

DETECTION OF PHISHING WEBSITES USING MACHINE LEARNING

BY

ADEDIRAN ADENIYI GOODNESS

(SSE/017/18295)

A PROJECT SUBMITTED

TO

COMPUTER SCIENCE PROGRAMME,

COLLEGE OF COMPUTING AND COMMUNICATION STUDIES,

BOWEN UNIVERSITY, IWO, OSUN STATE, NIGERIA.

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR

THE AWARD DEGREE OF BACHELOR OF SCIENCE (B.Sc.) IN

COMPUTER SCIENCE

AUGUST, 2021

CERTIFICATION

This is to certify that this project, “Detection of Phishing Websites Using Machine Learning” was carried out by Adediran Adeniyi Goodness (Matriculation Number: SSE/017/18295) of the Computer Science programme under my supervision

.....
Dr. A.O. Akinwunmi
(Supervisor)

.....
Date

.....
Dr. A.O. Akinwunmi
(Programme Coordinator)

.....
Date

DEDICATION

I dedicate this project work to Almighty God, for His unfailing love, grace and for being the Master Planner of my life and to my beloved parents Rev. Dr. & Mrs. Sina Adediran for their love and support, May the good Lord continue to bless and prosper your ways in Jesus name Amen.

ACKNOWLEDGEMENTS

I would like to acknowledge the sustaining power of God Almighty for giving me Grace, Strength, and Wisdom. I would like to acknowledge the effort of my supervisor Dr. A.O. Akinwunmi for taking the time to guide me through this work.

With a deep sense of appreciation, respect and gratitude, I would very much like to acknowledge the efforts of my lecturers in Computer Science and Information Technology programme of Bowen University: Dr. M.O. Oyelami, Dr. A.O. Akinwunmi, Dr. O.G. Lala, Dr. R.F. Famutimi, Dr. Mrs. O.N. Emuoyibofarhe, Mr. B.P. Ayanniyi, Mr. C. Agbonkhese, Dr. D.O. Lanloye, Dr. E.K. Olatunji, Dr. T.O. Olorunfemi, Dr. Segun Adebayo, Dr. A.O. Abiodun, Dr. A.O. Ibitoye, Dr. H.O. Aworinde, Dr. O.M. Awoniran, , Mr. A. Adeyemo and Mrs. Olaniran for giving me a solid footing in the field of Computer Science and Information Technology; I say God bless you richly.

I appreciate my parents Rev. Dr. S.A. & Mrs. O.G ADEDIRAN, and my siblings for their support. I cannot forget the contributions of my course mates to the successful completion of this work, I am forever grateful.

TABLE OF CONTENT

CERTIFICATION	ii
DEDICATION	iii
ACKNOWLEDGEMENTS	iv
TABLE OF CONTENT	v
LIST OF TABLES	ix
LIST OF FIGURES	x
GLOSSARY OF TERMS	xii
ABSTRACT	xiii
CHAPTER ONE	1
INTRODUCTION	1
1.1 Background Information	1
1.2 Statement of Problem	3
1.3 Aim of Study	3
1.4 Objectives of the Study	3
1.5 Methodology	4
1.6 Significance of the Project	5
1.7 Scope of the Study	5
1.8 Project Report Arrangement	6
CHAPTER TWO	7
LITERATURE REVIEW	7
2.1 Overview of the Study	7

2.2 Theoretical Review	7
2.2.1 Data imputation	11
2.2.2 Feature selection	15
2.2.3 Feature extraction	18
2.2.3.1 address bar based features	19
2.2.3.2 abnormal based features	25
2.2.3.3 html and javascript-based features	28
2.2.3.4 domain-based features	30
2.2.4 Algorithm and model evaluation (Performance Metrics)	36
2.2.4.1 implementation and result	39
2.3 Conceptual Review	44
2.3.1 phishing mechanism	44
2.3.2 taxonomy of phishing attack	46
2.3.3 antiphishing technique	48
2.3.4 visual similarity-based phishing detection and filtering approaches	50
2.4 Empirical Review	52
CHAPTER THREE	62
SYSTEM ANALYSIS AND DESIGN	62
3.1 Overview of System Analysis	62
3.2 Analysis of Existing System	63
3.3 Proposed System	63

3.3.1 Benefits of the new system	63
3.4 Model Development	64
3.5 System Modelling	68
3.5.1 System architecture	69
3.5.2 Use a case diagram of the system	69
3.5.3 Flowchart of the system	72
CHAPTER FOUR	75
SYSTEM IMPLEMENTATION AND RESULTS	75
4.1 Installation Requirements	75
4.1.1 Hardware requirements	75
4.1.2 Software requirements	76
4.2 Model Development	76
4.2.1 Data collection	76
4.2.2 Feature extraction on the datasets	79
4.2.2 Data Analysis & Visualization	84
4.2.3 Data pre-processing	89
4.2.4 Phishing detection model	92
4.3 General Working of The System	94
4.3.1 The home page	94
4.3.2 The about page	94
4.3.3 The use case page	94

4.3.4 Resource page	95
4.3.5 Web application Source Code	95
4.3.6 API (Application Programming Interface)	102
CHAPTER FIVE	108
SUMMARY, CONCLUSION, AND RECOMMENDATIONS	108
5.1 Summary	108
5.2 Contribution to Knowledge	109
5.3 Conclusion	109
5.4 Recommendation	109
REFERENCE	110
APPENDIX A: Web application Source code	116
APPENDIX B: Source code for API integration to web app	125

LIST OF TABLES

Table 2.1: Common ports to be checked	24
Table 2.2: Classifier's performance	41
Table 2.3: Outline of related algorithms used to detect phishing website	59

LIST OF FIGURES

Figure 2.1 Worldwide financial losses (in billion) due to phishing attacks	9
Figure 2.2 growth of phishing attacks from 2005 to 2015	10
Figure 2.3 Mean imputation	13
Figure 2.4 Zero Imputation	14
Figure 2.5 Feature selection process	16
Figure 2.6 Feature Selection Models	17
Figure 2.7 Detection accuracy comparison	40
Figure 2.8 Phishing Mechanism	45
Figure 2.9 Types of Phishing attack	47
Figure 2.10 Life cycle of phishing attack	49
Figure 2.11 (a) Legitimate PayPal webpage and (b) phishing webpage of PayPal	51
Figure 3.1 Machine Learning development process	66
Figure 3.2: Architectural Design of the Proposed System	70
Figure 3.3 Use Case diagram for Proposed System	71
Figure 3.4 Flowchart of the proposed System	73
Figure 3.5 Flowchart of the web interface	74
Figure 4.1 Dataset of Phishing URLs	77
Figure 4.2 Dataset of Legitimate URLs	78
Figure 4.3: Code for Address bar based feature extraction	80
Figure 4.4: Code for domain-based features extraction	81
Figure 4.5: Code for Html & java-script based features extraction	82
Figure 4.6: Code computation for all the feature extraction used dataset.	83
Figure 4.7: Distribution plot of dataset base on the features selected	85
Figure 4.8: Correlation heat map of the dataset	86

Figure 4.9: Feature importance for Decision Tree classifier	87
Figure 4.10: Feature importance for Random forest classifier	88
Figure 4.11: Summary of the dataset	90
Figure 4.12: Number of missing values in the dataset	91
Figure 4.13: Accuracy performance of models	93
Figure 4.14 (a): The Home page	96
Figure 4.14 (b): The home page footer	97
Figure 4.15: The About page	98
Figure 4.16 (a): The Use-case page	99
Figure 4.16 (b): The Use-case page	99
Figure 4.17: The Resource page	100
Figure 4.18: Code for the web application	101
Figure 4.19: python code to send a request on JSON	103
Figure 4.20: Code for API URL	104
Figure 4.21: Executing web-app on Django local server	105
Figure 4.22: Testing the API link on postman	106
Figure 4.23: JavaScript code for linking API	107

GLOSSARY OF TERMS

URL	Uniform Resource Locators
URL	Uniform Resource Locators
HTML	Hyper Text Markup Language
EDA	Exploratory Data Analysis
SVM	Support Vector Machine
CSV	Comma Separated Values
JSON	JavaScript Object Notation
API	Application Programming Interface
IDE	Integrated Development Environment

ABSTRACT

Phishing attacks are a rapidly expanding threat in the cyber world, costing internet users billions of dollars each year. It is a criminal crime that involves the use of a variety of social engineering tactics to obtain sensitive information from users. Phishing techniques can be detected using a variety of types of communication, including email, instant chats, pop-up messages, and web pages. This study develops and creates a model that can predict whether a URL link is legitimate or phishing.

The data set used for the classification was sourced from an open source service called ‘Phish Tank’ which contain phishing URLs in multiple formats such as CSV, JSON, etc. and also from the University of New Brunswick dataset bank which has a collection of benign, spam, phishing, malware & defacement URLs. Over six (6) machine learning models and deep neural network algorithms all together are used to detect phishing URLs.

This study aims to develop a web application software that detects phishing URLs from the collection of over 5,000 URLs which are randomly picked respectively and are fragmented into 80,000 training samples & 20,000 testing samples, which are equally divided between phishing and legitimate URLs. The URL dataset is trained and tested base on some feature selection such as address bar-based features, domain-based features, and HTML & JavaScript-based features to identify legitimate and phishing URLs.

In conclusion, the study provided a model for URL classification into phishing and legitimate URLs. This would be very valuable in assisting individuals and companies in identifying phishing attacks by authenticating any link supplied to them to prove its validity.

CHAPTER ONE

INTRODUCTION

1.1 Background Information

The Internet has become an important part of our lives for gathering and disseminating information, particularly through social media. According to Pamela (2021), the Internet is a network of computers containing valuable data, so there are many security mechanisms in place to protect that data, but there is a weak link: the human. When a user freely gives away their data or access to their computer, security mechanisms have a much more difficult time protecting their data and devices.

Therefore, Imperva (2021) defines social engineering (a type of attack used to steal user data, including login credentials and credit card numbers) as a type of attack that is one of the most common social engineering attacks. The attack happens when an attacker fools a victim into opening an email, instant message, or text message as if it were from a trusted source. Upon clicking the link, the recipient is fooled into believing that they've received a gift and unsuspectingly clicks a malicious link, resulting in the installation of malware, the freezing of the system as part of a ransomware attack, or the disclosure of sensitive information.

Computer security threats have increased substantially in recent years, owing to the rapid adoption of technology improvements, while simultaneously increasing the vulnerability of human exploitation. Users should know how the phishers do it, and they should also be aware of techniques to help protect themselves from becoming phished.

The strategies employed by cybercriminals are becoming more complex as technology advances. Other than phishing, there are a variety of methods for obtaining personal information from users. KnowBe4 (2021) stated the following techniques:

a) Vishing (Voice Phishing): This kind of phishing includes the phisher calling the victim to get personal information about the bank account. The most common method of phone phishing is to use a phony caller ID.

b) Smishing (SMS Phishing): Phishing via Short Message Service (SMS) is known as Smishing. It is a method of luring a target through the SMS text message service by sending a link to a phishing website.

c) Ransomware: A ransomware attack is a type of attack that prevents users from accessing a device or data unless they pay up.

d) Malvertising: Malvertising is malicious advertising that uses active scripts to download malware or push undesirable information onto your computer. The most prevalent techniques used in malvertisements are exploits in Adobe PDF and Flash.

Hence, this is a rapidly evolving threat to individuals as well as big and small corporations. Criminals now have access to industrial-strength services on the dark web, resulting in an increase in the amount of these phishing links and emails, and, more frighteningly, they are increasing in 'quality,' making them tougher to detect.

1.2 Statement of Problem

Phishing attacks have gotten increasingly complex, it is very difficult for an average person to determine if an email message link or website is legitimate. Cyber-attacks by criminals that employ phishing schemes are so prevalent and successful nowadays.

Hence, this project seeks to address fake URLs and domain names by identifying phishing website links. Therefore, having a web application that provides the user an interface to check if a URL is Phishing or legitimate will help decrease security risks to individuals and organizations.

1.3 Aim of Study

This project aims to detect phishing websites using machine learning and deep neural networks by developing a web application that allows users to check if a URL is phishing or legitimate and have access to resources to help tackle phishing attacks.

1.4 Objectives of the Study

To accomplish the project's purpose, the following particular objectives have been established:

- i. dataset collection and pre-processing;
- ii. machine-learning model selection and development ;
- iii. development of a web-based application for detection;
- iv. Integration of the developed model to a web application.

1.5 Methodology

An extensive review was done on related topics and existing documented materials such as journals, e-books, and websites containing related information gathered which was examined and reviewed to retrieve essential data to better understand and know how to help improve the system.

The methodology used to achieve the earlier stated objectives is explained below.

The dataset collection consists of phishing and legitimate URLs which were obtained from open-source platforms. The dataset was then pre-processed that is cleaned up from any abnormality such as missing data to avoid data imbalance. Afterward, expository data analysis was done on the dataset to explore and summarize the dataset. Once the dataset was free from all anomalies, website content-based features were extracted from the dataset to get accurate features to train and test the model. An extensive review was done on existing works of literature and machine learning models on detecting phishing websites to best decide the classification models to solve the problem of detecting phishing websites. Hence, Series of these machine learning classification models such as Decision Tree, Support Vector Machine, XGBooster, Multilayer perceptions, Auto encoder Neural Network and Random Forest was deployed on the dataset to distinguish between phishing and legitimate URLs. The best model with high training accuracy out of all the deployed models was selected then integrated into a developed web application. Thus, a user can enter a URL link on the web application to predict if it is phishing or legitimate.

1.6 Significance of the Project

According to Abdelhamid, Thabtah, and Abdel-jaber (2017), various fraudulent websites have been built on the World Wide Web in the previous decade to resemble reputable websites and steal financial assets from users and organizations. This type of online scam is known as phishing, and it has cost the internet community and other stakeholders hundreds of millions of dollars. As a result, robust countermeasures that can identify phishing are required.

