the detection of XSS attacks, thus reducing the statistic of the XSS attacks. The summary of project contributions is illustrated in Table 4.

Table 4: Summary of Project Contributions

PS	PQ	PO	PC	Project Contribution
PS1	PQ1	PO1	PC1	Thoroughly examine the existing XSS detection methods and identify their limitations and areas of enhancement.
		PO2	PC2	Proposed machine learning classifier and feature selection method for XSS attack detection purpose.
		PO3	PC3	Evaluate the performance and effectiveness of the proposed methods and algorithms.

1.7 REPORT ORGANIZATION

Chapter 1: Introduction

This chapter discusses the introduction, problem statement, project questions, project objectives, project scopes, and project contributions for this project.

Chapter 2: Literature Review

This chapter discusses several past research papers and studies, focusing on the implementation of machine learning methods and algorithms in detecting XSS attacks. It covers the review of terminology to the project issues based on related studies, their strengths, weaknesses and limits, the future development towards the project experimented.

Chapter 3: Project Methodology

Chapter 3 illustrates the methodology applied to this project, providing a step-by-step approach to the proposed work. The third chapter also includes a milestone, alongside the Gantt chart to monitor the continuation of project development and implementation.

Chapter 4: Analysis and Design

Chapter 4 discusses the design aspects of the projects, and other consideration design. In addition, we will investigate the expected outcomes of the system.

Chapter 5: Implementation

This chapter focuses on the project's implementation of various algorithms of machine learning towards the chosen dataset. This also includes how the system's input and output are generated prior to the final product.

Chapter 6: Testing

Chapter 6 describes the testing and evaluation of feature selection methods used in the project to obtain the project’s outcome.

Chapter 7: Project Conclusion

This chapter will conclude the whole project continuity and implementation, including research summary, contributions, and limitations. Hence, a significant yet comprehensive improvement will be made in this section.

1.8 CONCLUSION

Chapter 1 discussed the growing threat of Cross-Site Scripting (XSS) attacks and the limits of current detection technologies. The chapter defines explicit project goals and objectives, with an emphasis on analyzing, improving, integrating, and evaluating advanced XSS detection systems. The project's scope comprises key phases such as dataset collecting, data preprocessing, feature selection, classifier training and testing, and performance evaluation. This foundation lays the groundwork for a more in-depth examination in later chapters, with the goal of improving XSS detection and contributing practical solutions to web application security.

CHAPTER 2: LITERATURE REVIEW

2.1 INTRODUCTION

In this chapter, the research paper, articles and studies related to this project are presented and will be discussed meticulously. The literature review serves to describe and define the previous research conducted and is relatable to the project's topic. Thus, this chapter will elucidate the fundamentals of Cross-Site Scripting (XSS) Attacks, how they launched, the impact, as well as the comprehensive techniques and methods that are being taken to detect such malicious attacks. This also serves as acknowledgement of the previous researchers' unwavering dedication to their work which was evident in every hour spent in the lab, every data point analyzed, and hypothesis tested, embodying the 'blood, sweat, and tears' in the pursuit of knowledge and discovery. This extensive study will surely help readers obtain a complete grasp of the project and its context within the broader field of cybersecurity.

2.2 CROSS-SITE SCRIPTING (XSS) ATTACK

2.2.1 Nature and Impact of XSS attacks

The rising demand of the use of web applications in today's digital era awakened the public point of view, especially among the cybersecurity expert towards the integrity of user data and the level of security provided. Over the years, there have been numerous cases involving data breaches, phishing, ransomware, and more. Among these myriad types of cyber threats, Cross-Site Scripting (XSS) attacks have emerged as a significant concern, which often targets unsecured web applications.

Cross-Site Scripting (XSS) attacks have been recognized as one of the top 10 online application security risks by the Open Web Application Security Project (OWASP) for decades. As the latest research quote that Cross-Site Scripting (XSS) attack lists as the 2nd most critical web application security attack (Banerjee R., Baksi A., Singh N., Bishnu S.K., 2020). Cross-Site Scripting (XSS) is a process of malicious code injection attack where attackers insert code into a website or online application to gain access or do harm.

There is variety of payloads type that happen to be injected through the users' browsers, such as scripts and iframes. An occurrence of an attack like cross-site scripting allows an attacker to evade the same-origin policy, which is designed to isolate different websites from each other.

This kind of attack also enables attackers or hackers to steal the users' private and confidential information. In web-browsers or web applications, malicious script code is executed and used to transfer the sensitive data to the third-party domain. The attacker can also subsequently impersonate the user, performing illegal activities under the victim's account, such as social engineering attacks. The impact of this attack can be significant as it does perform various malicious things such as steal session cookies, impersonate users, deface websites, spread malware, phish for user credentials, and support social engineering techniques. Thus, detection and preventing XSS attacks is crucial to ensure the safety of data.

A cross-site Scripting (XSS) attack is clearly shown by the case of Timothy Barker, a Canadian guy caught up in a complicated e-commerce scam. Barker's story, which resembled triangulation fraud, involves unknowingly purchasing stuff from an internet merchant who used stolen payment card info to buy goods from other stores. This plot wrongly accused Barker and had serious ramifications, such as job loss and legal uncertainty due to his criminal record. His example underlines the crucial need for comprehensive detection and mitigation measures against XSS and other cyber threats, as well as the severe real-world consequences for individuals and communities affected by such security breaches.

2.2.2 Types of Cross-Site Scripting (XSS) Attacks

1. Non-Persistent (Reflected) XSS

This type of XSS attack occurs when the malicious script is embedded in a URL and reflected off the web server. When a user visits this specific URL, the attack code will be run and executed in the user's browser (Alenzi K.F., Abbas O.A.B., 2022). The malicious script is not stored on the target server, hence the term "non-persistent". It's important to note that the malicious script is executed when the user retrieves the manipulated URL. Figure 1 shows the reflected XSS attack flow.

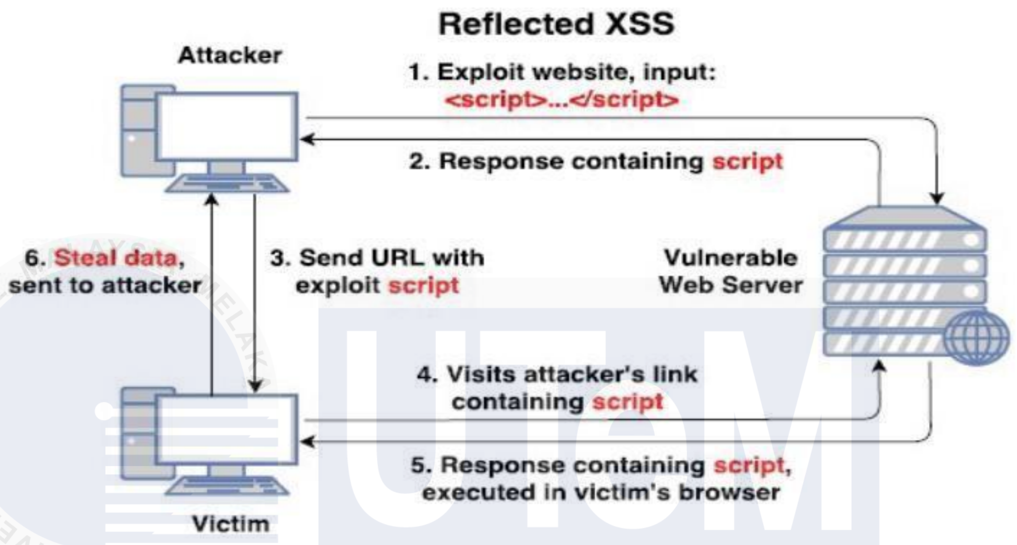

Reflected XSS
Attacker
1. Exploit website, input: <script>...</script>
2. Response containing script
6. Steal data, sent to attacker
3. Send URL with exploit script
Vulnerable Web Server
4. Visits attacker's link containing script
5. Response containing script, executed in victim's browser
Victim

Figure 1: Flow of Reflected XSS Attack

2. Persistent (Stored) XSS

In a Persistent XSS attack, the malicious script is permanently stored on the target server, such as in a database, message forum, comment field, etc. When a user visits any publicly accessible area of a websites, which might be injected by malicious script, the browser will retrieve and present the data. This will in turn execute the XSS attack stored in the browser content (Alenzi K.F., Abbas O.A.B., 2022). Because the malicious script is stored on the server and can affect multiple users, this type of attack can be more dangerous than a non-persistent XSS attack. Figure 2 illustrates the flow of a typical Stored XSS attack.

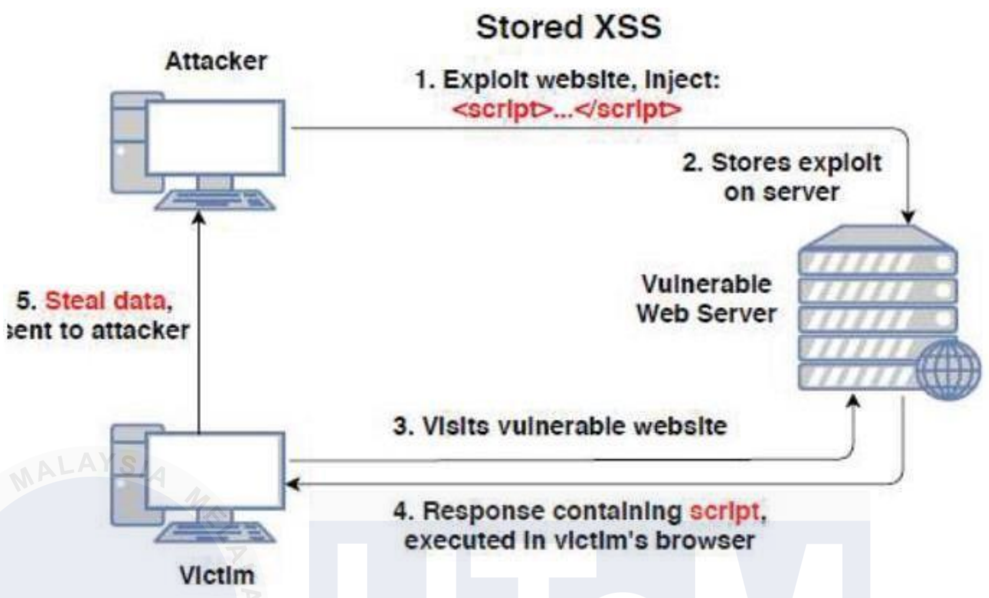

Stored XSS
Attacker
1. Exploit website, Inject: <script>...</script>
2. Stores exploit on server
Vulnerable Web Server
5. Steal data, sent to attacker
3. Visits vulnerable website
4. Response containing script, executed in victim's browser
Victim

Figure 2: Flow of Stored XSS Attack

3. DOM-Based XSS

DOM-Based XSS attacks occur when a malicious script manipulates the Document Object Model (DOM) of a webpage. The DOM allows the page to interact with JavaScript page code, making the page more dynamic (Alenzi K.F., Abbas O.A.B., 2022). If the JavaScript entry is not handled properly, then it also makes it possible for the malicious code to change or manipulate the page. This script manipulates the webpage's environment in the client-side script, allowing the attacker to run arbitrary code in the user's browser. Unlike the other two types, the server's response does not change in a DOM-based XSS attack. Figure 3 illustrates the flow of DOM-based attack.

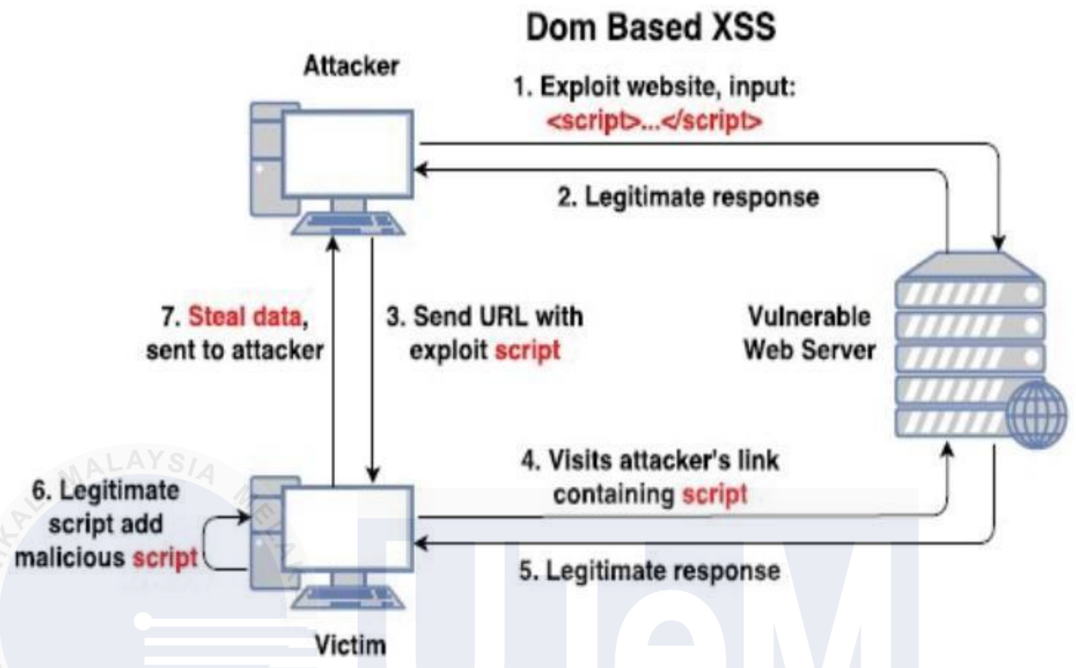

Dom Based XSS
Attacker
1. Exploit website, input: <script>...</script>
2. Legitimate response
7. Steal data, sent to attacker
3. Send URL with exploit script
Vulnerable Web Server
4. Visits attacker's link containing script
6. Legitimate script add malicious script
5. Legitimate response
Victim

Figure 3: Flow of DOM-Based Attack

2.2.3 History of Cross-Site Scripting (XSS) attacks

Cross-site scripting attacks have occurred since the 1990s. In January 2000, a Microsoft security engineer coined the term "cross-site scripting". Today, XSS remains a big danger to web applications. XSS attacks affect many popular social networking sites, including Facebook, Twitter, and YouTube. According to Netsparker web security data, cross-site scripting is a widespread vulnerability in web applications (Nagarjun, P., & Shakeel, S., 2020).

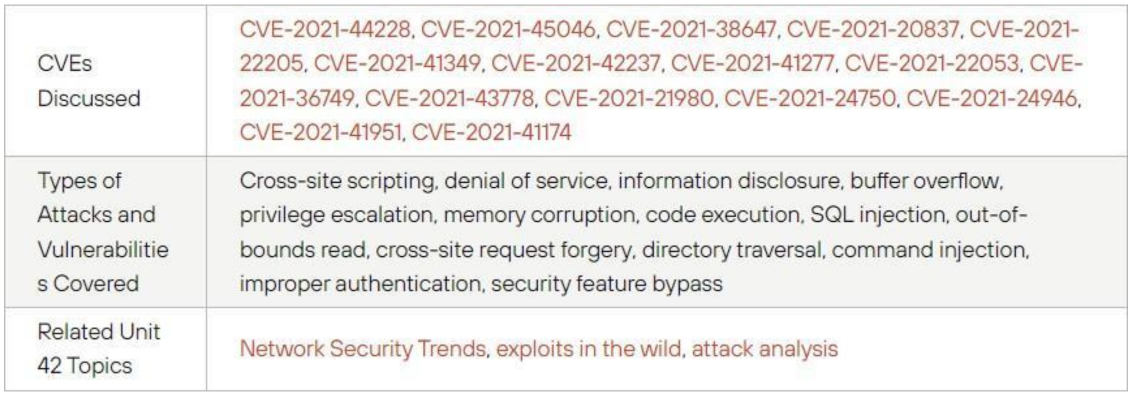
XSS has passed several key milestones during its evolution. Microsoft's formal reporting of XSS vulnerabilities in 1999 solidified its reputation in the security world. By 2003, OWASP had included XSS in its initial Top 10 list of online application vulnerabilities, sparking efforts to limit its impact. Between 2005 and 2010, there was an increase in awareness and study, which culminated in developments such as the W3C's Content Security Policy (CSP) in 2013. Even in 2021, XSS remains a significant concern, as evidenced by its presence in the latest OWASP Top 10 report. (Hannousse A., Yahiouche S., & Cherif M., 2022).

In its early phases, XSS took advantage of a basic understanding of web security, primarily targeting websites that lacked comprehensive input validation. Using the simplicity of script injections, attackers might execute arbitrary JavaScript in users' browsers, compromising sessions and sensitive data. Influential research articles, such as CERT's 2002 review and OWASP's inclusion of XSS in its 2003 Top 10 list, sparked advances in defense tactics. Seth Fogie et al.'s 2010 book "XSS Attacks: Cross Site Scripting Exploits and Defense" provides in-depth analysis and mitigation approaches, expanding the field's understanding. Countermeasures progressed over time from simple input sanitization to advanced approaches like output escaping and the implementation of Content Security Policy (CSP) in 2013, allowing developers to restrict browser resources effectively. Today, developments in Web Application Firewalls (WAFs) and browser security continue to strengthen defenses against real-time XSS attacks.

In conclusion, the history of XSS attacks demonstrates a constant conflict between changing attack strategies and the development of advanced defense systems. Understanding the historical backdrop of XSS, from its early detection to current online security initiatives, provides insights into the ongoing attempts to protect web applications from this persistent danger.

2.2.4 Network Security Trends

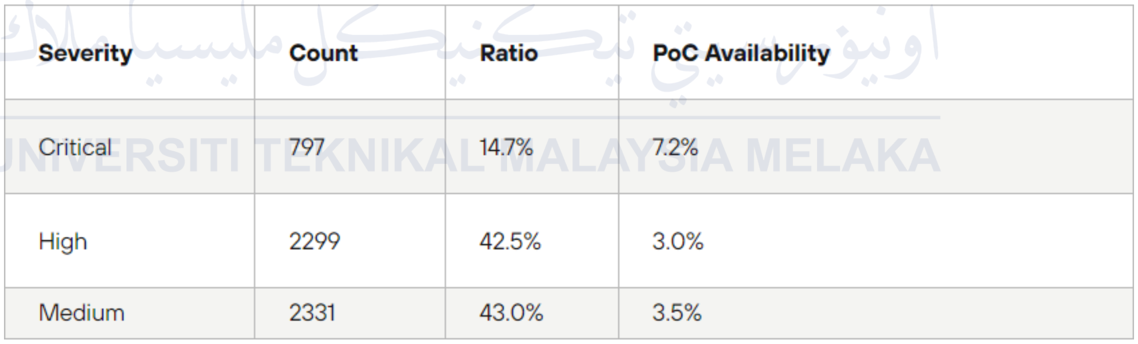
Between November 2021 and January 2022, XSS vulnerabilities accounted for approximately 10.6% of all newly reported security concerns. This demonstrates the prevalence of XSS and its continuous relevance as a threat vector in the cybersecurity landscape (Guan Y., 2022). The severity of XSS vulnerabilities fluctuated greatly during the observation period, with severe and high-severity vulnerabilities drawing a lot of attention due to their potential for widespread exploitation. For example, vulnerabilities listed in the research, such as CVE-2021-44228 and CVE-2021-45046, witnessed active exploitation shortly after their disclosure, emphasizing the need of mitigating such risks swiftly. Figure 4 shows the list of all CVEs that have been discussed.



CVEs Discussed	CVE-2021-44228, CVE-2021-45046, CVE-2021-38647, CVE-2021-20837, CVE-2021-22205, CVE-2021-41349, CVE-2021-42237, CVE-2021-41277, CVE-2021-22053, CVE-2021-36749, CVE-2021-43778, CVE-2021-21980, CVE-2021-24750, CVE-2021-24946, CVE-2021-41951, CVE-2021-41174
Types of Attacks and Vulnerabilities Covered	Cross-site scripting, denial of service, information disclosure, buffer overflow, privilege escalation, memory corruption, code execution, SQL injection, out-of-bounds read, cross-site request forgery, directory traversal, command injection, improper authentication, security feature bypass
Related Unit 42 Topics	Network Security Trends, exploits in the wild, attack analysis

Figure 4: CVEs (Guan Y., 2022)

From November 2021 to January 2022, 6443 new Common Vulnerabilities and Exposures (CVE) numbers were registered. The severity of those CVEs was examined by relies on proof-of-concept (PoCs). Figure 5 and 6 illustrates the potential impact of the vulnerabilities and their severity.



Severity	Count	Ratio	PoC Availability
Critical	797	14.7%	7.2%
High	2299	42.5%	3.0%
Medium	2331	43.0%	3.5%

Figure 5: CVEs severity registered from November 2021 to January 2022

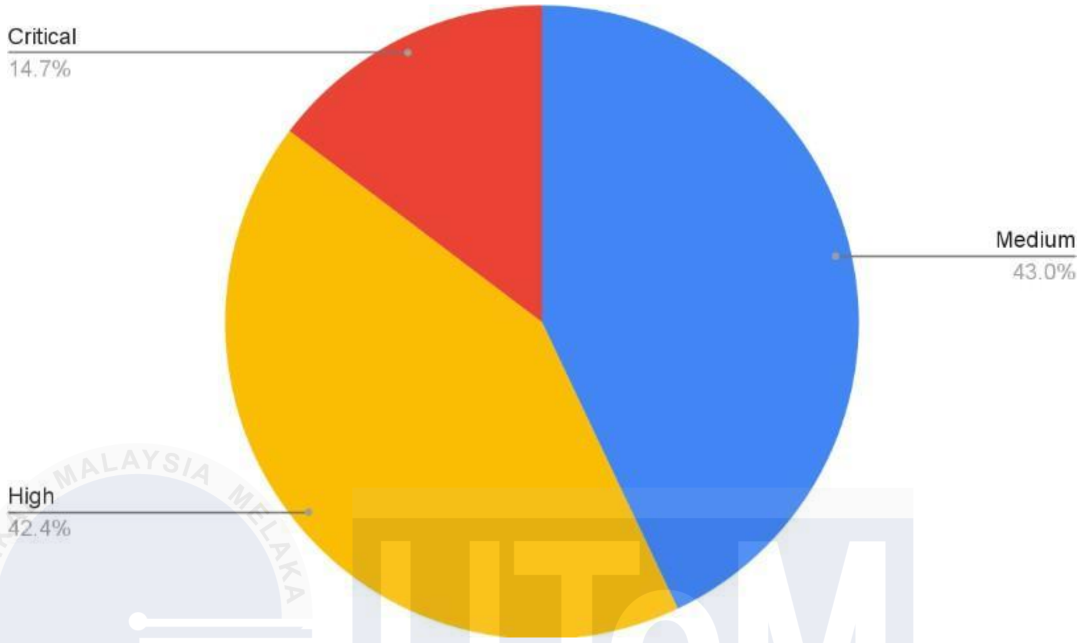


Figure 6: cont. of CVEs severity registered from November 2021 to January 2022

Among newly announced Common Vulnerabilities and Exposures (CVEs), 31.3% are local vulnerabilities that need prior access to compromised systems. The remaining 68.7% are remote vulnerabilities, which pose a risk to affected organizations worldwide. Based on figure 7, notably, cross-site scripting (XSS) remains the most common vulnerability, with an uptick in buffer overflow vulnerabilities throughout this period. Denial-of-service assaults also increased in number, while the majority of XSS and denial-of-service occurrences were medium or high intensity.

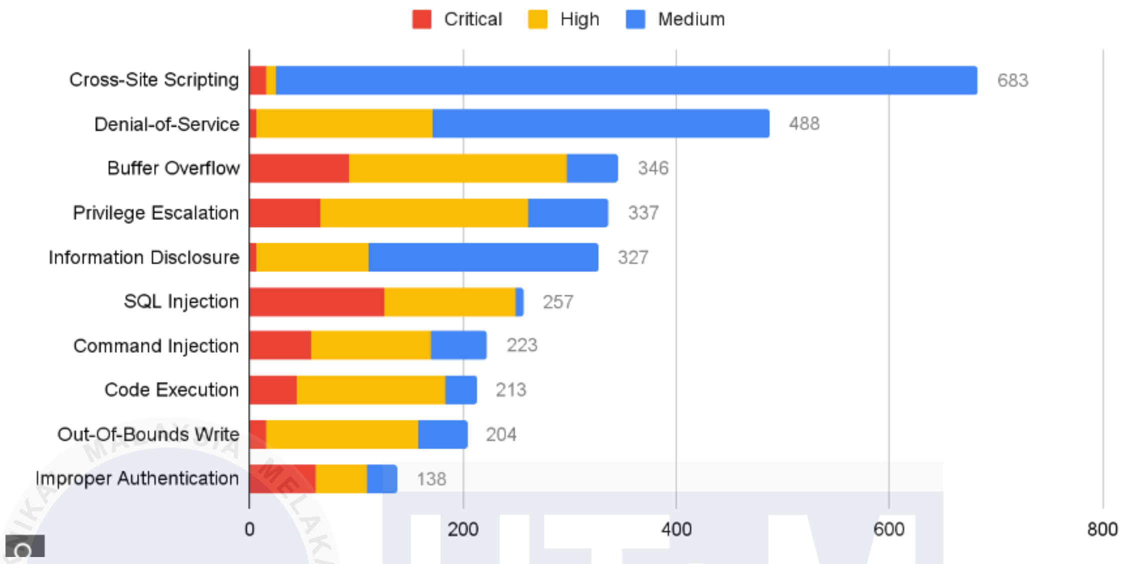


Figure 7: Vulnerability category distribution for CVEs registered from November 2021 to January 2022

To summaries, XSS attacks continue to pose serious dangers to web application security, as indicated by the prevalence and exploitation patterns highlighted in recent papers. Addressing XSS vulnerabilities necessitates a holistic approach that includes strong coding principles, regular upgrades, and constant monitoring. By adopting preventive measures and remaining informed about evolving threats, organizations can successfully manage the dangers posed by XSS attacks and protect the integrity of their web applications.

2.3 WEB APPLICATION SECURITY

2.3.1 Introduction and Background to Web Application Security

Web applications are the best Internet-based solution to provide online web services. However, they also bring significant security challenges. Due to the extensive use of websites and web applications, web vulnerabilities are continuously growing. A survey conducted in 2019 found that nine of 10 web applications are vulnerable and that sensitive data breaches are possible on 68% of web applications (Alaoui R. L. & Nfaoui E. H, 2022). Hence, the implementation of robust defenses towards web-based technologies are paramount against malicious attacks. Web applications security involves securing web applications from unauthorized access and malicious attacks that could compromise their functionality, data integrity, and user privacy. The defenses become vital as web applications are widely used for critical operations such as e-commerce and online banking, that could possibly lead to financial losses and reputational damage.

Web applications, by their nature, are vulnerable to different attacks due to flaws in design, development, and implementation. The research focuses on vulnerabilities induced by poor security practices during the software development lifecycle (Rafique S., Humayun M., Gul Z., Abbas A., & Javed H., 2015). These typical vulnerabilities include Cross-Site Scripting (XSS), SQL Injection, and others from the OWASP Top 10, as well as more advanced threats like Cross-Site Request Forgery (CSRF). Each of the risks and vulnerabilities represents a potential entry point for attackers to exploit, which might result in data breaches, unauthorized access, and service disruption.

