

**DETECTING PHISHING UNIFORM RESOURCE LOCATOR(URL) BY  
USING MACHINE LEARNING TECHNIQUES**



**UNIVERSITI TEKNIKAL MALAYSIA MELAKA**

DETECTING PHISHING UNIFORM RESOURCE LOCATOR(URL) BY USING  
MACHINE LEARNING TECHNIQUES

LIM CHIAN FANG



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

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY  
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2020/2021

## DECLARATION

I hereby declare that this project report entitled  
**DETECTING PHISHING UNIFORM RESOURCE LOCATOR(URL) BY USING  
MACHINE LEARNING TECHNIQUE**

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

STUDENT : \_\_\_\_\_ LIM CHIAN FANG \_\_\_\_\_ Date : 05/09/2021



UNIVERSITI TEKNIKAL MALAYSIA MELAKA

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.

SUPERVISOR : \_\_\_\_\_ ZAKIAH BINTI AYOP \_\_\_\_\_ Date : 05/09/2021

## DEDICATION

I would like to dedicate my work to my parents, who have always been a source of inspiration and determination for me, and who continue to support me morally, spiritually, and financially.

I also dedicate this dissertation to my brother, sister, mentor, and friends, who have always encouraged and supported me to finish my research.



## ACKNOWLEDGEMENTS

First and foremost, I want to express my gratitude to Universiti Teknikal Malaysia Melaka (UTeM) and the Faculty of Information and Communication Technology for providing me with an excellent opportunity to apply what I had learned to develop a project.

In addition, I would like to express my great appreciation and unlimited thanks to my supervisor, Madam Zakiah Binti Ayop for taking part in the useful decision and giving necessary advice and guidance during the planning and implementation of this project. Despite being extremely busy with her duties, she took the time to share her knowledge, guide, and keep me on track throughout the project. I was lucky to get consistent encouragement, support, and advice, which enabled me to complete my project sarjana muda successfully.

I would like to express my gratitude to my family and friends for their encouragement and support throughout the project's development. Hence, I would also like to thank everyone who involves directly or indirectly in the project. Without the support of the persons mentioned above, I may not have completed this project successfully.

## ABSTRACT

As the internet has grown in popularity, phishing websites have become more common and caused significant harm to online financial services such as online shopping and data security. Phishing is a type of fraud whereby an attacker sends a fake message or creates a phishing website to mislead web users into sharing confidential information or allowing malicious software to be installed on the victim's device. Many attackers started creating phishing websites to misled web users into thinking it's legitimate. So, web users may be exposed to common web attacks, which might result in the loss of money, personal information, and trust from online transactions. Hence, detecting phishing websites has become a critical task that requires more examination. The most commonly used blacklist- and whitelist-based methods have shown to be ineffective. Researchers have looked into using machine learning models to detect and prevent phishing attempts. The accuracy of the prediction can be increased using machine learning methods. CatBoost based URL classifiers for detecting phishing websites are proposed in this project. The first stage is dataset will be split to 80:20 ratio to train machine learning model. The second stage involves the comparison of 3 machine learning algorithms (Logistic Regression, Random Forest, and CatBoost) the third stage involves classification of the URL's legitimacy by using CatBoost. As a result, the URL will be classified as either a phishing or a legitimate URL.

## ABSTRAK

Dengan pertumbuhan internet, pancingan data melalui laman web menjadi peristiwa yang biasa dan membawa kesan negatif kepada perkhidmatan kewangan dalam talian seperti membeli-belah dalam talian dan keselamatan data. Phishing adalah satu bentuk penipuan di mana penggodam menghantar mesej palsu atau membuat laman web untuk mengelirukan pengguna web agar berkongsi maklumat sulit atau membenarkan perisian jahat dipasang pada peranti mangsa. Banyak penggodam mula membuat laman web palsu untuk mengelirukan pengguna web sehingga pengguna web menganggap laman web sah. Oleh itu, pengguna web mungkin terdedah kepada serangan web biasa, yang mungkin mengakibatkan kehilangan wang, maklumat peribadi, dan kepercayaan dari transaksi dalam talian. Oleh itu, mengesan laman web pancingan data telah menjadi tugas penting yang memerlukan lebih banyak pemeriksaan. Kaedah berdasarkan senarai hitam dan senarai putih yang paling biasa terbukti tidak berkesan. Penyelidik mengaji kaedah pembelajaran mesin untuk meramalkan dan mencegah pancingan data. Ketepatan ramalan dapat ditingkatkan dengan menggunakan kaedah pembelajaran mesin. Pengasingan URL berasaskan CatBoost untuk mengesan laman web pancingan data dicadangkan dalam projek ini. Langkah pertama adalah set data akan berpecah kepada 80:20 nisbah untuk melatih model pembelajaran mesin. Tahap kedua melibatkan perbandingan 3 algoritma pembelajaran mesin (Logistic Regression, Random Forest, dan CatBoost) tahap ketiga melibatkan klasifikasi kesahan URL dengan menggunakan CatBoost. Akibatnya, URL akan diklasifikasikan sebagai phishing atau URL yang sah.

## TABLE OF CONTENTS

<b>DECLARATION.....</b>	<b>II</b>
<b>DEDICATION.....</b>	<b>III</b>
<b>ACKNOWLEDGEMENTS.....</b>	<b>IV</b>
<b>ABSTRACT.....</b>	<b>V</b>
<b>ABSTRAK .....</b>	<b>VI</b>
<b>TABLE OF CONTENTS.....</b>	<b>VII</b>
<b>LIST OF TABLES .....</b>	<b>X</b>
<b>LIST OF FIGURES .....</b>	<b>XI</b>
<b>LIST OF ABBREVIATIONS .....</b>	<b>XIV</b>
<b>LIST OF ATTACHMENTS.....</b>	<b>XV</b>
<b>CHAPTER 1: INTRODUCTION.....</b>	<b>1</b>
1.1 Introduction.....	1
1.2 Project Background.....	2
1.3 Problem Statement .....	3
1.4 Project Question.....	4
1.5 Objective .....	4
1.6 Scope.....	5
1.7 Project Contribution.....	5
1.8 Report Organization.....	5
1.8.1 Chapter 1: Introduction.....	6



1.8.2	Chapter 2: Literature Review.....	6
1.8.3	Chapter 3: Project Methodology.....	6
1.8.4	Chapter 4: Implementation .....	6
1.8.5	Chapter 5: Testing & Analysis .....	6
1.8.6	Chapter 6: Project Conclusion.....	6
1.9	Conclusion .....	7
<b>CHAPTER 2: LITERATURE REVIEW AND PROJECT METHODOLOGY . 8</b>		
2.1	Introduction.....	8
2.2	Related Work .....	8
2.2.1	Heuristics approaches .....	9
2.2.2	Machine learning approaches .....	10
2.3	Critical review of existing algorithms/ techniques .....	12
2.4	Project solution .....	19
2.4.1	Logistic Regression .....	20
2.4.2	Random Forest.....	22
2.4.3	CatBoost.....	24
2.5	Conclusion .....	27
<b>CHAPTER 3: PROJECT METHODOLOGY ..... 28</b>		
3.1	Introduction.....	28
3.2	Methodology .....	28
3.3	Research Milestone .....	41
3.4	Conclusion .....	46
<b>CHAPTER 4: IMPLEMENTATION..... 47</b>		
4.1	Introduction.....	47

4.2	Implementation Steps.....	47
4.2.1	Dataset .....	48
4.2.2	Comparison.....	50
4.2.3	Classification .....	52
4.3	Conclusion .....	67
<b>CHAPTER 5: TESTING AND ANALYSIS .....</b>		<b>68</b>
5.1	Introduction.....	68
5.2	Result and Analysis.....	68
5.2.1	Result of Machine Learning Algorithms .....	71
5.2.2	Testing and Analysis Result in Graphs.....	73
5.2.3	Classification of URL.....	76
5.3	Testing Result Summary.....	83
5.4	Conclusion .....	85
<b>CHAPTER 6: RESEARCH CONCLUSION .....</b>		<b>87</b>
6.1	Introduction.....	87
6.2	Research Summarization .....	87
6.3	Research Contribution .....	88
6.4	Research Limitation .....	88
6.5	Future Works .....	89
6.6	Conclusion .....	89
<b>REFERENCES.....</b>		<b>90</b>

## LIST OF TABLES

	PAGE
Table 1.1: Project Question.....	4
Table 1.2: Objectives .....	4
Table 2.1: Comparison of the method used of previous work.....	17
Table 3.1: Feature Extraction.....	31
Table 3.2: Research Milestone of PSM 1 .....	42
Table 3.3: Research Milestone of PSM 2 .....	44
Table 5.1: Evaluation Table .....	83
Table 5.2: CatBoost's Classification Result.....	85

## LIST OF FIGURES

	PAGE
Figure 2.1 Sigmoid Function Graph .....	21
Figure 2.2 Hypothesis of logistic regression.....	22
Figure 2.3 How Random Forest Works .....	23
Figure 2.4 One Hot Encoding .....	25
Figure 2.5 OB Principle and Pseudocode .....	26
Figure 2.6 Building a tree in CatBoost.....	27
Figure 3.1 Flowchart for phishing URL detection.....	29
Figure 3.2 Gantt Chart of PSM 1.....	45
Figure 3.3 Gantt Chart PSM 2 .....	45
Figure 4.1 Block Flow Diagram .....	48
Figure 4.2 UCI Machine Learning Repository .....	48
Figure 4.3 Dataset of Phishing URL.....	49
Figure 4.4 Split Dataset.....	49
Figure 4.5 Logistic Regression Model.....	50
Figure 4.6 Random Forest Model .....	51
Figure 4.7 CatBoost Model.....	51
Figure 4.8 Confusion Matrix.....	52
Figure 4.9 URL Connection with Spyder .....	52
Figure 4.10 IP address, Long URL, and Tiny URL.....	53
Figure 4.11 AT symbol, redirecting using “//, prefix suffix separation, subdomains, and SSL .....	55
Figure 4.12 Domain Registration Length.....	56
Figure 4.13 Favicon, Non-Standard Port and Https Token.....	57
Figure 4.14 Request URL .....	58

Figure 4.15 URL of Anchor .....	58
Figure 4.16 Links in tags.....	59
Figure 4.17 SFH, Email Submit, and Abnormal URL.....	60
Figure 4.18 Redirect, onMouseover, Right Click, Popup Window, and Iframe.....	61
Figure 4.19 Age of Domain .....	62
Figure 4.20 DNS Record and Website Traffic.....	63
Figure 4.21 PageRank, Google Index, and Links Pointing to Page.....	64
Figure 4.22 Statistical Report.....	64
Figure 4.23 Lists of figures .....	65
Figure 4.24 Feature Extraction.....	66
Figure 4.25 Store Data in Excel .....	66
Figure 4.26 CatBoost's Classification.....	67
Figure 5.1 Values of Confusion Matrix .....	69
Figure 5.2 Formula of Accuracy .....	69
Figure 5.3 Formula of Precision.....	70
Figure 5.4 Formula of Recall .....	70
Figure 5.5 Formula of F1-score .....	70
Figure 5.6 Logistic Regression's Analysis Result .....	71
Figure 5.7 Random Forest's Analysis Result.....	72
Figure 5.8 CatBoost's Analysis Result.....	73
Figure 5.9 Graph of Detection Accuracy .....	74
Figure 5.10 Graph of Precision .....	74
Figure 5.11 Graph of Recall.....	75
Figure 5.12 Graph of F1-score .....	76
Figure 5.13 Test Case 1 .....	76
Figure 5.14 Result of Test Case 1 .....	77
Figure 5.15 Test Case 2.....	77
Figure 5.16 Result of Test Case 2 .....	77
Figure 5.17 Test Case 3.....	78
Figure 5.18 Result of Test Case 3 .....	78
Figure 5.19 Test Case 4.....	78
Figure 5.20 Result of Test Case 4 .....	79
Figure 5.19 Test Case 5.....	79
Figure 5.20 Result of Test Case 5 .....	79

Figure 5.21 Test Case 6.....	80
Figure 5.22 Result of Test Case 6 .....	80
Figure 5.23 Test Case 7.....	80
Figure 5.24 Result of Test Case 7 .....	80
Figure 5.25 Test Case 8.....	81
Figure 5.26 Result of Test Case 8 .....	81
Figure 5.27 Test Case 9.....	81
Figure 5.28 Result of Test Case 9 .....	82
Figure 5.29 Test Case 10.....	82
Figure 5.30 Result of Test Case 10 .....	82



## LIST OF ABBREVIATIONS

<b>AB</b>	<b>AdaBoost</b>
<b>CB</b>	<b>CatBoost</b>
<b>DS</b>	<b>Decision Stump</b>
<b>DT</b>	<b>Decision Tree</b>
<b>ELM</b>	<b>Extreme Learning Machine</b>
<b>HS</b>	<b>Harmony Search</b>
<b>HTTPS</b>	<b>Hyper Text Transfer Protocol with Secure Socket Layer</b>
<b>K*</b>	<b>K Star</b>
<b>KNN</b>	<b>K-Nearest Neighbors</b>
<b>LM</b>	<b>Linear Model</b>
<b>LR</b>	<b>Logistic Regression</b>
<b>NB</b>	<b>Naïve Bayes</b>
<b>NN</b>	<b>Neural Network</b>
<b>NR</b>	<b>Nonlinear Regression</b>
<b>RF</b>	<b>Random Forest</b>
<b>RT</b>	<b>Random Tree</b>
<b>SVM</b>	<b>Support Vector Machine</b>
<b>TWSVM</b>	<b>Twin Support Vector Machine</b>
<b>URL</b>	<b>Uniform Resource Locator</b>
<b>UTeM</b>	<b>Universiti Teknikal Malaysia Melaka</b>
<b>XGB</b>	<b>XGBoost</b>

**LIST OF ATTACHMENTS**

**PAGE**





## Chapter 1: INTRODUCTION

### 1.1 Introduction

Phishing is a type of web-based attack where attackers collect confidential data such as ID and passwords by sending a message that looks like it came from a trusted source, organization, or individual in other ways. The target of a phishing attack frequently receives e-mails that appear to be from a legitimate organization. The email usually includes web links that guide targets to the phishing web page to fool them into revealing personal or financial information such as ID, password, and card information. There are several reasons that people fall for phishing. First, a user unfamiliar with Uniform Resource Locators (URL) and URL usage. Second, users unable to differentiate between legitimate URLs and phishing URLs. Third, users unintentionally click on URLs or do not enough time to consult the URL. Forth, the user unable to access the target URL due to redirection or secret URLs. Last, users have no idea which of the URLs displayed can be trusted (Buber et al., 2017). Since phishing webpages focusing on banks, companies and web users are unavoidable, detecting web phishing attacks is critical. Because of various advanced techniques used by attackers to confuse web users, detecting a phishing website becomes difficult. To identify phishing webpages, several traditional strategies focused on set black and whitelisting databases. These methods, however, are ineffective since a new website can be created in a matter of seconds. As a result, most of these strategies are unable to determine if a new website is phishing or not in real-time (Ali, 2017). Machine learning is a multidisciplinary technique of learning that is mainly used in supervised learning to construct predictive models. Machine learning is suitable for detecting phishing webpage because machine learning can transform the problem into a typical

classification task. Machine learning can create models based on previously labeled websites, which can then be incorporated into a browser to detect phishing attempts. The dataset that contains website features, as well as the availability of enough websites to build realistic predictive models, are critical to build an automated anti-phishing machine learning model (Abdelhamid et al., 2017).

## 1.2 Project Background

In recent years, the development of various websites including online banking, education, and social media has been driven due to the growth of the internet. Phishing attacks have increased significantly and are now widely regarded as a most serious new internet crime, potentially causing people to not be trusted in e-business. As a result, phishing has adverse effects on internet banking, e-business, organizational revenues, client partnerships, and overall market operations. The development of various phishing websites enables hackers to access confidential personal or financial data. Phishing URLs are used to collect password and login information, as well as other account information, by sending attackers to target users via e-mail or other communication networks as a recognized individual or entity. Phishing URLs host unsolicited and trick users and become scam victims and result in losses (Yi et al., 2018). Phishing URLs are designed to look legitimate to successfully confuse the user. Since humans are so easily duped, automated techniques to identify phishing websites and legitimate are required as a second of defense (Kulkarni et al., 2019). Phishing URL detection has traditionally been done primarily through the use of blacklists. Blacklists, on the other hand, aren't exhaustive and can't detect newly created malicious URLs. Machine learning algorithms have gained popularity in recent years as a way to improve the generality of malicious URL detectors. (Sahoo et al., 2019). This project aims to build an URL classifier that can classify URLs as malicious or legitimate based on the most accurate algorithm after comparison. Three machine learning algorithms which are Logistic Regression, Random Forest, and Catboost are evaluated and compared in terms of accuracy. This research introduces a phishing URL detection machine learning algorithm based on CatBoost. Machine learning

algorithms for evaluating different features of URLs can help humans to distinguish between legitimate and phishing websites with high accuracy.

### 1.3 Problem Statement

Phishing attack becomes a threat to web users, governments, and companies. In a phishing attack, the attackers use spam email or fake websites to obtain the client's sensitive data such as user account login information, credit/debit card numbers, and passwords. Attackers are also able to create web pages that look and feel like legitimate websites, such as banking, to trick victims into providing confidential information. Even though phishing attacks do not necessitate specialized technical expertise and those users are becoming more aware of these attack tactics, they continue to cause significant financial losses (Basit et al., 2020).

- The traditional blacklist approach will never be completed, resulting in a false positive.

In traditionally, the method to detect phishing websites is updating blacklisted URLs to the database. As a result, it can take hours or even months to be added to a blacklist, giving scammers sufficient time to target several users. However, the blacklisted method can never complete since malicious URLs are created regularly. Genuine websites are often blacklisted inadvertently, whether purposely or not, resulting in a false positive. Given that a single website can be blocked through multiple browsers, such a situation can cause as much trouble for the parties as a good fraud (Silva et al., 2019). To solve these issues, a machine-learning model based on training datasets of previous phishing sites is used to categorize new phishing sites. Security researchers focus on machine learning techniques which consist of algorithms that require data to decide on future data. Machine learning algorithms can study numerous blacklists and legal URLs and characteristics to correctly identify phishing URLs.

## 1.4 Project Question

Project question are written based on the problem statement above, as shown in Table 1.1.

**Table 1.1: Project Question**

No.	Project Question
1	Which machine learning algorithms are used for phishing detection with higher accuracy?

## 1.5 Objective

Three objectives are written based on the project question above, as shown in Table 1.2.

**Table 1.2: Objectives**

No.	Objectives
1	To investigate the suitable machine learning algorithms to detect phishing URLs.
2	To compare the accuracy of machine learning algorithms such as Logistic Regression, Random Forest, and CatBoost.
3	To implement a URL classifier based on the most accurate algorithm that can detect between phishing and legitimate URLs.

## 1.6 Scope

Scope of the Project:

- Understand the characteristic of phishing websites and distinguishing features from legitimate websites.
- Determine dataset for designing machine-learning-based approaches.

## 1.7 Project Contribution

In the research domain, a comparison among CatBoost, Random Forest, and Logistic Regression algorithms is conducted in this project to determine which algorithm has the best performance in terms of accuracy, precision, recall, and f1-score. In addition, this project would benefit web users by allowing them to identify and stay alert to phishing websites in real-time, resulting in a more secure network experience. It can also be used in the security domain whereby cybersecurity authorities can apply it to prevent users from visiting these phishing websites and develop powerful security mechanisms that can identify and avoid phishing domains from reaching the user.

## 1.8 Report Organization

Report Organization will explain the summary of each chapter. This report contains six chapters.

### **1.8.1 Chapter 1: Introduction**

Chapter 1 explains detection of phishing URLs by using machine learning. Project background, problem statement, project question, objective, scope, project contribution and report organization are included in this chapter.

### **1.8.2 Chapter 2: Literature Review**

Chapter 2 discusses and reviews several journals, articles and books regarding the detection of phishing URL by using machine learning. This chapter included related work or previous work, critical review of existing algorithms or techniques, and project solution.

### **1.8.3 Chapter 3: Project Methodology**

Chapter 3 explains the methodology and process to complete this project. Research milestones, Gantt chart are included in this chapter to make sure that this project is completed on time.

### **1.8.4 Chapter 4: Implementation**

Chapter 4 highlights the development of detecting phishing URLs by using machine learning such as coding. The project's expected outcome will also include in this chapter.

### **1.8.5 Chapter 5: Testing & Analysis**

Chapter 5 examines and evaluates the results of detecting phishing URLs by using machine learning. In this chapter, a few tests and analyses will be carried out, and the results will be clarified.