These are the challenges to be addressed in this project:

- a. Reduce the rate of financial theft from users and organizations online.
- b. Educate Internet Users on the deception of phishers.
- c. Educate Internet users on the countermeasures of a phishing attack.

1.7 Scope of the Study

This study explores data science and machine learning models that use datasets gotten from open-source platforms to analyze website links and distinguish between phishing and legitimate URL links.

The model will be integrated into a web application, allowing a user to predict if a URL link is legitimate or phishing. This online application is compatible with a variety of browsers.

1.8 Project Report Arrangement

This report is structured into five chapters. The rest of the chapters are stated as follows:

Chapter two describes the summary of works of literature review on the research and related works which is divided into five areas which are: Introduction, Theoretical Review, Conceptual Review, and Empirical Review and Summary of the chapter. Chapter three describes extensively the methodology, system analysis, and design of the system model. Chapter four describes the implementation of the machine learning model in the methodology and discussion on the result. Chapter five discusses the Summary, Conclusion, Recommendation, Limitation of Study, and Contribution to knowledge.

CHAPTER TWO

LITERATURE REVIEW

2.1 Overview of the Study

This chapter offers an insight into various important studies conducted by excellent scholars from articles, books, and other sources relevant to the detection of phishing websites. It also provides the project with a theoretical review, conceptual review, and empirical review to demonstrate understanding of the project.

2.2 Theoretical Review

Ankit and Gupta (2017) mentioned that Statistics show that according to Internet world stats ("Internet world stats usage and population statistics", 2014), the total numbers of Internet users worldwide are 2.97 billion in 2014; that is, more than 38% of the world population uses the Internet. Hackers take advantage of the insecure Internet system and can fool unaware users to fall for phishing scams. A phishing e-mail is used to defraud both individuals and financial organizations on the Internet. ("RSA Anti-Fraud Command Center", n.d.) Said the Anti-Phishing Working Group (APWG) is an international consortium that is dedicated to promoting research, education, and law enforcement to eliminate online fraud and cyber-crime. In 2012, total phishing attacks increased by 160% over 2011, signifying a record year in phishing volumes. The total phishing attacks detected in 2013 were approximately 450,000 and led to financial losses of more than 5.9 billion dollars ("RSA Anti-Fraud Command Center", n.d.). Total attack increases by 1% in 2013 as compared to 2012. The total number of phishing attacks noticed in Q1 (first quarter) of 2014 was 125,215, a 10.7 percent increase over Q4 (fourth quarter) of 2013. More than 55% of phishing websites contain the name of the target site in some form to fool users and 99.4% of phishing websites use port 80 ("Anti-Phishing Working Group (APWG)Phishing activity trends report first quarter",

2014). According to the APWG report in the first quarter of 2014, the second-highest number of phishing attacks ever recorded was between January and March 2014 ("Anti-Phishing Working Group (APWG) Phishing activity trends report first quarter", 2014) and payment services are the most targeted industry. During the second half of 2014, 123,972 unique phishing attacks were observed ("APWG report", 2014).

In the year 2011, total financial losses were 1.2 billion, and they rose to 5.9 billion dollars in 2013. The financial losses due to phishing attacks in 2014 and 2015 were 4.5 and 4.6, respectively, as shown in Figure 2.1 ("The RSA Current State of Cybercrime", n.d.). The growth of phishing attacks from 2005 to 2015 is shown in Figure 2.2.



Figure 2.1 Worldwide financial losses (in billion) due to phishing attacks.

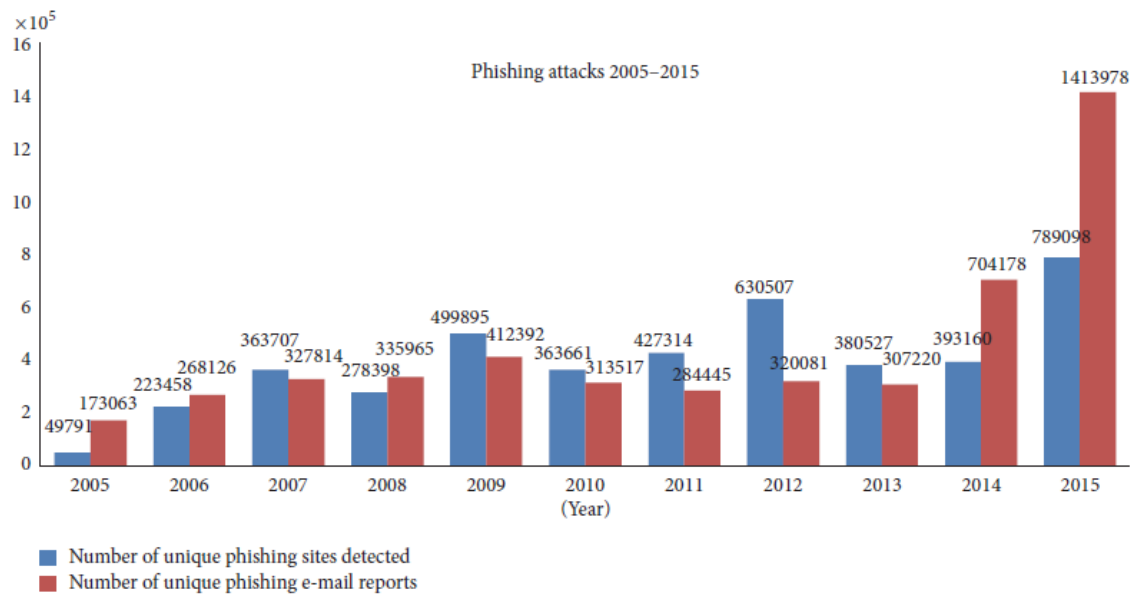


Figure 2.2 growth of phishing attacks from 2005 to 2015

2.2.1 Data imputation

Joachim (n.d). Has defined missing data imputation as a statistical method that replaces missing data points with substituted values. Real-world datasets can contain missing values for different reasons. They are often encoded as blanks, NaNs, or other placeholders. Training a model with a dataset with several missed values can have a dramatic impact on the quality of the machine learning algorithm.

There are three main types of missing data:

(1) Missing completely at random (MCAR)

MCAR occurs when the missing variable is completely unsystematic. When our dataset is missing values completely at random, the probability of missing data is unrelated to any other variable and unrelated to the variable with missing values itself. For example, MCAR would occur when data is missing because the responses to a research survey about depression are lost in the mail. (Kyaw, 2020)

(2) Missing at random (MAR)

MAR occurs when the probability of the missing data on a variable is related to some other measured variable but unrelated to the variable with missing values itself. For example, the data values are missing because males are less likely to respond to a depression survey. In this case, the missing data is related to the gender of the respondents. However, the missing data is not related to the level of depression itself. (Kyaw, 2020).

(3) Missing Not at Random (MNAR)

MNAR occurs when the missing values on a variable are related to the variable with the missing values itself. In this case, the data values are missing because the respondents failed to fill in the survey due to their level of depression. (Kyaw, 2020)

Popular ways for data imputation for cross-sectional datasets:

I. Imputation using (Mean/Median) Values:

It works by measuring the mean/median of the non-missing values in a column and extracting the missing values separately and independently from each other within each column. Only quantitative data can be used (Will, 2019) as shown in Figure 2.3.

II. Imputation using (Most Frequent) or (Zero/Constant) Values:

Most popular is another statistical technique for imputing missing values and this deals with categorical characteristics (strings or numerical representations) by replacing missing data from each column with its most frequent values (Will, 2019). As shown in Figure 2.4.

III. Imputation using k-NN: The k nearest neighbor is an algorithm that is used for simple classification. The algorithm uses ‘feature similarity’ to predict the values of any new data points. This means that the new point is assigned a value based on how closely it resembles the points in the training set. (Will, 2019).

	col1	col2	col3	col4	col5			col1	col2	col3	col4	col5
0	2	5.0	3.0	6	NaN	mean() →	0	2.0	5.0	3.0	6.0	7.0
1	9	NaN	9.0	0	7.0		1	9.0	11.0	9.0	0.0	7.0
2	19	17.0	NaN	9	NaN		2	19.0	17.0	6.0	9.0	7.0

Figure 2.3 Mean imputation

Source: <https://towardsdatascience.com/6-different-ways-to-compensate-for-missing-values-data-imputation-with-examples-6022d9ca0779>

	col1	col2	col3	col4	col5			col1	col2	col3	col4	col5	
0	2	5.0	3.0	6	NaN	df.fillna(0)		0	2	5.0	3.0	6	0.0
1	9	NaN	9.0	0	7.0			1	9	0.0	9.0	0	7.0
2	19	17.0	NaN	9	NaN			2	19	17.0	0.0	9	0.0

Figure 2.4 Zero Imputation

Source: <https://towardsdatascience.com/6-different-ways-to-compensate-for-missing-values-data-imputation-with-examples-6022d9ca0779>

2.2.2 Feature selection

Feature Selection is the method of reducing the input variable to your model by using only relevant data and getting rid of noise in data.

It is also the process of automatically choosing relevant features for your machine-learning model based on the type of problem you are trying to solve. We do this by including or excluding important features without changing them. It helps in cutting down the noise in our data and reducing the size of our input data. Figure 2.5 shows the feature selection process.

Feature selection models are of two types:

- i. **Supervised Models:**

Supervised feature selection refers to the method which uses the output label class for feature selection. They use the target variables to identify the variables which can increase the efficiency of the model.

- ii. **Unsupervised Models:**

Unsupervised Feature selection refers to the method which does not need the output label class for feature selection. We use them for unlabeled data. Figure 2.6 shows the flow of the feature selection model.

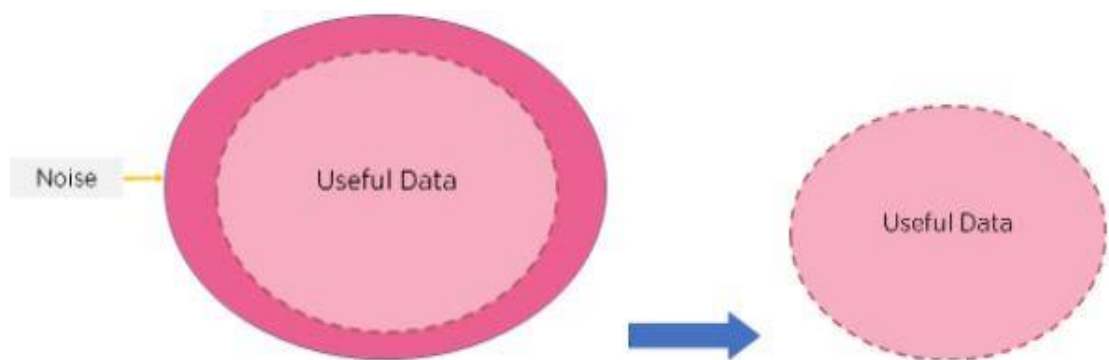


Figure 2.5 Feature selection process

Source: <https://www.simplilearn.com/tutorials/machine-learning-tutorial/feature-selection-in-machine-learning>

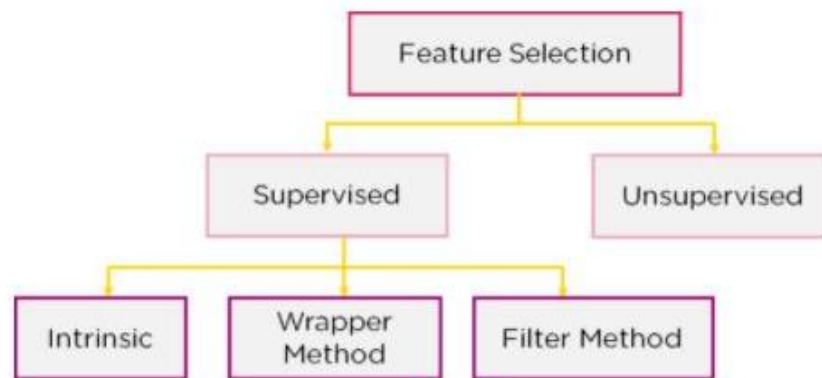


Figure 2.6 Feature Selection Models

Source: <https://www.simplilearn.com/tutorials/machine-learning-tutorial/feature-selection-in-machine-learning>

2.2.3 Feature extraction

Feature extraction is a process of dimensionality reduction by which an initial set of raw data is reduced to more manageable groups for processing. A characteristic of these large data sets is a large number of variables that require a lot of computing resources to process. Feature extraction is the name for methods that select and or combine variables into features, effectively reducing the amount of data that must be processed, while still accurately and completely describing the original data set (deepAI, n.d.).

Why is Feature Extraction Useful? The process of feature extraction is useful when you need to reduce the number of resources needed for processing without losing important or relevant information. Feature extraction can also reduce the amount of redundant data for a given analysis. Also, the reduction of the data and the machine's efforts in building variable combinations (features) facilitate the speed of learning and generalization steps in the machine learning process. (DeepAI, n.d.)

According to (Rami, Fadi & Lee, 2015), they have compounded important features that have proved to be sound and effective in predicting phishing websites. In addition, they have proposed some new features, experimentally assign new rules to some well-known features and update some other features.

These feature selections include:

- i. Address Bar based Features
- ii. Abnormal Based Features
- iii. HTML and JavaScript-based Features
- iv. Domain-based Features

All the listed feature selection above consists of feature extraction which are guided by rules. The feature extraction is as follows:

2.2.3.1 address bar based features

(1) Using the IP Address

If an IP address is used as an alternative to the domain name in the URL, such as “http://125.98.3.123/fake.html”, users can be sure that someone is trying to steal their personal information. Sometimes, the IP address is even transformed into hexadecimal code as shown in the following link “http://0x58.0xCC.0xCA.0x62/2/paypal.ca/index.html”.