2.3.2 Web Application Architecture

Web-based applications are essential network solutions for providing standard web services. They are developed using client and server-side approaches, with server-side applications utilizing backend scripting languages like NET, PHP, and JEE, and client-side applications using front-end scripting languages like CSS/HTML and JavaScript. These applications are typically interconnected via HTTP protocol. Web applications have become an integral part of daily life due to their accessibility and convenience.

However, their increased popularity can also lead to attackers compromising critical services in sectors like healthcare, education, banking, and e-commerce (Alaoui R. L. & Nfaoui E. H, 2022). Figure 8 shows the overview of the web architecture.

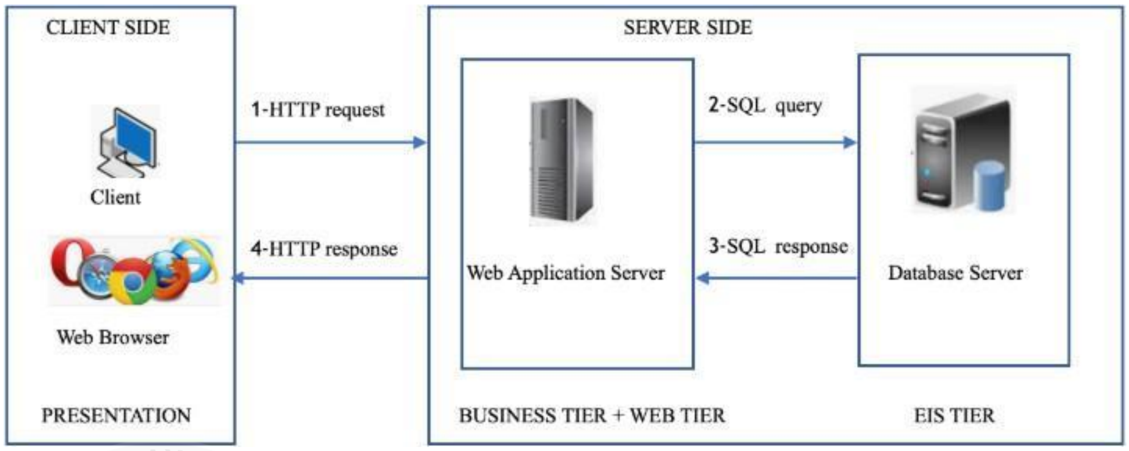


Figure 8: Overview of web architecture

2.3.3 Web Vulnerabilities

Web application vulnerabilities are weaknesses or loopholes that can be exploited to cause attacks. These vulnerabilities can be classified into three categories: improper input validation, which involves incorrect validation and sanitization of user input, leading to SQL injection and Cross-Site Scripting attacks. Improper session management, where web sessions are not secured correctly, can cause Cross-Site Request Forgery (CSRF) and session highjacking. Additionally, improper authorization and authentication vulnerabilities involve logic flaws in access control policies and authentication functions, resulting in broken access control and potential web attacks.

These vulnerabilities can lead to SQL injection, XSS, Cross-Site Request Forgery, and session highjacking. Figure 9 shows the types of web vulnerabilities.

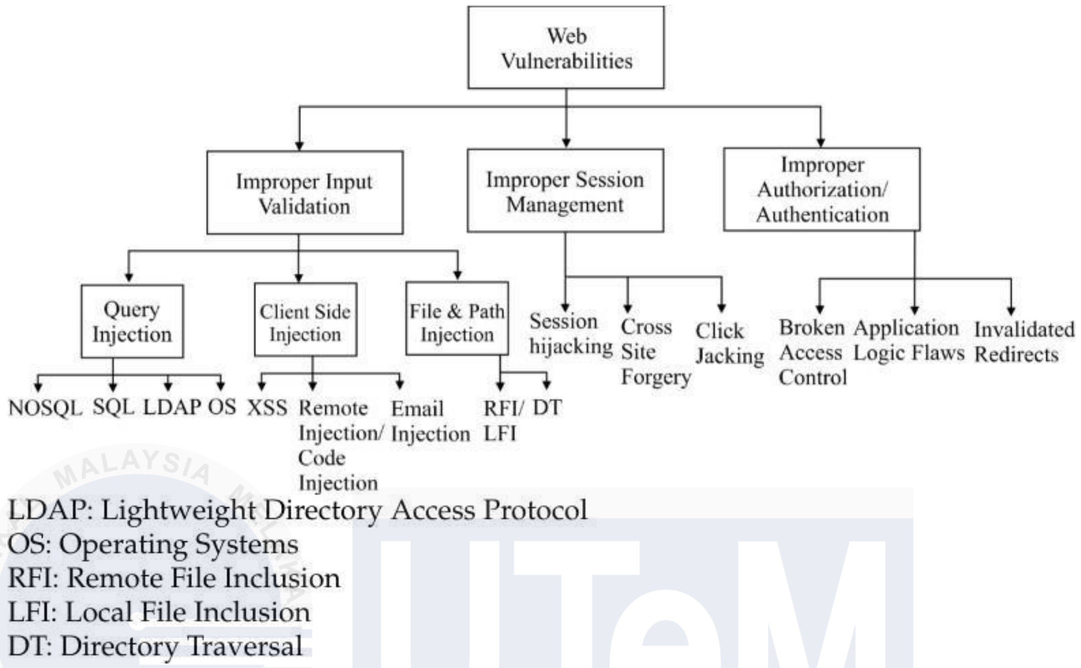


Figure 9: Types of web vulnerabilities

2.3.4 Mitigation

To mitigate these vulnerabilities, several security procedures and techniques are used throughout the web application's lifecycle. These include strict input validation to sanities user inputs, strong authentication systems to validate user identities, granular permission to manage access rights, and encryption to safeguard data at rest and in transit. Collectively, these measures improve the application's resilience to common attack vectors.

Recent advances in machine learning (ML) have shown promise for improving web application security. ML approaches can analyze large volumes of data to find unusual patterns that indicate an attack, providing proactive defense capabilities. However, important obstacles like model interpretability, training data quality, and adversarial assaults stand in the way of properly applying machine learning for web security.

To summarize, while web application security evolves with technological advancements and research breakthroughs, a comprehensive approach that incorporates robust security practices, effective use of tools, and leveraging the potential of machine learning remains critical in protecting web applications from emerging threats.

2.4 MACHINE LEARNING

2.4.1 Introduction to Machine Learning

Machine learning (ML) is a subset of artificial intelligence (AI) that allows computers to learn from data and improve performance over time without being explicitly programmed. Its significance across domains stems from its ability to analyze vast amounts of data, discover significant patterns, and generate data-driven predictions or judgements. Machine learning algorithms are learned by analyzing enormous volumes of data to detect patterns and make predictions or judgements. Rather than relying on explicit programming instructions, they iteratively enhance their task performance through experience. Key strategies include supervised learning, unsupervised learning, and reinforcement learning.

In the context for the project, machine learning (ML) to detect XSS attacks by identifying patterns in web page features that distinguish between benign and malicious information. Improving web security through automated detection is significant, as it reduces reliance on manual rule-based systems, which may overlook sophisticated threats.

2.4.2 Types of Machine Learning Techniques

Machine Learning algorithms are mainly divided into four categories: Supervised learning, Unsupervised learning, Semi-supervised learning, and Reinforcement learning (Sarker I. H., 2021). Figure 10 shows the types of Machine Learning Techniques, classified by their categories.

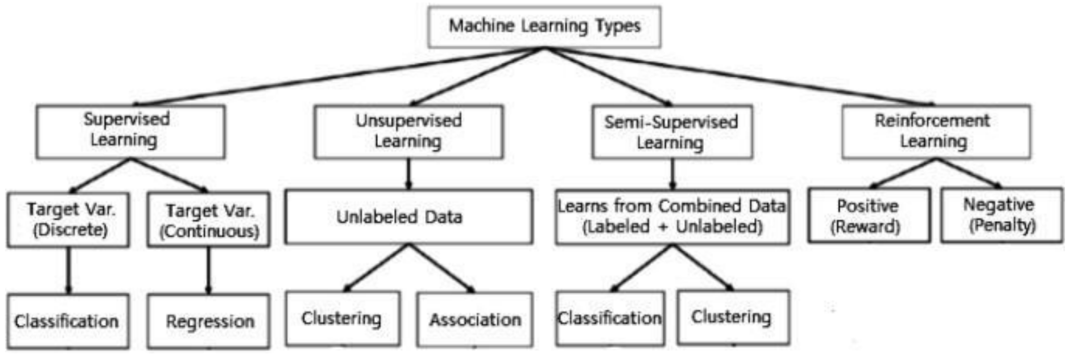


Figure 10: Machine Learning Types

1. Supervised Learning

Supervised learning is a machine learning process that converts input to output using sample input-output pairs. It employs labeled training data and examples to derive a function. Supervised learning is a task-driven strategy that achieves goals using a set of inputs. Common tasks include data separation (classification) and data fitting (regression). Text categorization, for example, predicts the class label or mood of a given piece of text, such as a tweet or product review.

For Classification Algorithms, there are five methods which are Linear Classifier, Support Vector Machines (SVM), Decision trees, k-Nearest Neighbors (k-NN), and Random Forest. Meanwhile, in Regression Algorithms, there are three methods which are Linear Regression, Logistic Regression, and Polynomial Regression.

2. Unsupervised Learning

Unsupervised learning analyses unlabeled datasets without human intervention, resulting in a data-driven approach. This technique is commonly used to extract generative features, find relevant trends and structures, group results, and conduct exploratory analyses. Common unsupervised learning problems include clustering, density estimation, feature learning, and dimensionality reduction. For Clustering Algorithms, there are three methods which are k-means Clustering, Hierarchical Clustering, and DBSCAN (Density-Based Spatial Clustering of Applications with Noise). Meanwhile Association Rule Learning includes Apriori Algorithm which focuses on finding frequent item sets.

3. Semi-supervised Learning

Semi-supervised learning combines supervised and unsupervised methods, operating on labelled and unlabeled data. This falls between learning "without supervision" and learning "with supervision." Semi-supervised learning can be effective in situations where labelled data is few and unlabeled data is abundant.

Semi-supervised learning models aim to improve prediction accuracy compared to labelled data alone. Thus, semi-supervised learning has applications such as machine translation, fraud detection, data labelling, and text classification.

4. Reinforcement Learning

Reinforcement learning is a machine learning method that automatically evaluates optimal behavior in a specific setting to increase efficiency. It is an environment-driven technique. This sort of learning, based on reward or punishment, aims to use environmental activists' insights to maximize benefit or reduce danger. This tool is effective for training AI models to improve automation and efficiency in complex systems like robotics, autonomous driving, manufacturing, and supply chain logistics. However, it is not suitable for solving simple problems.

These methods include Q-learning, SARSA (State-Action-Reward-Sate-Action), and Neural Networks.

2.4.3 Machine Learning Classifiers

1. Support Vector Machines (SVM) Classifier

Support vector machine (SVM) is a common technique in machine learning for classification, regression, and other tasks. SVM area unit the foremost strong prediction strategies supported applied math learning framework. It constructs a hyper-plane or set of hyper-planes in high- or infinite-dimensional space, achieving strong separation based on the greatest distance from the nearest training data points. SVM is effective in high-dimensional spaces and can behave differently based on different mathematical functions called kernels. Popular kernel functions include linear, polynomial, radial basis function (RBF), and sigmoid. However, SVM may not perform well when data sets contain more noise.

2. Decision Trees Classifier

Decision tree (DT) is a renowned non-parametric supervised learning method used for classification and regression tasks. It is based on ID3, C4.5, and CART algorithms. Recently, BehavDT and IntrudTree by Sarker et al. are effective in user behavior analytics and cybersecurity analytics. DT classifies instances by sorting down the tree from the root to leaf nodes, checking the attribute defined by that node. The most popular criteria for splitting are "gini" for Gini impurity and "entropy" for the mathematical information gain as Figure 11. Meanwhile for Figure 12, it shows the example of decision tree structure.

$$\text{Entropy}\ :\ H(x) = -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i)$$

$$\text{Gini}(E) = 1 - \sum_{i=1}^{c} p_i^2.$$

Figure 11: the calculation for Entropy and Gini

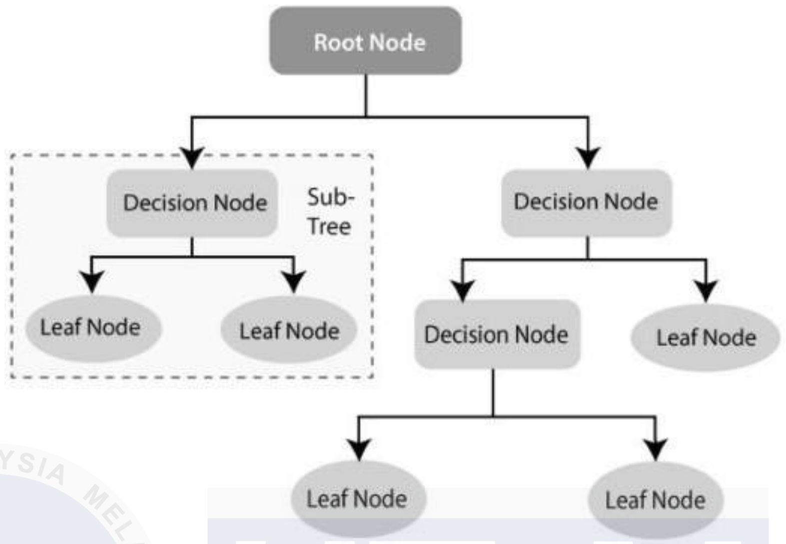


Figure 12: Example of a decision tree structure

3. k-Nearest Neighbors (k-NN) Classifier

K-Nearest Neighbours (KNN) is a type of "instance-based learning" or "lazy learning" method that does not generalize. The method maintains training data instances in n- dimensional space rather than creating a broad internal model. KNN classifies new data points using similarity metrics, such as the Euclidean distance function. The Classification is based on a simple majority vote among the k Nearest Neighbours of each point. Figure 13 and Figure 14 show calculation for the gap metric and calculation to gets x appoints to the category with the largest likelihood. It is relatively resistant to noisy training data, and accuracy is dependent on data quality. The main challenge with KNN is determining the ideal number of neighbors to consider. KNN can be used both for classification as well as regression. (Sarker I. H., 2021).

$$d(x,x') = \sqrt{(x_1 - x'_1)^2 + \ldots + (x_n - x'_n)^2}$$

Figure 13: Calculation for the gap metric by Euclidean metric

$$P(y=j \mid X=x) = 1/K \sum I(y^i = j)$$

Figure 14: x gets appointed to the category with the largest likelihood calculation

4. Random Forest Classifier

A random forest classifier is an ensemble classification technique used in machine learning and data science for various applications. It uses "parallel ensembling" to fit multiple decision tree classes in parallel on different dataset sub-samples, minimizing overfitting and increasing prediction accuracy and control. This approach is typically more accurate than a single decision tree-based model. The method combines bootstrap aggregation (bagging) and random feature selection to build a series of decision trees with controlled variation. It is adaptable to both classification and regression problems and works well for both categorical and continuous values. The RF learning model with multiple decision trees is typically more accurate than a single decision tree-based model. Figure 15 illustrates the Random Forest Classifier Structure.

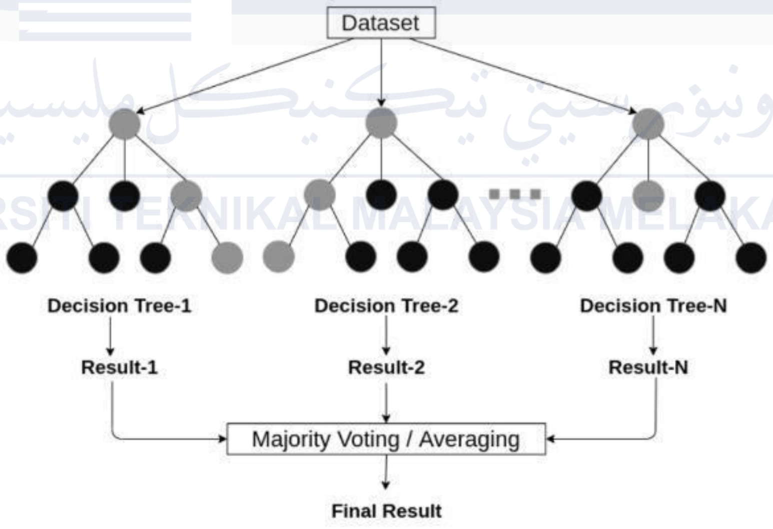


Figure 15: Random Forest structure

2.5 FEATURE SELECTION

Processing high-dimensional data in machine learning and data science is a hard issue for researchers and developers. Dimensionality reduction, an unsupervised learning technique, improves human interpretations, reduces computational costs, and simplifies models to prevent overfitting and redundancy (Sarker I. H., 2021).

2.5.1 Feature Selection Background

Feature selection, also known as variable attribute selection in data, is the process of selecting a subset of unique features (variables, predictors) for use in machine learning and data science models. The feature selection process aims to decrease the dimensionality, computational cost, training time of the prediction model, extreme information loss, and overfitting, which by selecting the optimal relevant features subset and ignoring the redundant or irrelevant features. (Chandrashekar & Sahin, 2014) (Thajeel I. K., Hshim F., Samsudin K., & Hashim S.J., 2023).

2.5.2 Classification of Feature Selection Techniques

1. Selection-based knowledge

Selection-based knowledge includes selecting features based on expertise in the field rather than using formal feature selection procedures to determine the importance of each feature. This method can result in biassed selection, perhaps ignoring the most important elements for detecting XSS attacks. It frequently results in a narrow feature set that may not capture all attack characteristics and lacks generalizability, as the chosen features may be exclusive to a certain dataset or data type. Furthermore, this strategy might be time-consuming when working with many features, making it ineffective.

Example of Selection-based Methods:

1. **Manual selection based on Domain Expertise**

   : Experts carefully select relevant features based on their understanding of the problem domain. This strategy is strongly reliant on expert judgement and may require iterative refinement based on trial and error.

2. **Expert-driven feature engineering**

: Domain specialists design new features or change existing ones based on their knowledge of the data and problem domain. This can include modifying variables, generating interaction terms, or deriving new features from raw data.

2. Filtering method

Filtering methods assess each attribute individually and assign scores or rankings using criteria such as correlation coefficient, Information Gain (IG), Chi-square test, and document frequency. These strategies exclude features that do not fit the criterion, hence decreasing the feature set. For example, the correlation coefficient measures the strength of the linear relationship between features and the target variable, IG measures entropy reduction, the Chi-square test assesses categorical variable dependence, and document frequency measures term occurrence in documents. Filtering approaches, on the other hand, have the potential to miss significant feature interactions because they evaluate features independently and may not perform well with huge feature sets. Figure 16 shows the Filter Methods Implementation.

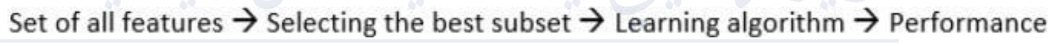

Set of all features → Selecting the best subset → Learning algorithm → Performance

Figure 16: Filter Methods Implementation

Example of Filtering Methods:

1. **Correlation Coefficients**

   : Determines the degree and direction of a linear relationship between two variables. During feature selection, features having low correlation coefficients with the target variable may be excluded.

2. **Information Gain (IG)**

   : Measures the reduction in entropy (uncertainty) of the target variable when a feature is known. The features that do the most to reduce ambiguity about the objective are kept.

3. **Chi-square Test**

: Assesses the independence of categorical variables. Feature selection discovers features that are significantly related to the target variable.

4. **Document Frequency (DF)**

: Counts how many times a term appears in a set of documents. It is used in text mining to exclude terms that are too uncommon or too prevalent.

3. Wrapper method

Wrapper approaches analyze subsets of features by training and testing the model on each subset to determine the most relevant characteristics. Although this strategy is computationally expensive due to the requirement to train and test the model several times, it is frequently more effective than other methods since it considers feature interactions and their impact on model performance. For example, the most significant features were selected using Sequential Backward Selection (SBS) in conjunction with IG. Despite their effectiveness, wrapper approaches' extensive search procedure has large processing costs. Figure 17 illustrates the Wrapper Methods Implementation.

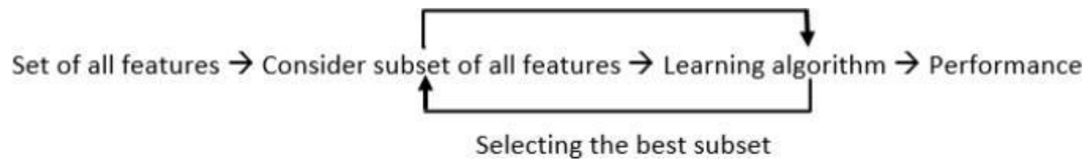

Set of all features → Consider subset of all features → Learning algorithm → Performance
Selecting the best subset

Figure 17: Wrapper Methods Implementation

Examples of Wrapper Methods:

1. **Recursive Feature Elimination (RFE)**

: Selects features by recursively considering smaller and smaller sets of features and rating them in order of significance. It trains the model repeatedly, removing the least significant characteristics until the desired number of features is achieved.

2. **Sequential Feature Selection (SFS)**

: Feature subsets are evaluated by adding features sequentially and analyzing the impact on model performance. It begins with an empty collection of features and adds the highest-performing feature at each phase.

3. **Genetic Algorithm for feature selection**

: The best collection of features is selected using evolutionary algorithms influenced by natural selection. It creates a population of candidate feature subsets, assesses their fitness using model performance, and develops the population over time to optimize feature selection.

4. **Embedded method**

Embedded approaches embed feature selection into the model training process, so it is part of the model creation rather than a separate pre-processing phase. Examples include Multi-Objective Evolutionary Feature Selection (MOEFS), which selects the fewest number of features, and approaches like CfsSubsetEval, which uses greedy search and reinforcement learning algorithms to choose features based on classification error rates. While embedded approaches can be effective and consider feature interactions, they are sensitive to model and hyperparameter selection, and feature selection outcomes are not always immediately interpretable. Figure 18 shows the Embedded Method Implementation.

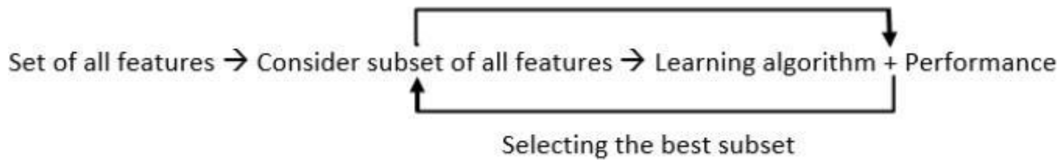

Set of all features
Consider subset of all features
Learning algorithm + Performance
Selecting the best subset

Figure 18: Embedded Methods Implementation

Example of Embedded Methods:

1. **LASSO (Least Absolute Shrinkage and Selection Operator)**

   : Adds a penalty proportionate to the absolute value of the coefficients, reducing less significant values to zero. Features with nonzero coefficients are chosen.

2. **Decision Trees**

   : Creates a tree structure by recursively separating data based on characteristics, with splits selected to maximize the homogeneity of the target variable within each subset. Features at higher levels of the tree are usually more important.

3. **Random Forest**

   : Decision tree ensembles are formed by training each tree on a bootstrap sample of data and a random selection of features. Feature significance is determined by how much each feature reduces impurity across all trees.

4. **Gradient Boosting Machines (GBM)**

   : Sequentially builds an ensemble of weak learners (typically decision trees), with each subsequent model fixing the preceding one's faults. Feature significance is determined by how frequently features are employed during boosting iterations.

5. **Attention model**

Attention models use weighted features to focus on the most relevant sections of the incoming data. They are particularly useful in Natural Language Processing (NLP) applications, where they are frequently used with techniques like as word2vec, Long Short-Term Memory (LSTM), and Bidirectional LSTM (Bi-LSTM) networks to model sequential data. Attention mechanisms solve the difficulty of collecting long-term dependencies in data by allowing the network to focus on the most relevant sections of the input sequence at each time step. Attention mechanisms have been shown to be effective in modelling language data in previous studies. However, the usefulness of attention models for feature selection in non-NLP domains is not well documented.

Example of Attention Model Methods:

1. Self-Attention Mechanisms (used in Transformer models)

: Computes attention scores for all pairs of words in a sequence, considering word dependencies regardless of location. It provides weights to words depending on their relationship to one another.

2. Attention Layers in recurrent Neural Networks (RNNs), LSTM, and Bi-LSTM

: Introduces attention mechanisms in recurrent neural networks for focusing on relevant parts of the input stream. It applies weights to different time steps or words based on their predictive value.

2.6 DATASET

Dataset in general is a structured collection of data points or observations. It serves as the foundation for various data-driven tasks, including machine learning, statistical analysis, and research. Meanwhile, in the context of Cross-Site Scripting (XSS) attacks detection, a dataset specifically refers to a collection of labeled examples used for training and evaluating machine learning models. Datasets in XSS attack detection are achieved and extracted from various sources like web pages, in the form of HTML tags, JavaScript code, and other myriad contextual information.

2.6.1 Labeling, Training, and Evaluation

Each instance in dataset is labeled, either as malicious (indicating an XSS attack) or benign (safe). This labeling is part of the process to ensure that the machine learning model learns to distinguish between harmful and harmless content. All the datasets will eventually split into two subsets, which into training set and a test set. This training set is to help machine learning recognize patterns that are associated with XSS attacks. This process is being evaluated by using performance metrics such as measuring accuracy, precision, recall, and other metrics.

2.6.2 Features/Attributes

Various XSS datasets have been found in various research papers and articles. Hence, the common datasets include HTML tags, Contextual Information, JavaScript code, and others.

2.6.3 Dataset sources

The paper ‘Detection and Prevention Techniques for cross-site scripting attacks: A Comprehensive review’ by Thajeel I. K., Samsudin K., Hashim S.J., & Hashim F., 2023’ provide a comprehensive review of several approaches for detecting and preventing XSS attack. Here are some of the most utilized datasets linked with XSS attacks:

1. CIC-ODS 2017 Dataset

This well-known and comprehensive dataset, compiled by the Canadian Institute for Cybersecurity (CIC), contains a variety of cyber threat data analysis, including XSS attacks. This dataset has been used extensively in cybersecurity research to assess intrusion detection systems and machine learning models. CIC-IDS 2017 serves as a benchmark dataset for evaluating IDS and machine learning models across different types of cyber threats, including XSS attacks.

Features/Attributes:

1. Network Traffic Metadata
   : This category includes details like source IP, destination IP, and port numbers. These attributes provide context about the communication flow.

2. Payloads

: This category is extracted from HTTP requests and responses. Payloads contain the actual content transmitted between client and server. Analyzing these payloads helps identify potential XSS attacks.

3. Attack labels, which indicate whether an instance is an XSS attacks or not

   : Each instance is labeled and either an XSS attack or benign. These labels serve as ground truth for training and evaluating machine learning models.

2. XSSD Dataset

This dataset is a curated dataset focused on XSS attacks, created to help researchers in this field. This dataset was used to train and evaluate machine learning models that detect XSS assaults. XSSD is often used alongside other cybersecurity datasets but is tailored specifically for XSS attacks, offering detailed examples that capture the nuances of XSS payloads and attack vectors.