### **1.8.6 Chapter 6: Project Conclusion**

Research summarization, research contribution, and research limitation will be addressed in this chapter. At the same time, this chapter will discuss future works.

## 1.9 Conclusion

In conclusion, detecting phishing URLs by using machine learning can help people and companies identify phishing URLs and avoid information leakage even financial losses. As a consequence, this study expects to produce a detection tool that can help people in preventing any criminal acts from occurring. In the next chapter, the literature review will discuss the papers that have been studied and analyzed regarding the detection of phishing URLs by using machine learning.



## **CHAPTER 2: LITERATURE REVIEW AND PROJECT METHODOLOGY**

### **2.1 Introduction**

This chapter explains several research and studies that are related to the title detecting phishing Uniform Resource Locator (URL) by using machine learning. Literature review involves finding relevant publications such as conference papers, journal papers, and books. This study will focus on the machine learning algorithms or techniques used. This chapter will review past or existing approaches or processes, as well as the shortcomings or limitations, compare the techniques used in previous studies and discuss the project solution.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

### **2.2 Related Work**

Phishing is a type of identity fraud that utilizes social engineering strategies as well as complex attack vectors to obtain financial details from unsuspecting users. A "phisher" usually sends out emails randomly to web users when trying to impersonate an individual or a respectable organization such as a bank. Users are tricked by phishers who use social engineering techniques to persuade them to enter the phishing website where the website is required to fill in personal or financial information. When web users are tricked to access a phishing website, they are duped into completing the website's goal (Ram, 2014). The topic of phishing URL detection has been solved using several approaches. Two groups of approaches are explained by depending on



the fundamental principles. The first is heuristics approaches and the second is machine learning approaches.

### 2.2.1 Heuristics approaches

A heuristic is a problem-solving technique that use a practical solution or a set of procedures to provide solutions that aren't ideal but are enough in the given timeframe. In computer science, the heuristic method is used to solve a problem faster than traditional methods and produce an estimated solution when traditional methods hard to find a solution. Heuristics are designed to generate a solution in a reasonable time that solves the problem at hand. Heuristic nonlinear regression (NR) strategy is implemented for identifying phishing websites. This project is implemented to distinguish phishing websites using a feature selection method and a meta-heuristic-based NR algorithm. To select the best feature subset, this study uses two feature selection methods which are decision tree and wrapper, with the latter achieving a detection accuracy rate of 96.32 percent. After selecting the best feature, two meta-heuristic algorithms are successfully applied to predict and detect fake websites: harmony search (HS), which uses a NR method, and SVM. A NR approach with HS algorithms was used, with the Generated New Harmony, Harmony Memory Considering Rate, and Pitch Adjusting Rate parameter to distinguish the phishing websites. The accuracy rates for the train and test processes were 94.13 and 92.80 %, respectively, using nonlinear regression based on HS. As a result, the study concludes that the nonlinear regression-based HS performs better than SVM (Babagoli et al., 2018). Association rule mining is proposed to detect phishing URLs. By using rule mining, significant features can be determined dan able to distinguish legitimate from phishing URLs. This method contains two phases. In the first phase, search for URLs from many sources. Second, extracted features from URLs that have been collected to determine different heuristics. The features include transport layer security, number of slashes in URL, keyword in the hostname of URL, number of terms in the hostname of the URL, and others. WEKA a data mining tool with apriori and predictive apriori rule generation algorithms is used for feature extraction. According to the results, the proposed method obtained an average accuracy of 93% (Jeeva et al., 2016). Aa heuristic technique which is Twin Support Vector Machine (TWSVM) classifier is

suggested to identify phishing websites. To overcome traditional heuristic limitations such as using HTTPS, search engines, and WHOIS information that may fail to identify malicious websites, a novel TWSVM heuristic strategy was developed to identify maliciously recorded phishing webpage as well as sites hosted on compromised servers. By matching the log-in page and the home page of the accessing website, this approach distinguishes phishing websites hosted on hacked domains. The URL-based functions are used to distinguish maliciously recorded phishing pages. In order to categorize phishing websites, researchers used various types of support vector machines (SVMs) such as SVM, Proximal support vector machine, and TWSVM. With a significant precision of 98.05 percent and recall of 98.33 percent, TWSVM outperformed the other models (Rao et al., 2019). A desktop application called PhishSaver is developed which is a heuristic approach to detect phishing websites. To check the legitimacy of the URL, a mixture of a blacklist and a variety of heuristic features are used. For the blacklist, researchers use GOOGLE API SERVICES, this is Google's safe browsing blacklist, and this list is modified and managed regularly by Google, and the heuristic features are classified into five modules such as the use of blacklist, detection of the login page, footer links pointing to NULL (#), use of copyright and title content, website identity. These five modules are considered as five identification stages. PhishSaver takes a URL as an input and returns the website's status, such as phishing, legitimate, or suspicious. (Suman et al., 2017).

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

### 2.2.2 Machine learning approaches

Machine learning uses data and expertise to automate the construction of analytical models. Machine learning is a subfield of AI that concentrates on computers that can study from data, recognize patterns, and decide things with very little human input. It aims to find patterns in data and then make predictions based on these often-complex patterns to answer questions, identify and analyze trends, and assist in problem-solving. Detecting phishing websites by using machine learning algorithms is proposed to compares each method's accuracy rate, false-positive rate, and false-negative rate in order to find the best machine learning algorithm for detecting

phishing URLs. First, features are extracted from URLs such as the existence of IP address in URL, exist of @ symbol in URL, the number of dots in the hostname, and others. Then, Dataset is split into training testing set in 50:50, 70:30, and 90:10 ratios respectively. Phishing websites are detected using Decision Tree (DT), RF, and SVM algorithms. As result, the RF algorithm is the most precise detection which achieved 97.14% of accuracy and the lowest false positive rate. Machine learning models perform better when there is more data as training data (Mahajan et al., 2018). A system is proposed by using a machine learning approach to detect and prevent Phishing Website using. First, URL is compared by using Blacklist and Whitelist Approach. If URL is found in Blacklist and Whitelist then the URL is a phishing website. If URL not found in Blacklist and Whitelist, then extract features of URL by using Heuristic and Visual Similarity Approach. Then, to predict the result, researchers use machine learning algorithms such as LR, DT, and RF to analyze these various features of URLs and webpages. The system emphasizes the efficiency gained by combining heuristic features, visual features, and a blacklist and whitelist approach with machine learning techniques. The result shows that the LR and DT achieved 96.23% accuracy while RF achieved 96.58% of accuracy (Patil et al., 2020). A detection model of phishing attacks calls Detection of Phishing Website Using Machine Learning is developed to focus on detecting phishing attacks by testing the functionality of phishing websites, as well as the Blacklist and WHOIS databases. URLs, source code, page style and content, web address bar, and other features that have been selected to distinguish between legitimate and spoofed websites. This research is limited to URLs and domain name features. IP addresses, long URL addresses, URLs presence of the @ are among the characteristics of URLs and that are verified. These characteristics are examined using a series of rules in order to differentiate phishing webpage URLs from legitimate website URLs. In the future, the system may be upgraded to be fully automated, determine the compatibility of the web page and the application with the web browser (Sampat et al., 2018). Pros and cons of many machine-learning-based phishing detection techniques, as well as their performance on a real-world dataset such as SVM, DT, RF, and AdaBoost (AB), is analyzed to determine the best machine learning algorithms. The classifier is SVM, and four measures related to the gamma parameter in the SVM algorithm are tested: accuracy, precision, f1, and recall. The experiment results shows that the accuracy is 96.95%. DT classifier are evaluated related to the parameter max\_depth in the DT

algorithm. The results indicate the accuracy of DT is 93.68%. AB classifier are evaluated regarding the parameter `n_estimators` in the AB algorithm. The accuracy is 94.5%. RF as the classifier and test four measures are evaluated related to the `parameter_estimators`. Based on experiment results, the accuracy of RF is highest which is 97.31% among the four-machine learning (Mao et al., 2019).

There consist of several types of machine learning algorithms which are supervised and unsupervised machine learning. Supervised learning known as all data is tagged to build algorithms so that algorithms are able to predict the result and classify data accurately while unsupervised learning is defined by using untagged data so that algorithms learn patterns from input data. Supervised learning solves two types of problems which are classification (grouping data into pre-determined categories) and regression (predicting values based on numerical data). Unsupervised learning solves clustering (data is grouped based on patterns) and association (identifying the rules that will characterize data patterns) problems. Supervised learning classification is effective in spam detection, revenue forecasting, and fraud detection. For example, a machine learning model provided data from thousands of emails and each data has classified as spam or not spam, and the model learns to recognize patterns that produce a "spam" or "not spam" result. The supervised learning process improves by continuously measuring the system's outputs and fine-tuning it to get similar to the target accuracy. A supervised learning algorithm examines the training phishing sites datasets and generates a predictor that can accurately predict the right category for unknown datasets and detect freshly generated phishing websites rapidly. (Ali,2017). The use of unsupervised learning has various drawbacks. Because the exact outcome is not known in advance, unsupervised learning models may produce less accurate results than supervised learning models. So, supervised learning is the most suitable method to detect phishing URLs.

### **2.3 Critical review of existing algorithms/ techniques**

For digits, recognition in the heuristic method, a set of rules, and some unlabeled pictures of digits are used to help model the data and use the above criteria to recognize patterns. For digits recognition in machine learning, machine learning

libraries, labeled pictures of digits, unlabeled pictures of digits are used to develop some predictive model and recognize the patterns. Normally, a heuristic is a hand-coded function. It is usually focused on some common-sense knowledge from domain experts, rather than a model gained from training on a data set. An algorithm that adapts to data is known as a machine learning algorithm. A machine learning algorithm, for example, adapts its model for each new example it encounters. The more the model is used, the better it becomes, but it depends on the class of models being used is sufficient for the task and on a sufficient training data set. A heuristic is a method of determining the answer to a problem without exploring all potential solutions or knowing the answer ahead of time. Instead of teaching the computer the technique that was discovered for solving the problem, machine learning is a technique that helps it to discover its technique for solving the problem. The heuristic approach can result in a high percentage of incorrect decisions, resulting in monetary and human life losses. Even though the heuristic approach can work faster than a machine learning approach but machine learning approach, on the other hand, maybe more flexible than the heuristic approach in terms of accuracy and performance. So, machine learning is used in this project in order to detect phishing websites. The machine learning techniques and programming languages used in existing projects are discussed in this section. Differences between the techniques, several comparisons will be presented. Justification will be presented from the comparison result and the proposed solution will discuss in the next section. Table 2.1 displays the comparison of the method used in previous work.

Extreme Learning Model Based Phishing Classifier is proposed to develop an useful method for identifying phishing websites using Extreme Learning Model (ELM). In the pre-processing stage, the groups in the dataset are assigned using classification model, followed by a search for null values. The model is made up of three layers: an input layer, a hidden layer, and an output layer. Activation can be done with linear or non-linear activation functions. In order for machine learning to identify the URL, the URL's features must be defined before classify by machine learning. The machine learnings such as RF, ELM, Naïve Bayes (NB), SVM then identifies the URL's features as phishing or legitimate. According to the result, RF achieved the highest accuracy of detecting phishing websites which is 98.80%. ELM achieved

97.00% of accuracy, NB achieved 93% of accuracy while SVM achieved 92% of accuracy (Tumuluru et al., 2019).

Classification of Phishing Web Sites Extreme Learning Machine (ELM)-based classification for 30 attributes of phishing websites data in the UC Irvine Machine Learning Repository database is used to detect phishing websites. To detect phishing website characteristics, first criteria are created for extracting features from a website. Bar-based features, abnormal-based features, HTML and JavaScript-based features, and domain-based features are all examples of features. After the data is ready to be processed, the learning algorithm modeling process begins. SVM, NB and ELM are used to classify URLs. According to the experimental result, ELM achieved 95.34% which is the highest, NB achieved 93.80% of accuracy while SVM achieved 92.98% of accuracy (Sonmez1 et al., 2018).

PhishMon is developed to identify phishing websites. PhishMon uses eighteen features, fifteen of which are novel, to determine whether a given URL is real or fraudulent.. It contains three categories of features which are HTTP Features, Code Complexity Features, and certificates feature. For feature selection, RF model is trained on Certificate Features and the accuracy is 92.6%, false-positive rate is 0.6%. For Code Complexity features the accuracy is 91.2% and the false positive rate is 4%. For HTML features the accuracy is 93.2% and the false positive rate is 2.7%. RF is chosen after conducting a sequence of testings on four different machine learning algorithms such as CART, KNN, AB and RF determine the performance. PhishMon achieved 95.4 % of accuracy to detect phishing websites, with a 1.3 percent false-positive rate (Niakanlahiji et al., 2018).

A project with the title Detection of Phishing Websites using Machine Learning Approach is proposed to identify phishing websites The information was gathered via the MillerSmiles archive, the Phish Tank archive, and Google's search operators. There are 30 characteristics in the data set. The integers -1, 0, and 1 are used to indicate the values of traits, with -1 representing phishing, 0 indicating suspicious, and 1 indicating legitimate. The data set is next processed to obtain mature data in the desired format, which is split into two sections: training 70% and testing 30%. Four machine learning algorithms is selected to detect phishing websites which are DT,

Neural Network (NN), Linear model (LM), and RF. Based on the result, RF achieved 95.7% of accuracy, LM achieved 92.10% of accuracy, NN achieved 90.70% of accuracy while DT achieved 90.4% of accuracy (Jalal et al., 2019).

A Decision Tree model is developed for the identification of phishing websites with information gain feature selection. Top feature subset 5, 10, 15, and 20 out of 30 features are selected. Then, using various top-selected feature subsets, develop a computationally efficient model for phishing website classification. Selected top feature subset 5, 10, 15 and 20 achieved 91.70%, 91.75%, 91.80% and 91.80% accuracy respectively with DT classifier. During the classification process, the data set is divided into two parts: training set and testing set. The classifier is trained using the training data set, and the classifier is tested using the testing data set. DT, Random Tree (RT), RF, and Decision Stump (DS) are used as a classifier for the classification of phishing websites. According to the result, DT achieved 91.80% of accuracy, DS achieved 84.73% of accuracy, RF achieved 78.85% of accuracy while Random Tree achieved 66.75% of accuracy (Shrivastava et al., 2017).

Seven different classification techniques and natural language processing (NLP) based features are used to construct a real-time anti-phishing system. The system is distinguished from other research in the literature by its language independence, use of a huge quantity of phishing and legitimate data, real-time execution, identification of new websites, independence from third-party providers, and use of feature-rich classifiers. A new dataset is created to measure the system's accuracy, and the experimental results are evaluated on it. Then, machine learning algorithms such as DT, AB, K\*, KNN, Sequential Minimal Optimization, NB, and RF algorithms are used to classify phishing and legitimate website. The result shows RF obtained the highest accuracy based on NLP features which is 97.98%. The second is DT which achieved 97.02% accuracy based on NLP features. KNN and NB achieved 95.86% accuracy based on Hybrid features and K star achieved 95.27% based on Hybrid features. Sequential Minimal Optimization achieved 94.92% accuracy based on NLP features while AdaBoost achieved 93.24% accuracy based on NLP features (Sahingoz et al., 2018).

Detection of phishing websites using an efficient machine learning framework is proposed to detect phishing websites. The proposed method is divided into two phases which are classification and detection of phishing. The classification phase consists of three sub-modules which are the data collection module, Feature selection module, classification module. In data collection module, phishing and legitimate URLs are collected then extract the features of URLs. The Address Bar, abnormal-based feature, HTML and JavaScript, and domain-based feature are all considered in the feature extraction module. The classification module's main objective is to accurately identify phishing websites. The dataset is split into training dataset and a testing dataset in the phishing detection phase, and five machine learning models such as KNN, LR, RF, DT, and SVM are used to distinguish legitimate URLs from phishing URLs using attributes extracted in the feature extraction module. To summarize, the RF classifier has the highest phishing detection accuracy (91.4%) (Kumar D et al., 2020).

A comparison of Machine Learning Techniques in Phishing Website Classification is implemented to identify the most accurate machine learning algorithm in classification. First, data set is collected in UCI Machine Learning Repository. Dataset phishing criteria contains 4 sections which has 30 attributes. Then several different machine learning algorithms such as RF, C4.5, Reduced Error Pruning (REPTree) Decision Stump, Hoeffding Tree, Rotation Forest, and Multilayer Perceptron (MLP) are used to detect phishing websites. The experiment result shows that Rotation Forest has the highest phishing detection rate which is 89.1% in test 1 and 88.5 % in test 2. REPTree achieved 88.4 % of accuracy in test 1 and 88% of accuracy in test 2 while C4.5 has the lowest accuracy of detecting phishing URLs with 74.6% and 73% for two test sets. (Hodžić et al., 2016).

RF algorithms achieved the highest accuracy in most of the projects and the highest it can reach is 96.80% which is on Tumuluru et al.'s project. Machine learning algorithms such as DT, ELM, NB, SVM, KNN can achieve an average accuracy of 90% or more. Random Tree in Shrivastava et al.'s project achieved the lowest detection accuracy which is 66.75%. C4.5 algorithms only achieved 73% of accuracy in detecting phishing websites. The paper of Hodžić et al. only achieves an average of



80% accuracy, the best accuracy is 89% on the MLP algorithm while the lowest accuracy is 73% on the C4.5 algorithm.

**Table 2.1: Comparison of the method used of previous work**

Journal	RF	NB	SVM	DT	KNN	ELM	AB	LR	DS	NN	K*	RT	LM
Tumuluru et al. (2019)	√	√	√			√							
Yasin Sönmez et al. (2018)		√	√			√							
Amirreza Niakanlahiji et al. (2018)	√				√		√						
Kahksha et al. (2019)	√			√						√			√
A. K. Shrivastava et al. (2017)	√			√					√			√	
Sahingoz et al. (2018)	√	√		√	√		√				√		
Naresh Kumar D et al. (2020)	√		√	√	√			√					
Adnan Hodžić et al. (2016)	√								√				

Our proposed model	√		√					√					
--------------------	---	--	---	--	--	--	--	---	--	--	--	--	--

- Improve Detection Accuracy by using Boosting algorithms

Among all other classification algorithms, the RF approach offers the best results with the highest accuracy. Several studies have shown that by implementing an RF classification model, more than 95% assault detection accuracy may be achieved. However, the Boosting algorithms are able to produce better experimental results compared to RF in terms of accuracy and other parameters for detecting phishing URLs. XGBoost (XGB) provides great accuracy (up to 97%) in a short amount of time, and the XGB classifier is the most accurate of all the classifiers. Together with accuracy, the Boosting algorithm is also consistent in terms of precision (Masurkar et al., 2020). Boosting is a collection of algorithms and the main goal is to turn weak learners into strong ones. Boosting algorithms can enhance the model's prediction accuracy by a significant number of features. CB is the latest boosting algorithm in machine learning and CB can increase the model's performance in Medicare Fraud Detection compared to XGB. In terms of AUC, the categorical feature for XGB and CB enhances performance, and CB's performance is statistically significantly higher. CB and XGB achieve nearly identical AUC on a purely numerical dataset. However, the XGB model having a faster training time compared to the CB model (Hancock et al., 2020). CB can perform better than XGB in fraud detection. CB model that has been trained makes prediction much quicker than XGB. Research about CB-based phishing URL detection is conducted to improve detection accuracy.

## 2.4 Project solution

This research is about phishing URL detection using machine learning algorithms. The comparison of algorithms such as LR, RF, and CB will be conducted to determine which algorithm is most suitable for phishing detection. The algorithm with the highest accuracy is chosen to build a URL classifier. Relevant data will be collected from UCI Machine Learning Repository. The related data (URL) will be classified into two categories: phishing and legitimate. The status of the output for phishing URL detection analysis will be tested and evaluated.

One of the most fundamental machine learning algorithms is LR. It's simple to set up and, in certain cases, provides good training results. Because of these variables, this approach does not require a large amount of processing power to train a model. LR is robust to small amounts of noise in the data and is unaffected by minor incidences of multi-collinearity. LR is most commonly used to solve binary classification such as determine email spam or not spam and online transactions fraud or not fraud, so LR was chosen to distinguish between phishing or legitimate URLs.

RF is created from multiple decision trees for modeling predictions and behavior analysis. RF can handle enormous datasets with thousands of variables and will classify uncommon data sets automatically. The method also works swiftly with variables, which makes it appropriate for more difficult problems. Each decision tree in the RF Classifier can use random selection to capture more complicated feature patterns and deliver the best accuracy. According to a study, RF has the highest accuracy for detecting phishing URLs. So, RF is used to detect phishing URLs in this project.