Rule: IF $\begin{cases} \text{If The Domain Part has an IP Address} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

(2) Long URL to Hide the Suspicious Part

Phishers can use a long URL to hide the doubtful part in the address bar. For example:

http://federmacedoadv.com.br/3f/aze/ab51e2e319e51502f416dbe46b773a5e/?cmd=_home&dispatch=11004d58f5b74f8dc1e7c2e8dd4105e811004d58f5b74f8dc1e7c2e8dd4105e8@phishing.website.html

To ensure the accuracy of our study, we calculated the length of URLs in the dataset and produced an average URL length. The results showed that if the length of the URL is greater than or equal to 54 characters then the URL is classified as phishing. By reviewing our dataset, we were able to find 1220 URL lengths equals 54 or more which constitute 48.8% of the total dataset size.

Rule: IF $\begin{cases} \text{URL length} < 54 \rightarrow \text{feature} = \text{Legitimate} \\ \text{else if URL length} \geq 54 \text{ and } \leq 75 \rightarrow \text{feature} = \text{Suspicious} \\ \text{otherwise} \rightarrow \text{feature} = \text{Phishing} \end{cases}$

We have been able to update this feature rule by using a method based on the frequency and thus improving its accuracy.

(3) Using URL Shortening Services “Tiny-URL”

URL shortening is a method on the “World Wide Web” in which a URL may be made considerably smaller in length and still lead to the required webpage. This is accomplished utilizing an “HTTP Redirect” on a short domain name, which links to the webpage that has a long URL. For example, the URL “http://portal.hud.ac.uk/” can be shortened to “bit.ly/19DXSk4”.

Rule: IF $\begin{cases} \text{TinyURL} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

(4) URL’s having “@” Symbol

Using the “@” symbol in the URL leads the browser to ignore everything preceding the “@” symbol and the real address often follows the “@” symbol.

Rule: IF $\begin{cases} \text{Url Having @ Symbol} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

(5) Redirecting using “//”

The existence of “//” within the URL path means that the user will be redirected to another website. An example of such URLs is: “http://www.legitimate.com//http://www.phishing.com”. We examine the location where the “//” appears. We find that if the URL starts with “HTTP”, that means the “//” should appear in the sixth position. However, if the URL employs “HTTPS” then the “//” should appear in the seventh position.

Rule: IF $\begin{cases} \text{the position of the Last Occurrence of "//" in the URL} > 7 \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

(6) Adding Prefix or Suffix Separated by (-) to the Domain

The dash symbol is rarely used in legitimate URLs. Phishers tend to add prefixes or suffixes separated by (-) to the domain name so that users feel that they are dealing with a legitimate webpage. For example, <http://www.Confirme-paypal.com/>.

Rule: IF $\begin{cases} \text{Domain Name Part Includes (-) Symbol} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

(7) Sub Domain and Multi-Sub Domains

Let us assume we have the following link: <http://www.hud.ac.uk/students/>. A domain name might include the country-code top-level domains (ccTLD), which in our example is “UK”. The “ac” part is shorthand for “academic”, the combined “ac. UK” is called a second-level domain (SLD) and “hud” is the actual name of the domain. To produce a rule for extracting this feature, we first have to omit the (www.) from the URL which is a subdomain in itself. Then, we have to remove the (ccTLD) if it exists. Finally, we count the remaining dots. If the number of dots is greater than one, then the URL is classified as “Suspicious” since it has one subdomain. However, if the dots are greater than two, it is classified as “Phishing” since it will have multiple subdomains. Otherwise, if the URL has no subdomains, we will assign “Legitimate” to the feature.

Rule: IF $\begin{cases} \text{Dots In Domain Part} = 1 \rightarrow \text{Legitimate} \\ \text{Dots In Domain Part} = 2 \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$

(8) HTTPS (Hypertext Transfer Protocol with Secure Sockets Layer)

The existence of HTTPS is very important in giving the impression of website legitimacy, but this is not enough. The authors in (Mohammad, Thabtah and McCluskey 2012) (Mohammad, Thabtah and McCluskey 2013) suggest checking the certificate assigned with HTTPS including the extent of the trust certificate issuer, and the

certificate age. Certificate Authorities that are consistently listed among the top trustworthy names include: “GeoTrust, GoDaddy, Network Solutions, Thawte, Comodo, Doster, and VeriSign”. Furthermore, by testing out our datasets, we find that the minimum age of a reputable certificate is two years.

Rule:

IF

$$\left\{ \begin{array}{l} \text{Use HTTPS and Issuer Is Trusted and Age of Certificate} \geq 1 \text{ Years} \rightarrow \text{Legitimate} \\ \text{Using https and Issuer Is Not Trusted} \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{array} \right.$$

(9) Domain Registration Length

Based on the fact that a phishing website lives for a short period, we believe that trustworthy domains are regularly paid for several years in advance. In our dataset, we find that the longest fraudulent domains have been used for one year only.

Rule: IF $\left\{ \begin{array}{l} \text{Domains Expire on} \leq 1 \text{ years} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{array} \right.$

(10) Favicon

A favicon is a graphic image (icon) associated with a specific webpage. Many existing user agents such as graphical browsers and newsreaders show favicon as a visual reminder of the website identity in the address bar. If the favicon is loaded from a domain other than that shown in the address bar, then the webpage is likely to be considered a Phishing attempt.

Rule: IF $\left\{ \begin{array}{l} \text{Favicon Loaded From External Domain} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{array} \right.$

(11) Using Non-Standard Port

This feature is useful in validating if a particular service (e.g., HTTP) is up or down on a specific server. To control intrusions, it is much better to merely open the ports that you need. Several firewalls, Proxy, and Network Address Translation (NAT) servers will, by default, block all or most of the ports and only open the ones selected. If all ports are open, phishers can run almost any service they want and as a result, user information is threatened. The most important ports and their preferred status are shown in Table 2.1

Rule: IF $\begin{cases} \text{Port \# is of the Preferred Status} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

(12) The Existence of “HTTPS” Token in the Domain Part of the URL

The phishers may add the “HTTPS” token to the domain part of a URL to trick users.

For example,

<http://https-www-paypal-it-webapps-mpp-home.soft-hair.com/>.

Rule: IF $\begin{cases} \text{Using HTTP Token in Domain Part of The URL} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

Table 2.1: Common ports to be checked

PORT	Service	Meaning	Preferred Status
21	FTP	Transfer files from one host to another	Close
22	SSH	Secure File Transfer Protocol	Close
23	Telnet	provide a bidirectional interactive text-oriented communication	Close
80	HTTP	Hyper text transfer protocol	Open
443	HTTPS	Hypertext transfer protocol secured	Open
445	SMB	Providing shared access to files, printers, serial ports	Close
1433	MSSQL	Store and retrieve data as requested by other software applications	Close
1521	ORACLE	Access oracle database from the web.	Close
3306	MySQL	Access MySQL database from the web.	Close
3389	Remote Desktop	allow remote access and remote collaboration	Close

2.2.3.2 abnormal based features

(1) Request URL

Request URL examines whether the external objects contained within a webpage such as images, videos, and sounds are loaded from another domain. In legitimate web pages, the webpage address and most of the objects embedded within the webpage are sharing the same domain.

$$\text{Rule: IF } \begin{cases} \% \text{ of Request URL} < 22\% \rightarrow \text{Legitimate} \\ \% \text{ of Request URL} \geq 22\% \text{ and } 61\% \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{feature} = \text{Phishing} \end{cases}$$

(2) URL of Anchor

An anchor is an element defined by the <a> tag. This feature is treated exactly as “Request URL”. However, for this feature we examine:

1. If the <a> tags and the website have different domain names. This is similar to the request URL feature.
2. If the anchor does not link to any webpage, e.g.:

A.

B.

C.

D.

$$\text{Rule: IF } \begin{cases} \% \text{ of URL Of Anchor} < 31\% \rightarrow \text{Legitimate} \\ \% \text{ of URL Of Anchor} \geq 31\% \text{ And } \leq 67\% \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$$

(3) Links in <Meta>, <Script> and <Link> tags

Given that our investigation covers all angles likely to be used in the webpage source code, we find that it is common for legitimate websites to use <Meta> tags to offer metadata about the HTML document; <Script> tags to create a client-side script; and <Link> tags to retrieve other web resources. It is expected that these tags are linked to the same domain of the webpage.

Rule:

IF

$$\left\{ \begin{array}{l} \% \text{ of Links in " < Meta > ", " < Script > " and " < Link > " } < 17\% \rightarrow \text{Legitimate} \\ \% \text{ of Links in " < Meta > ", " < Script > " and " < Link > " } \geq 17\% \text{ And } \leq 81\% \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{array} \right.$$

(4) Server Form Handler (SFH)

SFHs that contain an empty string or "about: blank" are considered doubtful because action should be taken upon the submitted information. In addition, if the domain name in SFHs is different from the domain name of the webpage, this reveals that the webpage is suspicious because the submitted information is rarely handled by external domains.

$$\text{Rule: IF} \left\{ \begin{array}{l} \text{SFH is "about blank" Or Is Empty} \rightarrow \text{Phishing} \\ \text{SFH Refers To A Different Domain} \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{array} \right.$$

(5) Submitting Information to Email

Web-form allows a user to submit his personal information that is directed to a server for processing. A phisher might redirect the user's information to his email. To that end, a server-side script language might be used such as the "mail ()" function in PHP. One more client-side function that might be used for this purpose is the "mailto:" function.

Rule:

IF $\begin{cases} \text{Using "mail ()" or "mailto:" Function to Submit User Information} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

(6) Abnormal URL

This feature can be extracted from the WHOIS database. For a legitimate website, identity is typically part of its URL.

Rule: IF $\begin{cases} \text{The Host Name Is Not Included In URL} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

2.2.3.3 html and javascript-based features

(1) Website Forwarding

The fine line that distinguishes phishing websites from legitimate ones is how many times a website has been redirected. In our dataset, we find that legitimate websites have been redirected one-time max. On the other hand, phishing websites containing this feature have been redirected at least 4 times.

$$\text{Rule: IF } \begin{cases} \text{ofRedirect Page} \leq 1 \rightarrow \text{Legitimate} \\ \text{of Redirect Page} \geq 2 \text{ And } < 4 \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$$

(2) Status Bar Customization

Phishers may use JavaScript to show a fake URL in the status bar to users. To extract this feature, we must dig out the webpage source code, particularly the “onMouseOver” event, and check if it makes any changes on the status bar.

$$\text{Rule: IF } \begin{cases} \text{onMouseOver Changes Status Bar} \rightarrow \text{Phishing} \\ \text{It Does't Change Status Bar} \rightarrow \text{Legitimate} \end{cases}$$

(3) Disabling Right Click

Phishers use JavaScript to disable the right-click function so that users cannot view and save the webpage source code. This feature is treated exactly as “Using onMouseOver to hide the Link”. Nonetheless, for this feature, we will search for the event “event.Button==2” in the webpage source code and check if the right-click is disabled.

$$\text{Rule: IF } \begin{cases} \text{Right Click Disabled} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

(4) Using Pop-up Window

It is unusual to find a legitimate website asking users to submit their personal information through a pop-up window. On the other hand, this feature has been used in some legitimate websites and its main goal is to warn users about fraudulent activities or broadcast a welcome announcement, though no personal information was asked to be filled in through these pop-up windows.

Rule: IF $\begin{cases} \text{Popup Window Contains Text Fields} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

(5) Iframe Redirection

The Iframe is an HTML tag used to display an additional webpage into one that is currently shown. Phishers can make use of the “iframe” tag and make it invisible i.e. without frame borders. In this regard, phishers make use of the “frame border” attribute which causes the browser to render a visual delineation.

Rule: IF $\begin{cases} \text{Using iframe} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

2.2.3.4 domain-based features

(1) Age of Domain

This feature can be extracted from the WHOIS database (Whois 2005). Most phishing websites live for a short period. By reviewing our dataset, we find that the minimum age of the legitimate domain is 6 months.

Rule: IF $\begin{cases} \text{Age Of Domain} \geq 6 \text{ months} \rightarrow \text{Legitimate} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$

(2) DNS Record

For phishing websites, either the claimed identity is not recognized by the WHOIS database (Whois 2005) or no records are found for the hostname (Pan and Ding 2006). If the DNS record is empty or not found then the website is classified as “Phishing”, otherwise it is classified as “Legitimate”.

Rule: IF $\begin{cases} \text{no DNS Record For The Domain} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

(3) Website Traffic

This feature measures the popularity of the website by determining the number of visitors and the number of pages they visit. However, since phishing websites live for a short period, they may not be recognized by the Alexa database (Alexa the Web Information Company., 1996). By reviewing our dataset, we find that in the worst scenarios, legitimate websites ranked among the top 100,000. Furthermore, if the domain has no traffic or is not recognized by the Alexa database, it is classified as “Phishing”. Otherwise, it is classified as “Suspicious”.

Rule: IF $\begin{cases} \text{Website Rank} < 100,000 \rightarrow \text{Legitimate} \\ \text{Website Rank} > 100,000 \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Phish} \end{cases}$

(4) PageRank

PageRank is a value ranging from “0” to “1”. PageRank aims to measure how important a webpage is on the Internet. The greater the PageRank value the more important the webpage. In our datasets, we find that about 95% of phishing web pages have no PageRank. Moreover, we find that the remaining 5% of phishing webpages may reach a PageRank value up to “0.2”.

Rule: IF $\begin{cases} \text{PageRank} < 0.2 \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

(5) Google Index

This feature examines whether a website is in Google’s index or not. When a site is indexed by Google, it is displayed on search results (Webmaster resources, 2014). Usually, phishing webpages are merely accessible for a short period, and as a result, many phishing webpages may not be found on the Google index.