Features/Attributes:

1. URL and Parameter Details

   : This category includes domain and path information, providing context about web requests.

2. HTTP Request and Response Headers

   : This header contains essential metadata exchanged during communications.

3. Content

   : HTML Extracted from web pages or scripts, this content helps identify potential vulnerabilities.

4. JavaScript code snippets

   : These snippets play a critical role in detecting XSS attacks.

3. Wooyun-Email-XSS-Dataset

This dataset included a sample of XSS attacks retrieved from email messages. This contains real-world instances of XSS assaults that originate from email sources, which may be utilized for analysis and model training. While specific to emails, it complements other datasets by offering insights into XSS propagation via different communication channels.

Features/Attributes:

1. Email headers

   : These include sender information, receiver details, and the subject of the email. Headers provide context about the communication and potential security risks.

2. Email body content

   : The dataset captures both HTML and text parts of email bodies. Analyzing these content segments helps identify any malicious payloads.

3. XSS payload examples extracted from emails

   : Extracted from actual emails, these examples represent real-world attack scenarios. They serve as valuable training data for detecting XSS vulnerabilities.

4. Kaggle XSS Dataset

Kaggle offers a dataset focused on XSS attacks that is frequently used by researchers and practitioners to design and test detection methods. This dataset allows for a comparative comparison of various detection approaches and algorithms. While specific to Kaggle, it aligns with broader cybersecurity datasets and provides a diverse set of XSS attack instances for training and evaluation.

Features/Attributes:

1. URLs and Associated Parameters

: These features capture information about web addresses (URLs) and any parameters associated with them. URLs play a crucial role in understanding the context of web requests.

2. HTTP Request and Response Headers

: These headers contain metadata exchanged during communication between clients (browsers) and servers. Analyzing headers helps identify potential security risks.

3. HTML Content and JavaScript Snippet

: The dataset includes actual HTML content from web pages and snippets of JavaScript code. These features allow for deeper analysis of potential vulnerabilities.

4. XSS Payload Examples

: The dataset contains examples of XSS payloads with varying levels of complexity and obfuscation. These real-world attack scenarios serve as valuable training data for detecting XSS vulnerabilities.

5. Personalized Datasets from Researchers

Many researchers develop their own datasets based on their experiments and research objectives. These datasets frequently include specialized samples or focus on specific features of XSS assaults that are of interest to researchers. Personalized datasets complement existing public datasets by providing tailored insights and addressing niche aspects of XSS attacks.

Features/Attributes:

1. Enhanced Metadata on HTTP Traffic and Payload Characteristics

   : Researchers collect detailed information about network traffic, including source IPs, destination IPs, ports, and payload sizes. Understanding traffic patterns aids in identifying anomalies or potential security threats.

2. Contextual Information Specific to the Researcher's Experimental Setup

   : This context may involve specific web applications, user behaviors, or environmental factors. Researchers tailor their datasets to match the conditions relevant to their experiments.

3. Fine-Grained Annotations and Label for Detailed Analysis

   : Labels go beyond binary (e.g., attack vs. benign) and provide nuanced information. Annotations may include severity levels, attack types, or specific vulnerability details.

6. Other publicly Available Datasets

Various other datasets are available on platforms such as GitHub, research repositories, and cybersecurity forums. These datasets help to increase the collective knowledge and understanding of XSS attack patterns and detection strategies. Below are the publicly available well-known datasets that researchers may explore for XSS attack detection:

1. OWASP Top Ten:
   : While not a dataset per se, OWASP provides a list of common web application security risks, including XSS, which can guide feature selection based on known attack vectors and mitigation strategies.

2. VX Vault, PhishTank, XSSed Archive

: These repositories provide real-world examples and statistics related to XSS attacks, offering insights into emerging threats and attack patterns.

3. Research Publications

: Often accompanied by supplementary materials, research papers publish datasets used in experiments, allowing replication and further exploration of findings.

2.7 CONSIDERATION IN DETERMINING FEATURE SELECTION METHODS FROM DATASET ATTRIBUTES

When designing feature selection approaches for XSS attack detection, it is critical to examine a variety of factors to ensure effective model performance and generalizability. Key aspects include dataset features, attack variability, data quality, model requirements, and community insights:

1. Dataset Features

Evaluating the types of characteristics available in a dataset is critical. XSS attacks are detected using features such as HTTP headers, URL structures, and HTML and JavaScript content. Understanding which features are most important and considering feature engineering to create more valuable features can dramatically improve detection capabilities.

2. Attack Variability

XSS attacks can vary greatly in sophistication and obfuscation. To effectively generalize feature selection algorithms, the dataset must include a wide range of XSS payloads and attack scenarios. Additionally, recording contextual information such as session details and user-agent data might aid in distinguishing between legitimate and fraudulent behavior.

3. Data Quality and Labeling

The accuracy of feature selection and model training depends on the quality and reliability of the dataset's labels or annotations. To reduce biases and optimize performance across both groups, imbalances in the dataset must be addressed, such as fewer XSS attack occurrences than benign traffic.

4. Model Requirements and Constraints

Consider the computational cost of feature selection approaches. Large datasets are best suited to computationally efficient techniques such as filtering. Model interpretability is also vital, therefore utilizing feature selection methods that provide information on feature importance and contributions to model predictions.

5. Experimental Validation and Benchmarking

Validate feature selection approaches with rigorous experimentation and comparison to known measures like precision, recall, and F1-score. Ensuring that selected features generalize effectively across XSS attack datasets and real-world scenarios improves model adaptability to a variety of contexts.

6. Community Insights and Best Practices

Use insights from cybersecurity community forums, research articles, and collaborative repositories such as Kaggle and GitHub. Community input can help identify beneficial methods and potential hazards, driving improved feature selection decisions.

Conclusion

Effective feature selection for detecting XSS attacks requires a thorough understanding of dataset properties, attack variability, data quality, and model specifications. By systematically examining these factors, researchers and developers can use robust feature selection approaches to increase the accuracy and reliability of XSS detection systems in real-world applications.

2.8 RELATED WORK/PREVIOUS WORK

2.8.1 Detection of XSS in web applications using Machine Learning Classifiers

This paper has been focused on detecting Cross-Site Scripting (XSS) attacks in web applications using machine learning classifiers. Cross-site scripting is known for the attack, in which malicious scripts are injected into trusted websites for illegal purposes like stealing cookies, impersonating users, and more. As the paper aims to develop an effective yet comprehensive method in identifying XSS attacks, both types of web pages, which are benign and malicious, are being operated throughout this study.

Machine Learning Classifier used in this paper includes Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Radom Forest, and Logistic Regression. In the meantime, a set of features selected were based on their relevance to XSS attacks, such as the presence of malicious JavaScript, doubled characters, special characters, server- side validation, content-length, and number of keywords.

The dataset comprised both benign and malicious web pages, with 70% used for training and 30% for testing in the first trial. A second experiment is being held using tenfold cross-validation to reduce sampling bias. Alongside this experiment, all the classifiers are being evaluated using performance metrics such as recall, precision, accuracy, and the F-measure. Thus, the Random Forest Classifier has the highest accuracy of 98%, with a recall of 0.99 and an F-measure of 0.955, showing greater effectiveness in detecting XSS attacks.

Despite the outstanding accuracy, the study recognizes opportunities for development by including more advanced features and investigating further machine learning and deep learning methods. While all classifiers performed outstandingly, with over 90% accuracy, the Random Forest classifier stood out as the most successful. The findings indicate that machine learning techniques can considerably improve online application security by correctly identifying and mitigating XSS threats.

In a nutshell, the study shows that machine learning classifiers, notably Random Forest, can detect XSS attacks with high accuracy. This method can be used in real time to detect dangerous scripts or URLs, helping to improve web application security. Future study will focus on increasing feature representation and developing more complex algorithms to improve detection accuracy and precision.

2.8.2 Machine Learning based Cross-site Scripting Detection in Online Social Network

The primary purpose of this study is to establish a machine learning-based method for detecting cross-site scripting (XSS) attacks in online social networks (OSNs). Considering the danger of XSS attacks towards OSNs, which are caused by user-generated content, the study concentrated on recognizing and classifying essential webpage elements to detect malicious content effectively. The study consists of processes such as extracting features, creating classification models, and assessing their performance.

There are two types of feature extraction that had been used in this study which are similarity-based features (keywords, JavaScript, HTML tags, and URLs) and difference-based features (which use OSN network topology to measure suspicious data distribution). The dataset consists of 29,046 benign samples from the DMOZ database and 21,998 malicious samples from the XSSed database and weibo.com, which represented XSS worm cases in OSNs.

Two machine learning algorithms are being used to create the classification models; ADTree and AdaBoost.M1. Performance metrics used are quite similar to the first paper, which are recall, recall, and F-measures. In result, both the algorithm resulting in high accuracy; ADTree is 0.938, meanwhile AdaBosst.M1 is 0.941. Both similarity-based and difference-based characteristics were useful, particularly in simulated OSN situations.

Despite the positive findings, the study admits limitations, such as the dependence for real-world testing and real-time detection capabilities. Future research directions include adding new features, increasing real-time identification, and investigating different machine learning algorithms. The study indicates that machine learning can greatly improve XSS detection in OSNs, indicating the feasibility of the proposed approach.

2.8.3 The Future of Web Security: XSS Detection through Machine Learning

This paper examines the growing importance of mitigating cross-site scripting (XSS) attacks in web applications using machine learning algorithm techniques. XSS attacks represent substantial concerns since they inject malicious scripts into web pages, jeopardizing user data and application integrity. The project intends to improve web security by utilizing machine learning algorithms such as neural networks, decision trees, SVM, random forests, naive Bayes, KNN, and logistic regression. These algorithms analyze information retrieved from URLs and JavaScript code, allowing for the detection of XSS attack patterns.

The study relies heavily on feature selection, which focuses on extracting significant characteristics from web content to determine between genuine and malicious inputs. While the precise datasets utilized for training and evaluation are not explicitly stated, the study emphasizes the ML models' efficiency in detecting XSS vulnerabilities. The performance of these models is assessed using evaluation metrics such as accuracy, precision, recall, and even AUC.

However, the study notices challenge such as the difficulty in responding to new XSS attack approaches, inherent false positive/negative rates, and the necessity for ongoing upgrades to retain effectiveness against developing threats. Future research directions suggest using AI and machine learning to construct more adaptable and preemptive web security measures. This includes improving natural language processing for better script analysis, safeguarding IoT devices with ML-driven defenses, utilizing blockchain for greater data integrity, and investigating quantum computing's potential for real-time threat detection.

Lastly, the study emphasizes the importance of machine learning (ML) in protecting web applications from XSS attacks, as well as ongoing efforts to develop proactive defense systems. By leveraging ML's adaptive capabilities and integrating with upcoming technologies, researchers hope to create a robust cybersecurity framework capable of efficiently tackling current and future concerns.

2.8.4 Detecting Cross-Site Scripting Attack using Machine Learning Algorithms

This paper starts by utilizing several supervised learning models to detect Cross-site Scripting (XSS) attacks. The models or methods consist of Decision Tree Classifier, Random Forest Classifier, Support Vector Machine (SVM), K-Nearest Neighbor (KNN), and Naive Bayes Classifier. The Decision Tree and Random Forest Classifier had the best result in the performance metrics which with highest precision, recall, and F1- scores, with just 19 and 12 misclassified samples, respectively. They also achieved precision of 99.905% and 99.940% respectively.

The whole performance of these models was assessed using various metrics, including accuracy, precision, recall, F! Score, mean absolute error, and root mean squared error. While the Decision Tree and Random Forest models performed admirably, the SVM and KNN models did equally well though they have slightly higher misclassification rates. However, the Naive Bayes Classifier has much poorer performance metrics, illustrating limitations in reliably detecting XSS attacks.

Future Work and Recommendations The study recommends future research directions, such as using sophisticated variations of Naive Bayes or ensemble techniques like stacking and boosting to increase classification performance. There is also interest in designing other Document Object Model (DOM)-based behaviors to identify other sorts of injection issues, such as SQL injection. Furthermore, the study looks at the possibilities of soft computing approaches like evolutionary algorithms and neural networks for future implementation tactics.

In conclusion, the study highlights the significance of supervised machine learning techniques in detecting XSS threats. Among the models tested, the Decision Tree classifier demonstrated the highest accuracy, highlighting machine learning's potential for improving web application security against XSS attacks.

2.8.5 A Detailed Survey on Recent XSS Web-Attacks machine Learning Detection Techniques

This study provides a comprehensive survey of machine learning methods employed in detecting Cross-site Scripting (XSS) attacks on web applications. This research aims to analyze the efficiency of various algorithms and approaches in minimizing these vulnerabilities. The methods involve thorough literature research, an examination of metaheuristic algorithms such as Genetic Algorithms mixed with machine learning, and a comparison of various algorithms and feature selection strategies.

The results reveal detection accuracies of up to 95.4% with minimal false positive rates, indicating encouraging results for XSS detection models. Despite these gains, the report illustrates many limits. Challenges include adjusting models to new and diversified XSS attack types, restricted access to extensive datasets, and the possibility of large false positive rates in some detection systems.

Main Algorithms and Features J48 Decision Tree, Naive Bayes, and metaheuristic algorithms such as Particle Swarm Optimization are among the key algorithms covered, with each being evaluated on measures such as accuracy, recall, and precision. The importance of features is stressed, emphasizing the important function of JavaScript events, URL patterns, and HTML elements in identifying XSS vulnerabilities.

Ahead to the future, the study proposes new research directions, such as investigating semi-supervised and deep learning algorithms to improve detection accuracy. It suggests further research into hybrid solutions that combine metaheuristics and machine learning to better flexibility and resilience to developing XSS attack strategies. In conclusion, while great progress has been made, further research is required to develop more effective and resilient XSS detection systems to protect online applications against malicious exploitation.

2.8.6 Multiclass Classification of XSS Web Page Attack using Machine Learning Techniques

This study “Multiclass Classification of XSS Web page Attack using Machine Learning Techniques” addresses the critical challenge of detecting XSS attacks on web pages using machine learning approaches. The primary objective is to identify whether web sites are benign or malicious by extracting information from both web documents and URLs. To do this, the study uses three classifiers: Naive Bayes, Decision Tree, and Multi-Layer Perceptron, and evaluates their performance using 10-fold cross-validation.

Results illustrate that MLP and Decision Tree classifiers beat Naive Bayes, with MLP obtaining high accuracy rates of 96.20%. Performance parameters such as accuracy, model building time, and error measurements such as mean absolute error and root mean squared error are used to evaluate classifier efficacy, demonstrating MLP's ability in handling XSS classification jobs. However, the study acknowledges many limitations, including potential constraints owing to dataset size and the specificity of the XSS attack types evaluated, which may affect the findings' generalizability.

In terms of algorithm selection, Naive Bayes, Decision Tree, and Multi-Layer Perceptron were chosen for their practicality for multiclass classification tasks and previous success in similar investigations. The report suggests future research approaches that focus on investigating fresh traits and assault kinds to improve detection skills. It also advises that studies be expanded to include more diverse datasets to increase accuracy and efficiency.

Overall, the study emphasizes the importance of feature extraction from web pages and URLs in improving XSS attack detection approaches. It provides useful insights into increasing web security using machine learning approaches, highlighting their potential for reducing XSS assaults.

2.8.7 Comparison of machine learning techniques for detecting malicious webpages

The study evaluates various machine learning algorithms for detecting potentially harmful websites, with a focus on their ability to distinguish between valid and malicious URLs or pages. It addresses the major cybersecurity dilemma by analyzing algorithms like SVM, decision trees, and neural networks. The study focuses on feature- rich datasets that include textual, structural, and visual features retrieved from online pages, which are critical for accurately modelling hazardous behaviors.

The models' efficiency in identifying malicious actions is evaluated using performance metrics such as accuracy, precision, recall, and F1-score. Cross-validation procedures are used to guarantee that the results are resilient by evaluating the models against several subsets of the dataset. Despite the hopeful results, the study admits some limitations, including potential biases in the dataset, which may affect the generalizability of the findings.

Furthermore, the interpretability of the models and the difficulty of describing feature importance are emphasized as issues that require further investigation. The conclusion emphasizes the importance of the findings for improving cybersecurity measures and offers future research paths, such as developing more interpretable models and incorporating emerging threat detection approaches.

Finally, the study provides useful insights into the use of machine learning to detect dangerous websites, including a comparison of several algorithms and datasets. It emphasizes the necessity for ongoing research to solve the stated shortcomings and increase the field's ability to successfully combat growing cybersecurity threats.

2.8.8 Summarization of papers

Based on the papers above, it is crystal clear that machine learning techniques, such as Decision Trees, Random Forests, Support Vector Machines, K-Nearest Neighbors, and Naive Bayes, are crucial in detecting Cross-Site Scripting (XSS) attacks. Random Forest method or classifier surprisedly had resulted for the highest accuracy in the detection of XSS attack, that been highlighted in most of the paper, which are 99.940% of the precision and lowest number of misclassifications. However, challenges like adapting models to diverse attack types, limited dataset availability, and high false positive rates persist. The papers emphasize the need for ongoing research to improve accuracy, efficiency, and adaptability of these models to evolving cybersecurity threats. Future directions include exploring semi-supervised and deep learning approaches, developing interpretable models, and incorporating emerging threat detection techniques.

2.9 CRITICAL VIEW ON CURRENT PROBLEM AND JUSTIFICATION

Table 5 shows some previous works that have been carried out by various researchers focusing on detecting Cross-Site Scripting (XSS) Attack using various methods in Machine Learning. These works have been categorized according to their methodologies, techniques, parameters/attributes, and software/hardware.

Table 5: Previous work

No	Paper	Methodologies	Techniques	Attributes	Software/Hardware
1	Detection of XSS in web applications using Machine Learning Classifiers by Raima Banerjee, Aritra Baksi, Nidhi Singh, Soham Kanti Bishnu (2020)	1. Text Mining-Based Approach 2. Client-Side Detection 3. Structures Supported Normal and Malicious JavaScript Code	1. Machine Learning classifiers (SVM, KNN, Random Forest, and Logistic Regression) 2. Feature Extraction 3. Dataset Extraction	1. URLs Features 2. JavaScript Features	1. Python 3.7.4 2. Sci-kit
2	Machine Learning based Cross-site Scripting Detection in Online Social Network by Rui Wang, Xiaoqi Jia, Qinlei Li,	1.Feature Extraction 2. Webpage Collection 3. Classification Model Building	1. ADTree and AdaBoost algorithms 2. 10-fold Cross-Validation Feature Grouping	1. Keyword Features 2. JavaScript Features 3. HTML Tag Features 4. URL Features	1. Weka (data mining toolkit)

	Shengzhi Zhang (2014)				
3	The Future of Web Security: XSS Detection through Machine Learning by Ritika Bansal (2023)	1. Supervised Learning Algorithms (SVM, neural networks, and decision trees) 2. Ensemble Learning Methods 3. Genetic Algorithms and Reinforcement Learning.	1. Feature Extraction 2. Anomaly Detection 3.Content Security Policy (CSP)	1. URL Features 2. JavaScript code Features User Interaction Patterns	1. Machine Learning Frameworks (TensorFlow, Scikit-learn) 2. Web Security Tools (Burp Suite, OWASP ZAP) 3. Dev Environments (VS Code, PyCharm)
4	Detecting Cross-Site Scripting Attack using Machine Learning Algorithms by S Karthika, G Padmavathi, Roshni A, and S Varshini (2024)	1. Data Collection 2. Data Pre-processing 3. Feature Selection	1. Machine Learning algorithms (Naive Bayes, KNN, Decision trees, Random Forest, SVM) 2. Performance Metrics 3. Data Importing	Datasets consist of 10100 rows and feature attributes of 68 columns	Datasets consist of 10100 rows and feature attributes of 68 columns
5	A Detailed Survey on Recent XSS Web-Attacks machine Learning Detection Techniques by Jasleen Kaur & Dr Urvashi Garg (2021)	1. Data Collection 2. Data Pre-processing 3. Feature Selection	1. Machine Learning algorithms (Naive Bayer, KNN, Decision Trees, Random Forest, and SVM) 2. Performance Metrics 3. Data Importing	Datasets consist of 10100 rows and feature attributes of 68 columns	Datasets: Kaggle Community
6	Multiclass Classification of XSS Web Page Attack using Machine Learning Techniques by S. Krishnaveni & K. Sathiyakumari (2013)	1. Cross-site Scripting (XSS) Attack Detection 2. Feature Extraction 3. Machine Learning Techniques (Naive Bayes, Decision Tree, and MLP)	1. Naive Bayes 2. Decision Tree 3. Multi-Layer Perceptron (MLP)	1. Script-based features 2. Core contents 3. DOM objects	1. Weka (open-source data mining tool)
7	Comparison of machine learning techniques for detecting malicious webpages by	1. Cross-site Scripting (XSS) Attack Detection 2. Feature Extraction 3. Machine Learning	1. Naive Bayes 2. Decision Tree 3. Multi-Layer Perceptron (MLP)	1. Script-based features 2. Core contents 3. DOM objects	1. Weka (open-source data mining tool)

	H.B. Kazemin, S. Ahmed (2014)	Techniques (Naive Bayes, Decision Tree, and MLP)			

2.10 PROPOSE SOLUTION/FURTHER PROJECT

Based on a critical review of existing work in Table 5, this proposal recommends employing the hybrid combinations of Random Forest algorithm and XGBoost to improve the identification of Cross-Site Scripting (XSS) threats. Random Forest, an ensemble learning method that combines multiple decision trees, results in a more accurate and robust model, lowering the danger of overfitting. Its ability to rate the value of various items is critical for detecting XSS, which involves identifying essential qualities like URL patterns, JavaScript features, and HTML tags. Meanwhile XGBoost is known for its high performance and accuracy. XGBoost includes Lasso and Ridge techniques, which will be help in encountering overfitting issues during the training.

For this project, a combination of comprehensive feature selection methods and Random Forest's built-in feature significance will be utilized to select features, ensuring that only the most relevant ones are used for model training. Consideration in choosing the best feature selection method for pairing with selected machine learning classifier is crucial in producing high accuracy of result. Thus, this matter has been properly highlighted in section 2.7.

The Random Forest classifier is justified by its accuracy and robustness, as it effectively handles complex and variable XSS attack patterns while minimizing overfitting. Its feature importance ranking determines the most important qualities for identification, and its capacity to manage imbalanced datasets using bootstrapping and majority voting eliminates bias towards the dominant class. Additionally, Random Forest's scalability makes it suitable for real-world applications with high data volumes, such as online security. Integration with well-known machine learning frameworks such as Scikit-learn and TensorFlow improves its practicality and usefulness.

Combining both methods as model training in the project could resulting in a powerful performance as both offering significant strength in the form of detecting XSS attacks. This jumping stone will help in enhancing the detection accuracy and efficiency, thus providing a comprehensive solution for web security.

2.11 CONCLUSION

In essence, this project advocates for the use of the Random Forest algorithm and XGBoost to improve the detection of Cross-Site Scripting (XSS) threats. Random Forest's precision and robustness, combined with XGBoost's high performance and accuracy, make this hybrid approach an excellent choice for dealing with complicated attack patterns while minimizing overfitting hazards. By integrating extensive feature selection methods with Random Forest's intrinsic feature relevance ranking, we ensure that only relevant features are used to train models. Furthermore, Random Forest's scalability and XGBoost regularization techniques makes it suitable for real-world applications, particularly in online security environments. Its smooth interaction with popular machine learning frameworks like Scikit-learn and TensorFlow makes it even more useful. When implementing this strategy, we prioritize accuracy, relevance, and adaptation to different situations.

CHAPTER 3: METHODOLOGY

3.1 INTRODUCTION

This chapter provides a comprehensive overview of the project methodology, which serves as the backbone for conducting our study. It meticulously outlines the various stages, processes, and strategies that have been strategically designed to ensure the efficient and successful achievement of the project's objectives. To provide a clear and dynamic understanding of the project's progression, this project employs the Agile approach. Renowned for its iterative and flexible design, the Agile approach allows us to tackle complex processes in manageable, incremental tasks. This ensures that each phase of the project is continuously refined based on feedback and learning, thereby maintaining a high standard of adaptability and quality throughout the study.

3.2 PROJECT METHODOLOGY

This image in Figure 19 presents the Agile methodology, a highly effective approach for managing machine learning projects. The Agile methodology is depicted as a circular diagram with six key phases: Plan, Design, Develop, Test, Deploy, and Review. This cyclical process emphasizes the iterative and incremental nature of Agile, making it ideal for the exploratory and experimental nature of machine learning. Each phase represents a critical step in the project lifecycle.

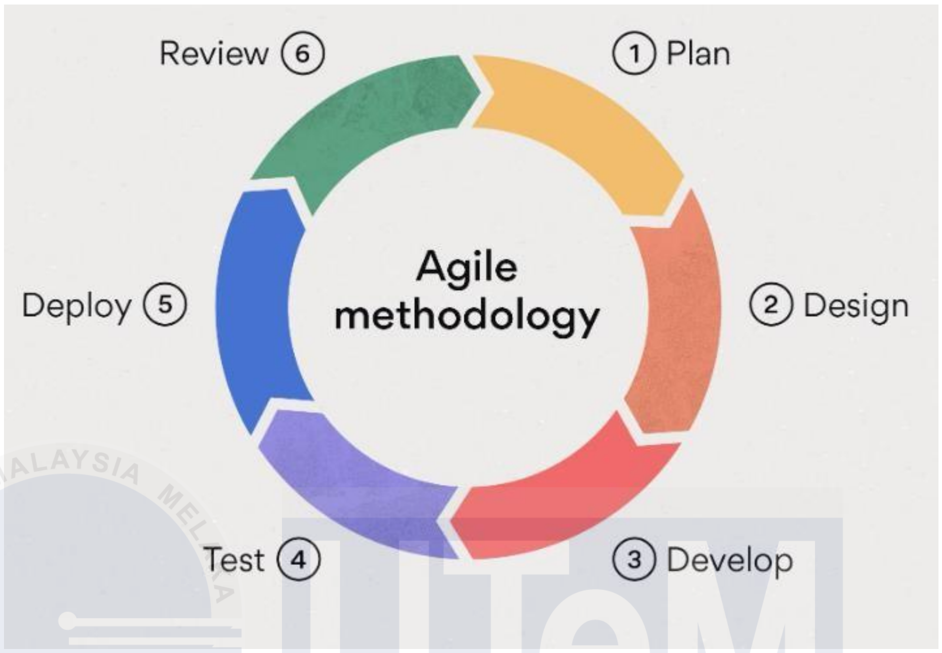


Figure 19: Agile Methodology

3.2.1 Plan

The planning stage is the foundation of the entire project. In the context of the project, this stage involves the critical step of 'Raw Data Collection'. This is where the project meticulously gathers all the necessary data for the project, which could range from logs and user inputs to any other form of data that is relevant to the project.