CB is a recently developed machine learning algorithm from Yandex. CB has a strong categorical data handling technique since it supports numerical, categorical and text features. CB can be used to solve business problems including fraud detection, recommendation systems, and forecasting. CB can produce a great result with a small amount of data. Other machine learning algorithms, on the other hand, only perform well after learning from a large amount of data. Based on previous research on CB algorithms, it can be proved that CB can achieve high accuracy of up to 95% when compared to other machine learning algorithms. However, it required long training and

optimization times for building a model. A machine learning model based on CB is used to predict fraud in financial transactions. This paper aims to enhance the detection's accuracy by using feature engineering to develop high-value features which are then fed into CB for prediction and classification. Then CB is compared with NB and SVM algorithms. CB achieved the highest accuracy in detecting fraud with 98.3%. This is due to CB's ability to overcome prediction shifts and better handle categorization features. As a result, CB outperformed other algorithms, proving its efficacy (Chen et al., 2021). CB model used for online transactions detection is implemented to overcome the risk of online transactions. Feature engineering is performed to exclude unnecessary features, so it can enhance the performance. Then, as a binary classification, the popular and effective GBDT method CB is applied. Several machine learning models, such as SVM, LR, and RF, are conducted to examine the accuracy. Since CB can deal with categorical features by transforming categorical data to numerical data before training, the CB-based model outperformed all other models. CB employs a more efficient technique that minimizes overfitting and allows the entire dataset to be used for training. CB achieved the highest accuracy with 98.7% and the highest Auc Roc score with 97.3% in this project (Li et al., 2020). CB algorithms are well performed in fraud detection which is quite similar to phishing detection, so it used for detecting phishing website.

#### **2.4.1 Logistic Regression**

One of the most commonly utilized statistical processes in research is Logistic Regression. Many analysts believe that LR is a crucial method in predictive analytics as well as the more established Six Sigma movement. LR analysis is a statistical method for determining the relationship between a set of categorical or continuous predictor variables and a binary outcome (dichotomous) (Ranganathan et al., 2017).

##### **2.4.1.1 Sigmoid Function**

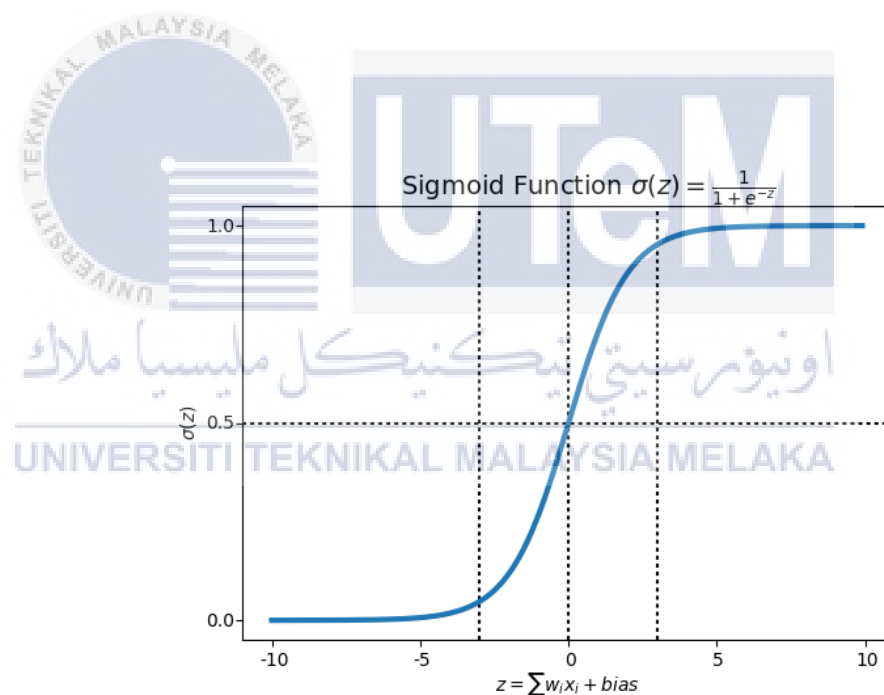
LR is used as a classification algorithm, where the value of the target variable is categorical. According to LR theory, the cost function may only have a value

between 0 and 1. The equation of the expectation of the Logistic regression hypothesis is shown as below (Pant A., 2019).

$$0 \leq h_{\theta} \leq 1$$

To convert predicted values to probabilities, the Sigmoid function is applied. This method converts any real value to a number between 0 and 1. The sigmoid function is used in machine learning to map predictions to probabilities. The equation of the sigmoid function shows as below and figure 2.1 shows the sigmoid function graph (Pant A., 2019).

$$f(x) = \frac{1}{1 + e^{-x}}$$



**Figure 2.1 Sigmoid Function Graph**

#### 2.4.1.2 Hypothesis Representation

$h_{\theta}(x) = \beta_0 + \beta_1 X$  is the formula of hypothesis while using linear regression. The formula of hypothesis is modifying,  $\sigma(Z) = \sigma(\beta_0 + \beta_1 X)$  for LR. The Hypothesis of logistic regression shows in figure 2.2 (Pant A., 2019).

$$Z = \beta_0 + \beta_1 X$$

$$h\Theta(x) = \text{sigmoid}(Z)$$

$$\text{i.e. } h\Theta(x) = 1/(1 + e^{-(\beta_0 + \beta_1 X)})$$

$$h\theta(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

**Figure 2.2 Hypothesis of logistic regression**

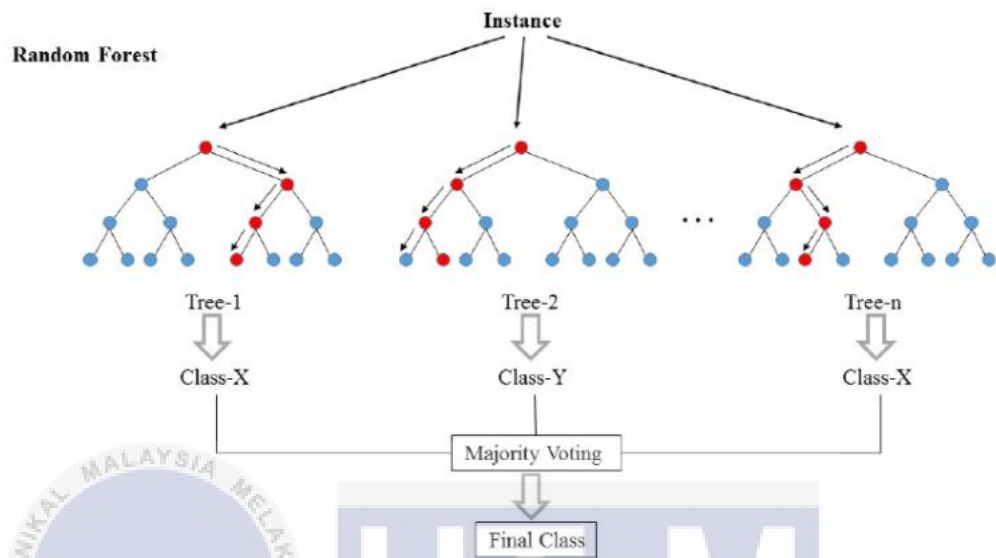
#### 2.4.2 Random Forest

Random Forest is a classification and regression ensemble learning strategy that generates numerous randomized decision trees during the training phase before averaging the results to predict. In practice, the strategy has evolved into a powerful data analysis tool that outperforms many established methods (Scornet et al., 2015). The ability of forests to handle a wide range of prediction problems with minimal parameters to alter has boosted their popularity. The technique is well-known for its accuracy and simplicity, as well as its ability to work with small sample sizes, high-dimensional feature spaces, and complex data frameworks.

##### 2.4.2.1 Random Forest Working Process

RF is built from several decision trees and merges them together in order to get a more precise and reliable forecast. The stages of the working process are outlined below. Step 1: Pick  $f$  data points at random from the training set. Step 2: For the data points you've picked, make decision trees (Subsets). Step 3: Decide on a  $N$  for the number of decision trees you wish to make. Step 4: Go through Steps 1 and 2 again. Step 5: For new data points, find the forecasts of each decision tree and assign them to

the category with the highest votes. Figure 2.3 illustrates the Random Forest algorithm (Koehrsen W.,2020).



**Figure 2.3 How Random Forest Works**

A decision tree's or a bagging classifier's hyperparameters are extremely similar to those of an RF. A decision tree or a bagging classifier have hyperparameters that are quite comparable to those of an RF. The RF provides more randomization to the model as the number of trees grows. Instead than looking for the most important feature when dividing a node, it seeks for the best feature from a random group of features. As a result, there are a lot more options, which leads to a superior model.

#### 2.4.2.2 Classification Formula

When doing RF based on classification results, it's crucial to consider how to use the Gini index, which is a method for deciding how nodes on a decision tree branch are linked. This formula determines which branch on a node is more likely to occur by calculating the Gini of each branch based on the class and probability. The number of classes in the dataset is represented by  $c$ , while the relative frequency of the class being

studied is represented by  $p_i$ . The equation of Gini index formula is shown as below (Tyagi N., 2020).

$$Gini = 1 - \sum_{i=1}^c (P_i)^2$$

In a decision tree, entropy can be applied to determine how nodes branch. The probability of a specific result is used by entropy to determine how the node should branch. It is more theoretically complex than the Gini index due to the logarithmic function used to calculate it. The Entropy formula is shown as below (Tyagi N., 2020).

$$Entropy = \sum_{i=1}^c -p_i * \log_2(p_i)$$

### 2.4.3 CatBoost

CatBoost is a supervised machine learning implementation that uses a Gradient Boosted Decision Tree. CB brings two new ideas which are Ordered Target Statistic (OTS) and Order Boosting (OB). CB is a great solution for problems involving heterogeneous data, but it might not be the ideal learner for situations involving homogeneous data (T. Hancock et al., 2020). CB's usage of OTS and OB makes it an excellent choice for datasets containing categorical variables that are sparse or infrequently occur with specified target values, because these techniques ensure that, given odd training data, CB can update its estimate for the unusual data consistently. To fight the prediction shift produced by a specific sort of target leakage inherent in all gradient boosting algorithms currently in use, OTS and OB use random permutations of the training instances.

#### 2.4.3.1 Ordered Target Statistic (OTS)

CB employs a more powerful approach to combat prediction change. It is based on the ordering theory and is influenced by online learning algorithms that provide training examples in a sequential sequence. To adapt this concept to a normal offline



context, CB adds an artificial “time”— a random permutation<sup>1</sup> of the training instances. Then, for each case, it computes the Target Statistic using all of the available “histories.” The preceding examples have a higher variance in the Target Statistic than the subsequent ones when only one random permutation is used. CB achieves this by using different permutations for each gradient boosting phases. CB utilizes a one-hot encoding for all features with at most one hot max size-specific value. The default value is 2. Figure 2.4 shows the concept of one-hot encoding (Peretz T.,2021).

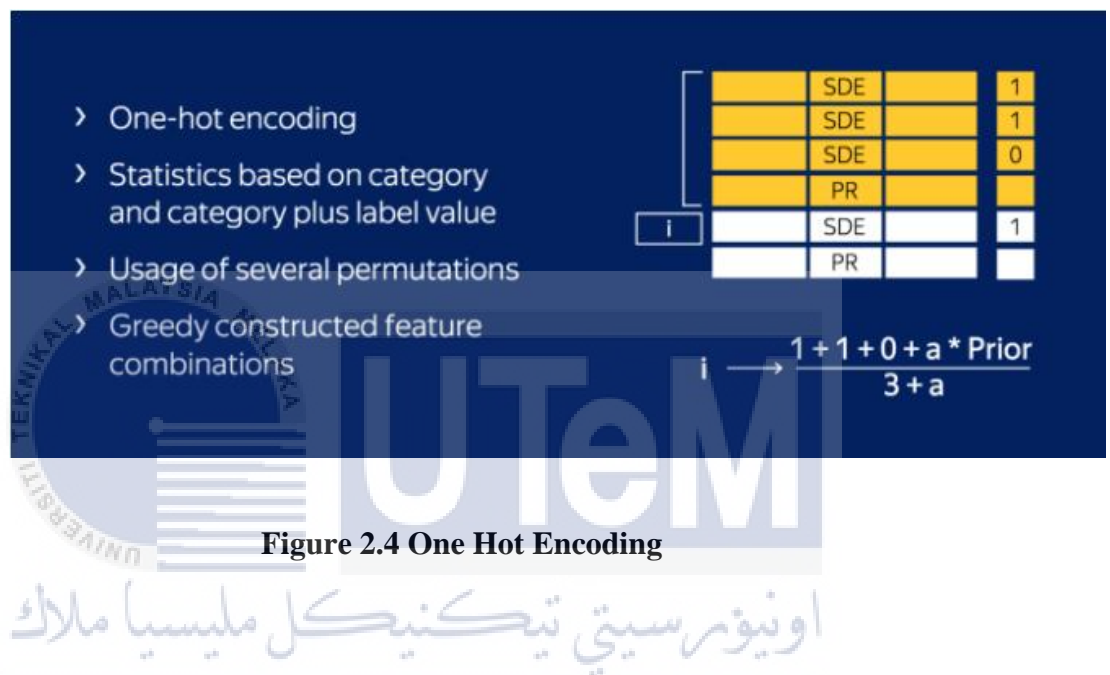
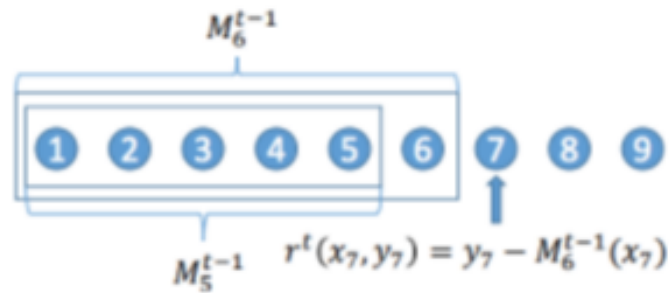


Figure 2.4 One Hot Encoding

#### 2.4.3.2 Ordering Boosting (OB)

CB contains two tree structure selection modes which are Ordered and Plain. Plain mode combines the standard GBDT method with the ordered Target Statistic. In Ordered Boosting mode, a random permutation of the training instances is used - 2, and n distinct supporting models are maintained - M1, M2... Mn such that just the first I sample in the permutation are used to train the model Mi. Because maintaining n distinct models increases the complexity and memory needs by n times, OB isn't realistic for most activities. Figure 2.5 shows the OB principle and pseudocode. Based on the gradient boosting algorithm, CB implements a modification of this algorithm that uses a single tree structure used by all the models to be created. Figure 2.6 shows the building of a tree in CB (Peretz T.,2021).




---

**Algorithm 1: Ordered boosting**


---

**input** :  $\{(x_k, y_k)\}_{k=1}^n, I$ ;

$\sigma \leftarrow$  random permutation of  $[1, n]$ ;

$M_i \leftarrow 0$  for  $i = 1..n$ ;

**for**  $t \leftarrow 1$  **to**  $I$  **do**

**for**  $i \leftarrow 1$  **to**  $n$  **do**

$r_i \leftarrow y_i - M_{\sigma(i)-1}(i)$ ;

**for**  $i \leftarrow 1$  **to**  $n$  **do**

$\Delta M \leftarrow$

$LearnModel((x_j, r_j) :$

$\sigma(j) \leq i$ ;

$M_i \leftarrow M_i + \Delta M$ ;

**return**  $M_n$

---

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

اونيورسيتي تكنولوجيكل مليسيا ملاك

Figure 2.5 OB Principle and Pseudocode

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

**Algorithm 2: Building a tree in CatBoost**


---

```

input :  $M, \{y_i\}_{i=1}^n, \alpha, L, \{\sigma_i\}_{i=1}^s, Mode$ 
 $grad \leftarrow \text{CaclGradient}(L, M, y);$ 
 $r \leftarrow \text{random}(1, s);$ 
 $G \leftarrow (grad_r(1), \dots, grad_r(n))$  for Plain;
 $G \leftarrow (grad_{r, \sigma_r(1)-1}(i) \text{ for } i = 1 \text{ to } n)$  for Ordered;
 $T \leftarrow$  empty tree;
foreach step of top-down procedure do
  foreach candidate split  $c$  do
     $T_c \leftarrow$  add split  $c$  to  $T$ ;
    if  $Mode == Plain$  then
       $\Delta(i) \leftarrow \text{avg}(grad_r(p)$  for
       $p : leaf(p) = leaf(i))$  for all  $i$ ;
    if  $Mode == Ordered$  then
       $\Delta(i) \leftarrow \text{avg}(grad_{r, \sigma_r(i)-1}(p)$  for
       $p : leaf(p) = leaf(i), \sigma_r(p) < \sigma_r(i) \forall i$ ;
     $loss(T_c) \leftarrow \|\Delta - G\|_2$ 
   $T \leftarrow \text{argmin}_{T_c}(loss(T_c))$ 
if  $Mode == Plain$  then
   $M_{r'}(i) \leftarrow M_{r'}(i) - \alpha \text{avg}(grad_{r'}(p)$  for
   $p : leaf(p) = leaf(i))$  for all  $r', i$ ;
if  $Mode == Ordered$  then
   $M_{r', j}(i) \leftarrow M_{r', j}(i) - \alpha \text{avg}(grad_{r', j}(p)$  for
   $p : leaf(p) = leaf(i), \sigma_{r'}(p) \leq j$  for all  $r', j, i$ ;
return  $T, M$ 

```

---

Figure 2.6 Building a tree in CatBoost

**2.5 Conclusion**

In conclusion, this chapter has explained previous works relating to methods or algorithms that have been used. The approach that will be used in this project and will be discussed in the following chapter.

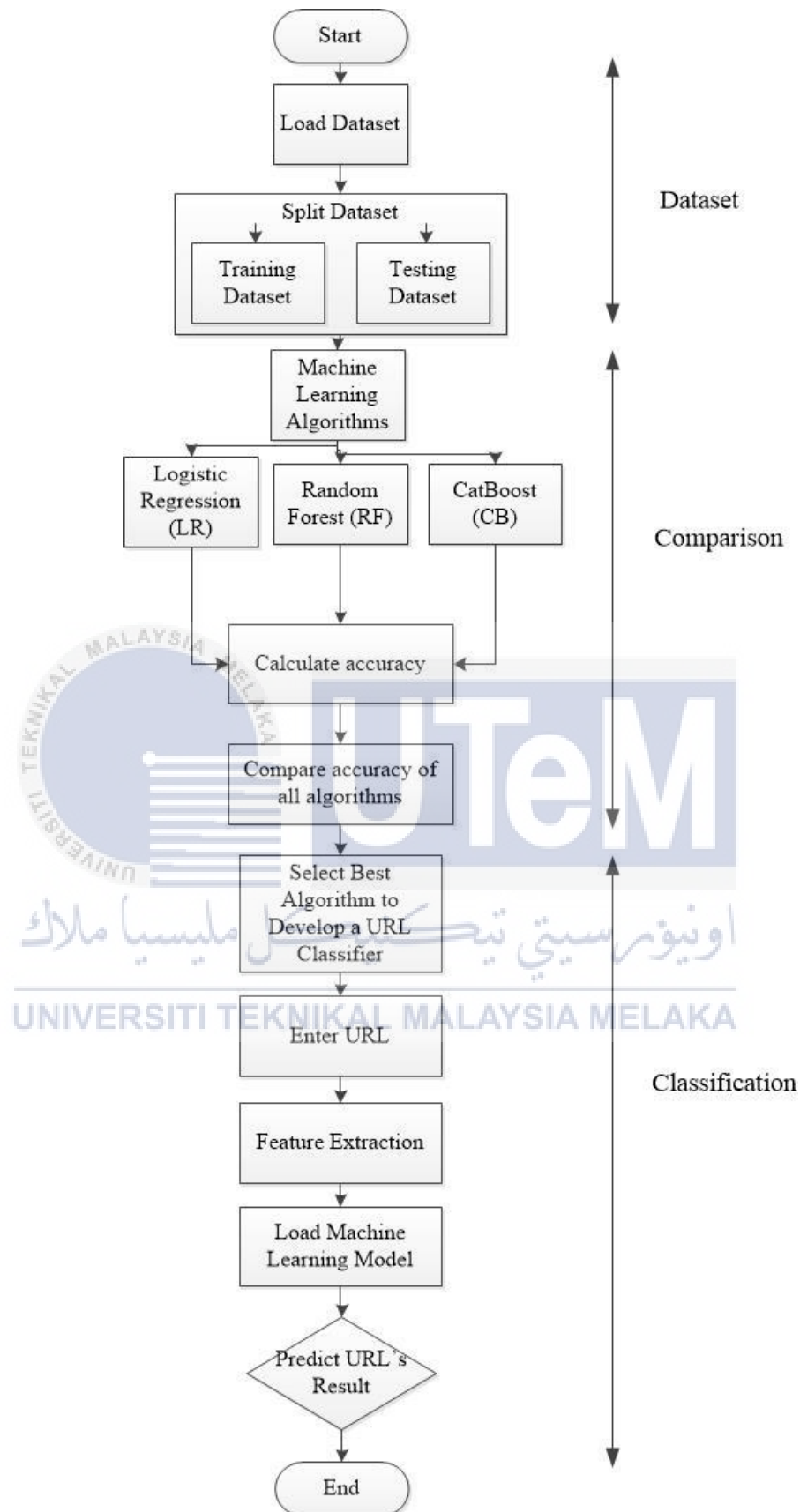
## CHAPTER 3: PROJECT METHODOLOGY

### 3.1 Introduction

Methodology refers to the techniques or methods used to conduct research and explain the process to result. This chapter will include data collection, methods to analyze data, tools used in this project. The aim of providing a methodology is to ensure that the project is implemented properly and also that the activities flow smoothly. In this project, machine learning techniques are chosen to detect phishing URLs. The dataset of this project is collected from the UCI Machine Learning Repository database while the tool that is used to analyze data is Spyder. Spyder is a free and open-source scientific environment for developers and data analysts written in Python. Python is suitable for detect phishing because it has a great library ecosystem for machine learning such as scikit-learn and pandas. It's also simple to use and allows for rapid data validation.