Rule: IF $\begin{cases} \text{Webpage Indexed by Google} \rightarrow \text{Legitimate} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$

(6) Number of Links Pointing to Page

The number of links pointing to the webpage indicates its legitimacy level, even if some links are of the same domain (Dean, 2014). In our datasets and due to their short life span, we find that 98% of phishing dataset items have no links pointing to them. On the other hand, legitimate websites have at least 2 external links pointing to them.

Rule: IF $\begin{cases} \text{Of Link Pointing to The Webpage} = 0 \rightarrow \text{Phishing} \\ \text{Of Link Pointing to The Webpage} > 0 \text{ and } \leq 2 \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

(7) Statistical-Reports Based Feature

Several parties such as Phish Tank (PhishTank Stats, 2010-2012), and Stop-Badware (StopBadware, 2010-2012) formulate numerous statistical reports on phishing websites at every given period; some are monthly and others are quarterly. In our research, we used 2 forms of the top ten statistics from Phish Tank: “Top 10 Domains” and “Top 10 IPs” according to statistical reports published in the last three years, starting in January 2010 to November 2012. Whereas for “Stop-Badware”, we used “Top 50” IP addresses.

Rule: IF $\begin{cases} \text{Host Belongs to Top Phishing IPs or Top Phishing Domains} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

Rishikesh and Irfan (2018), stated the feature extraction used for detecting phishing URLs, are as follows:

- i. **Presence of IP address in URL:** If IP address is present in URL, then the feature is set to 1 else set to 0. Most benign sites do not use an IP address as an URL to download a webpage. The use of an IP address in a URL indicates that the attacker is trying to steal sensitive information.
- ii. **Presence of @ symbol in URL:** If @ symbol is present in URL then the feature is set to 1 else set to 0. Phishers add special symbol @ in the URL leads the browser to ignore everything preceding the “@” symbol and the real address often follows the “@” symbol [4].
- iii. **The number of dots in Hostname:** Phishing URLs have many dots in URL. For example, <http://shop.fun.amazon.phishing.com>, in this URL phishing.com is an actual domain name, whereas the use of the “amazon” word is to trick users to click on it. The average number of dots in benign URLs is 3. If the number of dots in URLs is more than 3 then the feature is set to 1 else to 0.

- iv. **Prefix or Suffix separated by (-) to the domain:** If a domain name is separated by the dash (-) symbol then the feature is set to 1 else to 0. The dash symbol is rarely used in legitimate URLs. Phishers add a dash symbol (-) to the domain name so that users feel that they are dealing with a legitimate webpage. For example, the Actual site is <http://www.onlineamazon.com> but the phisher can create another fake website like <http://www.online-amazon.com> to confuse the innocent users.
- v. **URL redirection:** If “//” is present in the URL path then the feature is set to 1 else to 0. The existence of “//” within the URL path means that the user will be redirected to another website.
- vi. **HTTPS token in URL:** If HTTPS token is present in URL then the feature is set to 1 else to 0. Phishers may add the “HTTPS” token to the domain part of a URL to trick users. For example, <http://https-www-paypal-it-mpp-home.soft-hair.com>.
- vii. **Information submission to Email:** Phishers might use “mail ()” or “mailto:” functions to redirect the user’s information to his email. If such functions are present in the URL then the feature is set to 1 else to 0.
- viii. **URL Shortening Services “Tiny-URL”:** Tiny-URL service allows the phisher to hide long phishing URLs by making them short. The goal is to redirect the user to phishing websites. If the URL is crafted using shortening services (like bit.ly) then the feature is set to 1 else 0.
- ix. **Length of Hostname:** Average length of the benign URLs is found to be 25, If the URL’s length is greater than 25 then the feature is set to 1 else to 0
- x. **Presence of sensitive words in URL:** Phishing sites use sensitive words in its URL so that users feel that they are dealing with a legitimate webpage. Below

are the words that found in many phishing URLs: - 'confirm', 'account', 'banking', 'secure', 'ebyisapi', 'webscr', 'signin', 'mail', 'install', 'toolbar', 'backup', 'paypal', 'password', 'username', etc;

- xi. **The number of slashes in URL:** The number of slashes in benign URLs is found to be a 5; if the number of slashes in URL is greater than 5 then the feature is set to 1 else to 0.
- xii. **Presence of Unicode in URL:** Phishers can make use of Unicode characters in URL to trick users to click on them. For example, the domain “xn--80ak6aa92e.com” is equivalent to "apple.com". The visible URL to a user is "apple.com" but after clicking on this URL, the user will visit “xn--80ak6aa92e.com” which is a phishing site.
- xiii. **Age of SSL Certificate:** The existence of HTTPS is very important in giving the impression of website legitimacy. But minimum age of the SSL certificate of benign websites is between 1 year to 2 years.
- xiv. **URL of Anchor:** We have extracted this feature by crawling the source code on the URL. URL of the anchor is defined by <a> tag. If the <a> tag has a maximum number of hyperlinks that are from the other domain then the feature is set to 1 else to 0.
- xv. **IFRAME:** We have extracted this feature by crawling the source code of the URL. This tag is used to add another web page to the existing main webpage. Phishers can make use of the “iframe” tag and make it invisible i.e., without frame borders. Since the border of an inserted webpage is invisible, the user seems that the inserted webpage is also part of the main web page and can enter sensitive information.

- xvi. **Website Rank:** We extracted the rank of websites and compare it with the first One hundred thousand websites of the Alexa database. If the rank of the website is greater than 10, 0000 then the feature is set to 1 else to 0.

(Shreya, 2020), Stated feature selection used in her project which is based on the listed categories below which can be used to make predictions for phishing websites. These categories are similar to (Rami, Fadi & Lee, 2015) compounded principle for predicting phishing websites.

- i. Address Bar based Features
- ii. Domain-based Features
- iii. HTML & JavaScript based Feature

2.2.4 Algorithm and model evaluation (Performance Metrics)

Algorithms or Models are used when it comes to machine learning includes the most important topics in Artificial Intelligence. It is further divided into Supervised and Unsupervised learning which can be related to labeled and unlabeled data analysis or data prediction. In Supervised Learning, we have two more types of business problems called Regression and Classification.

Classification is a machine learning algorithm where we get the labeled data as input and we need to predict the output into a class. If there are two classes, then it is called Binary Classification. If there are more than two classes, then it is called Multi-Class Classification.

There are so many classification algorithms available but for this project, let focus on the commonly used algorithms use by researchers to predict phishing websites.

- I. According to Rishikesh and Irfan (2018), three machine learning classification models such as Decision Tree, Random forest, and Support vector machine was selected to detect phishing websites.

- i. Decision Tree Algorithm (Rahul, 2017)

One of the most widely used algorithms in machine learning technology.

The decision tree algorithm is easy to understand and also easy to implement. The decision tree begins its work by choosing the best splitter from the available attributes for classification which is considered as a root of the tree. The algorithm continues to build a tree until it finds the leaf node. Decision tree creates training model which is used to predict target value or class in tree representation each internal node of the tree belongs to attribute and each leaf node of the tree

belongs to the class label. In the decision tree algorithm, Gini index, and information gain methods are used to calculate these nodes.

ii. Random Forest Algorithm (Saimadhu, 2017)

Random forest algorithm is one of the most powerful algorithms in machine learning technology and it is based on the concept of decision tree algorithm. Random forest algorithm creates the forest with several decision trees. A high number of a tree gives high detection accuracy. The creation of trees is based on the bootstrap method. In the bootstrap method features and samples of a dataset are randomly selected with replacement to construct a single tree. Among randomly selected features, a random forest algorithm will choose the best splitter for the classification, and as the decision tree algorithm; the Random forest algorithm also uses Gini index and information gain methods to find the best splitter. This process will get continue until a random forest creates N number of trees. Each tree in the forest predicts the target value and then the algorithm will calculate the votes for each predicted target. Finally, the random forest algorithm considers the high-voted predicted target as a final prediction.

iii. Support Vector Machine Algorithm (Noel, 2016)

Support vector machine is another powerful algorithm in machine learning technology. In the support vector machine algorithm, each data item is plotted as a point in n-dimensional space, and the support vector machine algorithm constructs separating lines for the classification of two classes, this separating line is well known as a hyperplane.

Support vector machine seeks for the closest points called support vectors and once it finds the closest point it draws a line connecting to them. Support vector machine then constructs separating line which bisects and perpendicular to the connecting line. To classify data perfectly the margin should be maximum. Here the margin is a distance between hyperplane and support vectors. In a real scenario, it is not possible to separate complex and nonlinear data, to solve this problem support vector machine uses kernel trick which transforms lower-dimensional space to higher-dimensional space.

II. Kondeti, Konka, and Kavishree (2021) argued that phishing website detection is best done using Machine learning supervised classification models. Models such as Decision Tree Classifier, Random Forest Model, Multilayer Perceptron's, XGBoost Classifier, Auto Encoder, and Support Vector Machines are trained on a categorical input URL such as phishing (1) and legitimate (0). Since Rishikesh et al. (2018) has stated the significance of Decision tree classifier, Random Forest, and Support Vector Machines models on detecting phishing websites, let consider other classification models stated by Kondeti et al. (2021) for detecting phishing websites which are as follows:

- i. XGBooster is best used because it is not any different for classification or regression processes, it is meant for speed and performance. It will add gradient boosting to decision trees.
- ii. Auto Encoder Neural Network

It is like a neural network that has the same no of input neurons as that of output neurons. It has fewer neurons in the hidden layers of the

network that are called predictors. The input neurons pass information to the predictors and process the output.

iii. **Multilayer Perceptrons:**

Multilayer perceptrons are known as feed-forward neural networks. They are used to process multiple stages simultaneously and result in an optimal decision for the processed stage.

2.2.4.1 implementation and result

(Rishikesh & Irfan, 2018) stated the implementation and result for detecting phishing websites. Let consider and evaluate the models used by Rishikesh et al. (2018) to carry out phishing websites detection. Rishikesh et al. (2018) said that the scikit-learn tool is best used to import Machine learning algorithms for phishing detection. A Dataset is divided into a training set and testing set in 50:50, 70:30, and 90:10 ratios respectively. Each classifier is trained using the training set and a testing set is used to evaluate the performance of classifiers. The performance of classifiers has been evaluated by calculating the classifier's accuracy score, false-negative rate, and false-positive rate from Table 2.2.

The result shows that the Random forest algorithm gives better detection accuracy which is 97.14% with the lowest false-negative rate than decision tree and support vector machine algorithms. Result also shows that the detection accuracy of phishing websites increases as more datasets are used as the training dataset. All classifiers perform well when 90% of data is used as a training dataset. Fig. 2.7 shows the detection accuracy of all classifiers when 50%, 70%, and 90% of data is used as a training dataset, and the graph clearly shows that detection accuracy increases when 90% of data is used as a training dataset and random forest detection accuracy is maximum than other two classifiers.

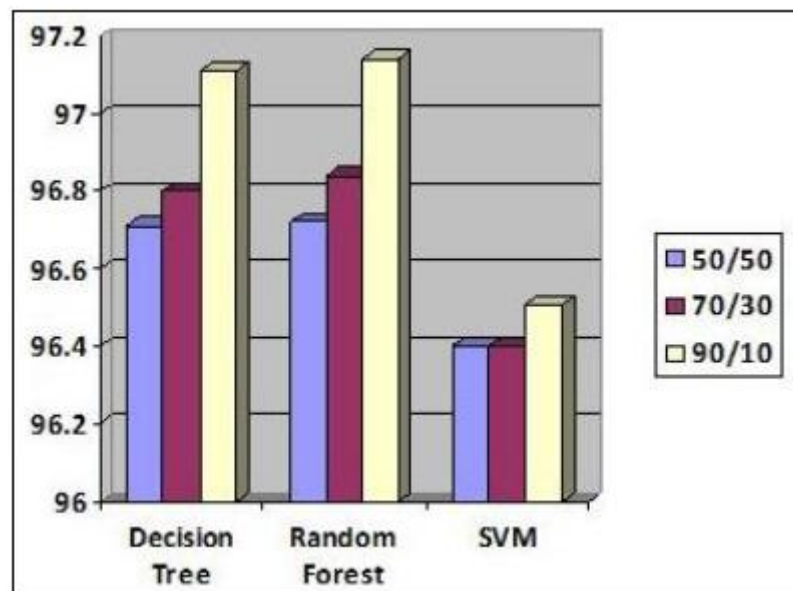


Figure 2.7 Detection accuracy comparison

(Source: Rishikesh and Irfan, 2018)

Table 2.2: Classifier's performance

Dataset Split ratio	Classifiers	Accuracy Score	False Negative Rate	False Positive Rate
50:50	Decision Tree	96.71	3.69	2.93
	Random Forest	96.72	3.69	2.91
	Support vector machine	96.40	5.26	2.08
70:30	Decision Tree	96.80	3.43	2.99
	Random Forest	96.84	3.35	2.98
	Support vector machine	96.40	5.13	2.17
90:10	Decision Tree	97.11	3.18	2.66
	Random Forest	97.14	3.14	2.61
	Support vector machine	96.51	4.73	2.34

- 2 According to (Ashritha, Chaithra, Mangala & Deekshitha, 2019), many algorithms are used to detect phishing websites accurately. Few of them are discussed which can be used to classify the URL as legitimate or phished. The publicly available phishing website's data set from the UCI machine learning repository can be used for training and testing. The features of the dataset are used to predict the result.