3.2.2 Design

The design stage is where the 'Feature Collection' and 'Feature Extraction' steps come into play. Here, the process is embarking on identifying and collecting the features from the raw data. This process involves steps like data cleaning, data transformation, upsampling minority, and feature scaling. Furthermore, engagement in the process of feature extraction, which is the reduction of the number of resources required to describe a large set of data accurately. This step is pivotal as it directly impacts the performance of the machine learning model. This model involves experimenting various feature selection methods to enhance and improving the performance of machine learning approach.

3.2.3 Develop

The development stage involves the ‘Feature Selection’ and ‘Data Splitting’ steps. During feature selection, this project strategically chooses the most important features that contribute significantly to the prediction variable or output. Having irrelevant features in the data can decrease the accuracy of the models and make your model learn based on irrelevant features. Once the features are selected, the data is split into “Training data” and “Test data”, setting the stage for the next phase.

3.2.4 Test

The testing stage corresponds to the ‘Model Training’ step. Here, various machine learning classifiers such as Logistic Regression, Random Forest, XGBoost, Multilayer Perceptron (MLP), Support Vector Machine (SVM) are employed to train the model using the training data. This stage ensures that the model is well-equipped to make accurate predictions.

3.2.5 Deploy

The deployment stage involves the ‘Classification’ step. This is the final step where the scripts are classified as either “Malicious/Benign” based on the trained model. This stage marks the transition of the model from the development phase to the real-world application.

3.2.6 Review

The review stage involves evaluating the performance of the model using the test data and refining the model based on the results. This could involve going back to previous stages like ‘Develop’ or ‘Design’ to adjust the features or the model parameters. This stage ensures that the model is continuously improved and adapted to meet the evolving requirements.

3.3 PROJECT MILESTONES

Project milestones as in Table 6 indicate substantial progress or completion of a key deliverable or phase. Milestones are defined during project planning and used to track progress throughout the lifecycle. They establish a defined plan and framework for project execution, ensuring it stays on track and meets objectives. Tables 6 and 7 display the project milestones for final year projects 1 and 2, respectively.

Table 6: Project Milestones

Week	Activity
1	Proposal Discussion Evaluation and verification of the proposal
2	Correction and Improvement of the proposal Confirmation of chosen supervisor
3	Submission of Proposal to ePSM system Beginning of article and research paper collection based on the chosen topic
4	Listing of summarizations of related research papers
5	Discussion on guidelines of Chapter 1 and Chapter 2
6	Chapter 1
7	**MID SEMESTER BREAK**
8	Chapter 2
9	Discussion on project progress and changes on report.
10	Discussion on guidelines of Chapter 3
11	Chapter 3 & Chapter 4
12	Project Demonstration & FYP1 Report
13	Project Demonstration & FYP1 Report
14	**FINAL FYP I PRESENTATION** **Report Submission & Presentation**
15	Make changes to project according to comments from evaluator during the presentation.

16	Chapter 5 and discussion regarding Machine Learning frameworks and detection methods
17	Chapter 6
18	Discussion and modification of Chapter 6
19	Chapter 7
20	Completing and making adjustment of final report
21	Project demonstration and final discussion with supervisor
22	**FINAL FYP II PRESENTATION** **Report Submission & presentation**

3.4 FYP I Gantt Chart

Table 7: FYP I Gantt Chart

No	Activity	Week														
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
1	Meeting with supervisor	🟦														
2	Proposal Improvement		🟦													
3	Proposal submission			🟦												
4	Design the system				🟦											
5	System development				🟦											
6	Implementation of project				🟦	🟦	🟦	🟦	🟦	🟦	🟦					
7	System Testing & Analysis											🟦	🟦			

8	System maintenance												🟦	🟦		
9	Final report preparations													🟦	🟦	
10	Presentation and report submission															🟦

3.5 FYP II Gantt Chart

Table 8: FYP II Gantt Chart

No	Activity	Week						
		16	17	18	19	20	21	22
1	System correction	🟦	🟦	🟦				
2	Meeting with supervisor and project demo			🟦				
3	Report Correction			🟦	🟦			
4	Meeting with supervisor				🟦			
5	Report Correction					🟦		
6	Meeting with supervisor					🟦	🟦	
7	Final report preparations					🟦	🟦	
8	Presentation and report submission							🟦

3.6 CONCLUSION

In conclusion, this chapter presents a full illustration of the project methodology, which precisely implements the Agile approach's six stages: plan, design, develop, test, deploy, and review. Each stage is critical to the project's progress, ensuring a structured and dynamic approach. Furthermore, this chapter includes a visual representation of the project milestones, providing a clear roadmap for the project's progress. It also introduces the planned Gantt chart, a strategic tool for monitoring project progress and ensuring its implementation. This thorough methodology and strategic planning highlight our commitment to achieve the project's objectives efficiently and precisely.

CHAPTER 4: DESIGN

4.1 INTRODUCTION

This chapter covers project problem analysis and gathering information of requirements. The analysis starts with a problem analysis. The analysis refines and clarifies an incomplete or informal statement, revealing ambiguities and inconsistencies. Analyzing the problem and identifying its features is crucial for improving product quality.

4.2 PROBLEM ANALYSIS

Cross-Site Scripting (XSS) attacks are becoming more common and dangerous in today's digital environment. As web applications become more complicated and interconnected, they provide a wider target for these attacks. XSS attacks leverage web application vulnerabilities to inject malicious scripts into web pages visited by unsuspecting visitors. These scripts can access sensitive data such as session cookies, allowing attackers to impersonate users and do activities on their behalf. The ramifications can vary from data theft to web content manipulation, all of which undermine user confidence and security. Despite efforts to counteract these attacks using input validation, output encoding, and security headers, the increasing sophistication of attack vectors remains a serious concern. In the digital age, where online interactions are pervasive, the possibility of XSS attacks emphasizes the crucial importance of strong web application security mechanisms and ongoing monitoring.

4.3 REQUIREMENT ANALYSIS

In developing this project using a comprehensive machine-learning approach, this project proposed a framework as shown in figure 20.

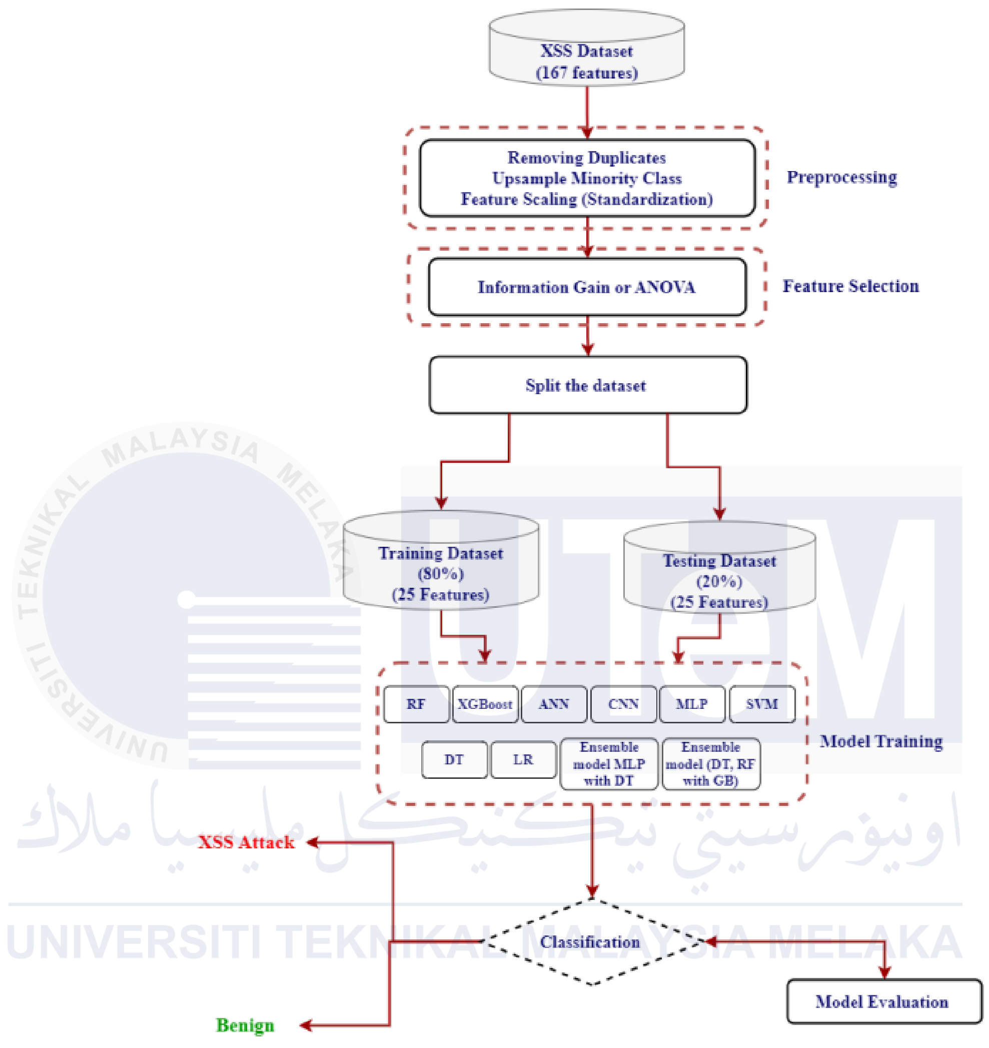


Figure 20: Proposed Framework

From figure 20, the proposed framework for the detection of XSS attack using machine learning consists of seven steps.

1. **Raw Data**: This is the initial stage where all the necessary data for the project is collected. This could be in the form of logs, user input, or any other form of data relevant to your project.
2. **Feature Collection**: In this step, we identify and collect the features from the raw data. This could involve processes like data cleaning, upsampling minority, and feature scaling.

3. **Feature Extraction**: This is the process of reducing the number of resources required to describe a large set of data accurately. This step is crucial as it directly impacts the performance of the machine learning model.

4. **Feature Selection**: This step involves selecting the most important features that contribute to the prediction variable or output in which you are interested. Having irrelevant features in the data can decrease the accuracy of the models and make the model learn based on irrelevant features.

5. **Data Splitting**: Once the features are selected, the data is split into "Training data" and "Test data". The training data is used to train the machine learning model while the test data is used to evaluate the model's performance.

6. **Model Training**: In this step, various machine learning classifiers such as Logistic Regression, Random Forest, XGBoost, Multilayer Perceptron (MLP), Support Vector Machine (SVM) are used to train the model using the training data.

7. **Classification**: The final step is to classify the scripts as either "Malicious/Benign" based on the trained model.

4.4 FEATURE SELECTION

In this project, various approach of feature selection was implemented. These methods include Information Gain (IG) with different number of features (25, 67, 167), IG and Recursive Feature Elimination (RFE), IG and Sequential Backward Selection (SBS), IG and Lasso, IG and Principal Component Analysis (PCA), PCA, Lasso, Lasso and PCA, Forward Selection and PCA, Elastic Net, Feature Importance, and RFE with different numbers of features (10, 15, 3, and 2). Among this approach, the combination of Information Gain (IG) with highest feature selected (167) as feature selection and hybrid combination of model training (Random Forest (RF) and XGBoost achieved the highest accuracy of performance.

All the list of approach was actually under three types of feature selection types, which are filter method, wrapper method, and embedded method. Each type of the feature selection is actually offering different strength but aims for the same goal. As for the filter method, this method is quick, computationally efficient, and easy to apply, making them ideal for huge datasets. They rank characteristics according to statistical criteria such as correlation, mutual information, and chi-squared scores. They are model agnostic since they do not use any machine learning algorithms, making them appropriate for preprocessing prior to applying any model. Examples include Information Gain (IG) and the Correlation Coefficient (CC), which quantify the reduction in entropy or uncertainty caused by separating data based on a characteristic.

Next, wrapper methods are machine learning approaches that enable personalized feature selection by training a model on each subset of a feature collection. This strategy improves forecast accuracy by taking into account the interactions between features and the model. Wrapper techniques are adaptable and may be applied with any machine learning algorithm. Some examples are Recursive Feature Elimination (RFE), Sequential Forward Selection, and Sequential Backward Selection. These approaches operate by analyzing feature subsets, modifying the number of features, and ensuring that the optimal number of features is achieved.

Furthermore, embedded methods are a model training methodology that incorporates feature selection into the model training process, decreasing overfitting by selecting only relevant features. They are computationally efficient since feature selection is part of the model training procedure. Embedded approaches include Lasso Regularization (L1), which adds a penalty to the absolute value of coefficients to successfully execute feature selection, and Ridge Regularization (L2), which prevents overfitting by introducing a penalty proportional to the square of coefficient magnitudes. Decision trees and ensemble approaches, such as Random Forest and XGBoost, assign feature relevance scores depending on their utility in lowering impurities.

In summary, this project used a variety of feature selection approaches to determine the most important attributes for identifying Cross-Site Scripting (XSS) threats. The combination of Information Gain (IG) with the hybrid model of Random Forest (RF) and XGBoost produced the best results and a great performance. The following chapter will go over further experiments, analyses, and outcomes.

4.5 MODEL TRAINING

In this project, various approach of model training was implemented. The method includes Random Forest classifier (RF), XGBoost, Support Vector Mechanism (SVM), and Multilayer Perceptron (MLP). Among this approach, the hybrid combination of model training (Random Forest (RF) and XGBoost, along with Information Gain (IG) as feature selection achieved the highest accuracy of performance.

Random Forest Classifier (RF) is a reliable and precise ensemble approach that mixes numerous decision trees to increase accuracy and prevent overfitting. It can handle complicated datasets and make reliable predictions. RF can prioritize feature importance, determine which features contribute the most to predictions, and deal with unbalanced datasets using techniques like as bootstrapping and majority voting. It is useful in machine learning for both classification and regression.

Furthermore, XGBoost is a popular choice for predictive modelling because to its great performance and efficiency. It incorporates L1 and L2 regularization strategies to reduce overfitting and improve model generalization. XGBoost is extremely scalable and can handle big datasets, making it ideal for a variety of machine learning tasks.

Next, Support Vector Mechanism (SVM) is a highly accurate and effective data processing technique, especially in high-dimensional spaces. It can handle non-linear relationships in data using kernel tricks and is less prone to overfitting, making it suitable for classification, regression, and outlier detection tasks.

Lastly, Multilayer Perceptron (MLP) is a flexible machine learning algorithm that can learn complicated data patterns using several layers and non-linear activation functions. It is utilized for tasks including as classification, regression, and pattern recognition, making it extremely flexible to many types of data and issues. MLPs also serve as the foundation for complicated neural network topologies in deep learning.

In summary, this project run the experiment towards all these models to test their performance. The result shows that the hybrid combination of model training (Random Forest (RF) and XGBoost, along with Information Gain (IG) as feature selection achieved the highest accuracy of performance. This thorough analysis emphasises each model's qualities and highlights the practicality of the chosen hybrid strategy in improving XSS threat detection.

4.6 CONCLUSION

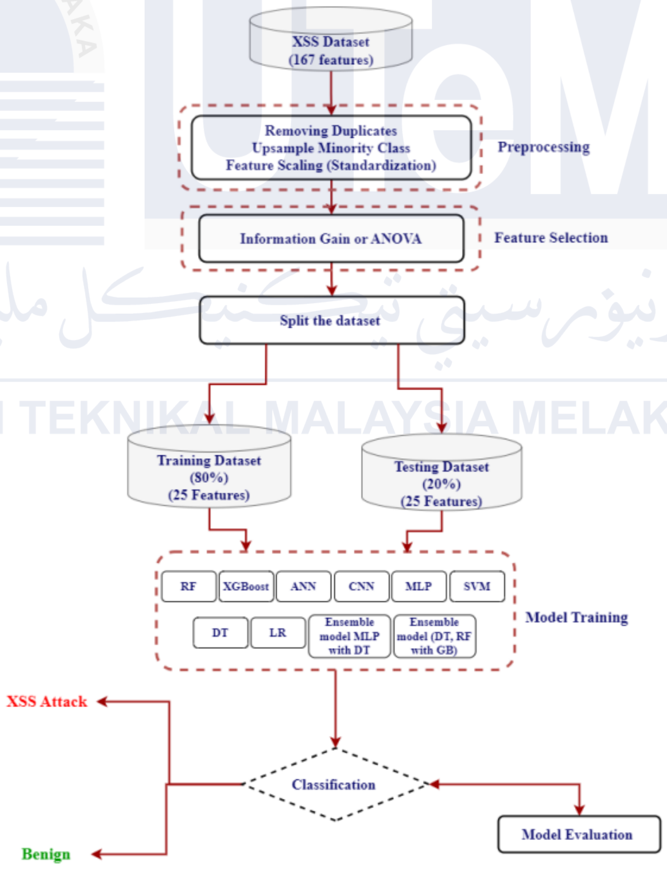
This chapter delves into the design approach for identifying Cross-Site Scripting (XSS) attacks with machine learning algorithm. The complexity and danger of XSS attacks in current online applications are addressed, revealing important needs for a strong detection system. The design process involves feature collection, extraction, selection, and model training. A blend of Information Gain (IG), Random Forest (RF), and XGBoost was discovered to be the most successful, with the best accuracy. Multiple machine learning models were examined, but the hybrid model that included RF and XGBoost and was augmented by IG outperformed them all. The hybrid model's successful implementation illustrates the efficacy of the chosen methodologies, paving the way for future experimentation, analysis, and development.

CHAPTER 5: IMPLEMENTATION

5.1 INTRODUCTION

In this chapter, we will go through to the all process that happen in this project. The processes are Dataset Collection, Data Preprocessing, Feature Selection, Model Training Phase, and Performance Evaluation. The evaluation matric will be measured to test the model accuracy in this project. The framework for this study is shown in Figure 21.

5.2 PROJECT ENVIRONMENT SETUP



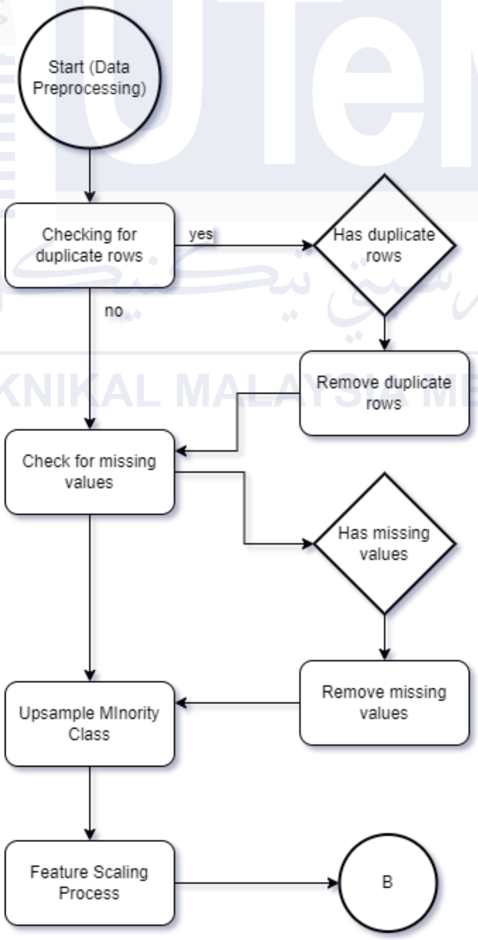
XSS Dataset (167 features)
Removing Duplicates
Upsample Minority Class
Feature Scaling (Standardization)
Preprocessing
Information Gain or ANOVA
Feature Selection
Split the dataset
Training Dataset (80%) (25 Features)
Testing Dataset (20%) (25 Features)
RF
XGBoost
ANN
CNN
MLP
SVM
DT
LR
Ensemble model MLP with DT
Ensemble model (DT, RF with GB)
Model Training
XSS Attack
Classification
Benign
Model Evaluation

Figure 21: Framework for XSS Detection

From Figure 21 of the XSS Detection Framework, the main process highlighted consist of five stages, which are Preprocessing (Data Preprocessing), Feature Selection, Split the dataset, Model Training, and Classification (Model Evalution).

5.2.1 Data Preprocessing

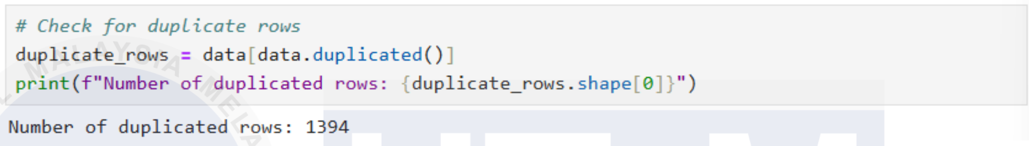
Data Preprocessing is one of the crucial processes in preparing raw dataset to ensure the uniformity, quality, and relevance. This stage is also purposed to resulting in a data cleaning. Figure 22 shown the detail process that are running in this stage, which are checking for duplicate rows, checking for missing values, unsampled minority class, and feature scaling process**.**



Feature 22: Flowchart for the Data Preprocessing stage

5.2.1.1 Check duplicate rows

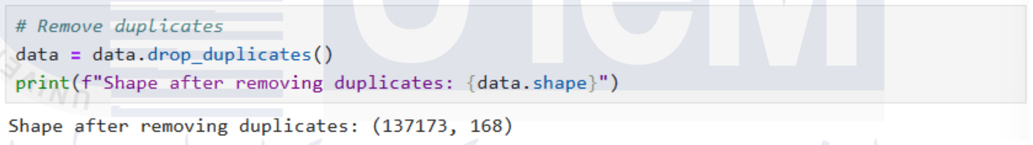
The occurrence of identical data instances in a dataset, known as duplicate rows, can have an impact on the training and assessment of models. Before stepping to other step, common techniques involve eliminating duplicates or retaining one instance. This are aimed to keep one occurrence based on particular columns and removing the others had been implemented to ensure the originality. Figure 23 and Figure 24 shows the process of checking and removing the duplicate rows.



```
# Check for duplicate rows
duplicate_rows = data[data.duplicated()]
print(f"Number of duplicated rows: {duplicate_rows.shape[0]}")
```

```
Number of duplicated rows: 1394
```

Figure 23: check duplicate row



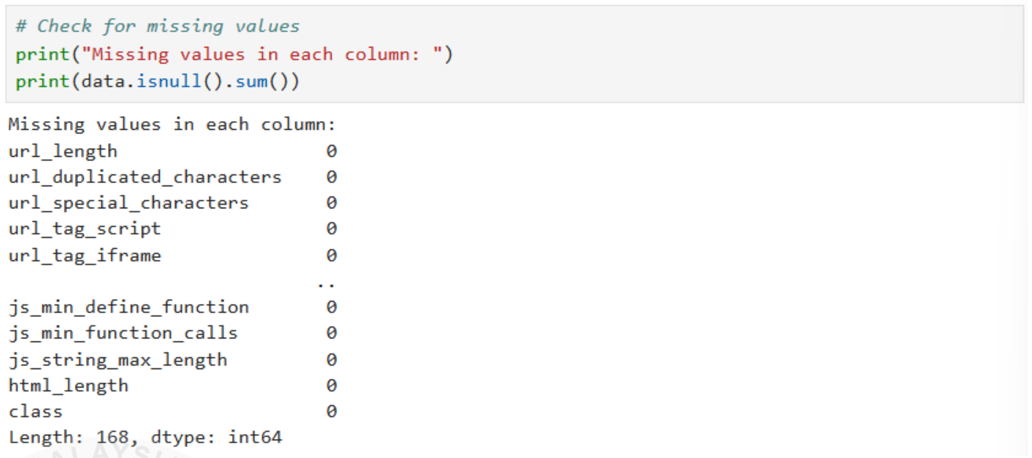
```
# Remove duplicates
data = data.drop_duplicates()
print(f"Shape after removing duplicates: {data.shape}")
```

```
Shape after removing duplicates: (137173, 168)
```

Figure 24: remove duplicate

5.2.1.2 Check for missing values

On the other hand, missing values or null can also impact the evaluation of model training. Missing values also referred as ‘null’ or ‘NaN’ in the dataset. There are several strategies to overcome this problem which are eliminating missing data, imputation using statistical measurements, and sophisticated techniques using machine learning algorithms. However, as my dataset does not have any missing values (shown in Figure xx), we continue to proceed to the next process.



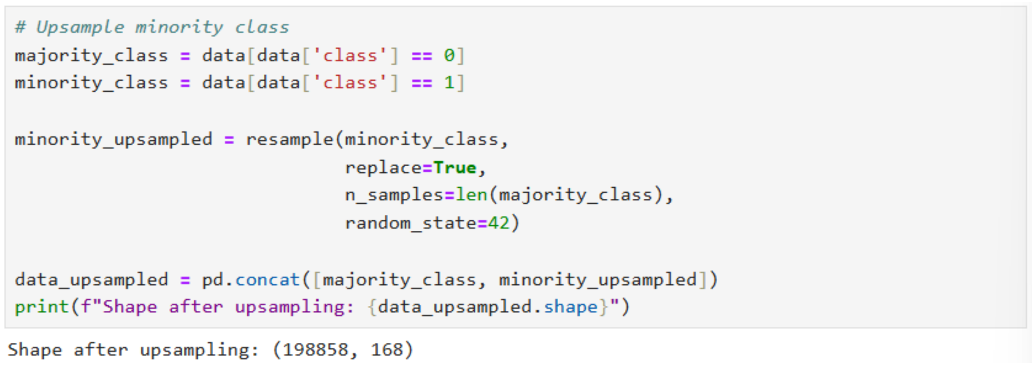
```
# Check for missing values
print("Missing values in each column: ")
print(data.isnull().sum())
```

```
Missing values in each column:
url_length                   0
url_duplicated_characters    0
url_special_characters       0
url_tag_script               0
url_tag_iframe               0
                            ..
js_min_define_function       0
js_min_function_calls        0
js_string_max_length         0
html_length                  0
class                        0
Length: 168, dtype: int64
```

Figure 25: check missing value

5.2.1.3 Upsample minority class

In imbalanced datasets, the minority class may have insufficient samples, such as rare events like XSS attacks. Techniques to address this include oversampling, undersampling, and SMOTE (Synthetic Minority Over-sampling Technique), which create synthetic instances of the minority class. In this project, as to balance with the majority class, we manage this procedure in the project by increasing the sample size in the minority class. Figure 26 shows the process of upsampling the minority class.