### 3.2 Methodology

. The methodology is a set of plans, concepts, guidelines, or recommendations on how to proceed with the preparation of a project or documentation. The process of detecting phishing URLs by machine learning algorithms such as dataset, comparison, and classification will explain at below. Figure 3.1 shows the Flowchart of phishing URL detection.



**Figure 3.1 Flowchart for phishing URL detection**

## 1) Dataset

- Load Dataset

This dataset is mainly obtained from the MillerSmiles archive, Phish Tank archive, and Google searching operators which consists of 30 attributes and 2456 instances. The value presented in each attribute is -1, 0, and 1. -1 represents as legitimate, 0 represented as suspicious while 1 presents as phishing.

- Spilt Dataset

After loading the dataset, the dataset is divided into two parts, 80% of the training set and 20% of the testing set so that the machine learning algorithm able to learn from the data and make predictions. The training test split ratio of 80:20 provides enough training data and the algorithm can perform better (Rácz et al., 2021).

## 2) Comparison

Comparison between 3 machine learning algorithms such as LR, RF, and CB are conducted to determine which machine learning algorithms achieved the highest accuracy on phishing URLs detection based on the dataset. The confusion matrix is used to evaluate the performance of a classification method. The confusion matrix compares actual data to the predictions of the machine learning model. This gives an overall view of the classification model's performance as well as the types of errors it makes. The following are the different confusion matrix values:

- True Positive (TP) indicating that the model accurately identified positive class data.
- True Negative (TN) indicating that the model accurately identified negative class data.

- False Positive (FP) indicating that the model wrongly identified negative class data as positive class data.
- False Negative (FN) indicating that the model wrongly identified positive class data as the negative class.

Besides, accuracy, precision, recall, and f1-score are also calculated to measure the performance of machine learning algorithms.

### 3) Classification

The most accurate machine learning algorithm will be used to create a URL Classifier. The features of the URL will be extracted once URL is entered. The features are divided into 4 parts which are address bar-based features, abnormal-based features, HTML and JavaScript features, and domain-based features. After extracting the features of the URL, the machine learning model will be loaded to classify the URL as phishing or legitimate. Table 3.1 shows the feature extraction which includes feature group, features factor indicator, and description.

**Table 3.1: Feature Extraction**

Feature group	Features Factor Indicator	Description
Address Bar based Features	Using the IP Address	To retrieve sensitive information from users, phishers will utilize an IP address rather than the URL's domain name. As seen in the link <a href="http://0x42.1.x.DA.0xH7.65/share.html">http://0x42.1.x.DA.0xH7.65/share.html</a> , sometimes an IP address is translated into hexadecimal characters. If the URL has an IP

		address it considers a phishing URL; otherwise, it is a legitimate URL.
	Long URL to Hide the Suspicious Part	By employing a long URL, phishers can hide the suspicious element of a URL in the address bar. An average URL length is obtained by measuring the length of URLs in the dataset, If URL length more than 75 characters it considers a phishing URL, if URL length between 54 and 75 it considers a suspicious URL, and if the URL length is less than 54, it considers as a legitimate URL.
	Using URL Shortening Services “Tiny URL”	URL shortening is a technique for making a URL longer while still directing to the target webpage on the "World Wide Web." This is accomplished by employing a "HTTP Redirect" on a short domain name that connects to a long URL website. An example of a URL is “bit.ly/25TGk9”. If URL is “Tiny URL, it considers a phishing URL. If not, it is a legitimate URL.
	Using “@” in the URL	When the “@” character appears in a URL, the browser ignores anything that comes before the “@,” and the right address is always discovered after the “@.” If URL contains “@”, it considers a



		phishing URL. If not, consider it a legitimate URL.
	Redirecting using “//”	The inclusion of the “//” in the URL means that the user will be redirected to many websites. The “//” in the HTTP URL should be inserted in the sixth position, while the “//” in the HTTPS URL should be placed in the seventh position. If the last occurrence of “//” in the URL is greater than 7, the URL is considered phishing. If not, the URL considers as legitimate.
	Adding Prefix or Suffix Separated by ‘-’ to the Domain	In legitimate URLs, the “-” symbol is seldom used. Phishers usually apply prefixes of domain name separated by the symbol “-” to make users believe they are accessing the legitimate URL. If URL contains a “-” symbol it considers a phishing URL. If not, it considers as legitimate URL.
	Sub Domain and Multi-Sub Domains	The example of <a href="http://www.ulearn.edu.uk/students/">http://www.ulearn.edu.uk/students/</a> is used to create a rule to extract this function. The “edu” part is education and combines the edu.us is called second-level domain. First, remove the (www.) from the URL, then remove the country code uk and

		<p>finally count the remaining dots. If the no of dots in the domain part more than 1, it considers a suspicious URL, if the no of dots is more than 2, it considers a phishing URL while the no of dots equal to 1, it considers a legitimate URL.</p>
	<p>HTTPS (HyperText Transfer Protocol with Secure Socket Layer)</p>	<p>The presence of HTTPS is important in conveying a website's validity, but it is insufficient; therefore, analyzing the certificate linked with HTTPS, including the trust certificate issuer's scope of trust and the certificate's age, is included to assess the website's legitimacy. After checking our datasets, we determined that a credible certificate must be at least two years old. If the URL uses HTTPS and trusts the issuer, as well as the certificate's age, it considers a legitimate URL. If URL uses HTTPS with an untrustworthy issuer, it considers a suspicious URL, otherwise, phishing URL.</p>
	<p>Domain Registration Length</p>	<p>Domains that are trusted are paid for in advance for many years. The longest fraudulent domains in the dataset were only active for a year. If the domain expires less than 1years, it considers a phishing URL. If not, it considers a legitimate URL.</p>

	F avicon	As a visual reminder of the website's identity, many modern user agents display the favicon in the address bar. The webpage is most likely a Phishing webpage if the favicon is loaded from a domain other than the one showing in the URL bar. If not, it considers a legitimate webpage.
	Using Non-Standard Port	It is much safer to only open ports that are needed to control intrusions. Several firewalls, proxy servers, and Network Address Translation (NAT) servers can block all or most ports by default, allowing only access to those that have been chosen. If the port is of the preferred status, it considers a phishing URL. If not, it considers a legitimate URL.
	The existence of “HTTPS” Token in the Domain Part of the URL	Phishers can insert the “HTTPS” token to the domain section of a URL to trick users. It is considered a phishing URL if HTTPS is included in the domain section of the URL. If not, it considers the URL to be legitimate.
Abnormal Based Features	Request URL	External elements on the page, such as photos, videos, and sounds, are checked to see if they are loaded from a different domain using the request URL. In legitimate web pages, the webpage address and the most of the objects on the page are all under the same domain. If the

		request URL less than 22%, it considers a legitimate URL. If the request URL between 22% and 61%, it considers a suspicious URL. Otherwise, phishing URL.
	URL of Anchor	An anchor is defined as an element by the <a> tag. This functionality works the same method as the "Request" URL. It considered a legitimate URL if the anchor URL is less than 31%. It considers a suspicious URL if the anchor URL is between 31% and %. Otherwise, phishing URL.
	Links in <Meta>, <Script> and <Link> tags	<Meta> tags are commonly used by legal websites to include information about the HTML document; <Script> tags are used to generate a client-side script; and <Link> tags are used to retrieve other web resources. If links in "<Meta>","<Script>" and "<Link>" less than 17% it considers a legitimate URL. If links in "<Meta>","<Script>" and "<Link>" between 17% and 81%, it considers a suspicious URL. Otherwise, phishing URL.
	Server Form Handler (SFH)	Action must be made based on the information provided, hence SFHs with an empty string or "about: blank" are suspicious. Furthermore, if the domain name in SFHs differs

		from the domain name of the webpage, the site is suspicious, as external domains rarely handle submitted data. If SFH equal to about: blank, it considers a phishing URL. If a different domain is defined by SFH, it considers a suspicious URL. Otherwise, it considers a legitimate URL.
	Submitting Information to Email	The user's details could be forwarded to the phisher's email address. This can be done with a server-side programming language like PHP's "mail()" capability. Another client-side feature that might be used for this is the "mailto:" function. If contains mail() and mailto() in PHP, it considers a phishing URL. If not, it considers a legitimate URL.
	Abnormal URL	The WHOIS database can be used to retrieve this information. A legitimate website's URL usually includes the user's identity. It considers a phishing URL if the hostname is not mentioned in the URL. If user's identity include in WHOIS database, it considers the URL to be legitimate.
HTML and JavaScript	Website Forwarding	The number of times a webpage is redirected determines if it is a legitimate or phishing website. In the dataset, the legitimate websites

-based Features		were only routed once. This functionality has been redirected at least four times on phishing websites. It is considered a legitimate URL if the number of web pages redirected is less than or equal to one. It is considered a suspicious URL if the number of web pages redirected is between 2 and 4. Otherwise, phishing URL.
	Status Bar Customization	Phishers can mislead visitors into reading a fake URL in the status bar by using JavaScript. By looking at the source code of the webpage, specifically the "onMouseOver" (oMO) case, and seeing if the status bar changes. If the status bar changes, oMO considers the URL to be a phishing URL. If not, it considers the URL to be legitimate.
	Disabling Right Click	Phishers apply JavaScript to disable the right-click feature, preventing users from reading and downloading the webpage source code. By viewing the webpage source code, "event.button==2" and the right click is blocked for this function. If right-click disabled, it considers a phishing URL. If not, it considers a legitimate URL.
	Using Popup Window	Popup Window is used in the legitimate website to alert users about fraudulent activities and most

		phishing websites required the user to fill in personal information through the popup window. If text fields in a Popup Window, it considers a phishing URL. If not, it considers a legitimate URL.
	Iframe Redirection	Iframe is a technique for embedding a link from another website inside the current one. Phishers can utilize the "iframe" tag to make a website transparent. In this situation, phishers use the "frameBorder" feature, which allows the browser to create a visual border. It considers a phishing URL when using an iframe. If not, it considers that URL to be legitimate.
Domain-based Features	Age of Domain	The most of phishing websites only exist for a brief period of time. After evaluating the dataset, the legitimate domain has at least 6 months. It considers a legitimate URL if the domain age is equal to or greater than 6 months. If it doesn't, it's a phishing URL.
	DNS Record	For phishing websites, there are no records for the hostname. It considers a legitimate URL if a DNS record is empty or not found. Otherwise, it considers the URL to be a phishing URL.
	Website Traffic	This function determines the popularity of a website by counting

		<p>the number of people and the number of pages they visit. Legitimate websites were placed among the top 100,000 in the worst-case scenarios. The dataset considers a website to be legitimate if its rank is less than 100,000. If the number of visitors to a website exceeds 100,000, it is deemed a suspicious URL. It is considered a phishing URL if the URL receives no traffic or is not listed in the Alexa database.</p>
	PageRank	<p>PageRank is a number between 0 and 1 that indicates how important a website is. PageRank's purpose is to determine the important of the webpage on the Internet. The better the PageRank value of a webpage, the more crucial it is. It considers a phishing URL if the page's ranking is less than 0.2. If not, it considers the URL to be legitimate.</p>
	Google Index	<p>A website's indexing by Google enables it to appear in search results. Many phishing webpages may not be listed in the Google index since they are only available for a short time. If Google has indexed this page, it is considered as a legitimate URL. If it doesn't, it's a phishing URL.</p>



	Number of Links Pointing to Page	The no of links pointing to a website shows its degree of trustworthiness. Due to their short lifespan, 98 percent of phishing dataset objects have no connections pointing to them in datasets. If no of links pointing to the page equal 0, it considers a phishing URL. If no of links pointing to a page more than 0 and less and equal to 2, it considers a suspicious URL. Otherwise, it considers a legitimate URL.
	Statistical Reports Based Feature	Several agencies, such as PhishTank and StopBadware, publish weekly or quarterly statistical data on phishing websites. There are two types of top ten statistics, according to PhishTank's statistical studies: "Top 10 Domains" and "Top 10 IPs." If the host belongs to a list of top phishing IPs or domains, it is considered a phishing URL. If not, it considers the URL to be legitimate.

### 3.3 Research Milestone

A project milestone is a systematic approach for designating a clear point in the timetable of a project. Milestones serve as checkpoints in the project, ensuring that it stays on track. It ensures that the project can be completed on time. Table 3.2 shows

the research milestone of PSM 1. Table 3.3 shows the research milestone of PSM 2. A Gantt chart is a type of bar chart that illustrates a project timeline and displays activity constraints as well as the current situation of the schedule. Figure 3.1 shows the Gantt Chart of PSM 1 while figure 3.2 shows the Gantt Chart of PSM 2.

**Table 3.2: Research Milestone of PSM 1**

Week	Date	Activity
1	15 March 2021 – 21 March 2021	<ul style="list-style-type: none"> <li>• Proposal discussion</li> <li>• Proposal assessment &amp; verification</li> <li>• Proposal Correction &amp; Improvement</li> </ul>
2	22 March 2021 – 28 March 2021	<ul style="list-style-type: none"> <li>• Chapter 1(System Development Begins)</li> </ul>
3	29 March 2021 – 4 April 2021	<ul style="list-style-type: none"> <li>• Chapter 1</li> <li>• Chapter 2</li> </ul>
4	5 April 2021 – 11 April 2021	<ul style="list-style-type: none"> <li>• Chapter 2</li> </ul>
5	12 April 2021 – 18 April 2021	<ul style="list-style-type: none"> <li>• Chapter 2</li> </ul>
6	19 April 2021 – 25 April 2021	<ul style="list-style-type: none"> <li>• Chapter 3</li> </ul>
7	26 April 2021 – 2 May 2021	<ul style="list-style-type: none"> <li>• Chapter 3</li> </ul>
8	3 May 2021 – 9 May 2021	<ul style="list-style-type: none"> <li>• Chapter 4</li> </ul>

9	10 May 2021 – 16 May 2021	<ul style="list-style-type: none"> <li>• Chapter 4</li> <li>• Project Demo</li> </ul>
10	17 May 2021 – 23 May 2021	Midterm Semester Break
11	24 May 2021 – 30 May 2021	<ul style="list-style-type: none"> <li>• Project Demo</li> </ul>
12	31 May 2021 – 6 June 2021	<ul style="list-style-type: none"> <li>• Project Demo</li> </ul>
13	7 June 2021 – 13 June 2021	<ul style="list-style-type: none"> <li>• Project Demo</li> <li>• PSM 1 Report</li> </ul>
14	14 June 2021 – 20 June 2021	<ul style="list-style-type: none"> <li>• Project Demo</li> </ul>
15	21 June 2021 – 27 June 2021	<ul style="list-style-type: none"> <li>• Final report submission and presentation</li> </ul>
16	28 June 2021 - 4 July 2021	<ul style="list-style-type: none"> <li>• Revision Week, correction on the draft report</li> </ul>

**Table 3.3: Research Milestone of PSM 2**

<b>Week</b>	<b>Date</b>	<b>Activity</b>
1	19 July 2021 – 25 July 2021	<ul style="list-style-type: none"> <li>• Meeting with supervisor</li> <li>• Discussion of PSM1 presentation</li> <li>• Correcting report</li> </ul>
2	26 July 2021 – 1 August 2021	<ul style="list-style-type: none"> <li>• Chapter 5</li> </ul>
3	2 August 2021 – 8 August 2021	<ul style="list-style-type: none"> <li>• Chapter 5</li> </ul>
4	9 August 2021 – 15 August 2021	<ul style="list-style-type: none"> <li>• Chapter 5</li> <li>• Journal.</li> </ul>
5	16 August 2021 – 22 August 2021	<ul style="list-style-type: none"> <li>• Chapter 5</li> <li>• Journal.</li> </ul>
6	23 August 2021 – 29 August 2021	<ul style="list-style-type: none"> <li>• Chapter 6</li> <li>• Journal</li> </ul>
7	30 August 2021 – 5 September 2021	<ul style="list-style-type: none"> <li>• Complete report, journal and log book</li> </ul>
8	6 September – 8 September 2021	<ul style="list-style-type: none"> <li>• Presentation week</li> <li>• Submission of report and logbook</li> </ul>



### 3.4 Conclusion

This chapter has discussed the methodology used in this research. The technique for this study is machine learning. A project milestone and a Gantt chart are also included in this section to ensure that the analysis can be completed on time. The research implementation will be discussed in the following chapter.



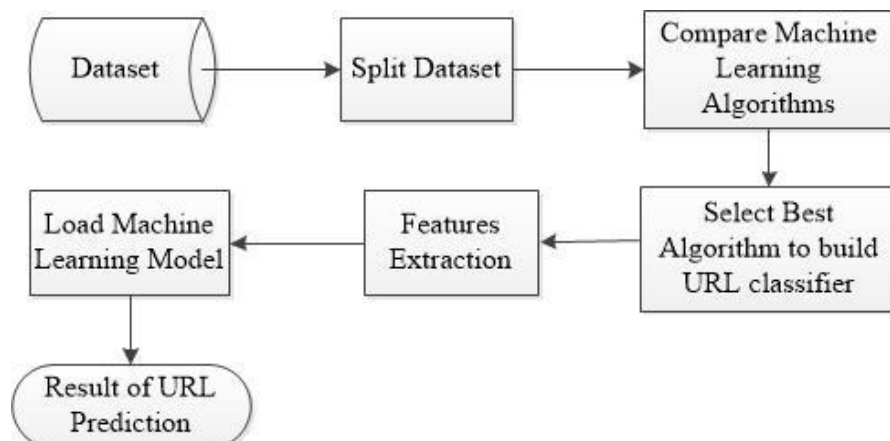
## CHAPTER 4: IMPLEMENTATION

### 4.1 Introduction

The previous chapter had covered the research methodology while this chapter will show the steps in implementation. This chapter will explain in detail the stage of phishing URL detection such as dataset, comparison, and classification. At the same time, the features used in the system are discussed in this chapter.