Different algorithms that can be used to detect phishing websites are:

i. Artificial Neural Networks (ANN)

An artificial neural network (ANN), inspired by biological neural networks, is a set of interconnected nodes (neurons). Each connection between nodes is typically assigned weights. The network learns by adjusting the weights, in the learning phase for the correct prediction process. ANNs were considered less suitable for data mining due to their poor interpretability and long training times. However, their advantages include the ability to classify patterns on which they have not been trained and a high tolerance for noisy data.

ii. K-Nearest Neighbor (k-NN)

Learning for K-NN classifiers occurs by analogy, that is, by comparing the test tuple to similar training tuples. These are distance-based comparisons that intrinsically assign equal weights to each attribute; therefore, accuracy could be poor when noisy or irrelevant data is presented. However, methods of editing and pruning have been introduced to solve the problem of useless and noisy data tuples respectively. The training tuples are described by n attributes. Each tuple represents a point in an n -dimensional space. The good value for the number of neighbors can be determined experimentally.

iii. Support Vector Machine (SVM)

Support vector machines (SVMs) are used for the classification of both linear and nonlinear data. In short, when given original training data, the algorithm uses a nonlinear mapping to transform it into a higher dimension. In this dimension, a linear optimal hyperplane is searched, to keep the data of any two classes separate. SVMs can be used for classification and numeric prediction as well. The simplest form of SVM is a two-class problem, where the classes are linearly separable. For a 2-D problem, a straight line can be drawn to separate the classes multiple lines could be drawn.

iv. Random Forests (RF)

Random Forests can be built in tandem with random attribute selection using bagging. Random Forests follow an ensemble approach to learning, which is a divide and conquer approach for improving performance. In a simple decision tree, the input or test is added at the top and it traverses down the tree, ending up in smaller subsets. In a random forest, the ensemble mechanism combines various random subsets of trees. The input/test traverses through all the trees. The result is calculated based on an average or weighted average of the individual results, or the voting majority in the case of categorical data. The accuracy of a random forest depends on a measure of the dependence between the classifier and the strength of the individual classifiers and they improve the problem of overfitting of the decision trees.

2.3 Conceptual Review

2.3.1 phishing mechanism

Figure 2.8 below shows a fake website is the clone of a targeted genuine website, and it always contains some input fields (e.g., textbox). When the user submits his/her details, the information is transferred to the attacker. An attacker steals the credential of the innocent user by performing the following steps:

- i. **Construction of Phishing Site.** In the first step, the attacker identifies the target as a well-known organization. Afterward, the attacker collects detailed information about the organization by visiting their website. The attacker then uses this information to construct the fake website.
- ii. **URL Sending.** In this step, the attacker composes a bogus e-mail and sends it to thousands of users. The attacker attached the URL of the fake website in the bogus e-mail. In the case of a spear-phishing attack, an attacker sends the e-mail to selected users. An attacker can also spread the link of phishing websites with the help of blogs, forums, and so forth (Kruegel, Kirda, Mutz, Robertson, & Vigna, 2005).
- iii. **Stealing of the Credentials:** when the user clicks on the attached URL, consequently, fake site is opened in the web browser. The fake website contains a fake login form that is used to take the credential of an innocent user. Furthermore, the attacker can access the information filled by the user.
- iv. **Identity Theft:** attacker uses this credential for malicious purposes. For example, an attacker purchases something by using the credit card details of the user. (Ankit & Gupta, 2017).

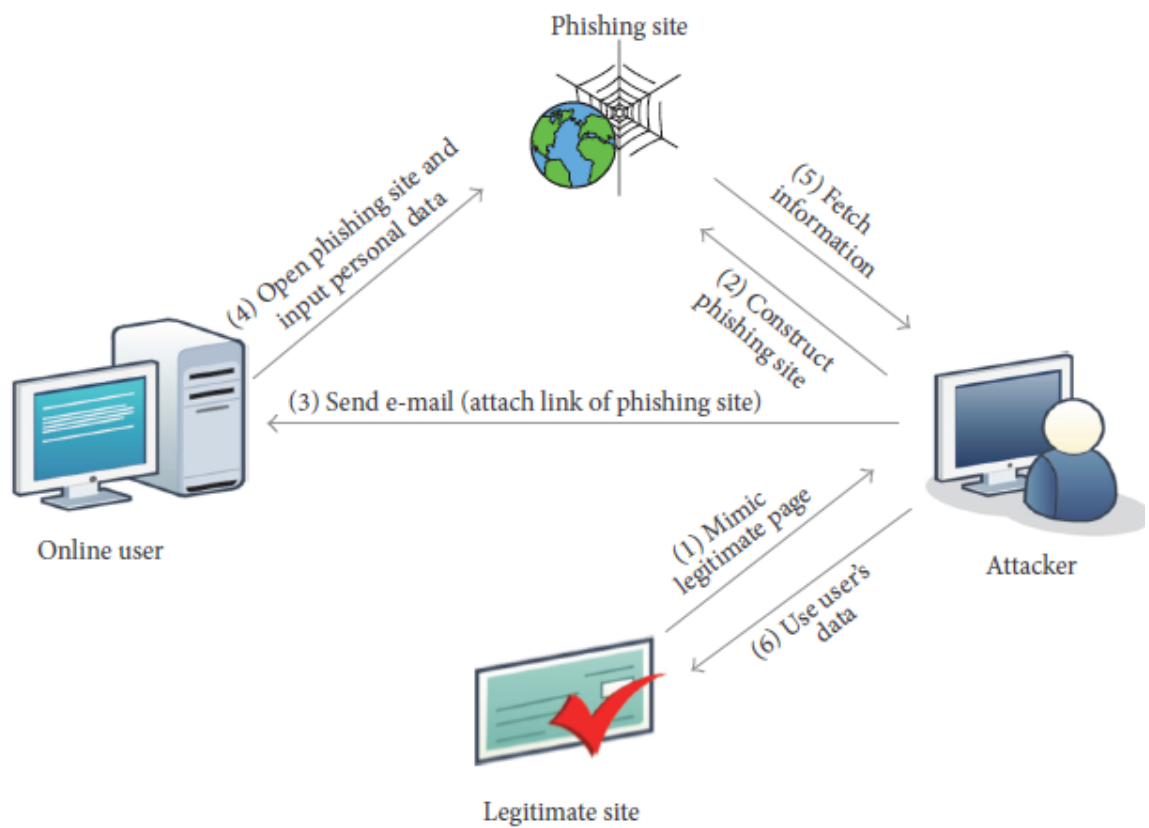


Figure 2.8 Phishing Mechanism

(Source: Ankit & Gupta, 2017)

2.3.2 taxonomy of phishing attack

An Attacker performs a phishing attack by utilizing technical subterfuge and social engineering techniques [(“RSA Anti-Fraud Command Center”, n.d.), (Almomani, Gupta, Atawneh, Meulenberg, & Almomani, 2013)]. In social engineering techniques, attackers carry out this attack by sending bogus e-mail. Attackers often convince recipients to respond using the names of banks, credit card companies, e-retailers, and so forth (Tewari, Jain, & Gupta, 2016). Technical subterfuge strategies install malware into the user’s system to steal credentials directly using Trojan and key logger spyware (Gupta, Tewari, Jain, & Agrawal, 2016). The malware also misaddresses users to fake websites or proxy servers. Attackers attached malware or embedded malicious links in the fraudulent emails and when the user opens the fraud hyperlink, malicious software is installed on the user’s system, which collected the confidential information from the system and sent it to the attacker (e.g., key logger software sends the details of every key hit by the user). Attackers may also get remote access to the victim’s computer and collect data whenever attackers want. In this paper, we focus on social engineering schemes, as it is the most popular way to steal victim’s information by phishing. Classification of various phishing attacks is shown in Figure 2.9. (Ankit & Gupta, 2017).

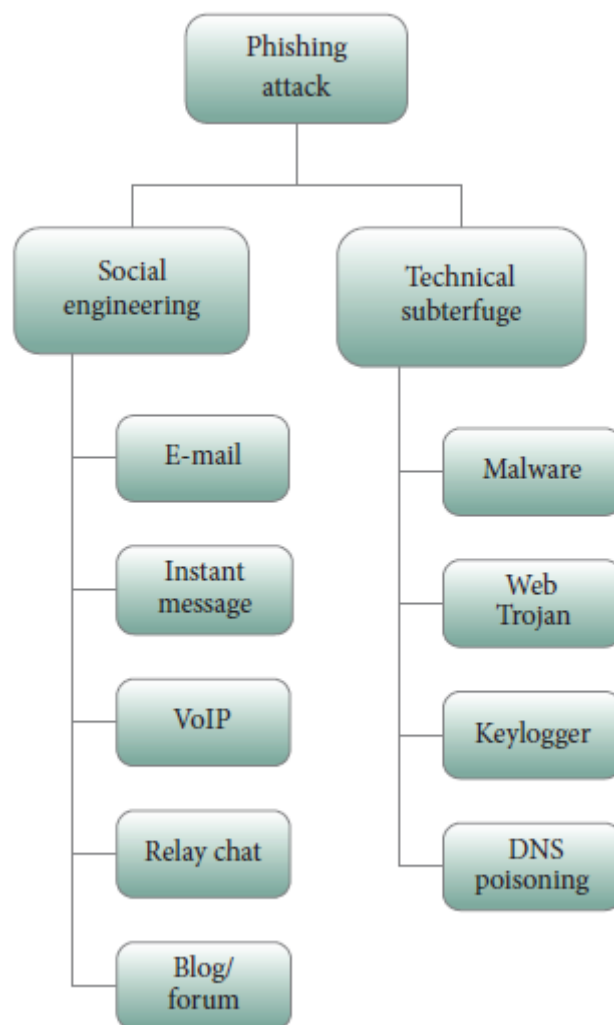


Figure 2.9 Types of Phishing attack

(Source: Ankit & Gupta, 2017).

2.3.3 antiphishing technique

The method to a phishing scam starts with spreading bogus e-mail. To prevent a phishing attack, after receiving such e-mail, antiphishing techniques need to be activated, either by redirecting the phishing mail in the spam folder or by showing a warning when an online user clicks on the link of the phishing URL. The lifecycle of phishing attacks is shown in Figure 2.10. The following steps are involved in the phishing lifecycle:

- i. Step 1. The attacker creates the fake copy of a popular organization and sends the URL of the fake website to a large number of Internet users using e-mail, blogs, social networking sites, and so forth.
- ii. Step 2. In the case of a fake e-mail, every e-mail is first to pass through the DNS-based blacklist filters. If the domain is found in the blacklist, then e-mail is blocked before it reached the SMTP mail server. There are also various solutions available which block fake e-mail based on structural features of mail (Almomani, Gupta, Atawneh, Meulenberg, & Almomani, 2013).
- iii. Step 3. If a fake e-mail bypasses the blacklist and features-based solutions and if the user opens the attached link in the email then some browser-based blacklist techniques block the site on the client-side.
- iv. Step 4. Some other solutions like the heuristic and visual similarities-based approaches also blocked the webpage only when the browser requests for any suspicious webpage.

- v. Step 5. If the phishing attack bypasses all the solutions then it steals the credential of innocent users and sends them to the attacker. The attacker uses this information for financial or some other benefits. (Ankit & Gupta, 2017).

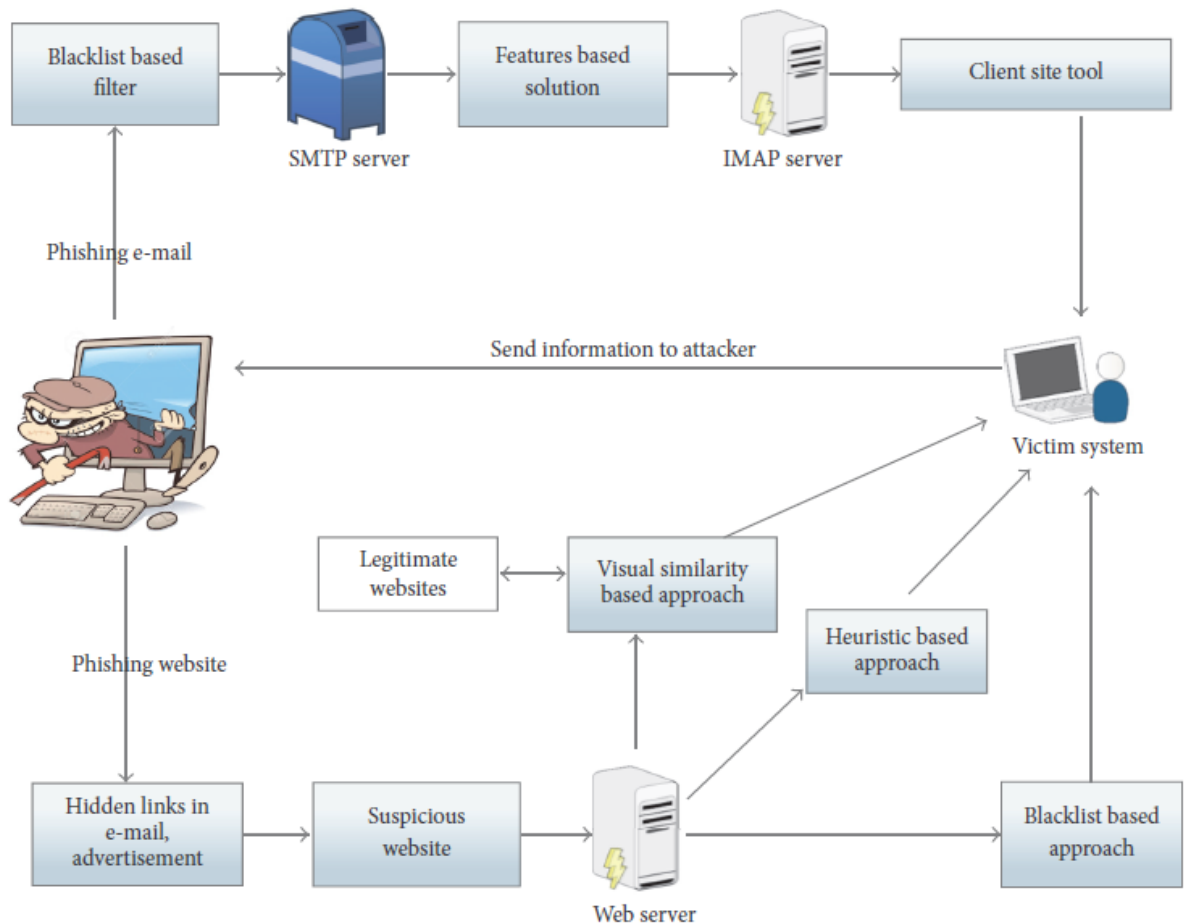


Figure 2.10 Life cycle of phishing attack

(Source: Ankit & Gupta, 2017).