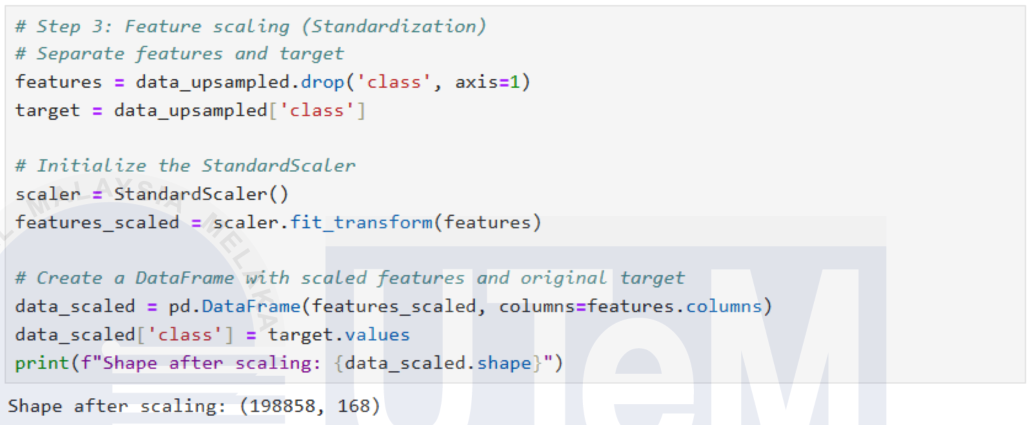
```
# Upsample minority class
majority_class = data[data['class'] == 0]
minority_class = data[data['class'] == 1]

minority_upsampled = resample(minority_class,
                              replace=True,
                              n_samples=len(majority_class),
                              random_state=42)

data_upsampled = pd.concat([majority_class, minority_upsampled])
print(f"Shape after upsampling: {data_upsampled.shape}")
```

```
Shape after upsampling: (198858, 168)
```

Figure 26: upsample minority class

5.2.1.4 Feature Scaling Process

During model training, feature scaling is a strategy that is used to make sure that all features have comparable scales. This can be accomplished by standardization, min-max scaling, or robust scaling, which entails changing features to zero mean and unit variance. Figure 27 shows the process of feature scaling towards the dataset.



```
# Step 3: Feature scaling (Standardization)
# Separate features and target
features = data_upsampled.drop('class', axis=1)
target = data_upsampled['class']

# Initialize the StandardScaler
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

# Create a DataFrame with scaled features and original target
data_scaled = pd.DataFrame(features_scaled, columns=features.columns)
data_scaled['class'] = target.values
print(f"Shape after scaling: {data_scaled.shape}")
```

```
Shape after scaling: (198858, 168)
```

Figure 27: feature scaling process

5.2.2 Feature Selection

In this stage, instead of focusing and relying on the same techniques or methods, that are commonly practiced among scholars, like Information Gain, this project introduces a new improvement and enhancement towards the XSS Detection. Various methods have been experimented, starting from an individual approach to the using of hybrid, or combination methods in feature selection stage. Figure 28 shows the complete process for the stage of feature selection.

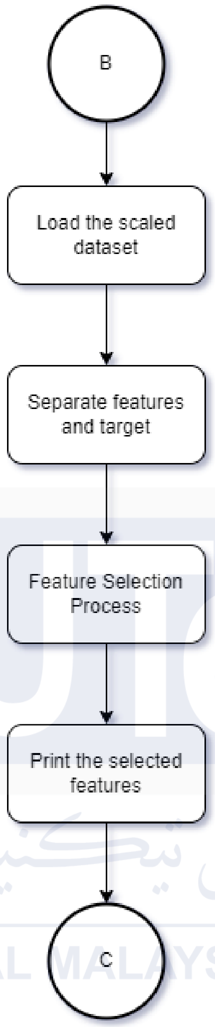
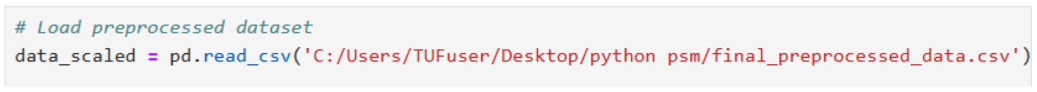

B
Load the scaled dataset
Separate features and target
Feature Selection Process
Print the selected features
C

Figure 28: Flowchart of Feature Selection process

5.2.2.1 Load the scaled dataset

Before further into more depth step, this project ensures to reload the scaled or final preprocessing dataset into environment. This process is crucial as we want to avoid to train the wrong or raw dataset. If we happen to train the raw dataset, it will badly impact the quality and consistency of the model output. Figure 29 shows the process of loading the scaled dataset.

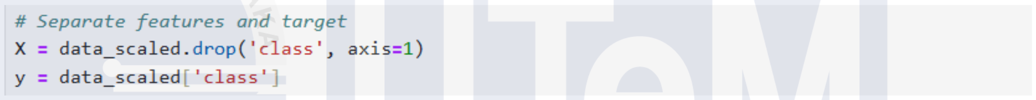


```
# Load preprocessed dataset
data_scaled = pd.read_csv('C:/Users/TUFuser/Desktop/python psm/final_preprocessed_data.csv')
```

Figure 29: Process of load the final dataset

5.2.2.2 Separate features and target

In this step, the dataset is divided into features (input variables) and target variables using the code in Figure 30. It is employed in machine learning to get data ready for regression and classification models. The feature matrix is represented by the X variable, the target vector is represented by the Y variable, and a column is dropped using the axis=1 parameter. When training models and assessing their effectiveness by contrasting predictions with real target values, this division is essential.

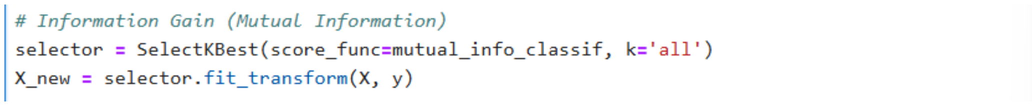


```
# Separate features and target
X = data_scaled.drop('class', axis=1)
y = data_scaled['class']
```

Figure 30: Process of separating the features and target

5.2.2.3 Feature Selection Process

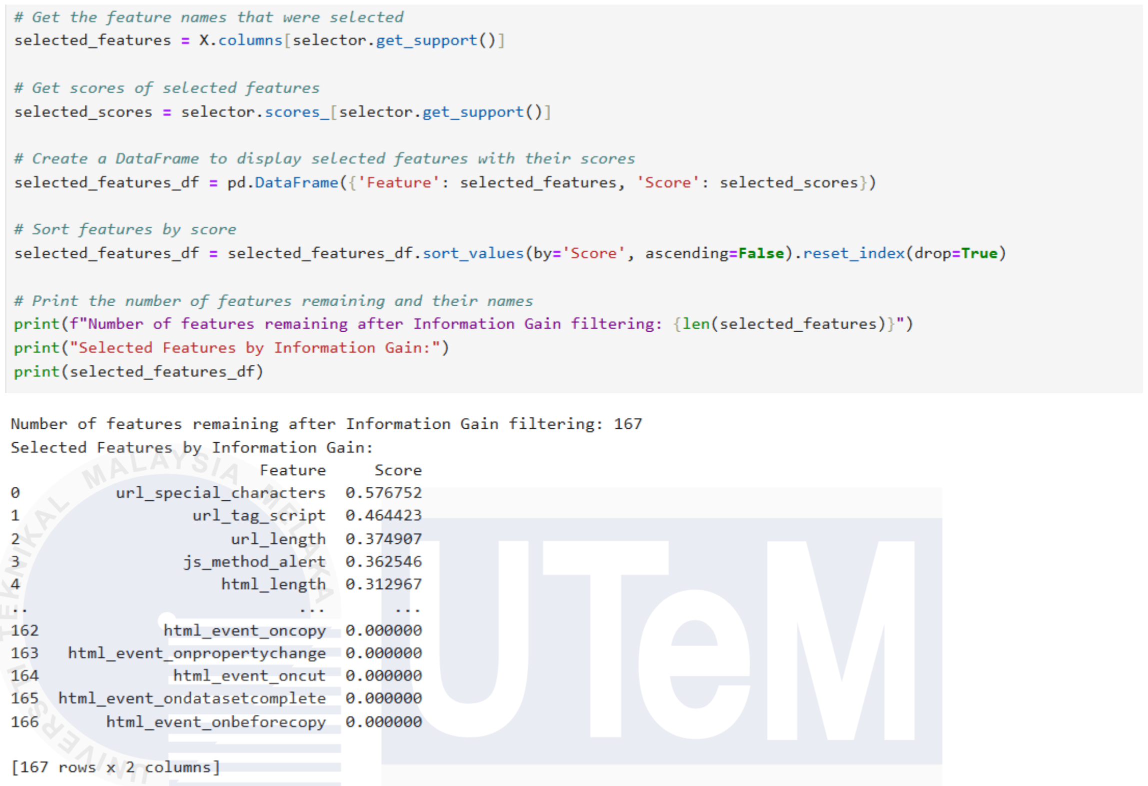
This the one of example for Feature Selection Process. In this example of Figure 31, the project use Information Gain or Mutual Information for scoring purpose. This project continues to utilized the SelectKBest method from scikit-learn. This approach evaluates each feature's statistical dependency on the target variable. All features in the converted dataset (X_new) were preserved by setting k='all'.



```
# Information Gain (Mutual Information)
selector = SelectKBest(score_func=mutual_info_classif, k='all')
X_new = selector.fit_transform(X, y)
```

Figure 31: One of example for feature selection process

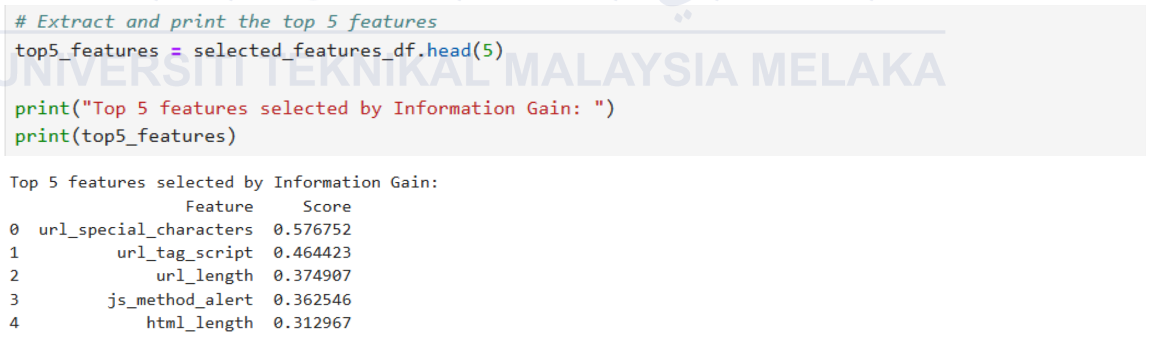
Figure 32 then shows the process of printing the features that are selected in the process. In this part the features are ranked by their score. Regarding their score, this project prints out the top five features that high likely to be selected during the selection process as shown in Figure 33.



```python
# Get the feature names that were selected
selected_features = X.columns[selector.get_support()]

# Get scores of selected features
selected_scores = selector.scores_[selector.get_support()]

# Create a DataFrame to display selected features with their scores
selected_features_df = pd.DataFrame({'Feature': selected_features, 'Score': selected_scores})

# Sort features by score
selected_features_df = selected_features_df.sort_values(by='Score', ascending=False).reset_index(drop=True)

# Print the number of features remaining and their names
print(f"Number of features remaining after Information Gain filtering: {len(selected_features)}")
print("Selected Features by Information Gain:")
print(selected_features_df)
```

```
Number of features remaining after Information Gain filtering: 167
Selected Features by Information Gain:
                         Feature     Score
0        url_special_characters  0.576752
1                url_tag_script  0.464423
2                    url_length  0.374907
3               js_method_alert  0.362546
4                   html_length  0.312967
..                           ...       ...
162            html_event_oncopy  0.000000
163  html_event_onpropertychange  0.000000
164             html_event_oncut  0.000000
165 html_event_ondatasetcomplete  0.000000
166       html_event_onbeforecopy  0.000000

[167 rows x 2 columns]
```

Figure 32: Process of printing the feature selected



```python
# Extract and print the top 5 features
top5_features = selected_features_df.head(5)

print("Top 5 features selected by Information Gain: ")
print(top5_features)
```

```
Top 5 features selected by Information Gain:
                  Feature     Score
0  url_special_characters  0.576752
1          url_tag_script  0.464423
2              url_length  0.374907
3         js_method_alert  0.362546
4             html_length  0.312967
```

Figure 33: Process of printing the top 5 features selected

5.2.3 Split the dataset

As we refer to the framework in Figure 34 and Figure 35, this stage is the process for the splitting dataset. The purpose of this step is to split a dataset into two subsets, which are a training set and a test set. This process is important as it will influence the performance of the model training.

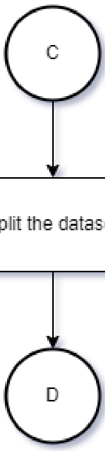
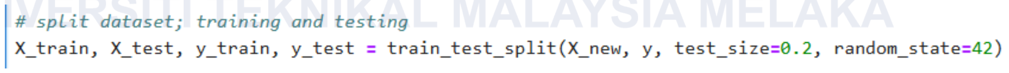


Figure 34: Flowchart of Splitting the dataset

From Figure 35, the machine learning model is trained using the training set (X_train, y_train). These data provide patterns for the model to learn.
Test Set (X_test, y_test): Applied to assess the performance of the model.
evaluates the model's ability to generalize to new data. Meanwhile test_size=0.2 is purposes to specifies that 20% of the data will be allocated to the test set, and the remaining 80% is for training.



```
# split dataset; training and testing
X_train, X_test, y_train, y_test = train_test_split(X_new, y, test_size=0.2, random_state=42)
```

Figure 35: Split dataset process

5.2.4 Model Training

A key component of machine learning is model training, which gives the model the ability to recognize patterns, optimize parameters, generalize to new data, and make precise predictions or classifications. It assures the model's efficacy in resolving issues in the actual world by assisting it in adapting to different conditions outside of the training set. Model training is a crucial stage in developing efficient machine learning systems since methods like gradient descent are used to fine-tune these parameters. Figure 36 shows the process of model training stages, meanwhile Figure 37 shows the implementation in the form of python code.

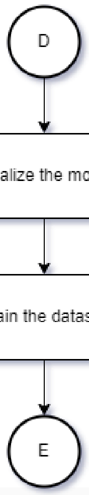
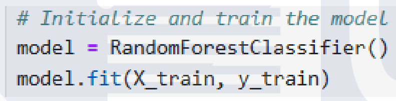


Figure 36: Flowchart of Model Training stage



```
# Initialize and train the model
model = RandomForestClassifier()
model.fit(X_train, y_train)
```

Figure 37: Process of initialize and training the model

5.2.5 Classification (Model Evaluation)

After training the dataset with the model, this project continues with the model evaluation to measure the overall performance of each model as shown is Figure 38. Model evaluation helps in identifying the strength and weakness of each approach.

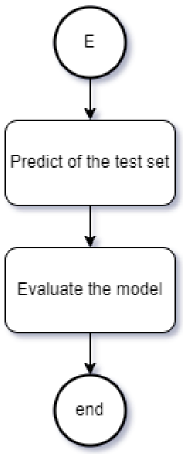
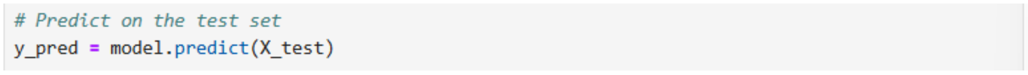


Figure 38: Flowchart of Model Evaluation stage

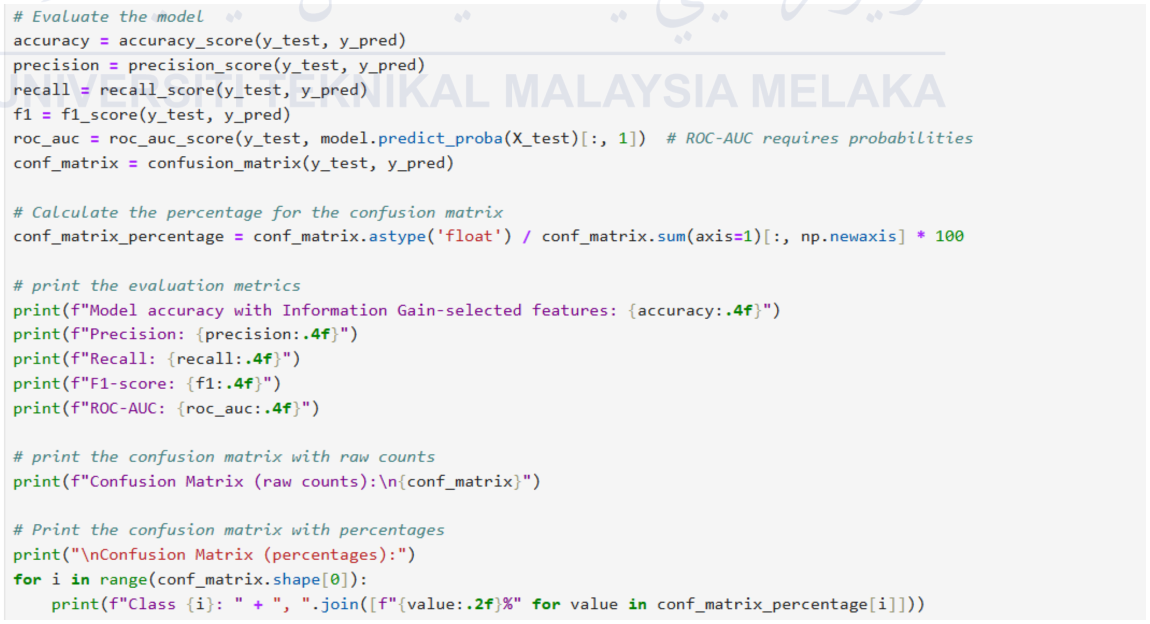
Figure 39, y_pred defined as varible that consists of the expected target values (output) derived from the test set's characteristics (X_test).



```
# Predict on the test set
y_pred = model.predict(X_test)
```

Figure 39: Predict of the test set

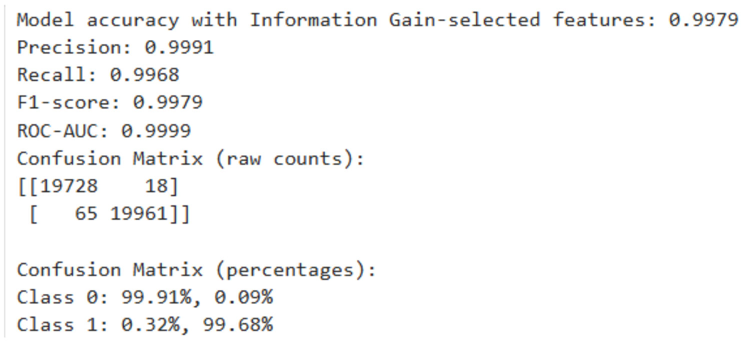
Meanwhile in Figure 40 is focusing on the evaluation metrices, such as *accuracy* which evaluates how well predictions are made overall, *precision* represents the proportion of actual positive cases that are anticipated to be positive. Meanwhile *recall* is a metric used to quantify how well real positive events are anticipated. *f1*: Consolidates recall and accuracy into a single score, and *roc_auc* is the area under the ROC curve, which measures receiver operating characteristic (predicted probabilities are required). True positive, true negative, false positive, and false negative counts are displayed via the conf_matrix. In this phase, we also implement calculation for the confusion matrix to get their percentage after the training phase.



```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, model.predict_proba(X_test)[:, 1])  # ROC-AUC requires probabilities
conf_matrix = confusion_matrix(y_test, y_pred)

# Calculate the percentage for the confusion matrix
conf_matrix_percentage = conf_matrix.astype('float') / conf_matrix.sum(axis=1)[:, np.newaxis] * 100

# print the evaluation metrics
print(f"Model accuracy with Information Gain-selected features: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1-score: {f1:.4f}")
print(f"ROC-AUC: {roc_auc:.4f}")

# print the confusion matrix with raw counts
print(f"Confusion Matrix (raw counts):\n{conf_matrix}")

# Print the confusion matrix with percentages
print("\nConfusion Matrix (percentages):")
for i in range(conf_matrix.shape[0]):
    print(f"Class {i}: " + ", ".join([f"{value:.2f}%" for value in conf_matrix_percentage[i]]))
```

Figure 40: Process of evaluating the model

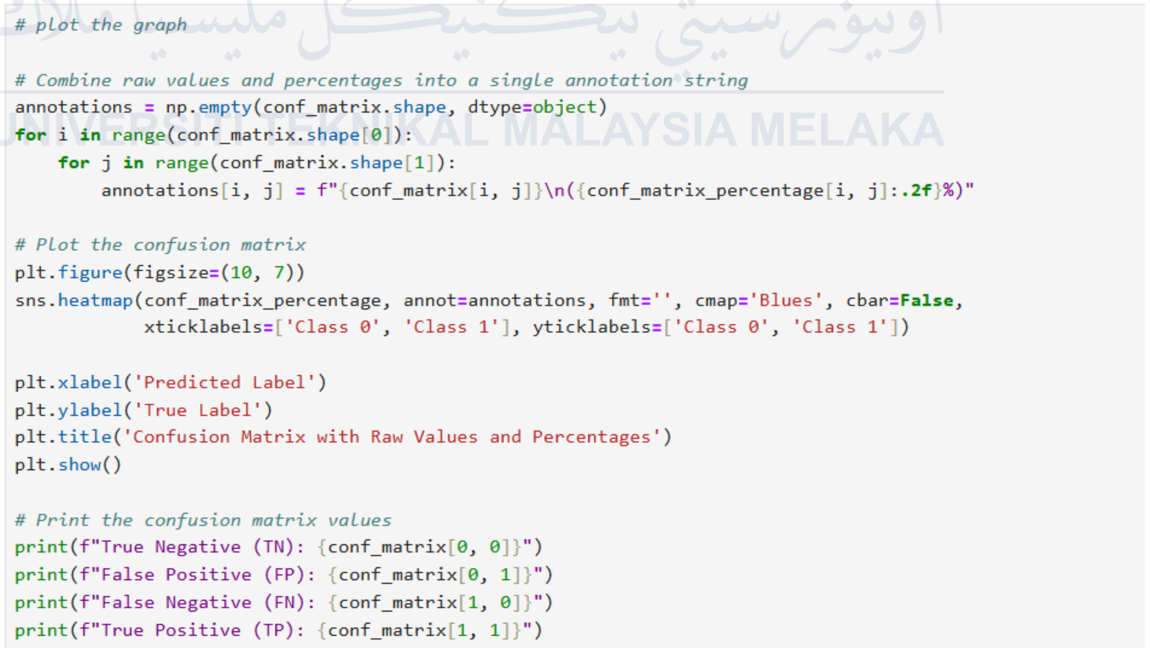
Figure 41 shows the example output of the model evaluation after running the code.



```
Model accuracy with Information Gain-selected features: 0.9979
Precision: 0.9991
Recall: 0.9968
F1-score: 0.9979
ROC-AUC: 0.9999
Confusion Matrix (raw counts):
[[19728    18]
 [   65 19961]]

Confusion Matrix (percentages):
Class 0: 99.91%, 0.09%
Class 1: 0.32%, 99.68%
```

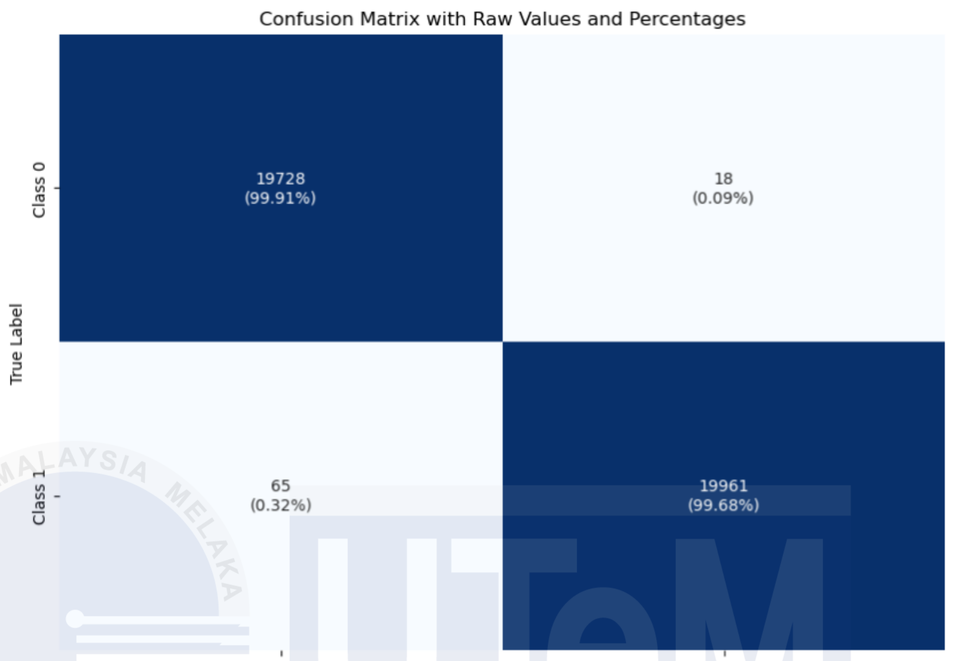
Figure 41: Example output for the model training

After gaining the result of the evaluation metrices and confusion matrix, this project continues with plotting the confusion matrix graph to have a clear visualization of the True Positive, True Negative, False Positive, and False Negative. The process is shown as Figure 42. Meanwhile at Figure 43 shows the graph of confusion matrix. Note that confusion matrix plays a crucial role in determining the evaluation scores of the experiment in this project.



```
# plot the graph

# Combine raw values and percentages into a single annotation string
annotations = np.empty(conf_matrix.shape, dtype=object)
for i in range(conf_matrix.shape[0]):
    for j in range(conf_matrix.shape[1]):
        annotations[i, j] = f"{conf_matrix[i, j]}\n({conf_matrix_percentage[i, j]:.2f}%)"

# Plot the confusion matrix
plt.figure(figsize=(10, 7))
sns.heatmap(conf_matrix_percentage, annot=annotations, fmt='', cmap='Blues', cbar=False,
            xticklabels=['Class 0', 'Class 1'], yticklabels=['Class 0', 'Class 1'])

plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix with Raw Values and Percentages')
plt.show()

# Print the confusion matrix values
print(f"True Negative (TN): {conf_matrix[0, 0]}")
print(f"False Positive (FP): {conf_matrix[0, 1]}")
print(f"False Negative (FN): {conf_matrix[1, 0]}")
print(f"True Positive (TP): {conf_matrix[1, 1]}")
```

Figure 42: Process of plotting the confusion matrix graph



```
True Negative (TN): 19728
False Positive (FP): 18
False Negative (FN): 65
True Positive (TP): 19961
```

Figure 43: Process of confusion matrix graph

5.3 CONCLUSION

This chapter describes an orderly approach to machine learning for the detection of cross-site scripting (XSS). Data gathering, preprocessing, feature selection, model testing and training, and performance evaluation are all steps in the process. The models evaluated for the accuracy, precision, recall, and F1-score using a raw dataset that was obtained from GitHub. The success of the system was quantified, laying the groundwork for further comparisons. The goal of the trip is to provide participants a thorough grasp of machine learning for XSS detection.

CHAPTER 6: TESTING AND ANALYSIS

6.1 INTRODUCTION

This chapter will explain and shows how the project underwent testing with the new improvement and enhancement for the XSS detection. The testing and analysis that conducted will be present the performance of the methods and model selected.

6.2 TEST PLAN

6.2.1 Test Environment

This experiment is conducted on a machine operating on Windows 11 Home Single Language 64-bit, the processor was Intel(R) Core (TM) i5-10300H CPU @ 2.50GHz, 2.50 GHz. The memory of the machine was 7900MB RAM, and the graphic card used was NVIDIA GeForce GTX 1650. The suggested model for this project was created in Jupyter Notebook and implemented in the Python programming language. 'Pandas' was the package used to load the data into the models. The "Sci-Kit Learn" package was used to implement the model, and it also supplied the matrix.