### 4.2 Implementation Steps

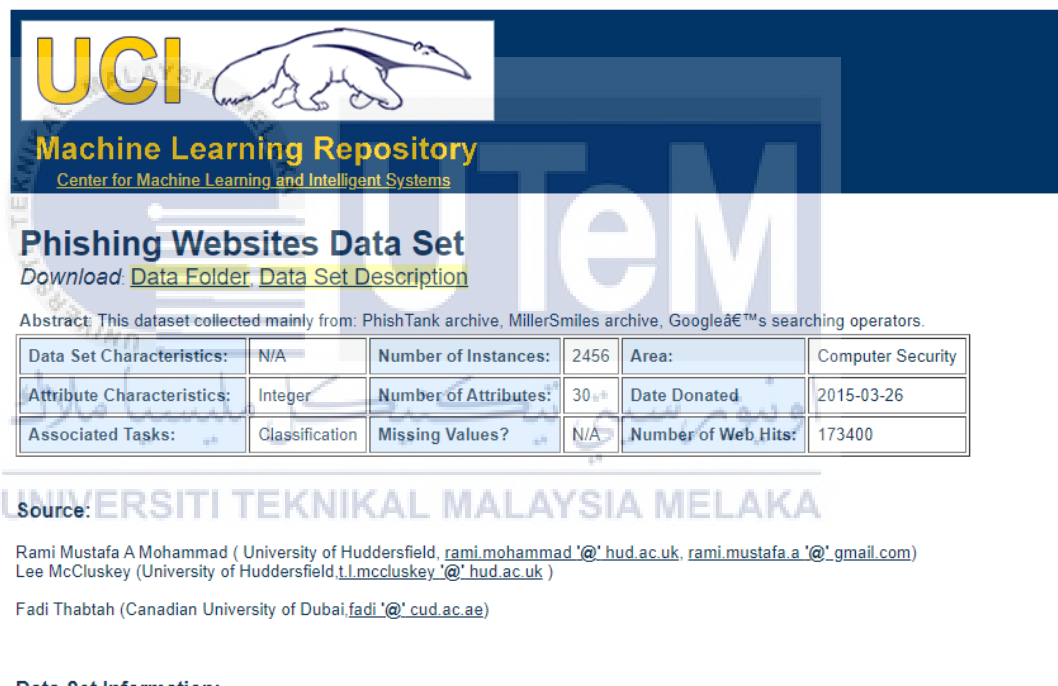
There contain three stages to detect phishing URLs. The stages included dataset, comparison, and classification as mentioned in the previous chapter. Figure 4.1 shows the block flow diagram of this project.



## Figure 4.1 Block Flow Diagram

### 4.2.1 Dataset

In the first phase, the phishing website dataset is collected from UCI Machine Learning Repository (Mohammad et al., 2015). This dataset contains 30 features to identify the phishing URL. Figure 4.2 shows the UCI Machine Learning Repository while figure 4.3 shows the dataset of a phishing website.



**UCI** Machine Learning Repository  
Center for Machine Learning and Intelligent Systems

**Phishing Websites Data Set**  
Download: [Data Folder](#), [Data Set Description](#)

Abstract: This dataset collected mainly from: PhishTank archive, MillerSmiles archive, Googleâ€™s searching operators.

Data Set Characteristics:	N/A	Number of Instances:	2456	Area:	Computer Security
Attribute Characteristics:	Integer	Number of Attributes:	30 <sub>+</sub>	Date Donated:	2015-03-26
Associated Tasks:	Classification	Missing Values?	N/A	Number of Web Hits:	173400

Source:  
 Rami Mustafa A Mohammad ( University of Huddersfield, [rami.mohammad@hud.ac.uk](mailto:rami.mohammad@hud.ac.uk), [rami.mustafa.a@gmail.com](mailto:rami.mustafa.a@gmail.com))  
 Lee McCluskey (University of Huddersfield, [l.mcccluskey@hud.ac.uk](mailto:l.mcccluskey@hud.ac.uk))  
 Fadi Thabtah (Canadian University of Dubai, [fadi@hud.ac.uk](mailto:fadi@hud.ac.uk))

Data Set Information:

**Figure 4.2 UCI Machine Learning Repository**



A	B	C	D	E	F	G	H	I	J	K	L	M
id	having_IP	URL_Leng	Shortning	having_At	double_slz	Prefix_Suft	having_Su	SSLfinal_St	Domain_r	Favicon	port	HTTPS_to
1	-1	1	1	1	-1	-1	-1	-1	-1	1	1	-1
2	1	1	1	1	1	-1	0	1	-1	1	1	-1
3	1	0	1	1	1	-1	-1	-1	-1	1	1	-1
4	1	0	1	1	1	-1	-1	-1	1	1	1	-1
5	1	0	-1	1	1	-1	1	1	-1	1	1	1
6	-1	0	-1	1	-1	-1	1	1	-1	1	1	-1
7	1	0	-1	1	1	-1	-1	-1	1	1	1	1
8	1	0	1	1	1	-1	-1	-1	1	1	1	-1
9	1	0	-1	1	1	-1	1	1	-1	1	1	-1
10	1	1	-1	1	1	-1	-1	1	-1	1	1	1
11	1	1	1	1	1	-1	0	1	1	1	1	1
12	1	1	-1	1	1	-1	1	-1	-1	1	1	1
13	-1	1	-1	1	-1	-1	0	0	1	1	1	-1
14	1	1	-1	1	1	-1	0	-1	1	1	1	1
15	1	1	-1	1	1	1	-1	1	-1	1	1	-1
16	1	-1	-1	-1	1	-1	0	0	1	1	1	1
17	1	-1	-1	1	1	-1	1	1	-1	1	1	-1
18	1	-1	1	1	1	-1	-1	0	1	1	-1	1
19	1	1	1	1	1	-1	-1	1	1	1	1	-1

**Figure 4.3 Dataset of Phishing URL**

After loading the dataset, the dataset is divided into 80% for the training dataset while 20% for the testing dataset so that machine learning algorithms are able to learn from the dataset to predict the result. Figure 4.4 shows the coding of the splitting dataset.

```

simplefilter(action='ignore', category=FutureWarning)

URLS = pd.read_csv("data.csv")
URLS = URLS.drop(URLS.columns[[0]], axis=1)

URLS = URLS.sample(frac=1).reset_index(drop=True)
URLS_Without_Labels = URLS.drop('Result', axis=1)

Labels = URLS['Result']

Training_Data, Testing_Data = train_test_split(URLS_Without_Labels, test_size=0.20, random_state=110)
Training_Labels, Testing_Labels = train_test_split(Labels, test_size=0.20, random_state=110)

```

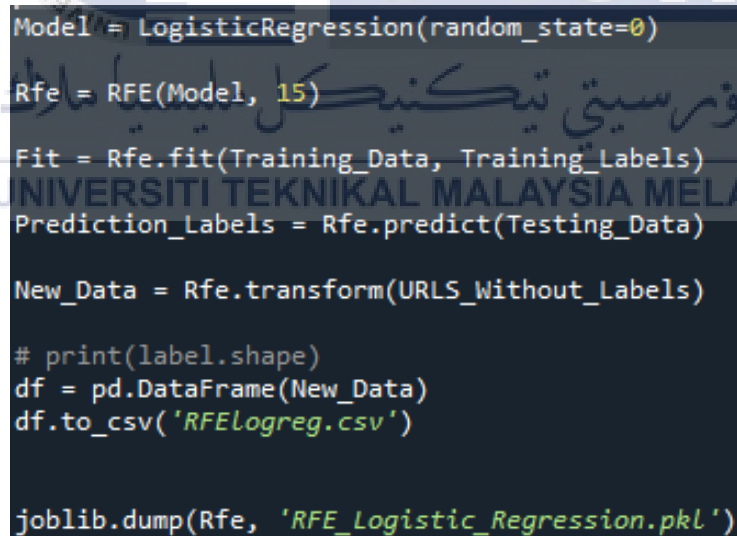
**Figure 4.4 Split Dataset**

### 4.2.2 Comparison

The libraries included in the comparison stage are joblib, pandas, and sklearn. After splitting the dataset for training and testing, machine learning models such as LR, RF, and CB learn information directly from the dataset.

- Train Machine Learning Model

Recursive Feature Elimination (RFE) is applied to each machine learning model to select the most suitable features for the dataset. Out of 30 features, 25 features are selected to make the machine learning algorithm more efficient and effective since irrelevant features will mislead the machine learning to learn and result in worse predictive performance. Then machine learning model is saved in a pkl file to reuse later for URL classification and prediction. Figure 4.5 shows the code for the Logistic Regression model. Figure 4.6 shows the code for the Random Forest model while Figure 4.7 shows the code for the CatBoost model.



```

Model = LogisticRegression(random_state=0)
Rfe = RFE(Model, 15)
Fit = Rfe.fit(Training_Data, Training_Labels)
Prediction_Labels = Rfe.predict(Testing_Data)

New_Data = Rfe.transform(URLS_Without_Labels)

# print(label.shape)
df = pd.DataFrame(New_Data)
df.to_csv('RFELogreg.csv')

joblib.dump(Rfe, 'RFE_Logistic_Regression.pkl')

```

**Figure 4.5 Logistic Regression Model**

```

Model = RandomForestClassifier(random_state=0)
Rfe = RFE(Model, 15)
Fit = Rfe.fit(Training_Data, Training_Labels)
Prediction_Labels = Rfe.predict(Testing_Data)
Confusion_Matrix = confusion_matrix(Testing_Labels, Prediction_Labels)
New_Data = Rfe.transform(URLS_Without_Labels)

# print(label.shape)
df = pd.DataFrame(New_Data)
df.to_csv('RFErandfor.csv')

joblib.dump(Rfe, 'RFE_Random_Forest.pkl')

```

**Figure 4.6 Random Forest Model**

```

Model = CatBoostClassifier(iterations=None, random_state=0)
Rfe = RFE(Model, 15)
Fit = Rfe.fit(Training_Data, Training_Labels)
Prediction_Labels = Rfe.predict(Testing_Data)
Confusion_Matrix = confusion_matrix(Testing_Labels, Prediction_Labels)
New_Data = Rfe.transform(URLS_Without_Labels)

# print(label.shape)
df = pd.DataFrame(New_Data)
df.to_csv('RFEcatboost.csv')

joblib.dump(Rfe, 'RFE_CatBoost.pkl')

```

**Figure 4.7 CatBoost Model**

- Calculate Accuracy & Evaluate the Performance of Machine Learning Algorithms

The performance of a classification algorithm is measured by using a confusion matrix. Accuracy, precision, recall, f1-score is calculated in each machine learning model (LR, RF, and CB). The comparison of machine learning models

is conducted to determine the machine learning model that performs the best in terms of accuracy. Figure 4.8 shows the code of the confusion matrix.

```
Confusion_Matrix = confusion_matrix(Testing_Labels, Prediction_Labels)
Acc= accuracy_score(Testing_Labels, Prediction_Labels)
print("Training Accuracy Score Obtained is: {0:.2f}%".format(Fit.score(Training_Data,
print("Testing Accuracy Score Obtained is: {0:.2f}%\n".format(accuracy_score(Testing_
print("Classification Report: \n",classification_report(Testing_Labels, Prediction_La
print("Confusion Matrix: \n", Confusion_Matrix)
print("\n Accuracy:\n", Acc)
```

**Figure 4.8 Confusion Matrix**

### 4.2.3 Classification

After the comparison stage, the best machine learning model is selected to build a URL classifier. The libraries included in this stage are re (regular expression), time, whois, socket, pandas, joblib, and urllib. Class Feature\_Extraction is created and all the function of features is stored in the class. Figure 4.9 URL connection with Spyder.

```
http_https = r"https://|http://"
class Feature_Extraction:
    def __init__(self):
        pass

    def Get_Protocol(self, url):
        return urlparse(url).scheme

    def Get_Domain(self, url):
        return urlparse(url).netloc

    def Get_Path(self, url):
        return urlparse(url).path

    def Get_Hostname_From_URL(self, url):
        hostname = url
        pattern = "https://|http://|www.|https://www.|http://www."
        pre_pattern_match = re.search(pattern, hostname)

        if pre_pattern_match:
            hostname = hostname[pre_pattern_match.end():]
            post_pattern_match = re.search("/", hostname)
            if post_pattern_match:
                hostname = hostname[:post_pattern_match.start()]

        return hostname
```

**Figure 4.9 URL Connection with Spyder**

The phishing website features are categorized into 4 groups which are address bar-based features, abnormal-based features, HTML and JavaScript-based features, and domain-based features. The previous chapter covered a detailed explanation about the 4 groups of phishing websites' features. Figures 4.10 until figure 4.13 will show the address bar-based features of phishing websites.

- IP\_address function: If URL contains IP address it considers a phishing URL (return 1). If not, it is legitimate URL (return -1).
- LongURL function: If URL length more than 75 characters it considers a phishing URL (return 1), if URL length between 54 and 75 it considers a suspicious URL (return 0), and if URL length is less than 54, it considers a legitimate URL (return -1).
- Tiny URL function: If URL is “Tiny URL” such as bit.ly/25TGk9, it considers a phishing URL (return 1). If not, it is a legitimate URL (return -1).

The code for extracting features such as using IP address, Long URL, and Tiny URL is shown in Figure 4.10.

```
def Ip_Address(self, url):
    match = re.search('(?:[0-9]{1,3}\.){3}[0-9]{1,3}|(?:[0-9]{1,3}\.){2}[0-9]{1,3}|(?:[0-9]{1,3}\.){1}[0-9]{1,3}|(?:[0-9]{1,3})', url)
    if match:
        return 1
    else:
        return -1

def LongURL(self, url):
    if len(url) < 54:
        return -1
    elif len(url) >= 54 and len(url) <= 75:
        return 0
    else:
        return 1

def TinyURL(self, url):
    match = re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cl|i\.gs|'
    'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|'
    'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|'
    'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|'
    'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|'
    'q\.gs|is\.gd|po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.or'
    'g\.co|prettylinkpro\.com|scrncr\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.me|', url)
    if match:
        return 1
    else:
        return -1
```

Figure 4.10 IP address, Long URL, and Tiny URL

- ATSymbol function: If the URL contains “@”, it considers a phishing URL (return 1). If not, consider it as a legitimate URL (return -1).
- Redirection function: If the position of the last occurrence of “//” in the URL more than 7, it considers a phishing URL (return 1). If not, it considers a legitimate URL (return -1).
- PrefixSuffixSeperation function: If URL contains “-” symbol it considers a phishing URL (return 1). If not, it considers a legitimate URL (return -1).
- SubDomain function: If the no of dots in the domain part more than 1, it considers a suspicious URL (return 0), if the no of dots is more than 2, it considers a phishing URL (return 1) while the no of dots equal to 1, it considers as legitimate URL (return -1).
- SSL function: If the URL use HTTPS and trust the issuer, as well as the certificate's age, it considers a legitimate URL (return -1). If URL uses HTTPS with an untrustworthy issuer, it considers a suspicious URL (return 0), otherwise, phishing URL (return 1).

Figure 4.11 shows the code of using AT (@) symbol in URL, redirecting using “//”, adding prefix or suffix separated by “-” to the domain, Subdomain, and Multi-subdomains and HyperText Transfer Protocol with Secure Socket Layer (SSL).

```

def ATSymbol(self, url):
    if "@" in url:
        return 1
    else:
        return -1

def Redirection(self, url):
    if "/" in urlparse(url).path:
        return 1
    else:
        return -1

def PrefixSuffixSeparation(self, url):
    if '-' in urlparse(url).netloc:
        return 1
    else:
        return -1

def SubDomains(self, url):
    if url.count(".") < 3:
        return -1
    elif url.count(".") == 3:
        return 0
    else:
        return 1

def SSL(self, url):
    return -1

```

**Figure 4.11** AT symbol, redirecting using “//, prefix suffix separation, subdomains, and SSL

- DomainRegistrationLength function: If the domain expires less than 1 year, it considers a phishing URL (return 1). If not, it considers a legitimate URL (return -1).

Figure 4.12 shows the code domain registration length.

```

def DomainRegistrationLength(self, url):
    dns = 0
    try:
        domain_name = whois.whois(urlparse(url).netloc)
    except:
        dns = 1

    if dns == 1:
        return 1 #phishing
    else:
        expiration_date = domain_name.expiration_date
        today = time.strftime('%Y-%m-%d')
        today = datetime.strptime(today, '%Y-%m-%d')
        if expiration_date is None:
            return 1
        elif type(expiration_date) is list or type(today) is list :
            return 0
        else:
            creation_date = domain_name.creation_date
            expiration_date = domain_name.expiration_date
            if (isinstance(creation_date,str) or isinstance(expiration_date,str)):
                try:
                    creation_date = datetime.strptime(creation_date, '%Y-%m-%d')
                    expiration_date = datetime.strptime(expiration_date, "%Y-%m-%d")
                except:
                    return 0
            registration_length = abs((expiration_date - today).days)
            if registration_length / 365 <= 1:
                return 1
            else:
                return -1

```

**Figure 4.12 Domain Registration Length**

- Favicon function: If the favicon is loaded from a domain other than the one shown in the address bar, the webpage is a phishing webpage (return 1). If not, it considers a legitimate webpage (return -1).
- Port function: If port # is of the preferred status, it considers a phishing URL (return 1). If not, it considers a legitimate URL (return -1).
- HttpsToken: If HTTPS is present in the domain part of the URL, it considers a phishing URL (return 1). If not, it considers a legitimate URL (return -1).

Figure 4.13 shows Favicon, Using Non-Standard Port, and the existence of the “HTTPS” token in the domain part of the URL.



```

def Favicon(self, url, soup, domain):
    for head in soup.find_all('head'):
        for head.link in soup.find_all('link', href=True):
            dots = [x.start() for x in re.finditer(r'\.', head.link['href'])]
            return -1 if url in head.link['href'] or len(dots) == -1 or domain in head.link['href'] else 1
        return -1

def Port(self, url):
    return -1

def HttpsToken(self, url):
    match = re.search('https://|http://',url)
    try:
        if match.start(0)==0 and match.start(0) is not None:
            url = url[match.end(0):]
            match = re.search('http/https',url)
            if match:
                return 1
            else:
                return -1
    except:
        return 1

```

**Figure 4.13 Favicon, Non-Standard Port and Https Token**

Figures 4.14 until figure 4.17 will show the address abnormal-based features of phishing websites.

- RequestURL function: If request URL less than 22%, it considers a legitimate URL (return -1). If request URL between 22% and 61%, it considers a suspicious URL (return 0). Otherwise, phishing URL (return 1).

Figure 4.14 shows the request URL feature.

```

def RequestURL(self, url, soup, domain):
    i = 0
    success = 0
    for img in soup.find_all('img', src=True):
        dots = [x.start() for x in re.finditer(r'\.', img['src'])]
        if url in img['src'] or domain in img['src'] or len(dots) == 1:
            success = success + 1
        i = i + 1

    for audio in soup.find_all('audio', src=True):
        dots = [x.start() for x in re.finditer(r'\.', audio['src'])]
        if url in audio['src'] or domain in audio['src'] or len(dots) == 1:
            success = success + 1
        i = i + 1

    for embed in soup.find_all('embed', src=True):
        dots = [x.start() for x in re.finditer(r'\.', embed['src'])]
        if url in embed['src'] or domain in embed['src'] or len(dots) == 1:
            success = success + 1
        i = i + 1

    for i_frame in soup.find_all('i_frame', src=True):
        dots = [x.start() for x in re.finditer(r'\.', i_frame['src'])]
        if url in i_frame['src'] or domain in i_frame['src'] or len(dots) == 1:
            success = success + 1
        i = i + 1
    try:
        percentage = success / float(i) * 100
    except:
        return 1

    if percentage < 22.0:
        return -1
    elif 22.0 <= percentage < 61.0:
        return 0
    else:
        return 1

```

Figure 4.14 Request URL

- URLOfAnchor function: If the URL of anchor less than 31%, it considers a legitimate URL (return -1). If the URL of anchor between 31% and 67%, it considers a suspicious URL (return 0). Otherwise, phishing URL (return 1).

Figure 4.15 shows the URL of Anchor.

```

def URLOfAnchor(self, url, soup, domain):
    i = 0
    unsafe = 0
    for a in soup.find_all('a', href=True):
        if "#" in a['href'] or "javascript" in a['href'].lower() or "mailto" in a['href'].lower()
            url in a['href'] or domain in a['href']):
            unsafe = unsafe + 1
        i = i + 1
    try:
        percentage = unsafe / float(i) * 100
    except:
        return -1
    if percentage < 31.0:
        return -1
    elif 31.0 <= percentage < 67.0:
        return 0
    else:
        return 1

```

Figure 4.15 URL of Anchor

- LinksInTags function: If links in "<Meta>","<Script>" and "<Link>" less than 17% it considers a legitimate URL (return 0). If links in "<Meta>","<Script>" and "<Link>" between 17% and 81%, it considers a suspicious URL. (return 2) Otherwise, phishing URL (return 1).