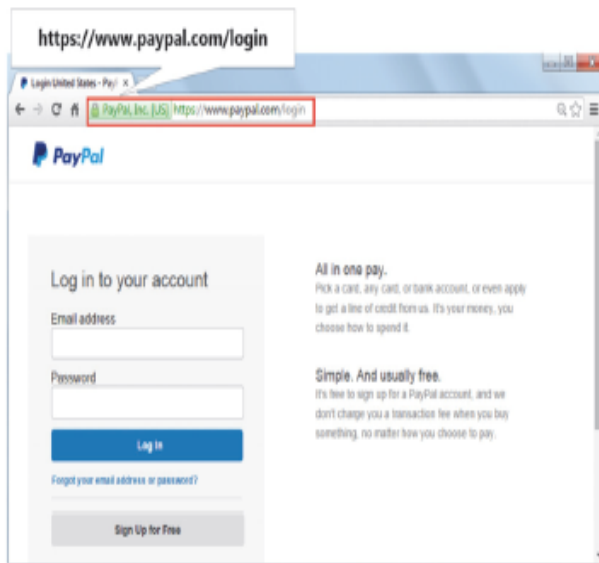
2.3.4 visual similarity-based phishing detection and filtering approaches

A user could become the victim of the phishing attack by looking at the high visual resemblance of phishing websites with the targeted legitimate site, such as page layouts, images, text content, font size, and font color. Take for example the fake and genuine web pages of PayPal as shown in Figure 2.11.

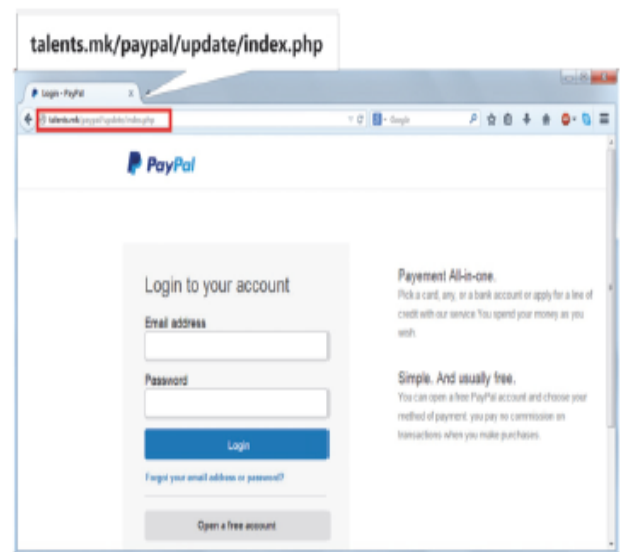
Both pages have the same visual appearance but different URLs. It is not always necessary that people carefully notice on URL and SSL (Secure Socket Layer) certificates of websites. If an attacker does not copy the visual appearance of a targeted website well, then the chances of inputting credentials by Internet users are very less.

An attacker fools the user in the following ways:

- a. Visual Appearance: The phishing website looks similar to its legitimate website. Attackers used to copy the HTML source code of a genuine website to build the fake website.
- b. Address Bar: Attackers also cover the address or URL bar of the website by script or image. The user would believe that they are inputting information on the right website. (Ankit & Gupta, 2017).



(a)



(b)

Figure 2.11 (a) Legitimate PayPal webpage and (b) phishing webpage of PayPal.

(Source: Ankit & Gupta, 2017).

2.4 Empirical Review

In 2017, the paper titled **A Theoretical Study on Different ways to Identify the Phishing URL and Its Prevention Approaches** by V. Karamchand Gandhi and Dr. M. Suriakala works on identifying the general format and procedure of phishing URLs by analyzing 200 phishing email archives provided by APWG. Based on the analysis, they find out the suitable approaches to prevent them from the fraudulent process.

The strengths of the paper are that the approaches, technical techniques, and classification of phishing hyperlinks in phishing e-mails enhance better results in the identification and prevention of phishing attacks.

The weakness of the paper is that the paper faces the challenge of inadequate data, as any machine learning algorithm can only give correct readings/predictions if it is applied to reliable data.

To address this weakness, this project will ensure proper and accurate dataset collection and pre-processing of updated phishing emails from the Anti-Phishing Working Group (APWG).

In 2016, the paper titled **A Literature Review on Phishing Crime, Prevention Review and Investigation of Gaps** by Anjum N. Shaikh, Antesar M. Shabat, and M.A. Hossain produced comprehensive research in which different reviews of previous pieces of literature are presented. The paper proposes an innovative approach of CRI - Crime, prevention Review and Investigation of research gap to combat the phishing in exploring the phishing problem.

The strengths of the paper state the theory of phishing crime, a review of the antiphishing techniques offered by different research, and an investigation of the research gaps. This research aims to develop a practice to understand the growing threats of phishing under a theoretical model of CRI and discover new literature.

The weakness of the paper is that there is need for a thorough research in terms of evaluating the effectiveness of countermeasures for phishing and developing a holistic technique to generating blacklists that are free from the demerits mentioned in the paper. Also, the paper didn't implement the research into a phishing detection system particularly against phishing emails since it is considered the most common way of attack.

To address this weakness, this project will ensure proper and accurate evaluation from different resources on Blacklisting and Countermeasures to a Phishing attack. Also, this project will develop a phishing detection system to validate phishing emails

In 2005, the paper titled **Protecting users against phishing attacks with AntiPhish** by Engin Kirda and Christopher Kruegel presents AntiPhish, a browser extension that aims to protect inexperienced users against spoofed website-based phishing attacks. AntiPhish keeps track of the sensitive information of a user and generates warnings whenever sensitive information is typed into a form on a website that is considered untrusted. The tool has been implemented as a Mozilla Firefox plug-in and is free for public use.

The strengths of the paper give an overview of the different types of phishing attacks and discuss a real-world example. It also describes the solution of AntiPhish, and provides details about its implementation, as well as presents an example that shows how AntiPhish mitigates a typical phishing attempt.

The weakness of the paper is that despite that AntiPhish is free for public use, the AntiPhish browser extension does not support other Web browsers such as Chrome, Brave, and Internet Explorer but is limited to only the Mozilla browser.

To address this weakness, a cross-platform plug-in is developed for any browser to ensure AntiPhish is not limited to a specific browser since it is free for public use.

In 2019, the paper titled **A Review Paper on Detection of Phishing Websites Using Machine Learning** by Ashritha Jain R, Chaithra Kulal, Deekshitha S, and Mrs. Mangala Kini proposed different algorithms (models) as well as different features of phishing attacks and techniques to detect phishing websites.

The strengths of the paper discuss works and different methods for phishing detection. It also introduces the proposed methodology that can be implemented to predict the phishing website accurately. The investigation gap provides more scope to study phishing detection.

The weakness of the paper is that it is limited to few feature selections of URL websites as well as four models for the prediction of phishing websites which might not produce the best accuracy of the model.

To address this weakness, several resources are sourced from updated published work to uncover different Machine learning models and features of a website to test and train the dataset for accuracy of the model. This will help find the best accuracy of the model for the detection of a phishing website.

In 2019, the paper titled **Phishing Websites Detection Using Machine Learning** by (Kiruthiga. R., & Akila, D.) Explained a novel approach to detect phishing websites using machine learning algorithms. They also compared the accuracy of five machine learning algorithms Decision Tree (DT), Random Forest (RF) (Shad & Sharma, 2018), Gradient Boosting (GBM), Generalized Linear Model (GLM), and Generalized Additive Model (GAM)(Shad & Sharma, 2018). Accuracy, Precision, and Recall evaluation methods were calculated for each algorithm and compared. Website attributes (30) are extracted with the help of Python and performance evaluation done with open-source programming language R. Top three algorithms namely Decision Tree, Random Forest, and GBM performance were compared in the table. From the tables of accuracy, recall, and performance, it is shown that the Random Forest algorithm has given the highest 98.4% accuracy, 98.59% recall, and 97.70% precision.

The strengths of the paper are that different authors proposed different approaches for feature selection algorithm, automatic feature extraction, and phishing detection model to detect phishing performance effectively by using different machine learning models.

In this paper, the authors Sönmez, Tuncer, Gökal, & Avci (2018) **proposes a classification model to classify phishing attacks**. This model comprises feature extraction from sites and the classification of websites. In feature extraction, 30 features have been taken from the UCI Irvine machine learning repository data set and phishing feature extraction rules have been clearly defined. To classification, these features, Support Vector Machine (SVM), Naïve Bayes (NB), and Extreme Learning Machine (ELM) (Sönmez et al., 2018) were used. In Extreme Learning Machine (ELM), six activation functions were used and achieved 95.34% accuracy than SVM and NB. The results were obtained with the help of MATLAB.

The Authors Peng, Harris, & Sawa, (2018) present an approach to **detect phishing email attacks using natural language processing and machine learning**. This is used to perform the semantic analysis of the text to detect malicious intent. A natural Language Processing (NLP) technique is used to parse each sentence and finds the semantic jobs of words in the sentence in connection to the predicate. In light of the job of each word in the sentence, this strategy recognizes whether the sentence is an inquiry or an order. Supervised machine learning (Peng et al., 2018) is used to generate the blacklist of malicious pairs. Authors defined the algorithm SEAHound (Peng et al., 2018) for detecting phishing emails and Net-craft Anti-Phishing Toolbar is used to verify the validity of a URL. This algorithm is implemented with Python scripts and the dataset Nazario phishing email set is used. Results of Net-craft and SEAHound (Peng et al., 2018) are compared and obtained precision 98% and 95% respectively.

The weakness of the paper is that it is limited to a few feature selections and extraction of URL websites.

To address this weakness, new techniques and features should be added or replaced to help detect new phishing URL attacks.

The Table below shows related algorithms proposed by several researchers in Machine Learning to detect phishing websites. On reviewing their papers, they concluded that most of the work done is by using familiar machine learning algorithms like Naïve Bayesian, SVM, Decision Tree, and Random Forest. Some authors proposed a new system like Phish Score and Phish Checker for detection. The combinations of features with regards to accuracy, precision, recall, etc. were used. Experimentally successful techniques in detecting phishing website URLs were summarized in Table 2.3. As phishing websites increase day by day, some features may be included or replaced with new ones to detect them.

Table 2.3: Outline of related algorithms used to detect phishing website

Algorithm used	Reference paper	No. of Features	Dataset	Language/Tools	Conclusion
Decision Tree (DT), Random Forest (RF), Gradient Boosting (GBM), Generalized Linear Model (GLM), Generalized Additive Model (GAM)	J. Shad and S. Sharma,	30	Not Mentioned	Python, R Language	Random Forest highest accuracy 98.4%
Support vector Machine (SVM), Naïve Bayes (NB) and Extreme Learning Machine(ELM)	Y. Sönmez, T. Tuncer, H. Gökal, and E. Avci,	30	UCI-Machine Learning Repository	MATLAB	ELM achieved 95.34% accuracy.
Natural Language Processing	T. Peng, I. Harris, and Y. Sawa,	-	Nazario Phishing Email set	Python	Proposed SEAHound provides 95% accuracy
Bayesian Network, Stochastic Descent (SGD), lazy. K. Star, Randomizable Filtered Classifier, Logistic model tree (LMT) and ID3 (iterative Dichotomiser), Multilayer	M. Karabatak and T. Mustafa	27	UCI-Machine Learning Repository	MATLAB, WEKA	Lazy. K.Star obtained 97.58% accuracy

Perception, JRip, PART, J48, Random Forest and Random Tree					
Random Forest	S. Parekh, D. Parikh, S. Kotak, and P. S. Sankhe	8	Phishtank,	RStudio	95% accuracy
Neural network model Adam, AdaDelta and SGD	K. Shima	URL length	Phishtank	Chainer	Accuracy of Adam 94.18%
Convolution neural network (CNN) and SNN long short-term memory (CNN-LSTM)	A. Vazhayil, R. Vinayakumar, and K. Soman,	-	Phishtank, OpenPhish, MalwareDomainlist, MalwareDomain	TensorFlow in conjunction with Keras	CNN-LSTM obtained 98% accuracy
Logistic Regression and Support Vector Machine (SVM)	W. Fadheel, M. Abusharkh, and I. Abdel-Qader,	19	UCI machine learning repository	Big Data	SVM accuracy 95.62%
AdaBoost, Bagging, Random Forest, and SMO	X. Zhang, Y. Zeng, X. Jin, Z. Yan, and G. Geng	11	DirectIndustry Anti-Phishing Alliance of China	Big Data	Only Semantic Features of word embedding obtained high accuracy.
C4.5 decision tree	L. MacHado and J. Gadge,	9 features and heuristic values	Phishtank Google	-	89.40%
KNN, SVM and Random Forest	A. Desai, J. Jatakia, R. Naik, and N. Raul	22	UCI-Machine Learning Repository	HTML, JavaScript, CSS, Python	Random Forest high accuracy

Naïve Bayes and Sequential Minimal Optimization (SMO)	M. Aydin and N. Baykal	133	Phishtank Google	C# programming and R programming WEKA	SMO Beat accuracy than NB
Heuristic feature root mean square Error (RMSE)	L. A. T. Nguyen, B. L. To, H. K. Nguyen, and M. H. Nguyen,	6	PhishTank	MYSQL. PHP	97%
PhishScore	S. Marchal, J. Francois, R. State,	12	PhishTank	-	94.91%
PhishChecker	A. A. Ahmed and N. A. Abdullah,	5	PhishTank and Yahoo directory set	Microsoft Visual Studio Express 2013 and C# language	96%

CHAPTER THREE

SYSTEM ANALYSIS AND DESIGN

3.1 Overview of System Analysis

This chapter describes the various process, methods, and procedures adopted by the researcher to achieve the set aim and objectives and the conceptual structure within which the research was conducted.