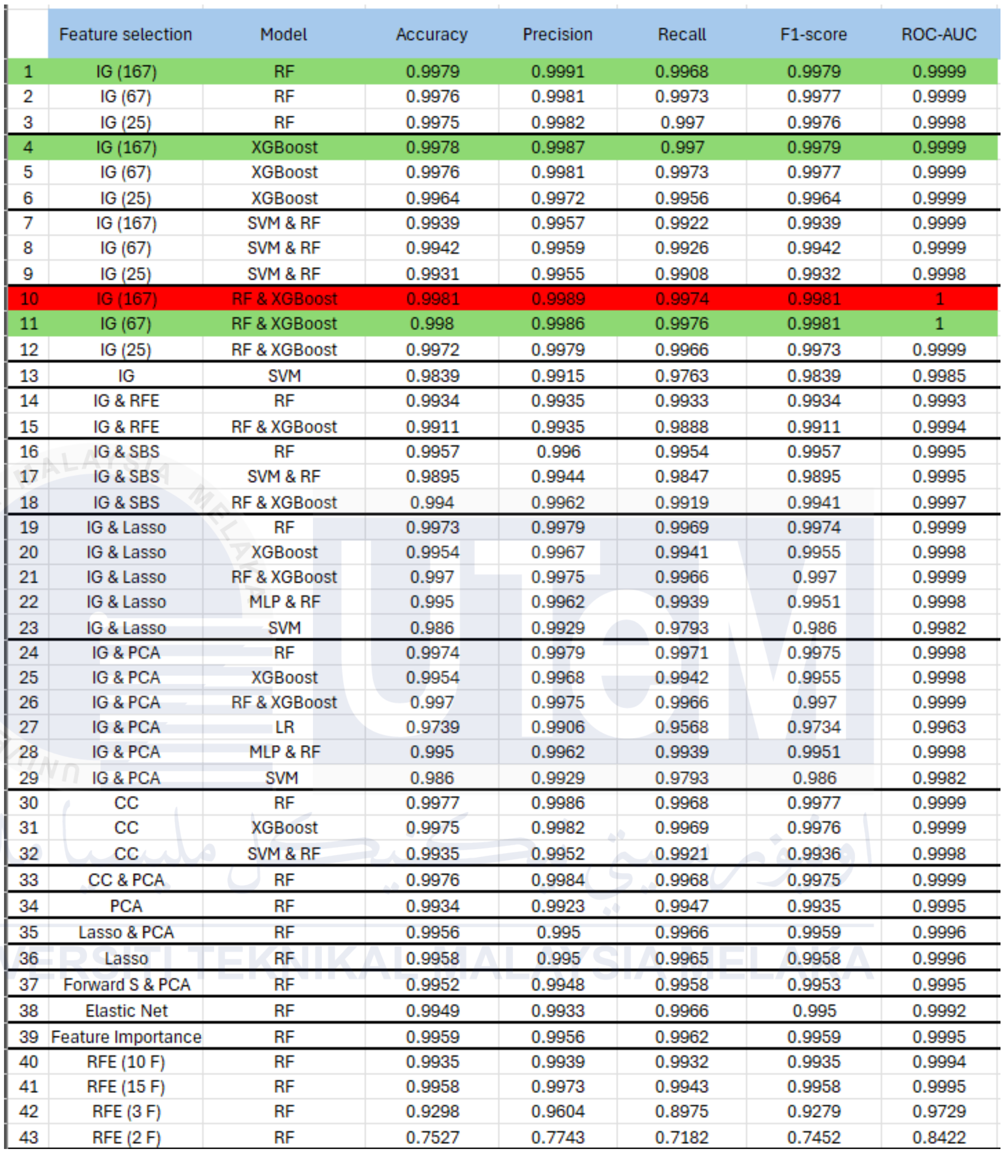
6.2.2 Test Strategy

This experiment underwent 43 approaches to complete all the testing. These approaches started from individual approach to hybrid combination of approaches. The approach for feature selection is Information Gain (IG), Recursive Feature Elimination (RFE), Sequential Backward Selection (SBS), Lasso, Principal Component Analysis (PCA), Correlation Coefficient (CC), Forward Selection (FS), Elastic Net, and Feature Importance. Meanwhile the approaches for model training consist of Random Forest Classifier (RF), XGBoost, Support Vector Machines (SVM), Multilayer Perceptrons (MLP), and Logistic Regression (LR). The outstanding result is presented after the trial-and-error for combination of approaches and many more.

6.3 TEST DESIGN

This project explored and proposed many combinations of feature selection-feature selection, and Model Training-Model Training. The result of the overall performance is presented in Figure 45. The overall performance resulting that most of them are in high accuracy. Only three out of 43 approaches not surpass 0.98 in accuracy, meanwhile the other 40 approaches are exceeded 0.99 in accuracy.

There are three color representatives of the result in Figure 44, which are red, green, and colorless table. The red table represent the proposed enhancement of the XSS detection machine learning algorithm, also resulting in a highest accuracy, which is 0.9981. The green one is the representation of the other top approaches that also resulting in a great performance, which including IG as feature selection and hybrid model training RF and XGBoost, with 167 features selected (0.998), IG as feature selection and model training RF, with 167 features selected (0.9979), and following with IG as feature selection and model training XGBoost, with 167 features selected (0.9978).



	Feature selection	Model	Accuracy	Precision	Recall	F1-score	ROC-AUC
1	IG (167)	RF	0.9979	0.9991	0.9968	0.9979	0.9999
2	IG (67)	RF	0.9976	0.9981	0.9973	0.9977	0.9999
3	IG (25)	RF	0.9975	0.9982	0.997	0.9976	0.9998
4	IG (167)	XGBoost	0.9978	0.9987	0.997	0.9979	0.9999
5	IG (67)	XGBoost	0.9976	0.9981	0.9973	0.9977	0.9999
6	IG (25)	XGBoost	0.9964	0.9972	0.9956	0.9964	0.9999
7	IG (167)	SVM & RF	0.9939	0.9957	0.9922	0.9939	0.9999
8	IG (67)	SVM & RF	0.9942	0.9959	0.9926	0.9942	0.9999
9	IG (25)	SVM & RF	0.9931	0.9955	0.9908	0.9932	0.9998
10	IG (167)	RF & XGBoost	0.9981	0.9989	0.9974	0.9981	1
11	IG (67)	RF & XGBoost	0.998	0.9986	0.9976	0.9981	1
12	IG (25)	RF & XGBoost	0.9972	0.9979	0.9966	0.9973	0.9999
13	IG	SVM	0.9839	0.9915	0.9763	0.9839	0.9985
14	IG & RFE	RF	0.9934	0.9935	0.9933	0.9934	0.9993
15	IG & RFE	RF & XGBoost	0.9911	0.9935	0.9888	0.9911	0.9994
16	IG & SBS	RF	0.9957	0.996	0.9954	0.9957	0.9995
17	IG & SBS	SVM & RF	0.9895	0.9944	0.9847	0.9895	0.9995
18	IG & SBS	RF & XGBoost	0.994	0.9962	0.9919	0.9941	0.9997
19	IG & Lasso	RF	0.9973	0.9979	0.9969	0.9974	0.9999
20	IG & Lasso	XGBoost	0.9954	0.9967	0.9941	0.9955	0.9998
21	IG & Lasso	RF & XGBoost	0.997	0.9975	0.9966	0.997	0.9999
22	IG & Lasso	MLP & RF	0.995	0.9962	0.9939	0.9951	0.9998
23	IG & Lasso	SVM	0.986	0.9929	0.9793	0.986	0.9982
24	IG & PCA	RF	0.9974	0.9979	0.9971	0.9975	0.9998
25	IG & PCA	XGBoost	0.9954	0.9968	0.9942	0.9955	0.9998
26	IG & PCA	RF & XGBoost	0.997	0.9975	0.9966	0.997	0.9999
27	IG & PCA	LR	0.9739	0.9906	0.9568	0.9734	0.9963
28	IG & PCA	MLP & RF	0.995	0.9962	0.9939	0.9951	0.9998
29	IG & PCA	SVM	0.986	0.9929	0.9793	0.986	0.9982
30	CC	RF	0.9977	0.9986	0.9968	0.9977	0.9999
31	CC	XGBoost	0.9975	0.9982	0.9969	0.9976	0.9999
32	CC	SVM & RF	0.9935	0.9952	0.9921	0.9936	0.9998
33	CC & PCA	RF	0.9976	0.9984	0.9968	0.9975	0.9999
34	PCA	RF	0.9934	0.9923	0.9947	0.9935	0.9995
35	Lasso & PCA	RF	0.9956	0.995	0.9966	0.9959	0.9996
36	Lasso	RF	0.9958	0.995	0.9965	0.9958	0.9996
37	Forward S & PCA	RF	0.9952	0.9948	0.9958	0.9953	0.9995
38	Elastic Net	RF	0.9949	0.9933	0.9966	0.995	0.9992
39	Feature Importance	RF	0.9959	0.9956	0.9962	0.9959	0.9995
40	RFE (10 F)	RF	0.9935	0.9939	0.9932	0.9935	0.9994
41	RFE (15 F)	RF	0.9958	0.9973	0.9943	0.9958	0.9995
42	RFE (3 F)	RF	0.9298	0.9604	0.8975	0.9279	0.9729
43	RFE (2 F)	RF	0.7527	0.7743	0.7182	0.7452	0.8422

Figure 44: Output of evaluation model for all approaches

6.4 RESULT AND ANALYSIS

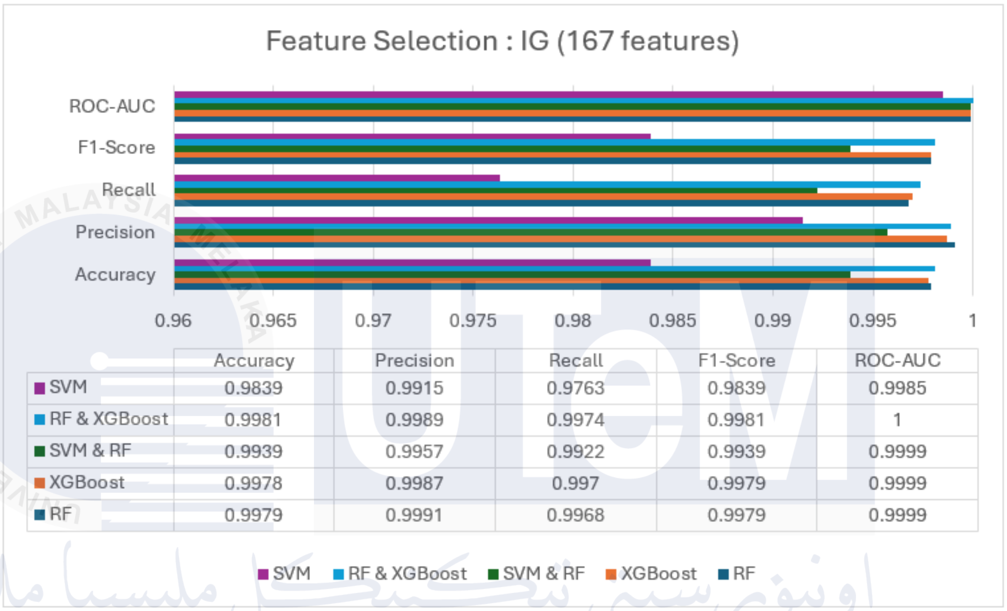
This section will explain in detail about the experiment in a divided section of feature selection.

6.4.1 Feature Selection: Information Gain

In this section, the project conducted by using Information Gain (IG) as feature selection. There are three experiments conducted, which select 167, 67, and 25 features.

The result shows a slightly different scores as the feature selected is changing in quantity. Thus, table 9 shows the top five features that has been selected from this experiment.

Information Gain with 167 features selected



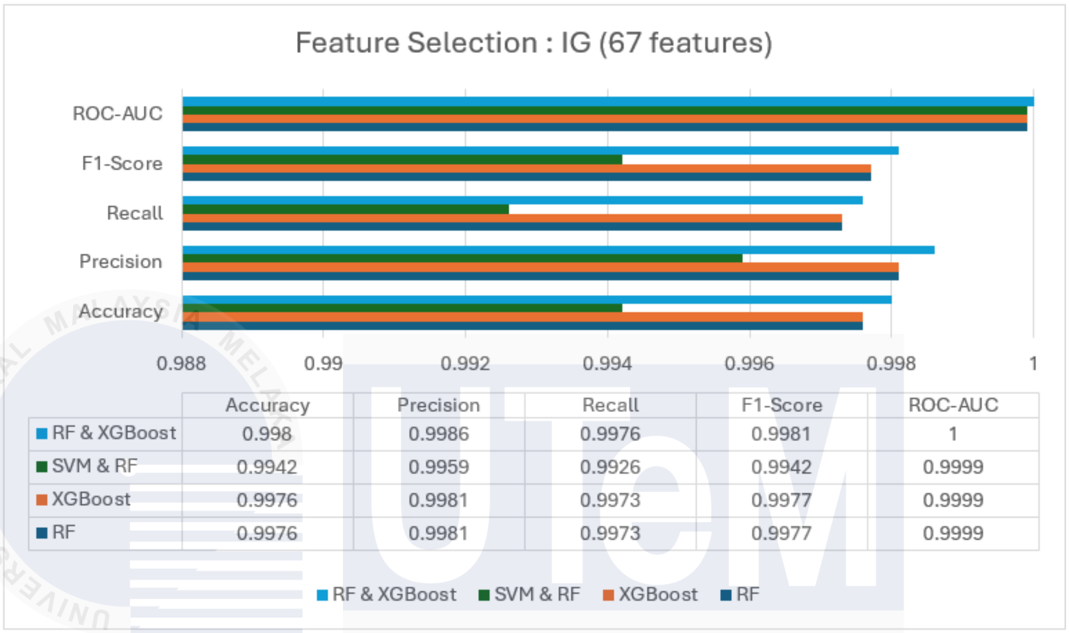
	Accuracy	Precision	Recall	F1-Score	ROC-AUC
SVM	0.9839	0.9915	0.9763	0.9839	0.9985
RF & XGBoost	0.9981	0.9989	0.9974	0.9981	1
SVM & RF	0.9939	0.9957	0.9922	0.9939	0.9999
XGBoost	0.9978	0.9987	0.997	0.9979	0.9999
RF	0.9979	0.9991	0.9968	0.9979	0.9999

Figure 45: Evaluation model for IG as feature selection (167 features)

Figure 45 above present various score for the combination of approaches. The highest score was hybrid of Random Forest (RF) and XGBoost as model training, with accuracy 0.9981, followed by RF (0.9979), XGBoost (0.9978), SVM and RF (0.9939), and lastly SVM (0.9839).

Random Forest is known as a robust and efficient technique that combines multiple decision trees to reduce variation and improve accuracy. It can be used for regression tasks or classification, and is resistant to overfitting and outliers. It trains quickly and considers random subsets of features. Meanwhile XGBoost is a fast and efficient gradient-boosting implementation, ideal for large datasets on multicore machines or clusters. It continuously outperforms boosted tree methods in terms of computation while delivering excellent accuracy and reliable performance. By combining both powerful model training, the performance is resulting in a good high accuracy.

Information Gain with 67 features selected



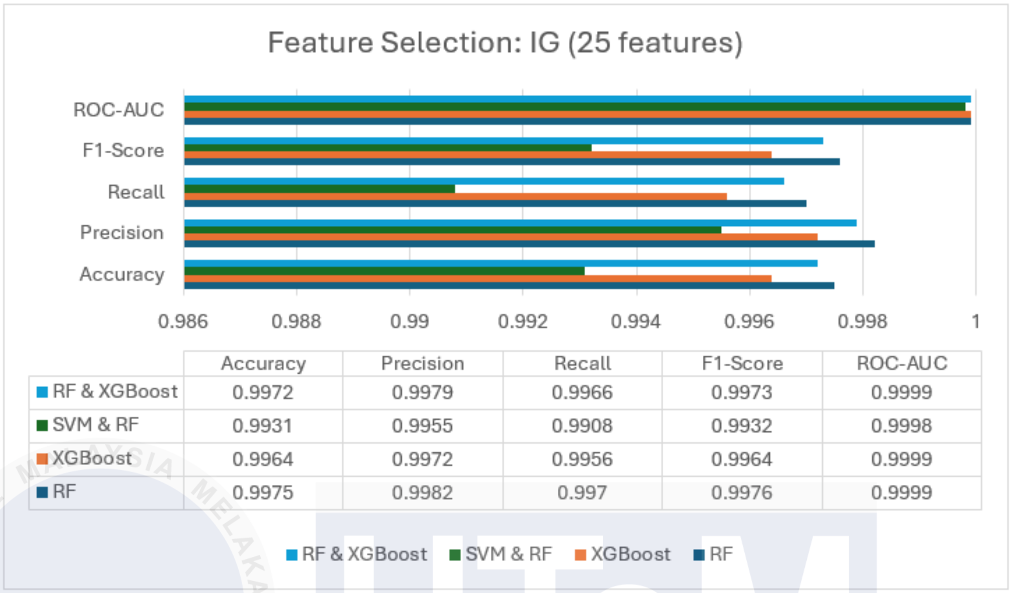
	Accuracy	Precision	Recall	F1-Score	ROC-AUC
🟦 RF & XGBoost	0.998	0.9986	0.9976	0.9981	1
🟩 SVM & RF	0.9942	0.9959	0.9926	0.9942	0.9999
🟧 XGBoost	0.9976	0.9981	0.9973	0.9977	0.9999
🟦 RF	0.9976	0.9981	0.9973	0.9977	0.9999

Figure 46: Evaluation model for IG as feature selection (67 features)

Figure 46 above present various score for the combination of approaches. The highest score was hybrid of Random Forest (RF) and XGBoost as model training, with accuracy 0.998, followed and tied by score, RF (0.9976) and XGBoost (0.9976), and lastly SVM and RF (0.9942).

The results show that the hybrid combination of RF and XGBoost outperforms other methods due to the complimentary capabilities of both algorithms. The highest accuracy is achieved using RF's ensemble learning and feature priority ranking, as well as XGBoost's gradient boosting and regularization approaches. Individual results of RF and XGBoost illustrate their usefulness, whereas the combination of SVM and RF performs well but with somewhat lower accuracy. These findings emphasize the need of combining several methods to improve model performance and resilience.

Information Gain with 25 features selected



	Accuracy	Precision	Recall	F1-Score	ROC-AUC
RF & XGBoost	0.9972	0.9979	0.9966	0.9973	0.9999
SVM & RF	0.9931	0.9955	0.9908	0.9932	0.9998
XGBoost	0.9964	0.9972	0.9956	0.9964	0.9999
RF	0.9975	0.9982	0.997	0.9976	0.9999

Figure 47: Evaluation model for IG as feature selection (25 features)

Figure 47 above present various score for the combination of approaches. The highest score was RF, with accuracy 0.9975, followed by hybrid of RF and XGBoost (0.9972), XGBoost (0.9964), and lastly SVM and RF (0.9931).

The Random Forest (RF) model outperforms other methods because to its ensemble learning capabilities, feature significance ranking, and good handling of skewed data. The hybrid combination of RF and XGBoost, which takes use of the capabilities of both algorithms, yields higher accuracy. XGBoost's great accuracy is due to its gradient boosting and regularization algorithms. The combination of SVM with RF, which is successful in high-dimensional spaces, increases the model's resistance to overfitting and its capacity to handle complicated data patterns. These findings highlight the value of combining methods to improve model performance and resilience.

Top 5 features selected from Information Gain

Table 9: Top 5 features selected from IG

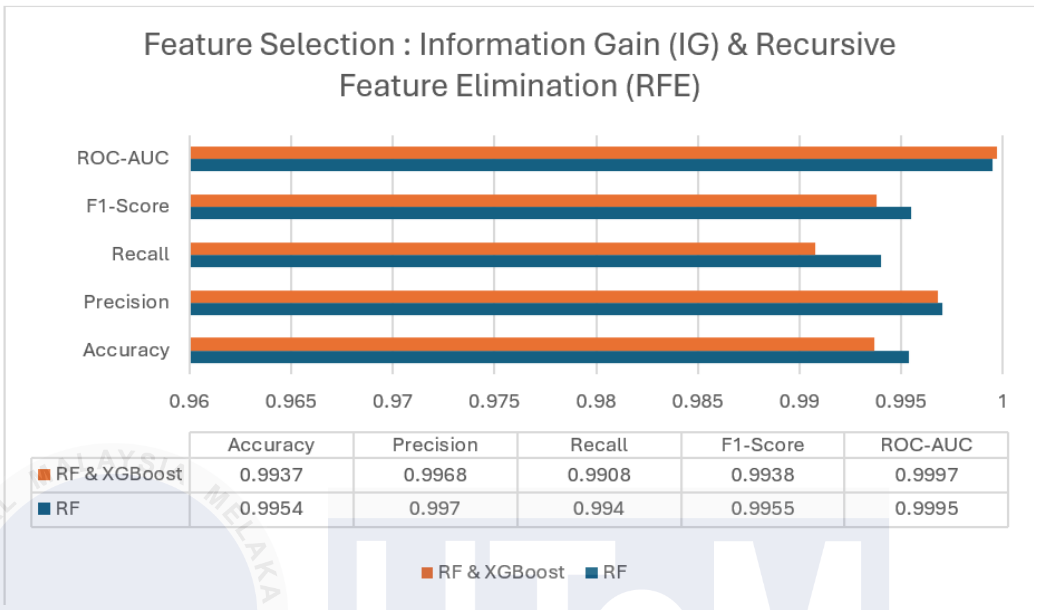
No	Features	Description

1	url_special_characters	Indicates the presence of special characters in URLs. The character often used in crafting malicious URLs.
2	url_tag_script	This feature will check the presence of ‘<script>’ Tags within URLs.
3	url_length	Measure the length of the URLs. Extremely long URLs might be sign of malicious attempt.
4	js_method_alert	Detects the use of ‘alert()’ JavaScript method, which often used as a successful sign of code injection.
5	html_length	Measures the length of HTML content. Long HTML might be a sign of malicious attempt or presence.

6.4.2 Feature Selection: Hybrid (IG and RFE)

In this section, the project conducted by using hybrid combination of Information Gain (IG) and Recursive Feature Elimination (RFE) as feature selection. There are 10 features that are selected from the process. Which are: 'url_special_characters', 'url_tag_script', 'url_cookie', 'html_tag_meta', 'html_tag_link', 'html_tag_div', 'js_method_alert', 'js_min_length', 'js_min_function_calls', 'js_string_max_length'. Meanwhile table 10 shows the top five of the features that has been selected from this process.

Figure 48 below present various score for the combination of approaches. The highest score was Random Forest (RF) as model training, with accuracy of 0.9954, followed by hybrid of RF and XGBoost (0.9937).



	Accuracy	Precision	Recall	F1-Score	ROC-AUC
🟧 RF & XGBoost	0.9937	0.9968	0.9908	0.9938	0.9997
🟦 RF	0.9954	0.997	0.994	0.9955	0.9995

Figure 48: Evaluation model for IG and RFE as feature selection

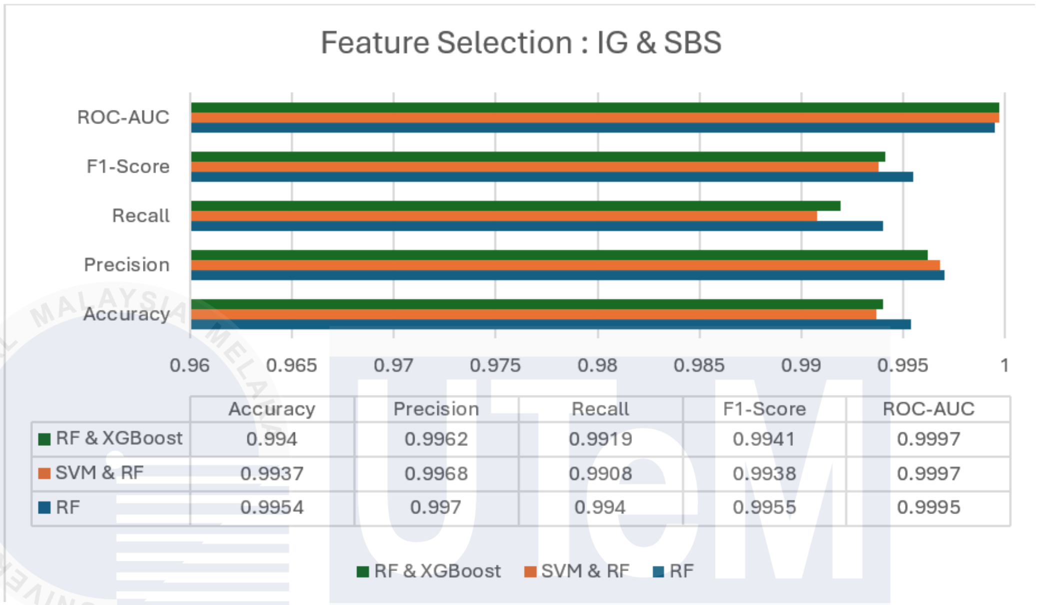
Top 5 features selected from Information Gain and Recursive Feature Elimination

Table 10: Top 5 features selected from IG and RFE

No	Features	Description
1	url_special_characters	Indicates the presence of special characters in URLs. The character often used in crafting malicious URLs.
2	js_method_alert	Detects the use of ‘alert()’ JavaScript method, which often used as a successful sign of code injection.
3	js_method_alert	Detects the use of ‘alert()’ JavaScript method, which often used as a successful sign of code injection.
4	html_attr_cite	Checks for ‘cite’ attribute in HTML, which often used to reference source for a quotation.
5	html_tag_form	Detects the presence of ‘<form>’ tags in HTML, which often used to create form for user input for the malicious target.

6.4.3 Feature Selection: IG and SBS

In this section, the project conducted by using hybrid combination of Information Gain (IG) and Sequential Backward Selection (SBS) as feature selection. There are 10 features that are selected from the process. Figure 49 below present various score for the combination of approaches. The highest score was Random Forest (RF) as model training, with accuracy 0.9954, followed by hybrid of IG and XGBoost (0.994), and RF (0.9937). Meanwhile table 11 shows the top five features that has been selected during this process.