Figure 4.16 shows links in <Meta>, <Script> and <Link> tags.

```
def LinksInTags(self, url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'Lxml')
        no_of_meta = 0
        no_of_link = 0
        no_of_script = 0
        anchors = 0
        avg = 0

        for meta in soup.find_all('meta'):
            no_of_meta = no_of_meta + 1
        for link in soup.find_all('link'):
            no_of_link = no_of_link + 1
        for script in soup.find_all('script'):
            no_of_script = no_of_script + 1
        for anchor in soup.find_all('a'):
            anchors = anchors + 1
        total = no_of_meta + no_of_link + no_of_script + anchors
        tags = no_of_meta + no_of_link + no_of_script

        if(total != 0):
            avg = tags / total
            if(avg < 0.17):
                return 0
            elif(0.17 <= avg <= 0.81):
                return 2
            else:
                return 1
        except:
            return 2
```

**Figure 4.16 Links in tags**

- SFH function: If SFH equal to about: blank, it considers a phishing URL (return 1). If a different domain is defined by SFH, it considers a suspicious URL (return 0). Otherwise, it considers a legitimate URL (return -1).
- EmailSubmit function: If contains mail() and mailto() in PHP, it considers as phishing URL (return 1). If not, it considers as legitimate URL (return -1).

- **AbnormalURL function:** If the hostname is not included in URL, it considers a phishing URL (return 1). If not, it considers a legitimate URL (return -1).

Figure 4.17 shows the Server Form Handler (SFH), submitting information to email, and abnormal URL.

```
def SFH(self, url, soup, domain):
    for form in soup.find_all('form', action=True):
        if form['action'] == "" or form['action'] == "about:blank":
            return 1
        elif url not in form['action'] and domain not in form['action']:
            return 0
        else:
            return -1
    return -1

def EmailSubmit(self, url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        if(soup.find('mailto:')):
            return 1
        else:
            return -1
    except:
        return 0

def AbnormalURL(self, hostname, url):
    match = re.search(hostname, url)
    return -1 if match else 1
```

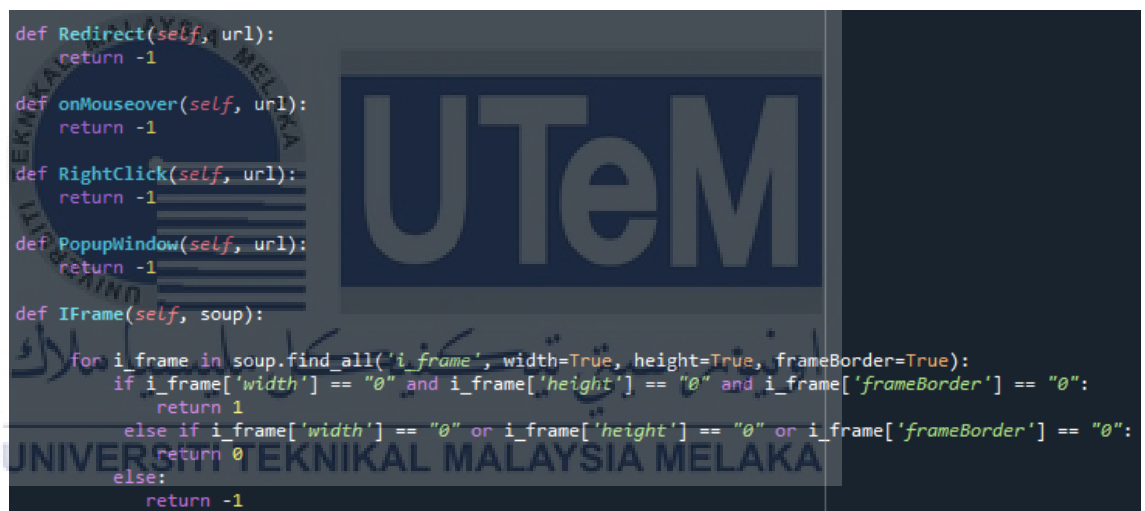
Figure 4.17 SFH, Email Submit, and Abnormal URL

Figures 4.18 will show the HTML and JavaScript-based features of phishing websites.

- **Redirect function:** If the number of web pages redirected less and equal to 1, it considers a legitimate URL (return -1). If the number of web pages redirected between 2 and 4, it considers a suspicious URL (return 0). Otherwise, phishing URL (return 1).
- **onMouseover(oMo) function:** If oMo change status bar, it considers a phishing URL (return 1). If not, it considers a legitimate URL (return -1).

- RightClick function: If right-click disabled, it considers a phishing URL (return 1). If not, it considers a legitimate URL (return -1).
- PopupWindow function: If text fields in a Popup Window, it considers a phishing URL (return 1). If not, it considers a legitimate URL (return -1).
- Iframe function: If using iframe, it considers a phishing URL (return 1). If not, it considers as legitimate URL (return -1).

Figure 4.18 shows the feature of website forwarding (Redirect), status bar customization (onMouseover), disabling right-click, using popup window, and IFrame Redirection.



```

def Redirect(self, url):
    return -1

def onMouseover(self, url):
    return -1

def RightClick(self, url):
    return -1

def PopupWindow(self, url):
    return -1

def IFrame(self, soup):
    for i_frame in soup.find_all('i_frame', width=True, height=True, frameBorder=True):
        if i_frame['width'] == "0" and i_frame['height'] == "0" and i_frame['frameBorder'] == "0":
            return 1
        else if i_frame['width'] == "0" or i_frame['height'] == "0" or i_frame['frameBorder'] == "0":
            return 0
        else:
            return -1

```

**Figure 4.18 Redirect, onMouseover, Right Click, Popup Window, and Iframe**

Figures 4.19 until figure 4.22 will show the domain-based features of phishing websites.

- AgeOfDomain function: If the age of the domain equal to or more than 6 months, it considers a legitimate URL (return -1). If not, it considers a phishing URL (return 1).

Figure 4.19 shows the feature of age of domain.

```
def AgeOfDomain(self, url):
    dns = 0
    try:
        domain_name = whois.whois(urlparse(url).netloc)
    except:
        dns = 1

    if dns == 1:
        return 1
    else:
        creation_date = domain_name.creation_date
        expiration_date = domain_name.expiration_date
        if (isinstance(creation_date, str) or isinstance(expiration_date, str)):
            try:
                creation_date = datetime.strptime(creation_date, '%Y-%m-%d')
                expiration_date = datetime.strptime(expiration_date, "%Y-%m-%d")
            except:
                return 0
            if ((expiration_date is None) or (creation_date is None)):
                return 1
            elif ((type(expiration_date) is list) or (type(creation_date) is list)):
                return 0
            else:
                ageofdomain = abs((expiration_date - creation_date).days)
                if ((ageofdomain/30) < 6):
                    return 1
                else:
                    return -1
```

**Figure 4.19 Age of Domain**

- DnsRecord function: If the DNS record is empty or not found, it considers a legitimate URL (return -1). Otherwise, it considers a phishing URL (return 1).
- WebTraffic function: If the website rank less than 100,000, it considers a legitimate URL (return -1). If, website more than 100,000 is considered a suspicious URL (return 0). If the URL does not receive any traffic or is not included in the Alexa database, it considers a phishing URL (return 1).

Figure 4.20 shows the DNS record and website traffic.

```

def DnsRecord(self, url):
    dns = 0
    try:
        domain_name = whois.whois(urlparse(url).netloc)
    except:
        dns = 1

    if dns == 1:
        return 1
    else:
        return dns

def WebTraffic(self, url):
    try:
        rank = BeautifulSoup(urllib.request.urlopen("http://data.alexa.com/data?cli=10&dat=s&url="
    except TypeError:
        return 1
    except HTTPError:
        return 0
    rank = int(rank)
    if rank < 100000:
        return -1
    else:
        return 0

```

**Figure 4.20 DNS Record and Website Traffic**

- PageRank function: If the website is included in page rank, it considers a legitimate URL (return -1). Otherwise, it considers a phishing URL (return 1).
- GoogleIndex function: If Google has indexed this webpage, it considers a legitimate URL (return -1). If not, it considers a phishing URL (return 1).
- LinksPointingToPage function: If no of links pointing to page equal 0, it considers as phishing URL (return 1). If no of links pointing to a page more than 0 and less and equal to 2, it considers a suspicious URL (return 0). Otherwise, it considers a legitimate URL (return -1).

Figure 4.21 shows the PageRank, Google Index, and the number of links pointing to the page.

```

def PageRank(self, url):
    return -1

def GoogleIndex(self, url):
    return 1

def LinksPointingToPage(self, url):
    count = 0
    try:
        r = urllib.request.urlopen(url)
        soup = BeautifulSoup(r, "html.parser")
        for link in soup.find_all('a'):
            count += 1
    except TypeError:
        return 1
    if(count == 0):
        return 1
    elif(count == 1):
        return 0
    else:
        return -1

```

**Figure 4.21 PageRank, Google Index, and Links Pointing to Page**

- **StatisticalReport** function: If Host Belongs to Top Phishing IPs or Top Phishing Domains it considers a phishing URL (return 1). If not, it considers a legitimate URL (return -1).

Figure 4.22 shows the statistical report.

```

def StatisticalReport(self, url):
    hostname = url
    h = [(x.start(0), x.end(0)) for x in re.finditer('https://|http://|w
    z = int(len(h))
    if z != 0:
        y = h[0][1]
        hostname = hostname[y:]
        h = [(x.start(0), x.end(0)) for x in re.finditer('/', hostname)]
        z = int(len(h))
        if z != 0:
            hostname = hostname[:h[0][0]]
    url_match=re.search('at\.ua|usa\.cc|baltazarpresses\.com\.br|pe\.h
    try:
        ip_address = socket.gethostbyname(hostname)
        ip_match=re.search('146\.112\.61\.108|213\.174\.157\.151|121\.50
    except:
        return 1

    if url_match:
        return 1
    else:
        return 0

```

**Figure 4.22 Statistical Report**



List of each feature is created to store the extracted feature's data and fe function is created to call the class Feature Extraction. The list can store different types of data such as string, integer, and double. Figure 4.23 shows the list of features.

```
fe = Feature_Extraction()

Protocol = []
Domain = []
Path = []

HavingIP = []
URLLength = []
TinyURL = []
HavingAtSymbol = []
RedirectionSymbol = []
PrefixSuffixSeparation = []
SubDomains = []
SSL = []
DomainRegistrationLength = []
Favicon = []
Port = []

HttpTokens = []
RequestURL = []
URLofAnchor = []
LinksInTags = []
SFH = []
EmailSubmit = []
AbnormalURL = []
Redirect = []
OnMouseover = []
RightClick = []
PopUpWindow = []
IFrame = []
AgeOfDomain = []
DNSRecord = []
WebTraffic = []

PageRank = []
GoogleIndex = []
LinksPointingToPage = []
StatisticalReport = []
```

Figure 4.23 Lists of figures

Once the URL is entered, the features of the websites are extracted and add to the created list. Figure 4.24 shows the feature extraction.

```
url = input("Enter URL you want to check: ")
print("Processing, please wait a minute...")

hostname = fe.Get_Hostname_From_URL(url)
opener = urllib.request.urlopen(url).read()
soup = BeautifulSoup(opener, 'Lxml')
domain = fe.Get_Domain(url)
HavingIP.append(fe.Ip_Address(url))
URLLength.append(fe.LongURL(url))
TinyURL.append(fe.TinyURL(url))
HavingAtSymbol.append(fe.ATSymbol(url))
RedirectionSymbol.append(fe.Redirection(url))
PrefixSuffixSeparation.append(fe.PrefixSuffixSeparation(url))
SubDomains.append(fe.SubDomains(url))
SSL.append(fe.SSL(url))
DomainRegistrationLength.append(fe.DomainRegistrationLength(url))
Favicon.append(fe.Favicon(url, soup, hostname))
Port.append(fe.Port(url))
HttpTokens.append(fe.HttpsToken(url))
RequestURL.append(fe.RequestURL(url, soup, hostname))
URLOfAnchor.append(fe.URLOfAnchor(url, soup, hostname))
LinksInTags.append(fe.LinksInTags(url))
SFH.append(fe.SFH(url, soup, hostname))
AbnormalURL.append(fe.AbnormalURL(hostname, url))
Redirect.append(fe.Redirect(url))
OnMouseover.append(fe.onMouseover(url))
RightClick.append(fe.RightClick(url))
PopUpWindow.append(fe.PopupWindow(url))
IFrame.append(fe.IFrame(url))
AgeOfDomain.append(fe.AgeOfDomain(url))
DNSRecord.append(fe.DnsRecord(url))
WebTraffic.append(fe.WebTraffic(url))
PageRank.append(fe.PageRank(url))
GoogleIndex.append(fe.GoogleIndex(url))
LinksPointingToPage.append(fe.LinksPointingToPage(url))
EmailSubmit.append(fe.EmailSubmit(url))
StatisticalReport.append(fe.StatisticalReport(url))
```

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

**Figure 4.24 Feature Extraction**

After all the features of the websites are extracted, it will store all the data in excel to analyze later. Figure 4.25 shows store data in Excel.

```
d = {'having_IP_Address':pd.Series(HavingIP), 'URL_Length':pd.Series(URLLength), 'Shortining_Service':pd.Series(TinyURL),
'having_At_Symbol':pd.Series(HavingAtSymbol), 'double_slash_redirecting':pd.Series(RedirectionSymbol),
'Prefix_Suffix':pd.Series(PrefixSuffixSeparation), 'having_Sub_Domain':pd.Series(SubDomains),
'SSLfinal_State':pd.Series(SSL), 'Domain_registration_Length':pd.Series(DomainRegistrationLength),
'Favicon':pd.Series(Favicon), 'port':pd.Series(Port), 'HTTPS_token':pd.Series(HttpTokens),
'Request_URL':pd.Series(RequestURL), 'URL_of_Anchor':pd.Series(URLOfAnchor), 'Links_in_tags':pd.Series(LinksInTags),
'SFH':pd.Series(SFH), 'Submitting_to_email':pd.Series(EmailSubmit), 'Abnormal_URL':pd.Series(AbnormalURL),
'Redirect':pd.Series(Redirect), 'on_mouseover':pd.Series(OnMouseover), 'RightClick':pd.Series(RightClick),
'popUpWindow':pd.Series(PopUpWindow), 'Iframe':pd.Series(IFrame), 'age_of_domain':pd.Series(AgeOfDomain),
'DNSRecord':pd.Series(DNSRecord), 'web_traffic':pd.Series(WebTraffic), 'Page_Rank':pd.Series(PageRank),
'Google_Index':pd.Series(GoogleIndex), 'Links_pointing_to_page':pd.Series(LinksPointingToPage),
'Statistical_report':pd.Series(StatisticalReport)}
```

```
data = pd.DataFrame(d)
data.to_csv("D:\FYP\Phishing-URL-Detection-master\Check.csv", index=False, encoding='UTF-8')
```

**Figure 4.25 Store Data in Excel**

The machine learning model CB is loaded and classify URL as legitimate URL or phishing URL. Figure 4.26 shows the code of CatBoost's classification.

```
URL = pd.read_csv(r'D:\FYP\Phishing-URL-Detection-master\Check.csv')
joblib = joblib.load(r'D:\FYP\Phishing-URL-Detection-master\RFE_CatBoost.pkl')
if (joblib.predict(URL)) == -1 :
    print("\n"Result: Legitimate URL")
else :
    print("\n"Result: Phishing URL")
```

**Figure 4.26 CatBoost's Classification**

### 4.3 Conclusion

This chapter has discussed the aspects of the implementation of the Detecting Phishing Uniform Resource Locator (URL) by using machine learning algorithms. This chapter covers the process of detect phishing URLs. The procedure for setting up the machine learning model was also presented in this chapter. The testing and analysis part will be carried out on the system to detect phishing URLs in the next chapter.

## CHAPTER 5: TESTING AND ANALYSIS

### 5.1 Introduction

The implementation of phishing URL detection using machine learning algorithms is discussed in the previous chapter. This chapter includes the testing and analysis of the system. The accuracy and efficiency of the classifiers are evaluated and compared in the testing and analysis phase by running different classification models/algorithms on the test data. Machine learning algorithm with the best performance in terms of accuracy will be chosen to develop an URL classifier. In order to verify and ensure that the URL classifier will categorize the URL accurately, known phishing and legitimate URLs are evaluated in the URL classifier.

### 5.2 Result and Analysis

The confusion matrix is a metric for evaluating machine learning classification performance where the output can be more than two classes. There are four values in the confusion matrix. The four values are used to calculate accuracy, precision, recall, and f1 score.

- True Positive (TP) known as the positive value is predicted correctly.

- False Positive (FP) known as the positive value is predicted incorrectly where the true value is negative but it predicts as a positive value.
- False Negative (FN) known as the negative value is predicted incorrectly where the true value is positive but it predicts as a negative value.
- True Negative (TN) defines as the negative value is predicated correctly.

Figure 5.1 shows the values of the confusion matrix.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Figure 5.1 Values of Confusion Matrix

Accuracy is known as the number of correctly identified data instances divided by the total number of data instances. Figure 5.2 shows the formula of calculate accuracy.

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

Figure 5.2 Formula of Accuracy

Precision defines as the percentage of classes that are positive out of all those predicted to be positive. Figure 5.3 shows the formula of calculate precision.

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

**Figure 5.3 Formula of Precision**

Recall represents the rightly predicted positive overall the positive classes. Figure 5.4 shows the formula of calculate recall.

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

**Figure 5.4 Formula of Recall**

The weighted average of Precision and Recall knowns as F1-score. Figure 5.5 shows the formula of calculate F1-score.

$$F1 - score = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$$

**Figure 5.5 Formula of F1-score**

## 5.2.1 Result of Machine Learning Algorithms

### 5.2.1.1 Logistic Regression

The figure below shows the LR's classification result. In the testing dataset, the total phishing website is 933 while the legitimate website is 1278. Based on the confusion matrix, TP value is 861. FP value is 78. FN is 72 and TN is 1200. The accuracy of LR to predict phishing URL dataset is 93.22%. The precision of phishing websites is 92% and legitimate websites is 94%. The recall of phishing websites is 92% while legitimate websites is 94%. The F1 score of phishing websites is 92% while legitimate websites are 94%. Figure 5.6 shows the Logistic Regression's analysis result.