The methodology of any research work refers to the research approach adopted by the researcher to tackle the stated problem. Since the efficiency and maintainability of any application are solely dependent on how designs are prepared. This chapter provides detailed descriptions of methods employed to proffer solutions to the stated objectives of the research work.

According to the Merriam-Webster dictionary (11th.Ed), system analysis is "the process of studying a procedure or business to identify its goals and purposes and create systems and procedures that will efficiently achieve them". It is also the act, process, or profession of studying an activity (such as a procedure, a business, or a physiological function) typically by mathematical means to define its goals or purposes and to discover operations and procedures for accomplishing them most efficiently. System analysis is used in every field where the development of something is done. Before planning and development, you need to understand the old systems thoroughly and use the knowledge to determine how best your new system can work.

3.2 Analysis of Existing System

The existing system of phishing detection techniques suffers low detection accuracy and high false alarm especially when different phishing approaches are introduced. Above and beyond, the most common technique used is the blacklist-based method which is inefficient in responding to emanating phishing attacks since registering a new domain has become easier, no comprehensive blacklist can ensure a perfect up-to-date database for phishing detection.

3.3 Proposed System

The proposed phishing detection system utilizes machine learning models and deep neural networks. The system comprises two major parts, which are the machine learning models and a web application. These models consist of Decision Tree, Support Vector Machine, XGBooster, Multilayer Perceptions, Auto Encoder Neural Network, and Random Forest.

These models are selected after different comparison-based performances of multiple machine learning algorithms. Each of these models is trained and tested on a website content-based feature, extracted from both phishing and legitimate dataset.

Hence, the model with the highest accuracy is selected and integrated into a web application that will enable a user to predict if a URL link is phishing or legitimate.

3.3.1 Benefits of the new system

- i. Will be able to differentiate between phishing(0) and legitimate(1) URLs
- ii. It Will help reduce phishing data breaches for an organization
- iii. It Will be helpful to individuals and organizations
- iv. It is easy to use

3.4 Model Development

The model development method takes several models, tests them, and adds them to an iterative process until a model that meets the required requirements is developed. Figure 3.1 shows the steps used in the development of machine learning models using both supervised and unsupervised learning.

The following are the stages to machine learning model development for phishing detection systems:

i. Data Collection

The data used to generate the datasets on which the models are trained are gotten from different open-source platforms. The dataset collection consists of phishing and legitimate URL dataset.

The set of phishing URLs are collected from an open-source service called Phish Tank. This service provides a set of phishing URLs in multiple formats like CSV, JSON and so on that gets updated hourly. This dataset is accessible from the phishtank.com website. From this dataset, over 5000 random phishing URLs are collected to train the ML models.

The set of legitimate URLs are obtained from the open datasets of the University of New Brunswick. This dataset is accessible on the university website. This dataset has a collection of benign, spam, phishing, malware & defacement URLs. Out of all these types, the benign URL dataset is considered for this project. From this dataset, Over 5000 random legitimate URLs are collected to train the ML models.

ii. Preprocessing

Data preprocessing is the first and crucial step after data collection. The raw dataset obtained for phishing detection was prepared by removing redundant and irregular data and also encoded using the One-Hot Encoding technique into a useful and efficient format suitable for the machine learning model.

iii. Exploratory data analysis

Exploratory data analysis (EDA) technique was used on the dataset after series of data cleaning. The data visualization method was employed to analyze, explore and summarize the dataset. These visualization consist of heat-map, histograms, box plots, scatter plots, and pair plots to uncover patterns and insights within data.

iv. Feature Extraction

Feature Extraction aims to reduce the number of features in a dataset by creating new features from the existing ones. Thus, Website content-based features were extracted from phishing and legitimate datasets such as the Address bar-based feature which consists of 9 features, Domain-based feature which consists of 4 features, and HTML & JavaScript-based Feature which consists of 4 features. So, altogether 17 features were extracted for phishing detection.

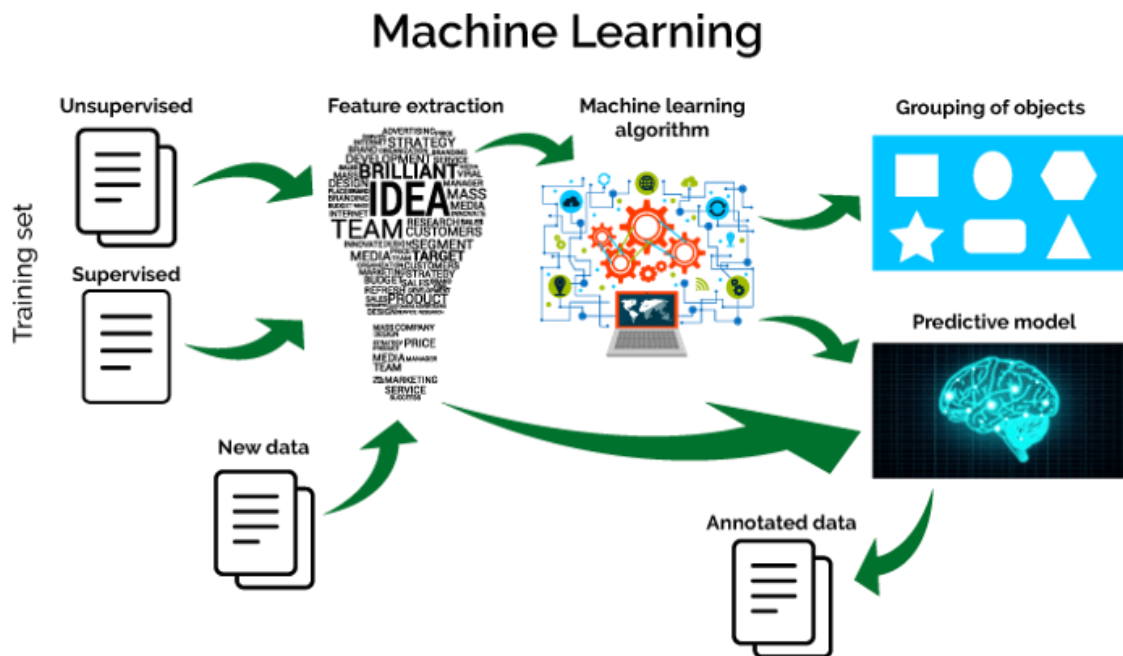


Figure 3.1 Machine Learning development process

(Source: Ayush, 2019)

v. Model Training

Model Training involves feeding Machine learning algorithms with data to help identify and learn good attributes of the dataset.

This research problem is a product of supervised learning, which falls under the classification problem. The algorithms used for phishing detection consist of supervised machine learning models (4) and deep neural network (2) which was used to train the dataset. These algorithms include Decision Tree, Random Forest, Support Vector Machines, XGBooster, Multilayer Perceptron, and Auto-encoder Neural Network. All these models were trained on the dataset. Thus, the dataset is spitted into a training and testing set. The training model consists of 80% of the dataset to enable the machine learning models to learn more about the data and be able to distinguish between phishing and legitimate URLs.

vi. Model Testing

Model Testing involves the process where the performance of a fully trained model is evaluated on a testing set.

Thus, after 80% of data has been trained, 20% of the dataset is used to evaluate the trained dataset to see the performance of the models.

vii. Model Evaluation

Model Evaluation involves estimating the generalization accuracy of models and deciding whether the model performs better or not.

Thus, Scikit-learn (sklearn metrics) module was used to implement several score and utility functions to measure the classification performance to properly evaluate the models deployed for phishing detection.

3.5 System Modelling

System modeling involves the process of developing an abstract model of a system, with each model presenting a different view or perspective of the system. It is the process of representing a system using various graphical notations that shows how users will interact with the system and how certain parts of the system function. The proposed system was modeled using the following diagrams:

- i. Architecture diagram
- ii. Use case diagram
- iii. Flowcharts

The proposed system will be implemented using Python Programming language along with different machine learning models and libraries such as pandas, scikit-learn, python who-is, beautiful-Soup, NumPy, seaborn, and matplotlib. Etc.

3.5.1 System architecture

Architectural design is concerned with understanding how a system should be organized and designing the overall structure of that system, it shows how different components of the system work together to achieve its main objectives. It is the process for identifying sub-systems making up a system and the framework for sub-system control and communication. The diagram below represents a graphical overview of the architectural design of the proposed system.

Figure 3.1 shows the architecture view of the proposed phishing detection system such that a user enters a URL link and the link moves through different trained machine learning and deep neural network models and the best model with the highest accuracy is selected. Thus, the selected model is deployed as an API (Application Programming Interface) which is then integrated into a web application. Hence, a user interacts with the web application which is accessible across different display devices such as computers, tablets, and mobile devices.

3.5.2 Use a case diagram of the system

The Use Case diagram describes the functionality of the system as designed from the requirements, it summarizes the details of a system and the users within the system. It is a behavior diagram and visualizes the observable interactions between actors and the system under development. The Use case diagram consists of the system, the related use cases, and actors and relates to each other.

Figure 3.2 shows the Use case scenarios that a user can carry out on the phishing detection system.

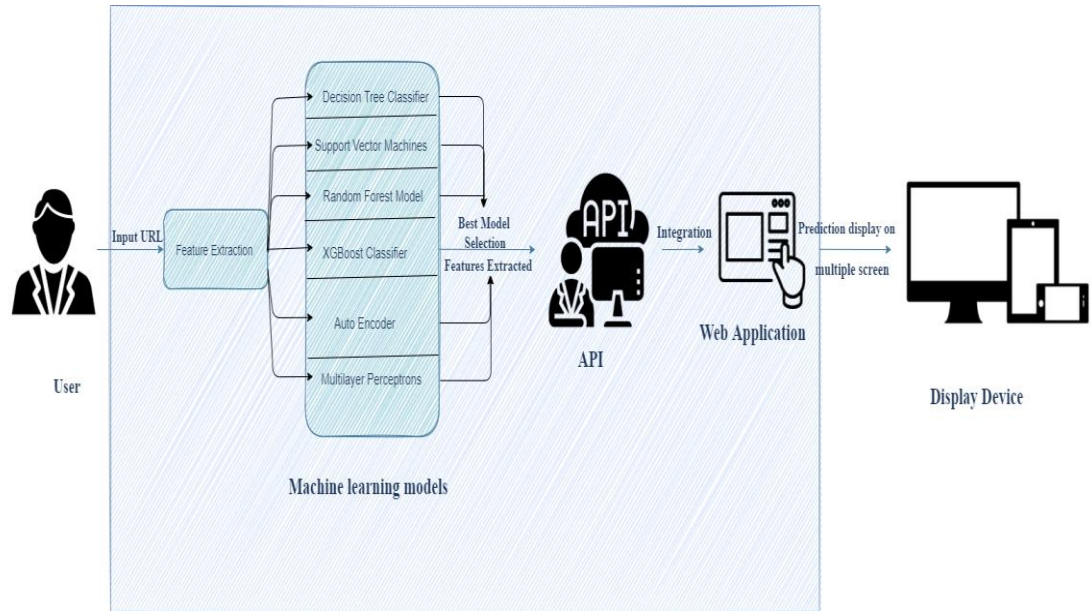


Figure 3.2: Architectural Design of the Proposed System

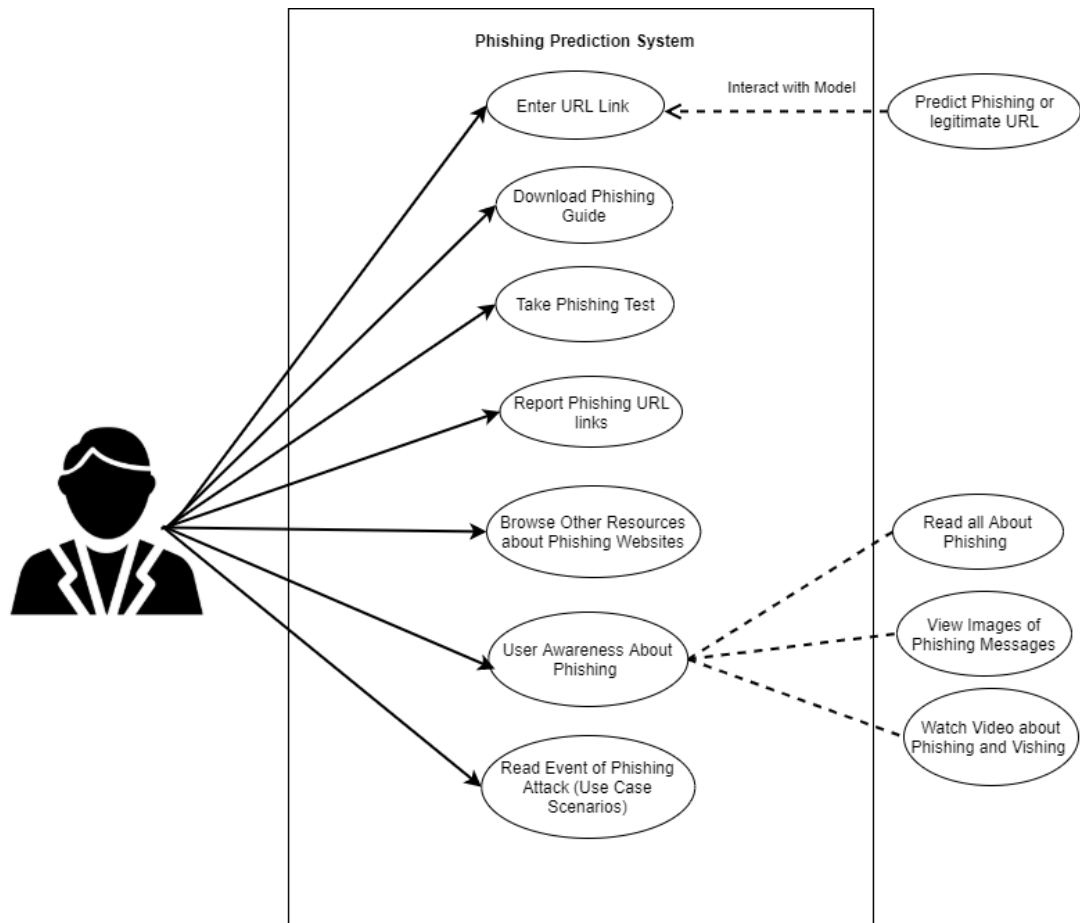


Figure 3.3 Use Case diagram for Proposed System

3.5.3 Flowchart of the system

A flowchart is a diagram that depicts a process, system, or computer algorithm. It is a graphical representation of the steps that are to be performed in a system, it shows the steps in sequential order. It is used in presenting the flow of algorithms and to communicate complex processes in clear, easy-to-understand diagrams.