	Accuracy	Precision	Recall	F1-Score	ROC-AUC
🟩 RF & XGBoost	0.994	0.9962	0.9919	0.9941	0.9997
🟧 SVM & RF	0.9937	0.9968	0.9908	0.9938	0.9997
🟦 RF	0.9954	0.997	0.994	0.9955	0.9995

Figure 49: Evaluation model for IG and SBS as feature selection

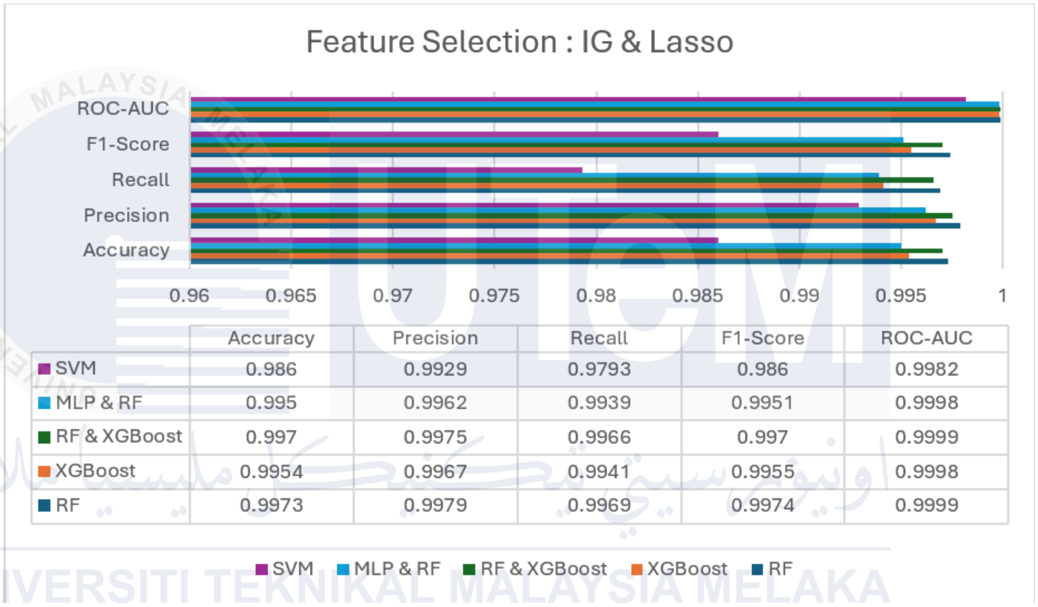
Top 5 features selected from Information Gain and Sequential Backward Selection

Table 11: Top 5 features selected from IG and SBS

No	Features	Description
1	url_length	Measure the length of the URLs. Extremely long URLs might be sign of malicious attempt.
2	url_tag_script	This feature will check the presence of '<script>' Tags within URLs.
3	html_event_onclick	Identify the presence of 'onclick' attribute in HTML, which often used to execute JavaScript when element is clicked.
4	html_tag_div	Detects the presence of '<div>' in HTML, which often used to define sections in HTML that could include hidden or malicious content.
5	html_tag_link	Identify the presence of '<link>' in HTML, which often used to define relationship of the current document and external sources.

6.4.4 Feature Selection: IG and Lasso

In this section, the project conducted by using Information Gain (IG) and Lasso as feature selection. There are 10 features that are selected from the process. Figure 50 below present various score for the combination of approaches. The highest score was Random Forest (RF) as model training, with accuracy 0.9973, followed by RF and XGBoost (0.997), XGBoost (0.9954), MLP and RF (0.995), and lastly SVM (0.986). Meanwhile table 12 shows the top five features that has been selected during this process.



	Accuracy	Precision	Recall	F1-Score	ROC-AUC
SVM	0.986	0.9929	0.9793	0.986	0.9982
MLP & RF	0.995	0.9962	0.9939	0.9951	0.9998
RF & XGBoost	0.997	0.9975	0.9966	0.997	0.9999
XGBoost	0.9954	0.9967	0.9941	0.9955	0.9998
RF	0.9973	0.9979	0.9969	0.9974	0.9999

Figure 50: Evaluation model for IG and Lasso as feature selection

Top 5 features selected from Information Gain and Lasso

Table 12: Top 5 features selected from IG and Lasso

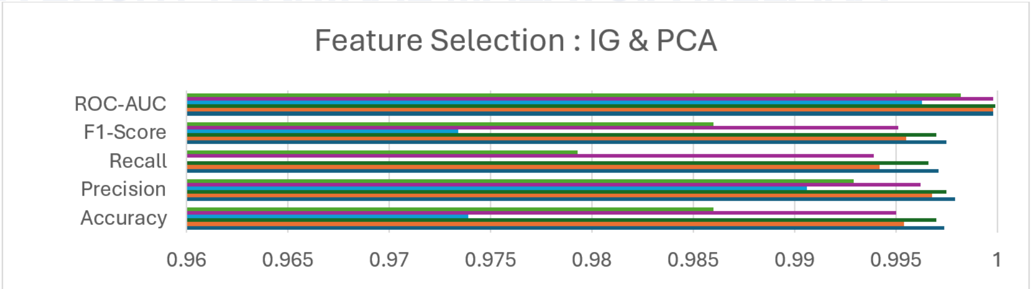
No	Features	Description
1	url_length	Measure the length of the URLs. Extremely long URLs might be sign of malicious attempt.
2	url_special_characters	Indicates the presence of special characters in URLs. The character often used in crafting malicious URLs.
3	url_number_keywords _param	Counts the number of keyword present in URL parameter, which often used to attempt malicious scripts.
4	html_tag_link	Identify the presence of '<link>' in HTML, which often used to define relationship of the current document and external sources.

5	html_tag_div	Detects the presence of '<div>' in HTML, which often used to define sections in HTML that could include hidden or malicious content.

6.4.5 Feature Selection: IG and PCA

In this section, the project conducted by using Information Gain (IG) and PCA as feature selection. There are 20 features that are selected from the process, which are: 'url_number_keywords_param', 'js_string_max_length', 'js_dom_location', 'html_attr_background', 'js_min_function_calls', 'js_file', 'js_min_length', 'url_length', 'html_attr_href', 'url_special_characters', 'html_tag_meta', 'html_tag_script', 'js_method_getElementsByTagName', 'js_method_alert', 'html_tag_div', 'html_tag_link', 'url_tag_script', 'url_cookie', 'url_number_domain', 'html_length'. Meanwhile table 13 shows the top five features that has been selected from this process.

Figure 51 below present various score for the combination of approaches. The highest score was Random Forest (RF) as model training, with accuracy 0.99734, followed by RF and XGBoost (0.997), XGBoost (0.9954), MLP and RF (0.995), SVM (0.986), and lastly LR (0.9739).



	Accuracy	Precision	Recall	F1-Score	ROC-AUC
🟩 SVM	0.986	0.9929	0.9793	0.986	0.9982
🟪 MLP & RF	0.995	0.9962	0.9939	0.9951	0.9998
🟦 LR	0.9739	0.9906	0.9568	0.9734	0.9963
🟩 RF & XGBoost	0.997	0.9975	0.9966	0.997	0.9999
🟧 XGBoost	0.9954	0.9968	0.9942	0.9955	0.9998
🟦 RF	0.9974	0.9979	0.9971	0.9975	0.9998

🟩 SVM 🟪 MLP & RF 🟦 LR 🟩 RF & XGBoost 🟧 XGBoost 🟦 RF

Figure 51: Evaluation model for IG and PCA as feature selection

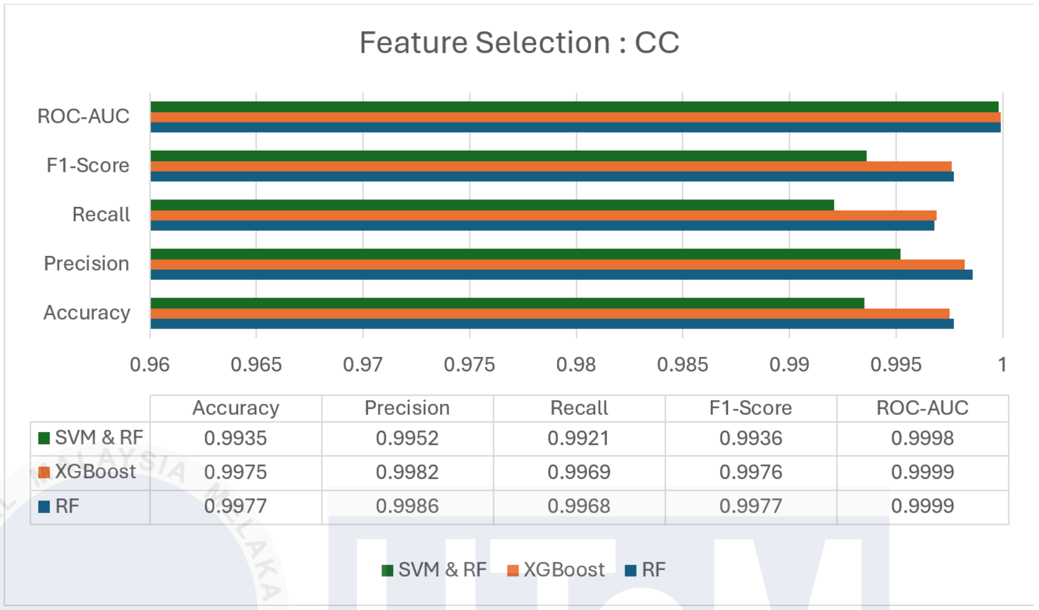
Top 5 features selected from Information Gain and PCA

Table 13: Top 5 features selected from IG and PCA

No	Features	Description
1	url_number_keywords_param	Counts the number of keyword present in URL parameter, which often used to attempt malicious scripts.
2	js_string_max_length	Measures the maximum length of the strings within JavaScript code, which could contain malicious scripts.
3	js_dom_location	Identify the use of 'location' attribute in JavaScript, which can be manipulated to redirect target victim to the malicious page.
4	html_attr_background	Checks the presence of 'background' attribute in HTML, which could be used to include external resources.
5	js_min_function_calls	Counts the minimum number of function call within the JavaScript.

6.4.6 Feature Selection: CC

In this section, the project conducted by using Correlation Coefficient (CC) as feature selection. There are 157 features that are selected from the process. Table 14 shows the top five features that has been selected from this process. Meanwhile Figure 52 present various score for the combination of approaches. The highest score was Random Forest (RF) as model training, with accuracy 0.9977, followed by XGBoost (0.9975), and lastly SVM and RF (0.9935).



	Accuracy	Precision	Recall	F1-Score	ROC-AUC
🟩 SVM & RF	0.9935	0.9952	0.9921	0.9936	0.9998
🟧 XGBoost	0.9975	0.9982	0.9969	0.9976	0.9999
🟦 RF	0.9977	0.9986	0.9968	0.9977	0.9999

Figure 52: Evaluation model for CC as feature selection

Top 5 features selected from CC

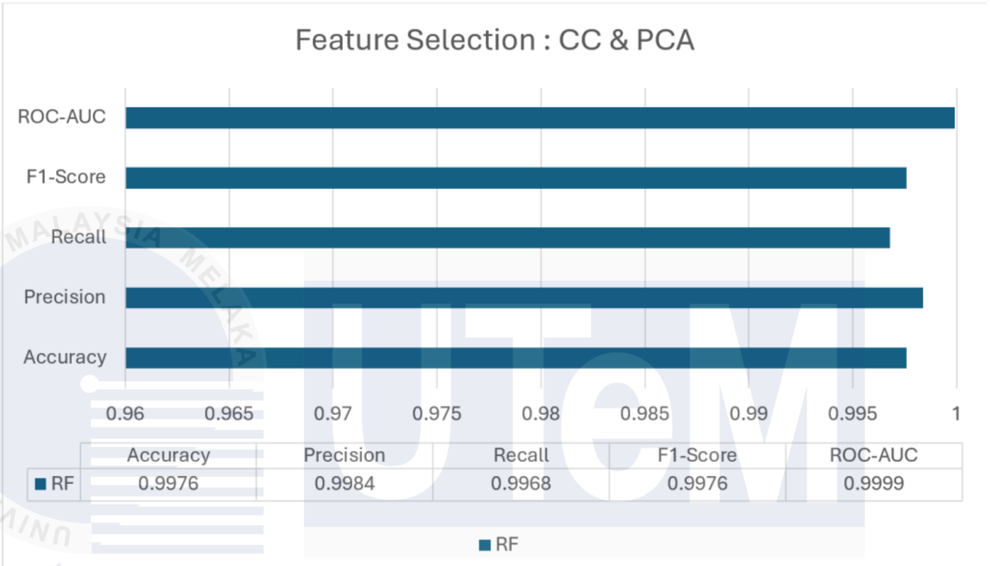
Table 14: Top 5 features selected CC

No	Features	Description
1	html_event_oninput	Identify the presence of 'oniput' attribute in HTML, which often used as the trigger to JavaScript code once any element is changes.
2	html_tag_video	Detect the presence of '<video>' in HTML, which often used as to embed video content with malicious scripts.
3	html_event_onmoused own	Check for 'onmousedown' attribute in HTML, which will trigger JavaScript code when a mouse is pressed on an element.
4	html_event_oncopy	Identify the presence of 'oncopy' attribute in HTML, which trigger JavaScript code once a content is copied to the clipboard.
5	html_event_onselect	Identify the presence of 'onselect' attribute in HTML, which could trigger the JavaScript when text within element is selected.

6.4.7 Feature Selection: CC and PCA

In this section, the project conducted by using Correlation Coefficient (CC) and PCA as feature selection. There are 50 features that are selected from the process.

Meanwhile table 15 shows the top five features that has been selected from this experiment. Figure 53 below present the score for the combination of approaches. The score was conducted for Random Forest (RF) as model training, with accuracy 0.9976, which is quite high in accuracy.



	Accuracy	Precision	Recall	F1-Score	ROC-AUC
■ RF	0.9976	0.9984	0.9968	0.9976	0.9999

Figure 53: Evaluation model for CC and PCA as feature selection

Top 5 features selected from CC and PCA

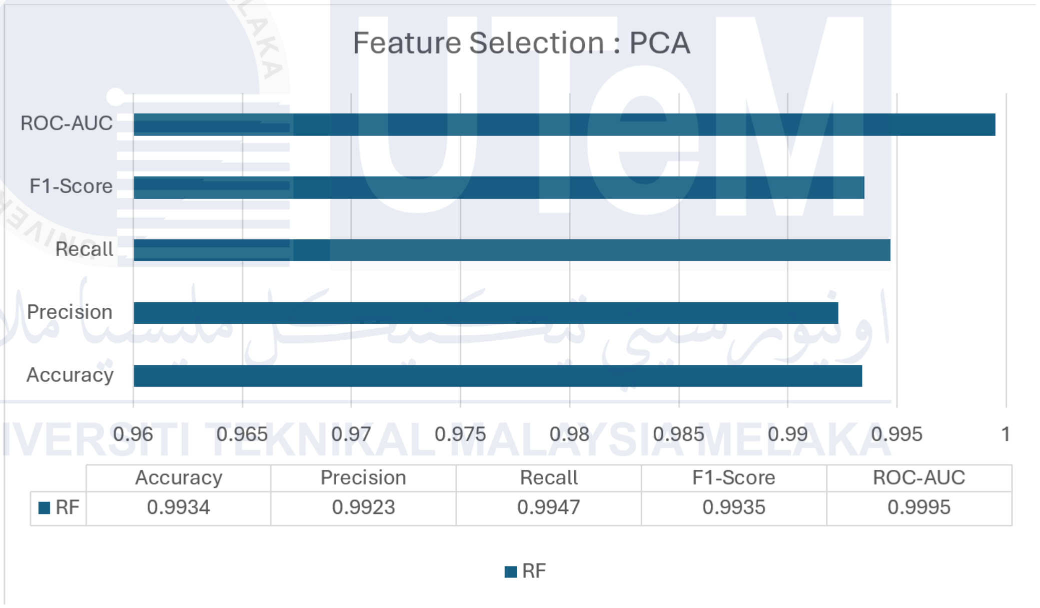
Table 15: Top 5 features selected from CC and PCA

No	Features	Description
1	url_attr_src	Detects the presence of 'src' attribute in URLs, which could be the source if an external resource.
2	html_event_onmouseleave	Identify the presence of 'onmouseleave' attribute in HTML, whihc trigger the JavaScript code once the mouse pointer leaves an element.
3	js_min_function_calls	Counts the minimum number of functions calls within JavaScript code. The higher the number, the higher the possibility of the calls to turns out to be malicious content.
4	js_min_define_function	Counts the minimum number of function definitions within JavaScript code. The higher the number, the higher the possibility of the calls to turns out to be malicious content.

5	url_tag_img	Identify the presence of '<img>' in URLs, which often used to embed image and could be illegally manipulated

6.4.8 Feature Selection: PCA

In this section, the project conducted by using Correlation Coefficient (CC) as feature selection. There are 10 features that are selected from the process. Figure 54 below present the score for the combination of approaches. The score was conducted for Random Forest (RF) as model training, with accuracy 0.9934. Table 16 then shows the top five features that has been selected from this process.



	Accuracy	Precision	Recall	F1-Score	ROC-AUC
■ RF	0.9934	0.9923	0.9947	0.9935	0.9995

Figure 54: Evaluation model for PCA as feature selection

Top 5 features selected from PCA

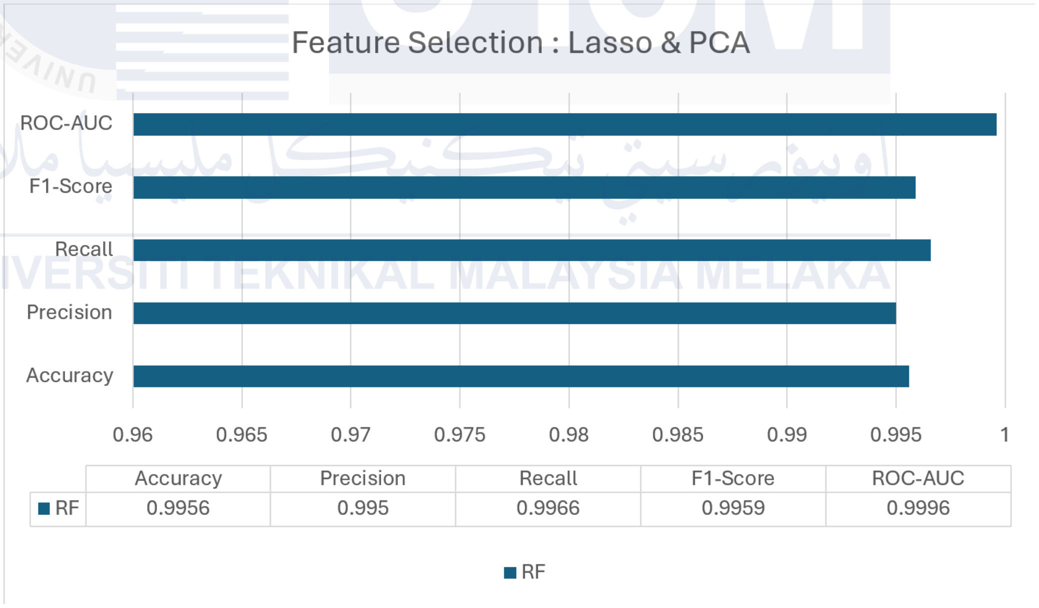
Table 16: Top 5 features selected from PCA

No	Features	Description
1	html_attr_action	Detects for 'action' attribute in HTML, which specifies the location the form is submitted that could be a way of manipulating content.
2	html_attr_archive	Identify the presence of 'archive' attribute in HTML, which specifies the location of the archive file.

3	html_attr_background	Identify the presence of the 'background' attribute in HTML, which often used to include external resources.
4	html_attr_cite	Checks the presence of 'cite' attribute in HTML, which often used as reference a source for blockqoute.
5	html_attr_classid	Identify the presence of 'classid' attribute in HTML, which often used to specify the object implementation location.

6.4.9 Feature Selection: Lasso and PCA

In this section, the project conducted by using Lasso and PCA as feature selection. There are 10 features that are selected from the process. Figure 55 below present the score for the combination of approaches. The score was conducted for Random Forest (RF) as model training, with accuracy 0.9956. Meanwhile table 17 shows the top five features that has been selected from this process.



	Accuracy	Precision	Recall	F1-Score	ROC-AUC
■ RF	0.9956	0.995	0.9966	0.9959	0.9996

Figure 55: Evaluation model for Lasso and PCA as feature selection

Top 5 features selected from Lasso and PCA

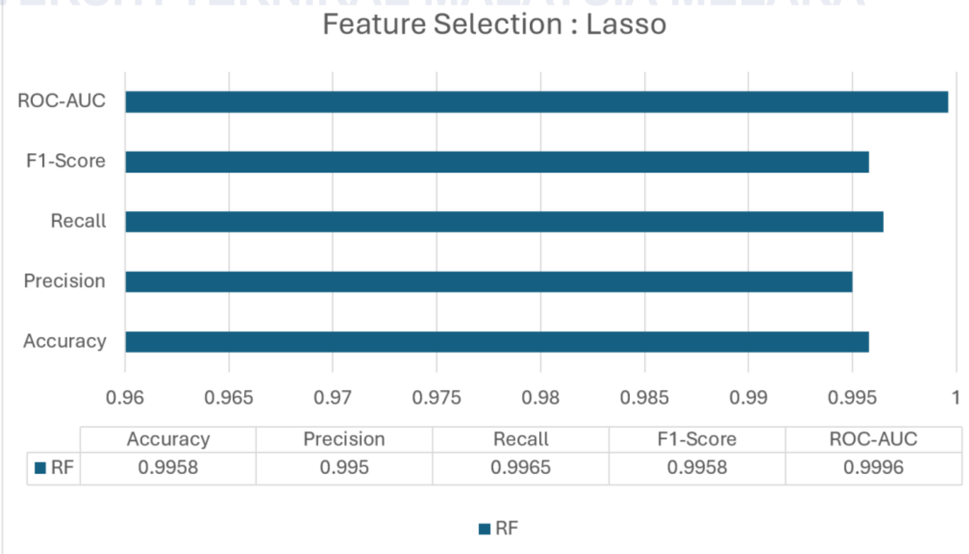
Table 17: Top 5 features selected from Lasso and PCA

No	Features	Description

1	html_length	Measures the length of HTML content. Long HTML might be a sign of malicious attempt or presence.
2	html_tag_div	Detects the presence of '<div>' in HTML, which often used to define sections in HTML that could include hidden or malicious content.
3	html_tag_link	Identify the presence of '<link>' in HTML, which often used to define relationship of the current document and external sources.
4	js_dom_location	Identify the use of 'location' attribute in JavaScript, which can be manipulated to redirect target victim to the malicious page.
5	js_file	Identify the presence of JavaScript files within the webpages, which often includes malicious content.

6.4.10 Feature Selection: Lasso

In this section, the project conducted by using Lasso as feature selection. There are 10 features that are selected from the process, and table 18 shows the top five features that has been selected from this experiment. Figure 56 below present the score for the combination of approaches. The score was conducted for Random Forest (RF) as model training, with accuracy 0.9958.



	Accuracy	Precision	Recall	F1-Score	ROC-AUC
■ RF	0.9958	0.995	0.9965	0.9958	0.9996

Figure 56: Evaluation model for Lasso as feature selection

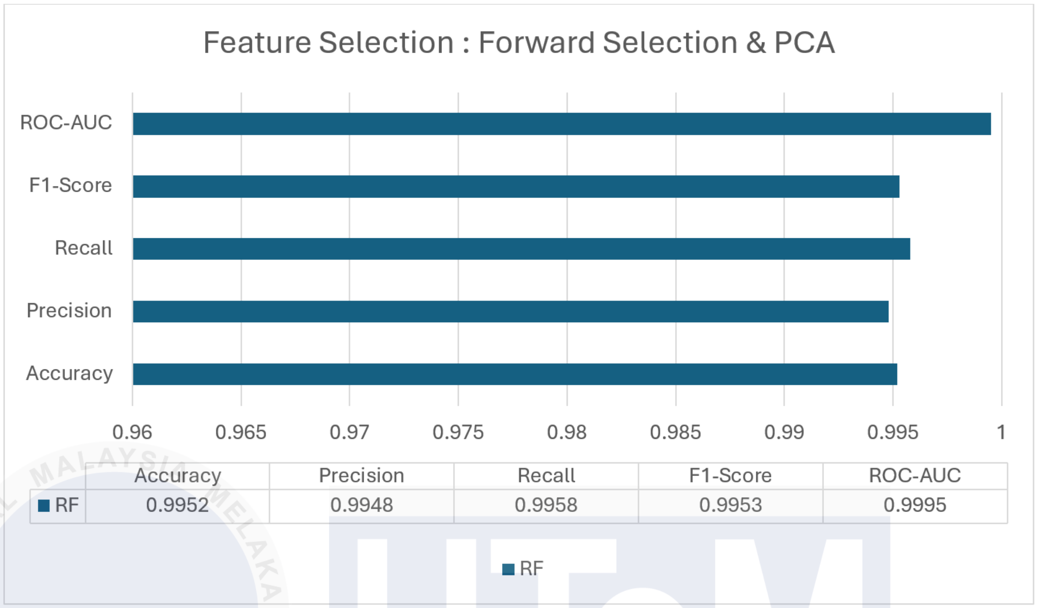
Top 5 features selected from Lasso

Table 18: Top 5 features selected from Lasso

No	Features	Description
1	url_special_characters	Indicates the presence of special characters in URLs. The character often used in crafting malicious URLs.
2	html_tag_link	Identify the presence of '<link>' in HTML, which often used to define relationship of the current document and external sources.
3	url_number_keywords _param	Counts the number of keyword present in URL parameter, which often used to attempt malicious scripts.
4	js_file	Identify the presence of JavaScript files within the webpages, which often includes malicious content.
5	js_method_getElement sByTagName	Detects the use of 'getElementByTagName' attribute within JavaScript, which often used to access tag name in HTML elements. This also allows the attackers to inject malicious content.

6.4.11 Feature Selection: Forward Selection and PCA

In this section, the project conducted by using Forward Selection and PCA as feature selection. There are 10 features that are selected from the process, and table 19 shows the top five features that has been selected from this experiment. Figure 57 below present the score for the combination of approaches. The score was conducted for Random Forest (RF) as model training, with accuracy 0.9952.



	Accuracy	Precision	Recall	F1-Score	ROC-AUC
■ RF	0.9952	0.9948	0.9958	0.9953	0.9995

Figure 57: Evaluation model for Forward Selection and PCA as feature selection

Top 5 features selected from Forward Selection and PCA

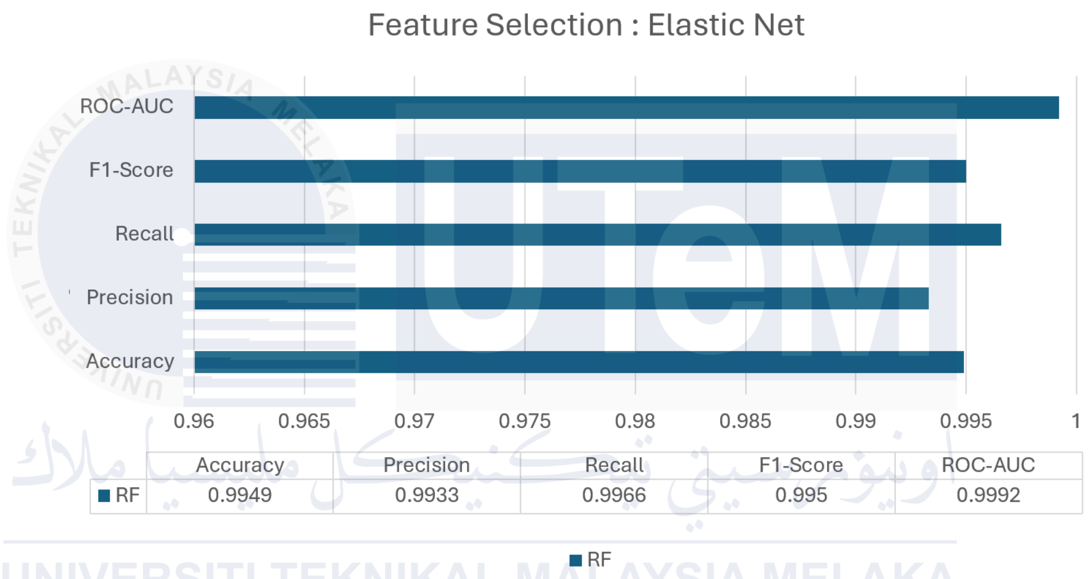
Table 19: Top 5 features selected from FS and PCA

No	Features	Description
1	url_special_characters	This feature will check the presence of '<script>' Tags within URLs.
2	url_tag_script	This feature will check the presence of '<script>' Tags within URLs.
3	url_attr_src	Detects the presence of 'src' attribute in URLs, which could be the source if an external resource.
4	url_cookie	Identify the presence of cookies in URLs, which can store session data
5	url_number_keywords _param	Counts the number of keyword present in URL parameter, which often used to attempt malicious scripts.