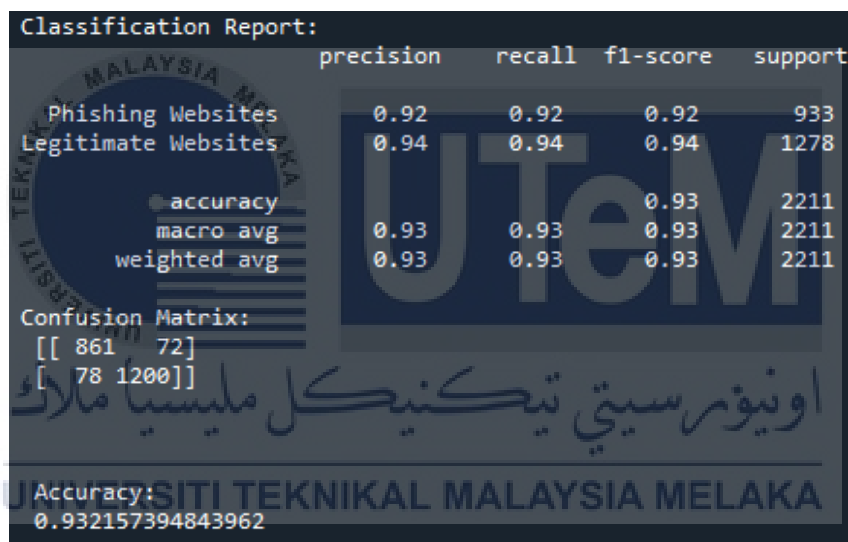


Figure 5.6 Logistic Regression's Analysis Result

### 5.2.1.2 Random Forest

The figure below shows the RF's classification result. In the testing dataset, the total phishing website is 986 while the legitimate website is 1225. Based on the confusion matrix, TP value is 940. FP value is 33. FN is 46 and TN is 1192. The accuracy of RF to predict phishing URL dataset is 96.43%. The precision of phishing websites is 97% and legitimate websites is 96%. The recall of phishing websites is

95% while legitimate websites is 97%. The F1 score of phishing websites is 96% while legitimate websites are 97%. Figure 5.7 shows the Random Forest's analysis result.

```

Classification Report:
              precision    recall  f1-score   support

   Phishing Websites         0.97      0.95      0.96       986
  Legitimate Websites         0.96      0.97      0.97      1225

   accuracy                   0.96       2211
  macro avg                   0.96      0.96      0.96       2211
 weighted avg                   0.96      0.96      0.96       2211

Confusion Matrix:
[[ 940   46]
 [   33 1192]]

Accuracy:
0.9642695612844867

```

**Figure 5.7 Random Forest's Analysis Result.**

### 5.2.1.3 CatBoost

The figure below shows the CB's classification result. In the testing dataset, the total phishing website is 984 while the legitimate website is 1227. Based on the confusion matrix, the TP value is 956. FP value is 28. FN is 19 and TN is 1208. The accuracy of CB to predict phishing URL dataset is 97.87%. The precision of phishing websites and legitimate websites is 98%. The recall of phishing websites is 97% while legitimate websites is 98%. The F1 score of phishing websites and legitimate websites both are 98%. Figure 5.8 shows the CB's analysis result.



```

Classification Report:
              precision    recall  f1-score   support

   Phishing Websites      0.98      0.97      0.98       984
  Legitimate Websites      0.98      0.98      0.98      1227

   accuracy                0.98                2211
  macro avg                0.98      0.98      0.98      2211
  weighted avg            0.98      0.98      0.98      2211

Confusion Matrix:
[[ 956  28]
 [  19 1208]]

Accuracy:
0.9787426503844414

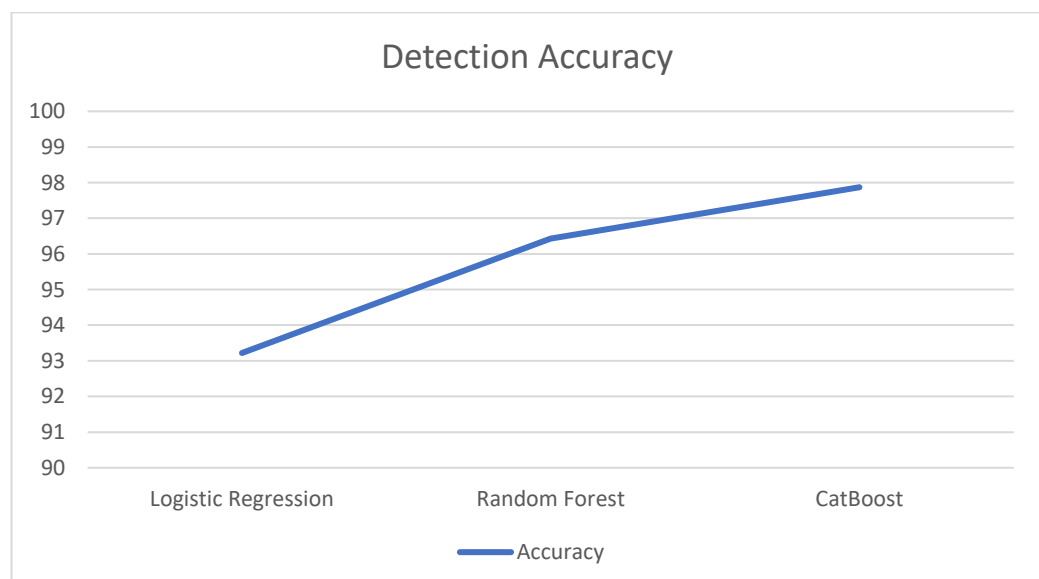
```

**Figure 5.8 CatBoost's Analysis Result**

## 5.2.2 Testing and Analysis Result in Graphs

### 5.2.2.1 Accuracy

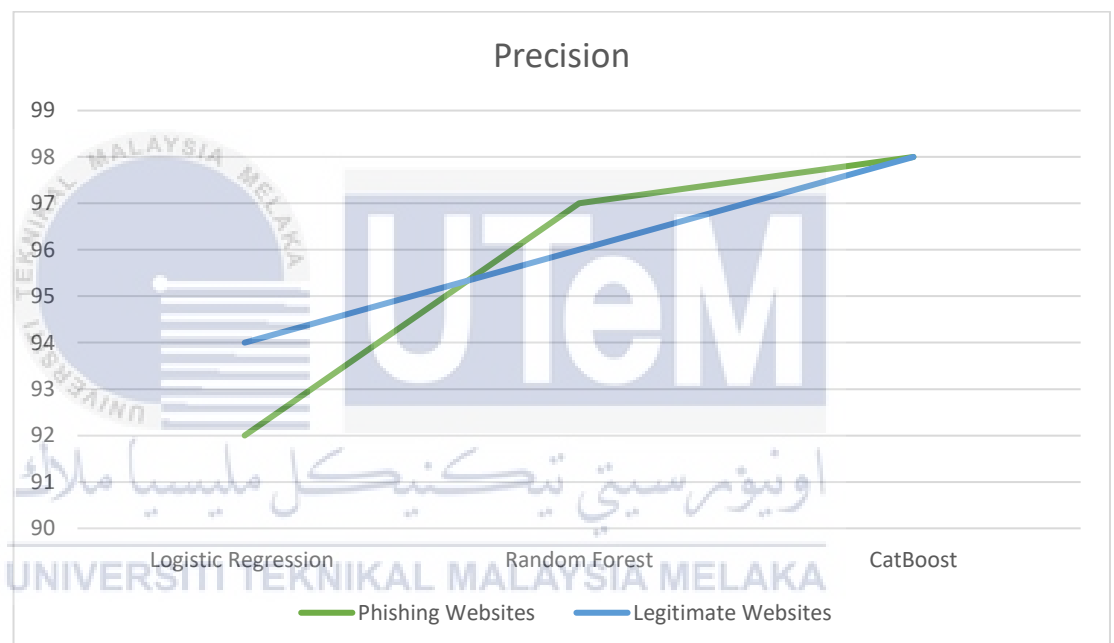
According to the graph, CB has the highest accuracy for detecting phishing URLs, with a result of 97.87%. The least accurate algorithm for detecting phishing URLs is LR, which has a 93.22% of accuracy rate while the accuracy rate of RF is between CB and LR which is 96.43%. Figure 5.9 shows the graph of detection accuracy.



**Figure 5.9 Graph of Detection Accuracy**

### 5.2.2.2 Precision

Based on the graph, CB has the highest precision rate which the precision of phishing websites and legitimate websites both are 98%. The second highest precision rate is RF. The precision of RF in phishing websites is 97% and legitimate websites is 96%. The lowest precision rate is LR, in which the precision of phishing websites is 92% and legitimate websites is 94%. Figure 5.10 shows the graph of precision.

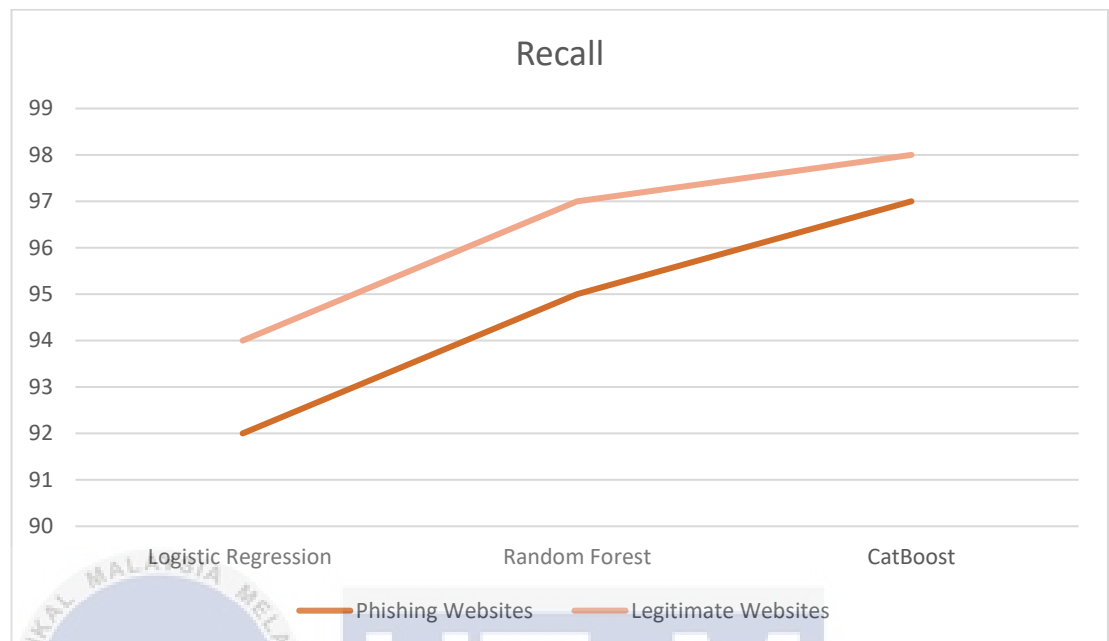


**Figure 5.10 Graph of Precision**

### 5.2.2.3 Recall

According to the graph, CB has the highest recall rate which the recall of phishing websites is 97% while legitimate websites is 98%. The second highest recall rate is RF. The recall of phishing websites is 95% while legitimate websites is 97%.

LR has the lowest recall rate which the recall of phishing websites is 92% while legitimate websites is 94%. Figure 5.11 shows the graph of recall.



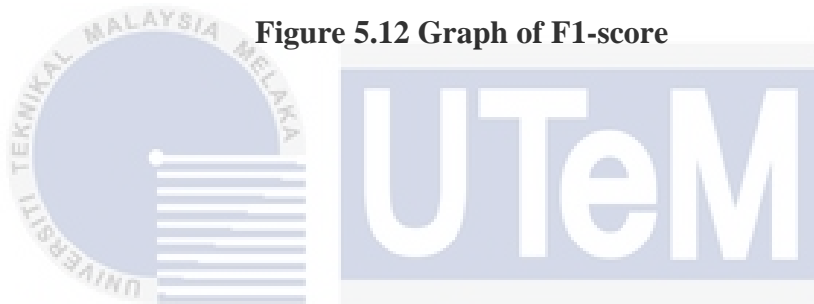
**Figure 5.11 Graph of Recall**

#### 5.2.2.4 F1-score

Based on the graph, CB has the highest F1-score which the F1-score of phishing websites and legitimate websites both are 98%. RF has the second-highest F1-score which the F1 score of phishing websites is 96% while legitimate websites are 97%. The lowest F1-score is LR. The F1 score of phishing websites is 92% while legitimate websites are 94%. Figure 5.12 shows the graph of the F1-score.



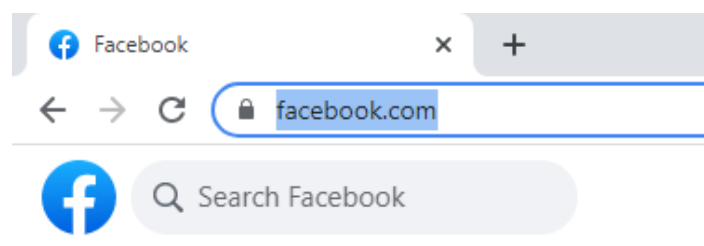
**Figure 5.12 Graph of F1-score**



### 5.2.3 Classification of URL

Based on the result of testing and analysis, CB has the best performance for detecting phishing URLs in terms of accuracy, precision, recall, and F1-score. So, CB is chosen to build an URL classifier. Test cases below show the result of a CB-based URL classifier to predict the phishing or legitimate URL. From the figures below, a CB-based URL classifier can predict the URL correctly and accurately.

Test Case 1: <https://www.facebook.com/>



**Figure 5.13 Test Case 1**

Result: Legitimate

```
In [1]: runfile('C:/Users/acerLim/anaconda3/Lib/urllib/FE_CatBoosts.py', wdir='C:/Users/acerLim/anaconda3/Lib/urllib')

Enter URL you want to check: https://www.facebook.com/
Processing, please wait a minute...

Result: Legitimate URL
```

Figure 5.14 Result of Test Case 1

Test Case 2: <https://ov.kredit24.com/gcNMpCtl>



Phish URL	Submitted	Valid?
<a href="http://jobster.co.nz/jobs?ts=ya&amp;q=dhl%20express%20job...">http://jobster.co.nz/jobs?ts=ya&amp;q=dhl%20express%20job...</a> added on Aug 3rd 2021 2:09 PM	by <a href="#">cleanmx</a>	Unknown
<a href="http://www.jobfinder.pt/jobs?ts=ya&amp;q=dhl%20express%20job...">http://www.jobfinder.pt/jobs?ts=ya&amp;q=dhl%20express%20job...</a> added on Aug 3rd 2021 2:08 PM	by <a href="#">cleanmx</a>	VALID PHISH
<a href="https://ov.kredit24.com/gcNMpCtl">https://ov.kredit24.com/gcNMpCtl</a> added on Aug 3rd 2021 2:08 PM	by <a href="#">cleanmx</a>	VALID PHISH

Figure 5.15 Test Case 2

Result: Phishing

```
In [11]: runfile('C:/Users/acerLim/anaconda3/Lib/urllib/FE_CatBoosts.py', wdir='C:/Users/acerLim/anaconda3/Lib/urllib')

Enter URL you want to check: https://ov.kredit24.com/gcNMpCtl
Processing, please wait a minute...

Result: Phishing URL
```

Figure 5.16 Result of Test Case 2

Test Case 3: <http://shawntownsend.art/belees/index.php>

### Archive

Phish URL	Submitted	Valid?
<a href="http://shawntownsend.art/belees/index.php">http://shawntownsend.art/belees/index.php</a> <small>added on Aug 3rd 2021 2:06 PM</small>	by <a href="#">cleanmx</a>	<b>VALID PHISH</b>

**Figure 5.17 Test Case 3**

Result: Phishing

```
In [14]: runfile('C:/Users/acerLim/anaconda3/Lib/urllib/FE_CatBoosts.py', wdir='C:/Users/acerLim/anaconda3/Lib/urllib')

Enter URL you want to check: http://shawntownsend.art/belees/index.php
Processing, please wait a minute...

Result: Phishing URL
```

**Figure 5.18 Result of Test Case 3**

Test Case 4: <https://encryptedmessage.weebly.com/>

### Archive

Phish URL	Submitted	Valid?
<a href="https://encryptedmessage.weebly.com/">https://encryptedmessage.weebly.com/</a> <small>added on Aug 3rd 2021 1:59 PM</small>	by <a href="#">rodiqvabuse</a>	<b>VALID PHISH</b>

**Figure 5.19 Test Case 4**

Result: Phishing

```
In [23]: runfile('C:/Users/acerLim/anaconda3/Lib/urllib/FE_CatBoosts.py', wdir='C:/Users/acerLim/anaconda3/Lib/urllib')

Enter URL you want to check: https://encryptedmessage.weebly.com/
Processing, please wait a minute...

Result: Phishing URL
```

**Figure 5.20 Result of Test Case 4**

Test Case 5: <https://ulearn.utem.edu.my/>



**Figure 5.19 Test Case 5**

Result: Legitimate

```
In [2]: runfile('C:/Users/acerLim/anaconda3/Lib/urllib/FE_CatBoosts.py', wdir='C:/Users/acerLim/anaconda3/Lib/urllib')
Enter URL you want to check: https://ulearn.utem.edu.my/
Processing, please wait a minute...
Error trying to connect to socket: closing socket
Error trying to connect to socket: closing socket
Error trying to connect to socket: closing socket
Result: Legitimate URL
```

**Figure 5.20 Result of Test Case 5**

Test Case 6: <http://thesdfsdfsdfsdf.com/mazon/e7fe1/>

<a href="http://pancakeswap.ch">http://pancakeswap.ch</a> added on Aug 12th 2021 4:20 AM	by <a href="#">r3g3rsec</a>	Unknown
<a href="https://nttdocomo.jp/winnerchoose.com/">https://nttdocomo.jp/winnerchoose.com/</a> added on Aug 12th 2021 4:15 AM	by <a href="#">wagawaga</a>	Unknown
<a href="https://www.cutt.ly/OQWKPBR/">https://www.cutt.ly/OQWKPBR/</a> added on Aug 12th 2021 4:00 AM	by <a href="#">cleanmx</a>	Unknown
<a href="http://thesdfsdfsdfsdf.com/mazon/e7fe1/">http://thesdfsdfsdfsdf.com/mazon/e7fe1/</a> added on Aug 12th 2021 4:00 AM	by <a href="#">cleanmx</a>	<b>VALID PHISH</b>

### Figure 5.21 Test Case 6

Result: Phishing

```
Enter URL you want to check: http://thesdfsfsdfsdfs.com/mazon/e7fe1/
Processing, please wait a minute...

Result: Phishing URL
```

### Figure 5.22 Result of Test Case 6

Test Case 7: <https://clive-smallman.questai.app/docuSign/>

<a href="https://chfac0ahcwagomkd-dot-gap1-322613-808-klef.ew.r.appspot.com/...">https://chfac0ahcwagomkd-dot-gap1-322613-808-klef.ew.r.appspot.com/...</a> added on Aug 12th 2021 3:19 AM	by <a href="#">kkalmus</a>	Unknown
<a href="https://www.japanetbank.co/">https://www.japanetbank.co/</a> added on Aug 12th 2021 3:06 AM	by <a href="#">wagawaga</a>	Unknown
<a href="https://clive-smallman.questai.app/docuSign/">https://clive-smallman.questai.app/docuSign/</a> added on Aug 12th 2021 3:04 AM	by <a href="#">cleanmx</a>	<b>VALID PHISH</b>

### Figure 5.23 Test Case 7

Result: Phishing

```
Enter URL you want to check: https://clive-smallman.questai.app/docuSign/
Processing, please wait a minute...

Result: Phishing URL
```

### Figure 5.24 Result of Test Case 7



Test Case 8: <https://www.wikipedia.org/>

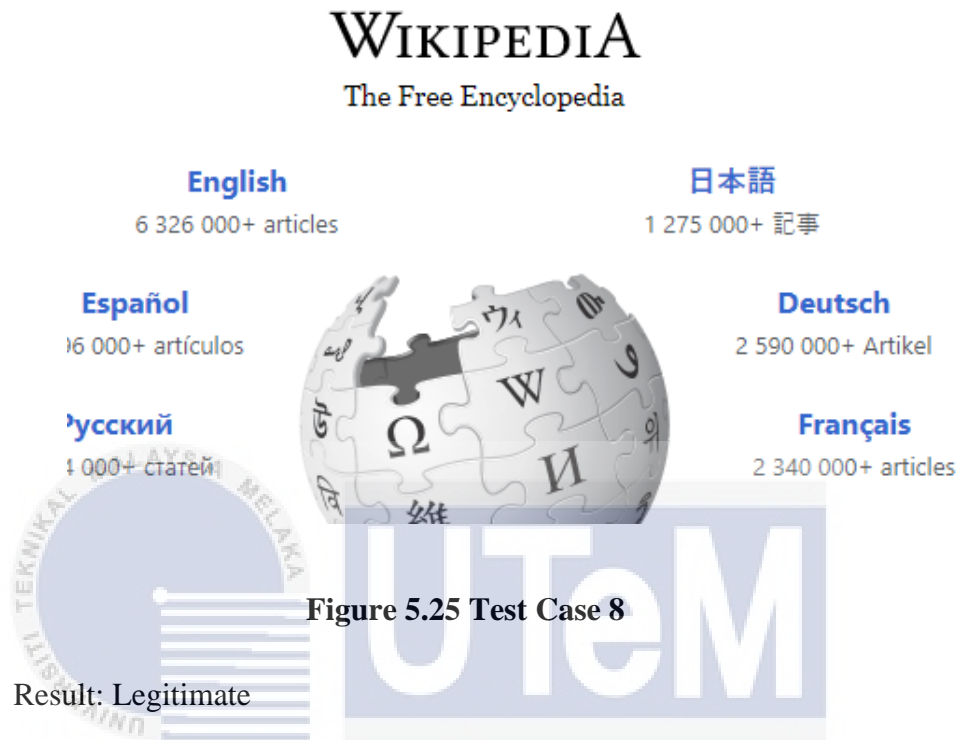


Figure 5.25 Test Case 8

Result: Legitimate

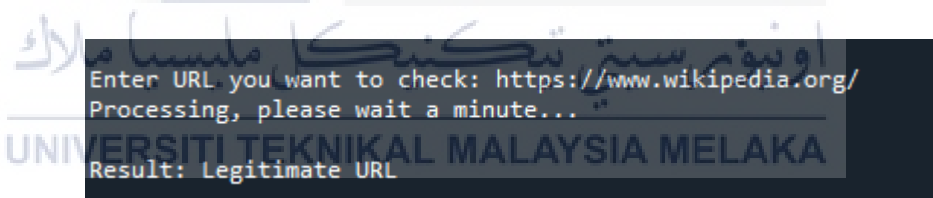


Figure 5.26 Result of Test Case 8

Test Case 9: <https://smbc.card-tr.club/m>

<a href="https://smbc.card-tr.club/m">https://smbc.card-tr.club/m</a> added on Aug 11th 2021 10:51 PM	by <a href="#">wagawaga</a>	<b>VALID PHISH</b>
<a href="https://registrodatosmultire1bn.xyz/">https://registrodatosmultire1bn.xyz/</a> added on Aug 11th 2021 10:49 PM	by <a href="#">PhishVerifier</a>	Unknown

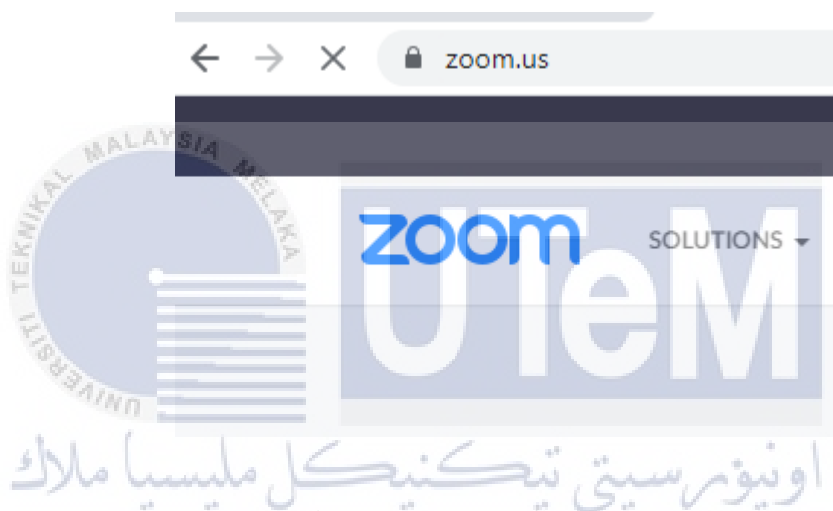
Figure 5.27 Test Case 9

Result: Phishing

```
Enter URL you want to check: https://co-jp.rakeunior.com/  
Processing, please wait a minute...  
  
Result: Phishing URL
```

**Figure 5.28 Result of Test Case 9**

Test Case 10: https://zoom.us/



**Figure 5.29 Test Case 10**

UNIVERSITI TEKNIKAL MALAYSIA MELAKA  
Result: Legitimate

```
Enter URL you want to check: https://zoom.us/  
Processing, please wait a minute...  
  
Result: Legitimate URL
```

**Figure 5.30 Result of Test Case 10**

### 5.3 Testing Result Summary

Table 5.1 summarize the overall result of phishing URL detection based on the UCI repository dataset. From 11055 URL dataset, 2211 URL is tested in order to predict the result. CB has achieved the highest accuracy score which is 97.87%, compared to RF and LR. CB also obtained the highest precision, recall, and F1-score in phishing and legitimate URL, with an average of 98%, as compared to RF and LR. CB classifier outperformed all other performance indicators such as precision, recall, and f1-score. As a result, a CB classifier is used as the final URL classification model because it has the best performance in terms of accuracy, precision, recall, and F1-score.

**Table 5.1: Evaluation Table**

Classifiers	Evaluation Parameter						
	Accuracy	Precision		Recall		F1-score	
		Phishing URL	Legitimate URL	Phishing URL	Legitimate URL	Phishing URL	Legitimate URL
<b>Logistic Regression</b>	93.22%	92%	94%	92%	94%	92%	94%
<b>Random Forest</b>	96.43%	97%	96%	95%	97%	96%	97%
<b>CatBoost</b>	97.87%	98%	98%	97%	98%	98%	98%

The false positive in CB is 28, RF is 46 while LR is 72. The false-negative in CB is 19, RF is 33 and LR is 78. CB has the lowest false positive and false negative so the accuracy, precision, recall, and F1-score of CB is highest with an average of 98%. LR has the highest false positive and false negative so the accuracy, precision, recall, and F1-score of LR is lowest with an average of 93%. The false-positive and false-negative of RF are between CB and LR, the accuracy, precision, recall, and F1-

score with an average of 96%. From the result, the CB model outperforms the LR and RF. This may be due to the LR model overfitting on the training set, exaggerating the accuracy of predictions on the training set, and so preventing the model from accurately predicting results on the test set. Because this approach is sensitive to outliers, the inclusion of data that differ from the acceptable range in the dataset may result in inaccurate outcomes. When classes are distinct, the estimation method in LR becomes incorrect because of a logistic function that forces the derivatives to be endless and therefore becoming computationally unstable (Grover, 2020). RF does not have a problem when classes are distinct. Instead, when adequate tree pruning methods are utilized, it assists in the reduction of computations. So, the performance of RF is better than LR. The performance of RF is lower than CB because RF may lack readability due to the ensemble of decision trees and fails to evaluate the importance of each feature. Data with categorical features with varying amounts of features can be a major issue because the RF method prefers those with more values, posing a risk of incorrect prediction (Holy Python, 2021). CB eliminates the need for intensive hyper-parameter adjustment because the default parameter in CB produces a good result. It also decreases the risk of overfitting, resulting in more flexible and accurate models (Mwiti, 2020). When determining the tree structure, CB uses a strategy to calculate leaf values, that greatly reduces overfitting. CB can evaluate then select the important feature. One of the methods used in CB is Prediction Values Change. It shows the prediction changes on average when the feature value changes. When a change in the feature value causes a large change in the anticipated value, the feature becomes more important. For non-ranking metrics, this is the default technique of calculating feature importance. CB provides a novel technique for analyzing category features. Some of the most prevalent approaches for encoding categorical data, such as one-hot encoding, result in an infeasibly large number of additional features in the case of features with high cardinality (Nahon, 2020). As a result, CB's accuracy would be superior for data with categorical attributes compared to RF.

In conclusion, a CB-based URL classifier can produce an accurate result when predicting the phishing or legitimate URL as shown in table 5.2.

**Table 5.2: CatBoost's Classification Result**

URL	Actual Result	CatBoost's Prediction
<a href="https://www.facebook.com/">https://www.facebook.com/</a>	Legitimate	Legitimate
<a href="https://ov.kredit24.com/gcNMpCtl">https://ov.kredit24.com/gcNMpCtl</a>	Phishing	Phishing
<a href="http://shawntownsend.art/belees/index.php">http://shawntownsend.art/belees/index.php</a>	Phishing	Phishing
<a href="https://encryptedmessage.weebly.com/">https://encryptedmessage.weebly.com/</a>	Phishing	Phishing
<a href="https://ulearn.utem.edu.my/">https://ulearn.utem.edu.my/</a>	Legitimate	Legitimate
<a href="http://thesdfsdfsdfsdf.com/mazon/e7fe1/">http://thesdfsdfsdfsdf.com/mazon/e7fe1/</a>	Phishing	Phishing
<a href="https://clive-smallman.questai.app/docuSign/">https://clive-smallman.questai.app/docuSign/</a>	Phishing	Phishing
<a href="https://www.wikipedia.org/">https://www.wikipedia.org/</a>	Legitimate	Legitimate
<a href="https://smbc.card-tr.club/m">https://smbc.card-tr.club/m</a>	Phishing	Phishing
<a href="https://zoom.us/">https://zoom.us/</a>	Legitimate	Legitimate

#### 5.4 Conclusion

The datasets are evaluated through three different classification algorithms which are LR, RF, and CB. In this chapter, figures, graphs, and tables represent the test and analysis results. This chapter also includes the results of the URL classifier to

classify URLs. Project summarization, project contribution, project limitation and future works of the research will discuss in the next chapter.



## CHAPTER 6: RESEARCH CONCLUSION

### 6.1 Introduction

The dataset has been evaluated, the results of class precision, recall, and accuracy of phishing and legitimate URL in three different classification models are presented and compared and the results of CB-based URL classification is shown in the previous chapter. This chapter summarizes, completes, and provides the research's conclusion. This chapter also examines the project's summary, contribution, highlighting limitations, and include future works.

### 6.2 Research Summarization

This research aims to investigate suitable machine learning algorithms to detect phishing URLs. LR, RF, and CB algorithms are used to detect phishing URLs in this research. Then the performance of machine learning algorithms such as LR, RF, and CB are compared. The dataset for this study was the UCI machine learning repository, which contains 11055 URLs. The datasets are organized into two categories: training datasets and testing datasets, with an 80:20 split. The training datasets used to train the URL classifier models, and testing datasets predict the results. CB has the best performance of phishing URL detection in terms of accuracy, precision, recall, and f1-score with an average of 98%. LR has the lowest accuracy, precision, recall, and f1-

score in phishing URL detection with an average of 93%. The performance of RF is between CB and LR and the accuracy, precision, recall, and f1-score of RF with an average of 96%. The outcomes of each classifier model's test and accuracy are then compared, analyzed, and displayed in figures and graphs. Lastly, an URL classifier is implemented based on the most accurate algorithm that can detect between phishing and legitimate URLs. The most accurate algorithms, CB is used to implement an URL classifier to classify legitimate and phishing URLs. The results of CB based classifier is shown in the table. The research objectives are met and the project is successfully concluded. Spyder is used to detect phishing URLs using machine learning techniques based on this research.

### 6.3 Research Contribution

In the research domain, a comparison among CatBoost, Random Forest, and Logistic Regression algorithms is conducted in this project to determine which algorithm has the best performance in terms of accuracy, precision, recall, and f1-score. Furthermore, this research will assist web users by allowing them to detect and keep alert to phishing websites in real-time, resulting in a more secure network experience. In other ways, this research can be applied in the security domain where cybersecurity authorities can use it to prevent users from visiting phishing websites and to develop powerful security mechanisms that can detect and prevent phishing domains from reaching users.

### 6.4 Research Limitation

There are some limitations to this research. The first limitation of this research is time-consuming. Scikit-learn machine learning models take a long time to learn the training dataset, and it also takes a long time for the classification of URLs. Second, it contains client errors during the classification of URLs. The server receives the request but refuses to approve it from Spyder. Some websites will check the User-Agent (browser) to avoid abnormal visits.



## 6.5 Future Works

Apache Spark framework can be used to improve the Sickit-learn library called Sk-dist. Sk-dist has overcome the limitation of Sickit-learn library such as being time-consuming and can speed up the model training (Djediden et al., 2019). In addition, develop an online URL scanning website that allows web users to detect phishing websites in real-time.

## 6.6 Conclusion

This research evaluated the performance of machine learning algorithms in detecting phishing URLs. As a result, a new phishing URL detection model CB-based URL classifier is implemented in this project. The CB algorithm has the highest accuracy, followed by the RF and the LR. CB algorithms outperform the RF and LR in terms of accuracy, precision, and recall. The effectiveness of the CB algorithm has been demonstrated through its higher performance over competing algorithms. The analysis is presented. The research can go further with future works.

اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

## REFERENCES

- Ali, W. (2017). Phishing Website Detection based on Supervised Machine Learning with Wrapper Features Selection. *International Journal of Advanced Computer Science and Applications*, 8(9). <https://doi.org/10.14569/ijacsa.2017.080910>
- Abdelhamid, N., Thabtah, F. and Abdel-jaber, H. (2017). Phishing detection: A recent intelligent machine learning comparison based on models content and features. 2017 IEEE International Conference on Intelligence and Security Informatics (ISI). <https://doi.org/10.1109/isi.2017.8004877>
- A. K. Shrivastava and Ramkishun Suryawanshi. (2017) Decision Tree Classifier for Classification of Phishing Website with Info Gain Feature Selection, *International Journal for Research in Applied Science & Engineering*, ISSN: 2321-9653
- Buber, E., Demir, O. and Sahingoz, O. K. (2017). Feature selections for the machine learning based detection of phishing websites. 2017 International Artificial Intelligence and Data Processing Symposium (IDAP). <https://doi.org/10.1109/idap.2017.8090317>
- Basit, A., Zafar, M., Liu, X., Javed, A. R., Jalil, Z., and Kifayat, K. (2020). A comprehensive survey of AI-enabled phishing attacks detection techniques. *Telecommunication Systems*, 76(1), 139–154. <https://doi.org/10.1007/s11235-020-00733-2>
- Babagoli, M., Aghababa, M. P. and Solouk, V. (2018). Heuristic nonlinear regression strategy for detecting phishing websites. *Soft Computing*, 23(12), 4315–4327. <https://doi.org/10.1007/s00500-018-3084-2>

- Chen, Y. and Han, X. (2021). Catboost for fraud detection in financial transactions. 2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE). <https://doi.org/10.1109/iccece51280.2021.9342475>
- Djediden , M. S. O., Mekkakia Maaza, and Reguieg, H. (2019). A distributed intrusion detection system based on apache spark and scikit-learn library. *Journal of Applied and Physical Sciences*, 5(1). <https://doi.org/10.20474/japs-5.1.4>
- D, N. K., Hemanth, N. S. R., S, P. and V, N. K. (2020). Detection of Phishing Websites using an Efficient Machine Learning Framework. *International Journal of Engineering Research And*, V9(05). <https://doi.org/10.17577/ijertv9is050888>
- Grover, K. (2020, June 23). Advantages and Disadvantages of Logistic Regression. OpenGenus IQ: Computing Expertise & Legacy. <https://iq.opengenus.org/advantages-and-disadvantages-of-logistic-regression/>
- Hodžić, A., Kevric, J. and Karadag, A. (1970, January 1). [PDF] COMPARISON OF MACHINE LEARNING TECHNIQUESIN PHISHING WEBSITE CLASSIFICATION: Semantic Scholar. undefined. <https://www.semanticscholar.org/paper/COMPARISON-OF-MACHINE-LEARNING-TECHNIQUESIN-WEBSITE-Hod%C5%BEi%C4%87-Kevric/c2264f962fdf7f207340d2d69811092d4f1f96c1>.
- Hancock, J. T. and Khoshgoftaar, T. M. (2020). CatBoost for big data: an interdisciplinary review. *Journal of Big Data*, 7(1). <https://doi.org/10.1186/s40537-020-00369-8>
- Hancock, J., & Khoshgoftaar, T. M. (2020). Performance of catboost and xgboost in medicare fraud detection. 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA). <https://doi.org/10.1109/icmla51294.2020.00095>

- Holy Python. (2021b, June 29). Random Forest Pros & Cons. HolyPython.Com.  
<https://holypython.com/rf/random-forest-pros-cons/>
- Jalal, K., and Naaz, S. (2019). Detection of phishing websites using machine learning approach. SSRN Electronic Journal. <https://doi.org/10.2139/ssrn.3357736>
- Jeeva, S. C. and Rajsingh, E. B. (2016). Intelligent phishing url detection using association rule mining. Human-Centric Computing and Information Sciences, 6(1). <https://doi.org/10.1186/s13673-016-0064-3>
- Koehrsen, W. (2020, August 18). Random Forest Simple Explanation - Will Koehrsen. Medium. <https://williamkoehrsen.medium.com/random-forest-simple-explanation-377895a60d2d>
- Kulkarni, A., and L., L. (2019). Phishing Websites Detection using Machine Learning. International Journal of Advanced Computer Science and Applications, 10(7). <https://doi.org/10.14569/ijacsa.2019.0100702>
- Li, Y., Mai, Y., Lin, Z., and Liang, S. (2020). Online transaction detection method using catboost model. 2020 International Conference on Communications, Information System and Computer Engineering (CISCE). <https://doi.org/10.1109/cisce50729.2020.00053>
- Mahajan, R. and Siddavatam, I. (2018). Phishing Website Detection using Machine Learning Algorithms. International Journal of Computer Applications, 181(23), 45–47. <https://doi.org/10.5120/ijca2018918026>
- Mao, J., Bian, J., Tian, W., Zhu, S., Wei, T., Li, A. and Liang, Z. (2019). Phishing page detection via learning classifiers from page layout feature. EURASIP Journal on Wireless Communications and Networking, 2019(1). <https://doi.org/10.1186/s13638-019-1361-0>

- Masurkar, S., and Dalal, V. (2020). Enhanced model for detection of phishing url using machine learning. *ETHICS AND INFORMATION TECHNOLOGY*.  
<https://doi.org/10.26480/etit.02.2020.158.163>
- Mohammad, R. M., Thabtah, F., and McCluskey, L. (2015). *UCI Machine Learning Repository: Phishing Websites Data Set*. UCI Machine Learning Repository.  
<https://archive.ics.uci.edu/ml/datasets/phishing+websites>
- Mwiti, D. (2020). Fast Gradient Boosting with CatBoost. *KDnuggets*.  
<https://www.kdnuggets.com/2020/10/fast-gradient-boosting-catboost.html>
- Nahon, A. (2020, February 10). XGBoost, LightGBM or CatBoost — which boosting algorithm should I use? *Medium*. <https://medium.com/riskified-technology/xgboost-lightgbm-or-catboost-which-boosting-algorithm-should-i-use-e7fda7bb36bc>
- Niakanlahiji, A., Chu, B.-T. and Al-Shaer, E. (2018). PhishMon: A Machine Learning Framework for Detecting Phishing Webpages. 2018 IEEE International Conference on Intelligence and Security Informatics (ISI).  
<https://doi.org/10.1109/isi.2018.8587410>
- Pant, A. (2019, January 22). Introduction to Logistic Regression - Towards Data Science. *Medium*. <https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148>
- Patil, V., Thakkar, P., Shah, C., Bhat, T. and Godse, S. P. (2018). Detection and Prevention of Phishing Websites Using Machine Learning Approach. 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA). <https://doi.org/10.1109/iccubea.2018.8697412>

- Peretz, T. (2021, July 12). Mastering The New Generation of Gradient Boosting - Towards Data Science. Medium. <https://towardsdatascience.com/https-medium-com-talperetz24-mastering-the-new-generation-of-gradient-boosting-db04062a7ea2>
- Ram. B. (2014). LEARNING TO DETECT PHISHING URLS. *International Journal of Research in Engineering and Technology*, 03(06), 11–24. <https://doi.org/10.15623/ijret.2014.0306003>
- Rao, R. S., Pais, A. R. and Anand, P. (2020). A heuristic technique to detect phishing websites using TWSVM classifier. *Neural Computing and Applications*, 33(11), 5733–5752. <https://doi.org/10.1007/s00521-020-05354-z>
- Rácz, A., Bajusz, D. and Héberger, K. (2021). Effect of Dataset Size and Train/Test Split Ratios in QSAR/QSPR Multiclass Classification. *Molecules*, 26(4), 1111. <https://doi.org/10.3390/molecules26041111>
- Ranganathan, P., Pramesh, C. S., & Aggarwal, R. (2017). Common pitfalls in statistical analysis: Logistic regression. *Perspectives in clinical research*. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5543767/>.
- Sahoo, D, Chenghao Liu and Steven C.H. Hoi. (2019). Malicious URL Detection using Machine Learning: A Survey. *arXiv:1701.07179v3*
- Sahingoz, O. K., Buber, E., Demir, O. and Diri, B. (2019). Machine learning based phishing detection from URLs. *Expert Systems with Applications*, 117, 345–357. <https://doi.org/10.1016/j.eswa.2018.09.029>
- Sampat, H., Saharkar, M., Pandey, A. and Lopes, H. (2018, March). Detection of Phishing Website Using Machine Learning. *International Research Journal of Engineering and Technology*. <https://www.irjet.net/archives/V5/i3/IRJET-V5I3580.pdf>.

- Silva, C. M., Feitosa, E. L. and Garcia, V. C. (2020). Heuristic-based strategy for Phishing prediction: A survey of URL-based approach. *Computers & Security*, 88, 101613. <https://doi.org/10.1016/j.cose.2019.101613>
- Scornet, E., Biau, G. and Vert, J.-P. (2015). Consistency of random forests. *The Annals of Statistics*, 43(4). <https://doi.org/10.1214/15-aos1321>
- Sonmez, Y., Tuncer, T., Gokal, H. and Avci, E. (2018). Phishing web sites features classification based on extreme learning machine. 2018 6th International Symposium on Digital Forensic and Security (ISDFS). <https://doi.org/10.1109/isdfs.2018.8355342>
- Suman, B., Chetan, K.C. and Praveen, K.P. (2017). Detecting Phishing Websites, a Heuristic Approach. *International Journal of Latest Engineering Research and Applications (IJLERA)* ISSN: 2455-7137. Volume – 02, Issue – 03, PP – 120-129
- Tumuluru, P., Ramani, B. L., Samineni, V., Konatham, D. S. S. and Jonnalagadda, R. M. (2019). Extreme Learning Model Based Phishing Classifier. *International Journal of Recent Technology and Engineering*, 8(4), 9606–9612. <https://doi.org/10.35940/ijrte.d9984.118419>
- Tyagi, N. (2020, September 30). Understanding the Gini Index and Information Gain in Decision Trees. *Medium*. <https://medium.com/analytics-steps/understanding-the-gini-index-and-information-gain-in-decision-trees-ab4720518ba8>
- Yi, P., Guan, Y., Zou, F., Yao, Y., Wang, W., and Zhu, T. (2018). Web Phishing Detection Using a Deep Learning Framework. *Wireless Communications and Mobile Computing*, 2018, 1–9. <https://doi.org/10.1155/2018/4678746>