6.4.12 Feature Selection: Elastic Net

In this section, the project conducted by using Elastic Net as feature selection. There are 10 features that are selected from the process, which are 'url_special_characters', 'url_number_keywords_param', 'html_tag_link', 'js_file', 'js_method_getElementsByTagName', 'js_method_alert', 'html_tag_div', 'url_tag_script', 'url_attr_src', 'html_length'. Meanwhile table 20 below, shows the top five features that has been selected from this process.

Figure 58 below present the score for the combination of approaches. The score was conducted for Random Forest (RF) as model training, with accuracy 0.9949.



	Accuracy	Precision	Recall	F1-Score	ROC-AUC
■ RF	0.9949	0.9933	0.9966	0.995	0.9992

Figure 58: Evaluation model for Elastic Net as feature selection

Top 5 features selected from Elastic Net

Table 20: Top 5 features selected from Elastic Net

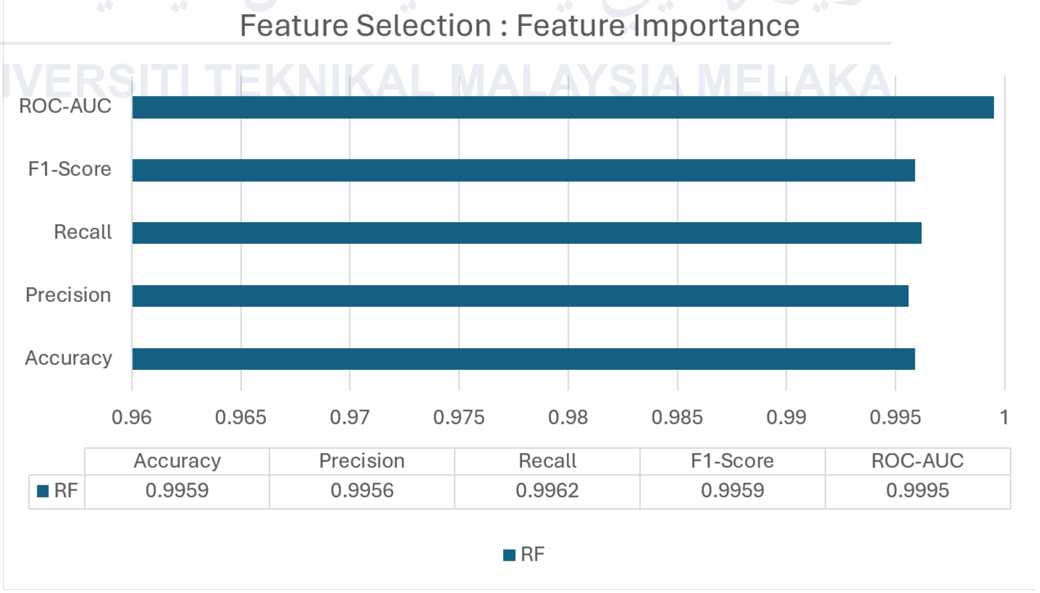
No	Features	Description
1	url_special_characters	Indicates the presence of special characters in URLs. The character often used in crafting malicious URLs.
2	url_number_keywords _param	Counts the number of keyword present in URL parameter, which often used to attempt malicious scripts.
3	html_tag_link	Identify the presence of '<link>' in HTML, which often used to define relationship of the current document and external sources.

4	js_file	Identify the presence of JavaScript files within the webpages, which often includes malicious content.
5	js_method_getElementsByTagName	Detects the use of ‘getElementByTagName’ attribute within JavaScript, which often used to access tag name in HTML elements. This also allows the attackers to inject malicious content.

6.4.13 Feature Selection: Feature Importance

In this section, the project conducted by using Feature Importance as feature selection. There are 10 features that are selected from the process, which are 'url_tag_script', 'url_special_characters', 'url_length', 'js_method_alert', 'html_tag_div', 'html_tag_link', 'js_method_getElementsByTagName', 'html_tag_meta', 'html_length', 'html_attr_href'. The top five features that has been selected from this process is displayed in table 21.

Figure 59 below present the score for the combination of approaches. The score was conducted for Random Forest (RF) as model training, with accuracy 0.9959.



	Accuracy	Precision	Recall	F1-Score	ROC-AUC
■ RF	0.9959	0.9956	0.9962	0.9959	0.9995

Figure 59: Evaluation model for Feature Importance as feature selection

Top 5 features selected from Feature Importance

Table 21: Top 5 features selected from Feature Importance

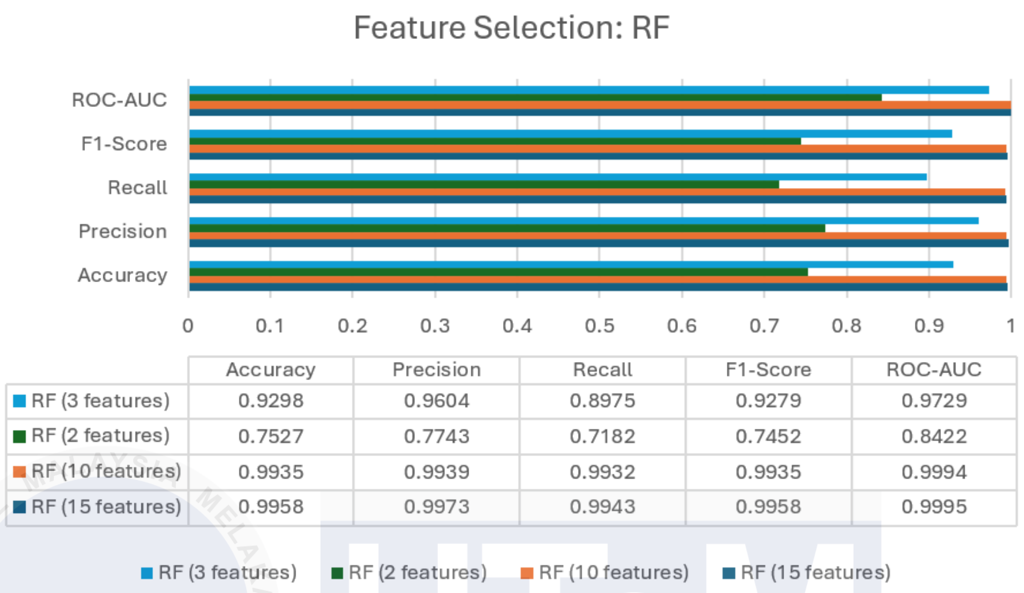
No	Features	Description
1	url_tag_script	This feature will check the presence of '<script>' Tags within URLs.
2	url_special_characters	Indicates the presence of special characters in URLs. The character often used in crafting malicious URLs.
3	url_length	Measure the length of the URLs. Extremely long URLs might be sign of malicious attempt.
4	js_method_alert	Detects the use of 'alert()' JavaScript method, which often used as a successful sign of code injection.
5	html_tag_div	Identify the presence of '<div>' attribute in HTML, which often used for structuring webpages.

6.4.14 Feature Selection: RFE

In this section, the project conducted by using RFE as feature selection. There is different approach that are using features as the key to influence the evaluation matrices value. The experiment using 2,3, 10 and 15 features for each process.

The score was conducted for Random Forest (RF) as model training, but with the present of different quantity on features selected, the score was quite different in the level of accuracy. The highest score of accuracy was lies for 15 features selected as shown in Figure 60, which is 0.9958, followed by 10 features (0.9935), 3 features (0.9298), and the last one is 2 features with 0.7527, which has the biggest gap compared to all approaches that was experimented within this paper.

This can be concluded that the lower the number of features been used in this experiment, or in general, the lower the score for the accuracy. Including more feature provides richer representation for the data training.



	Accuracy	Precision	Recall	F1-Score	ROC-AUC
🟦 RF (3 features)	0.9298	0.9604	0.8975	0.9279	0.9729
🟩 RF (2 features)	0.7527	0.7743	0.7182	0.7452	0.8422
🟧 RF (10 features)	0.9935	0.9939	0.9932	0.9935	0.9994
🟦 RF (15 features)	0.9958	0.9973	0.9943	0.9958	0.9995

Figure 60: Evaluation model for RFE as feature selection

Top 5 features selected from Recursive Feature Elimination

Table 22: Top 5 features selected from RFE

No	Features	Description
1	url_special_characters	Indicates the presence of special characters in URLs. The character often used in crafting malicious URLs.
2	url_tag_script	This feature will check the presence of '<script>' Tags within URLs.
3	url_cookie	Identify the presence of cookies in URLs, which can store session data
4	html_tag_meta	Detects the presence of '<meta>' attribute in HTML, which often used to include malicious content.
5	html_tag_svg	Identify the presence of '<svg>' attribute to define vector-based graphics, which often used to execute malicious script.

6.5 COMPARISON WITH EXISTING WORKS

In this section, we briefly explain and compare the techniques used and result by previous work and this project's approach. As shown in Table 23, we compare two works that used the same dataset with this project, to ensure a fair comparison. As for the first paper authored by Alhamyani R., Alshammari M. (2024), they utilized Information Gain

(IG) as feature selection techniques and select 25 features, achieving 0.9978 accuracy using Random Forest algorithm (RF). Next is the paper written by Mokbal F., Dan W., Xioaxi W., Wenbin Z., Lihua F. (2021), where they use hybrid combination of feature selection methods which are Information Gain (IG) and Sequential Backward Selection (SBS), selecting 67 of features from the dataset. In result, they achieve 0.9950 of accuracy by utilizing XGBoost as model training.

In this project, we achieve several result with highest accuracy, all are using Information Gain (IG) as feature selection method. The highest was utilizing the hybrid combination of two powerful model training, which are Random Forest algorithm (RF) and XGBoost, as a result, this approach select 167 features for continuation to complete the process, and achieve the highest accuracy of 0.9981. Next approach is quite similar with the first one, however, it select only 67 features to be train with hybrid model training, Random Forest (RF) and XGBoost, achieving 0.998 in accuracy.

Next approach is utilizing model training Random Forest, with 167 features selected, achieving 0.9979 in accuracy. Lastly is utilizing XGBoost as the model training, with also selecting 167 of the features. This approach resulting in 0.9978 of accuracy.

Table 23: Result comparison with previous work

References	**Method of Feature Selection**	**No of Selected Features**	**Best Algorithm**	**Accuracy**
Alhamyani R., Alshammari M., 2024	IG	25	Random Forest	0.9978
Mokbal F., Dan W., Xioaxi W., Wenbin Z., Lihua F., 2021	IG & SBS	67	XGBoost	0.9950
This project proposed approach	IG	167	Random Forest & XGBoost	0.9981
	IG	67	Random Forest & XGBoost	0.998
	IG	167	Random Forest	0.9979
	IG	167	XGBoost	0.9978

6.6 SUMMARIZATION

Combining Random Forest (RF) and XGBoost in a hybrid model offers several advantages that significantly enhance model performance and robustness. RF, an ensemble method that combines multiple decision trees, improves accuracy and reduces overfitting by averaging the results of different trees trained on various data subsets. It effectively handles complex datasets and provides insights into feature importance, making it suitable for real-world applications with high data volumes. XGBoost, known for its high performance and efficiency, uses gradient boosting to optimize model accuracy and includes regularization techniques to prevent overfitting. Its ability to handle imbalanced datasets and focus on hard-to-classify examples further enhances its robustness.

The hybrid combination of RF and XGBoost leverages the strengths of both algorithms, resulting in improved accuracy, robustness, and efficiency. RF's capability to manage noisy data and provide robust predictions through ensemble learning complements XGBoost's optimization and regularization techniques. This synergy enhances the model's ability to detect XSS threats accurately and efficiently, providing a comprehensive solution for web security. By integrating these methods, the hybrid model ensures better generalization to new data, making it a powerful tool for various machine learning tasks, including classification, regression, and ranking.

6.7 CONCLUSION

In this chapter of Testing and Analysis, we conducted various approach of feature selection methods and model training to identify the most effective techniques for Cross-Site Scripting (XSS) detection. Our finding displayed that the integration of Information Gain (IG) as feature selection method, followed by the hybrid combination of Random Forest algorithm and XGBoost, resulting in the highest accuracy, which are a superior and powerful performance. The result is closely monitored through the evaluation metrices such as accuracy, precision, recall, f1-score, and ROC-AUC.

CHAPTER 7: PROJECT CONCLUSION

7.1 INTRODUCTION

This chapter will give a thorough summary of the whole project, beginning with the way it was started and developed up until this point in the chapter. We will discuss the project's contributions to accomplishing each of the suggested goals. Next, we will be discussing about all the obstacles and limitations that arose while the project was being carried out. Lastly, we will discuss future work that may be done to address and improve the project's constraints.

7.2 PROJECT SUMMARIZATION

This project proposed enhancement to the existed Cross-Site Scripting (XSS) Detection method. This project focusing on the development of the improvise approach, implying variety combination feature selection methods and the model training, that will help in achieving the best result and scores in detection rate. The implementation of Information Gain for the feature selection method are seen and proved to have the best and optimal techniques, which achieving the accuracy of 0.9981 when using all the features. However, it shows a slightly low accuracy rate when fewer number of features were selected. Information Gain has proved to significantly enhance the performance of the overall performance by ensures only the most informative features are being selected, thus increase the speed in training process and avoid overfitting while training the model.

Information Gain are resulting in a much great and powerful performance after combined with the hybrid combination of the model training approach, which is Random Forest Classifier and XGBoost. The resilience and performance of machine learning are improved when Random Forest (RF) and XGBoost are combined into a hybrid model. It is scalable and economical, works well in a variety of domains, including text mining and picture classification, and enhances accuracy by avoiding overfitting, handling unbalanced data, and offering insights into the significance of features.

The dataset was initially collected from the GitHub sources, then underwent 43 different approaches in this project. These approaches were experimented as to check their accuracy compared to the previous existed work. To evaluate each approach, there are several performance metrices that has been closely monitored in achieving the best result of performance, which are accuracy, precision, recall, f1-score, ROC-AUC, and confusion matrix. In summary, the proposed project, which utilizes machine learning for the detection of Cross-Site Scripting (XSS) attack has achieved optimal accuracy of 0.9981 and resulting in a great performance when compared to the previous research.

7.3 PROJECT CONTRIBUTION

In the context of Cross-Site Scripting (XSS) Detection in the digital era, especially towards the web applications, it is crucial to have a concrete and robust system to detect and avoid any attack that occurred. As the uses of website application has gradually increased, they often contain of great number of privacy and sensitive details and information of the user, such as phone numbers, email addresses, bank details, and much more. Without effective detection and defending mechanisms, this will just bring a 'party' for the hackers, as this matter will simplify their works in pirating and stealing all the information. Therefore, the proposed approach in this project will definitely be a great help in this industry, which offering a good performance in feature selection and model training approaches.

7.4 PROJECT LIMITATION

One of the project limitation or challenges is matter towards the confusion matrix scores. Since the scores is actually influence the performance evaluation like accuracy, sometimes the displayed accuracy might be slightly different or not entirely accurate.

7.5 FUTURE WORKS

For the future works, it should focus more to the influence of confusion matrix scores on performance evaluation metrics such as accuracy, as understanding this can aid in a better evaluating the model's performance. Furthermore, addressing on issues of overfitting and underfitting will improve the model's generalization capabilities. Finally, exploring the idea to integrate the proposed method into an application or scanner to effectively detecting cross-site scripting (XSS) attacks. This might be creating a browser extension, standalone scanner, web application firewall, or even API service for real-time security and detection.

REFERENCES

A defensive framework for reflected XSS in Client-Side applications. (2022, October 1). River Publishers Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/document/10246948

A detailed survey on recent XSS Web-Attacks Machine Learning detection techniques. (2021, October 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/9587569

Alaoui, R. L., & Nfaoui, E. H. (2022b). Deep Learning for Vulnerability and attack Detection on Web Applications: A Systematic Literature Review. *Future Internet*, *14*(4), 118. https://doi.org/10.3390/fi14040118

Alhamyani, R., & Alshammari, M. (2024). Machine Learning-Driven detection of Cross-Site scripting attacks. *Information*, *15*(7), 420. https://doi.org/10.3390/info15070420

Detecting Cross-Site Scripting Attack using Machine Learning Algorithms. (2024, February 28). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/10499119

Detection of XSS in web applications using Machine Learning Classifiers. (2020, October 2). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/9270052

Guan, Y. (2022, May 31). Network Security Trends: November 2021 to January 2022. *Unit 42*. https://unit42.paloaltonetworks.com/network-security-trends-cross-site- scripting/

Insights2Techinfo. (2023b, November 15). *The Future of Web Security: XSS Detection through Machine Learning*. https://insights2techinfo.com/the-future-of-web- security-xss-detection-through-machine-learning/

Ivanova, M., & Rozeva, A. (2021). Detection of XSS Attack and Defense of REST Web Service – Machine Learning Perspective. *Detection of XSS Attack and Defense of REST Web Service – Machine Learning Perspective*. https://doi.org/10.1145/3453800.3453805

Kazemian, H., & Ahmed, S. (2015). Comparisons of machine learning techniques for detecting malicious webpages. *Expert Systems With Applications*, *42*(3), 1166–1177. https://doi.org/10.1016/j.eswa.2014.08.046

Machine Learning based Cross-Site Scripting detection in online social network. (2014, August 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/7056839

MLPXSS: An integrated XSS-Based attack detection scheme in web applications using multilayer perceptron technique. (2019b). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/8756243

Mokbal, F. M. M., Dan, W., Xiaoxi, W., Wenbin, Z., & Lihua, F. (2021). XGBXSS: an extreme gradient boosting detection framework for Cross-Site scripting attacks based on hybrid feature selection approach and parameters optimization. Journal of Information Security and Applications, 58, 102813. https://doi.org/10.1016/j.jisa.2021.102813

Nagarjun, P., & Shakeel, S. (2020). Cross-site Scripting Research: a review. *International Journal of Advanced Computer Science and Applications/International Journal of Advanced Computer Science & Applications*, *11*(4). https://doi.org/10.14569/ijacsa.2020.0110481

Sarker, I. H. (2021). Machine learning: algorithms, Real-World applications and research directions. *SN Computer Science/SN Computer Science*, *2*(3). https://doi.org/10.1007/s42979-021-00592-x

Shahid, M. (2023b, April 27). *Machine learning for detection and mitigation of web vulnerabilities and web attacks*. arXiv.org. https://arxiv.org/abs/2304.14451

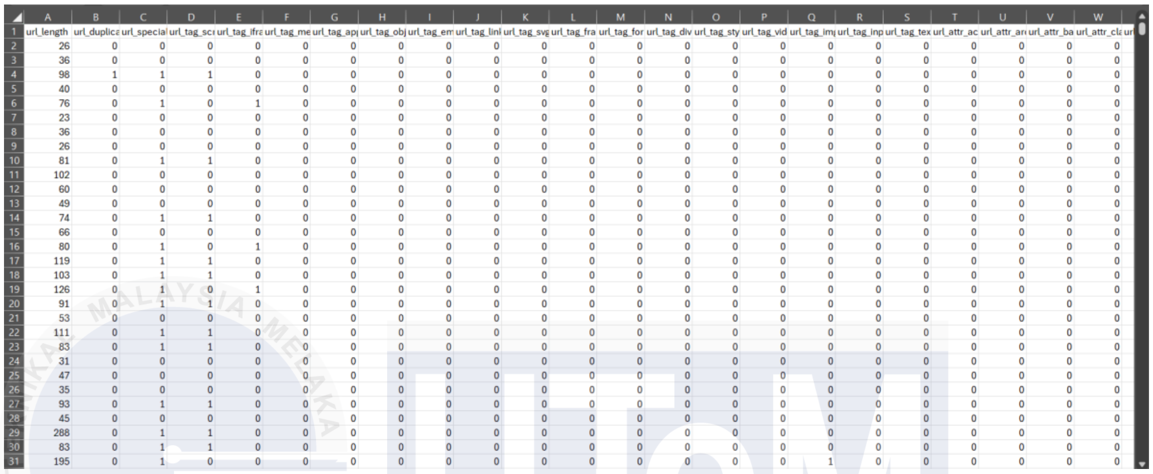
Thajeel, I. K., Hashim, F., Samsudin, K.., & Hashim, S. J. (2023, June). *Machine and Deep Learning-based XSS Detection Approaches: A Systematic literature review*. https://www.researchgate.net/publication/371731791_Machine_and_Deep_Le arn ing-based_XSS_Detection_Approaches_A_Systematic_Literature_Review

Vishnu, B. A., & Jevitha, K. P. (2014). Prediction of Cross-Site Scripting Attack Using Machine Learning Algorithms. *Prediction of Cross-Site Scripting Attack Using Machine Learning Algorithms*. https://doi.org/10.1145/2660859.2660969

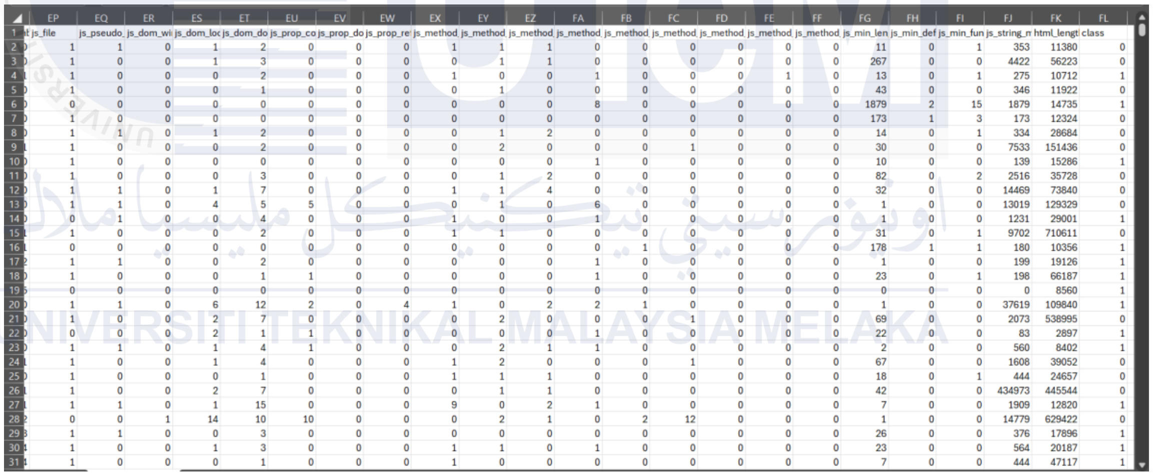
Web Server Attack Detection using Machine Learning. (2020, October 20). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/9292393

APPENDIX

Dataset from GitHub



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	url_length	url_duplica	url_special	url_tag_sc	url_tag_ifra	url_tag_me	url_tag_ap	url_tag_obj	url_tag_em	url_tag_link	url_tag_svg	url_tag_fra	url_tag_for	url_tag_div	url_tag_sty	url_tag_vid	url_tag_im	url_tag_inp	url_tag_tex	url_attr_ac	url_attr_ar	url_attr_ba	url_attr_cl
2	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	98	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	76	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	81	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	102	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	49	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	74	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	66	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	80	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	119	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	103	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	126	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	91	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	53	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	111	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	83	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	93	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	288	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	83	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	195	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0



	EP	EQ	ER	ES	ET	EU	EV	EW	EX	EY	EZ	FA	FB	FC	FD	FE	FF	FG	FH	FI	FJ	FK	FL
1	js_file	js_pseudo_	js_dom_wi	js_dom_loc	js_dom_do	js_prop_co	js_prop_do	js_prop_re	js_method_	js_method_	js_method_	js_method_	js_method_	js_method_	js_method_	js_method_	js_method_	js_min_len	js_min_def	js_min_fun	js_string_m	html_lengt	class
2	1	1	0	1	2	0	0	0	1	1	1	0	0	0	0	0	0	11	0	1	353	11380	0
3	1	0	0	1	3	0	0	0	0	1	1	0	0	0	0	0	0	267	0	0	4422	56223	0
4	1	0	0	0	2	0	0	0	1	0	0	1	0	0	0	1	0	13	0	1	275	10712	1
5	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	43	0	0	346	11922	0
6	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	1879	2	15	1879	14735	1
7	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	173	1	3	173	12324	0
8	1	1	0	1	2	0	0	0	0	1	2	0	0	0	0	0	0	14	0	1	334	28684	0
9	1	0	0	0	2	0	0	0	0	2	0	0	0	1	0	0	0	30	0	0	7533	151436	0
10	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	10	0	0	139	15286	1
11	1	0	0	0	3	0	0	0	0	1	2	0	0	0	0	0	0	82	0	2	2516	35728	0
12	1	1	0	1	7	0	0	0	1	1	4	0	0	0	0	0	0	32	0	0	14469	73840	0
13	1	1	0	4	5	5	0	0	0	1	0	6	0	0	0	0	0	1	0	0	13019	129329	0
14	0	1	0	0	4	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	1231	29001	1
15	1	0	0	0	2	0	0	0	1	1	0	0	0	0	0	0	0	31	0	1	9702	710611	0
16	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	178	1	1	180	10356	1
17	1	1	0	0	2	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	199	19126	1
18	1	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	23	0	1	198	66187	1
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8560	1
20	1	1	0	6	12	2	0	4	1	0	2	2	1	0	0	0	0	1	0	0	37619	109840	1
21	1	0	0	2	7	0	0	0	0	2	0	0	0	1	0	0	0	69	0	0	2073	538995	0
22	0	0	0	2	1	1	0	0	0	0	0	1	0	0	0	0	0	22	0	0	83	2897	1
23	1	1	0	1	4	1	0	0	0	2	1	1	0	0	0	0	0	2	0	0	560	8402	1
24	1	0	0	1	4	0	0	0	1	2	0	0	0	1	0	0	0	67	0	0	1608	39052	0
25	1	0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	0	18	0	1	444	24657	0
26	1	0	0	2	7	0	0	0	0	1	1	0	0	0	0	0	0	42	0	0	434973	445544	0
27	1	1	0	1	15	0	0	0	9	0	2	1	0	0	0	0	0	7	0	0	1909	12820	1
28	0	0	1	14	10	10	0	0	0	2	1	0	2	12	0	0	0	1	0	0	14779	629422	0
29	1	1	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	26	0	0	376	17896	1
30	1	0	0	1	3	0	0	0	1	1	0	1	0	0	0	0	0	23	0	0	564	20187	1
31	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	7	0	0	444	47117	1