Figure 3.3 shows the flow of phishing detection systems using the machine learning process.

Figure 3.4 shows the phishing detection web interface system. The user inputs a URL link and the website validates the format of the URL and then predicts if the link is phishing or legitimate.

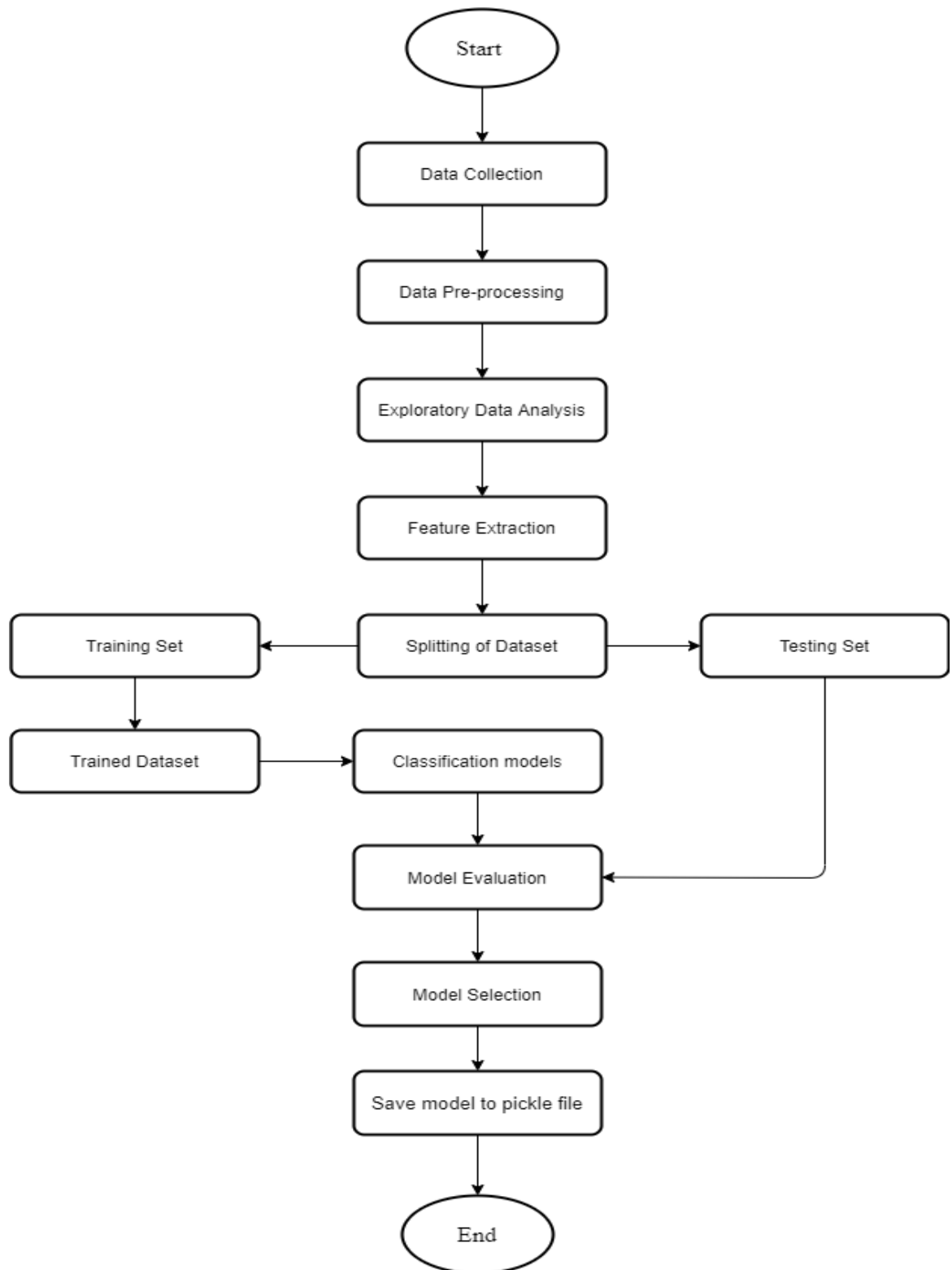


Figure 3.4 Flowchart of the proposed System

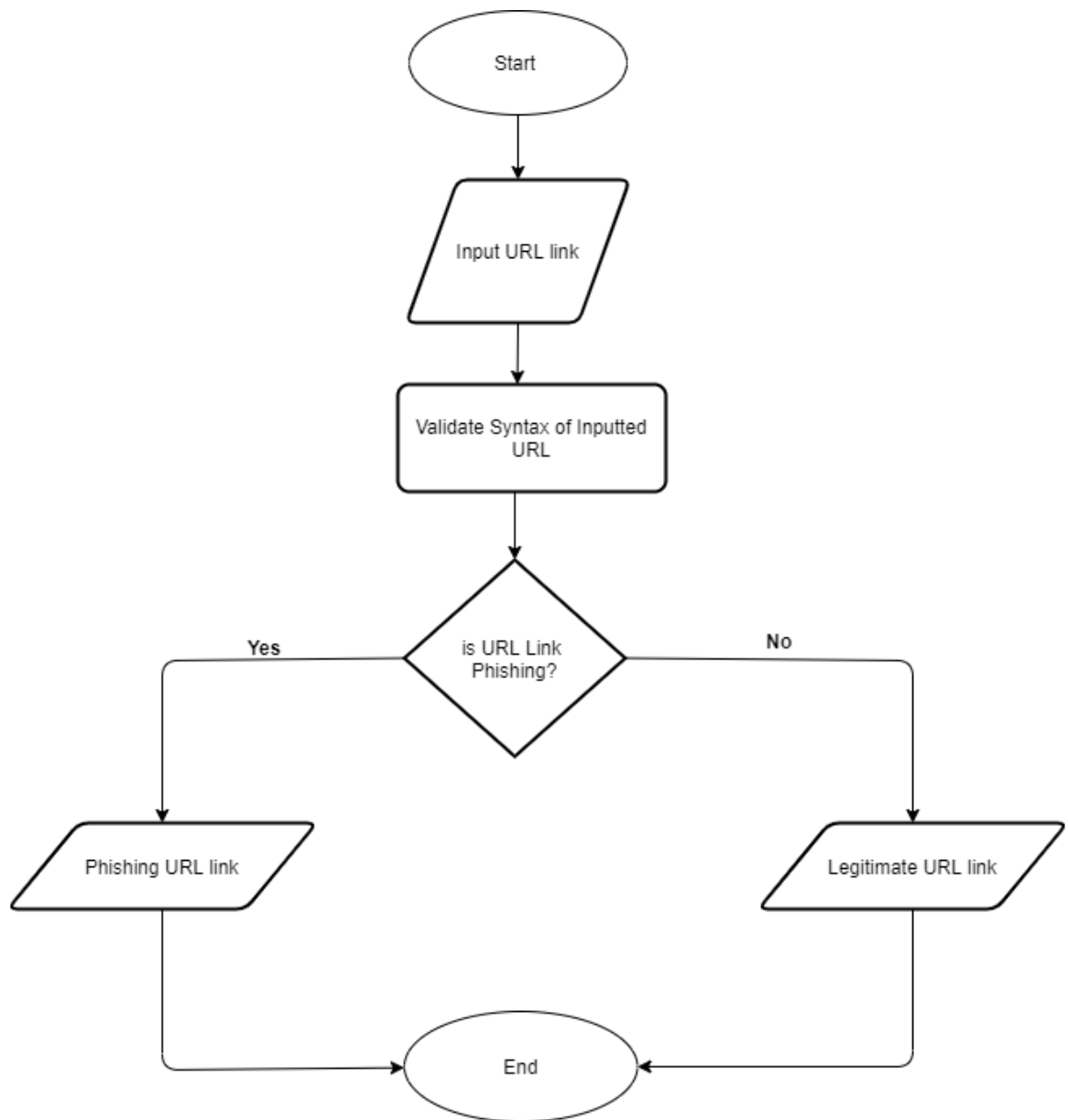


Figure 3.5 Flowchart of the web interface

CHAPTER FOUR

SYSTEM IMPLEMENTATION AND RESULTS

This chapter deals with the implementation of multiple machine learning models for the detection of underlying diseases and illnesses as earlier designed in the preceding chapters.

The implementation is concerned with all the activities that took place to put up the newly developed system into operation (using the approach that was stated in the methodology (e.g. architectural diagram, flowchart, uses case, etc.)) to achieve the objectives of the project to convert the theoretical design into a working system. The components of the system were also tested and evaluated.

4.1 Installation Requirements

The hardware (physical components of a computer system that can be seen, touched, or felt) and software (both system software and the application software installed and used in the system development) tools needed to satisfy these objectives highlighted below:

4.1.1 Hardware requirements

The hardware requirement includes:

- i. A laptop or desktop computer (Preferably 64bit)
- ii. Random Access Memory (RAM): 8 Gigabytes Minimum
- iii. Processor: Intel Core i5, 2.4 GHz Minimum

4.1.2 Software requirements

The software requirements for the development of this system include:

- i. Windows Operating System (8/10)
- ii. Anaconda Navigator (Jupyter Notebook)
- iii. PyCharm Community edition
- iv. Web browser (Preferably Chrome)
- v. Visual Studio Code
- vi. Postman

4.2 Model Development

The model for detecting phishing URL websites was built using a python programming language with over six (6) machine learning models and deep neural network algorithms altogether and the most accurate test score on the tested 5,000 datasets were used.

4.2.1 Data collection

The dataset used for the classification was sourced from was gotten from multiple sources listed in the earlier stated methodology.

The dataset used for classifying the dataset into phishing and legitimate URLs was sourced from open source websites, samples of which are shown below in figure 4.1 and 4.2 respectively.

	A	B	C	D	E	F	G	H	I	J
1	http://1337x.to/torrent/1048648/American-Sniper-2014-MD-ITALIAN-DVDSCR-X264-BST-MT/									
2	http://1337x.to/torrent/1110018/Blackhat-2015-RUSSIAN-720p-WEB-DL-DD5-1-H264-RUFGT/									
3	http://1337x.to/torrent/1122940/Blackhat-2015-x264-1080p-WEB-DL-eng-nl-sub-sharky/									
4	http://1337x.to/torrent/1124395/Fast-and-Furious-7-2015-HD-TS-XVID-AC3-HQ-Hive-CM8/									
5	http://1337x.to/torrent/1145504/Avengers-Age-of-Ultron-2015-CAM-New-Audio-x264-CPG/									
6	http://1337x.to/torrent/1160078/Avengers-age-of-Ultron-2015-HQ-CAM-H264-AC3-MURD3R/									
7	http://1337x.to/torrent/294349/American-Idol-S11E04-Auditions-4-HDTV-XviD-FQM-ettv/									
8	http://189.cn/dqmh/userCenter/myOrderInfoList.do?method=listMyOrderInfo_new&isVs=no									
9	http://2gis.ru/moscow/search/%D0%9F%D0%BE%D0%B5%D1%81%D1%82%D1%8C/tab/firms/zoom/11									
10	http://abc.go.com/shows/general-hospital/episode-guide/2015-05/08-friday-may-8-2015									
11	http://abc.go.com/shows/the-muppets/video/new-abc-comedy-trailers?cid=abchp_muppets									
12	http://abcnews.go.com/US/wireStory/regulators-delays-georgia-nuclear-plant-31020059									
13	http://adultfriendfinder.com/css/live_cd/ffadult/english/0/font_face-1427390957.css									
14	http://akhbarelyom.com/news/newdetails/410322/1/%D8%A8%D9%88%D8%B6%D9%88%D8%AD.html									
15	http://allegro.pl/amadeus-quartet-haydn-string-quartets-collectors-i5207998383.html									
16	http://allegro.pl/narzedzia-i-sprzet-warsztatowy-18554?ref=simplified-category-tree									
17	http://allegro.pl/royal-string-quartet-music-for-string-quartet-cd-i5262876831.html									
18	http://allegro.pl/sexy-g-string-stringi-z-koralikami-must-have-uni-i5039087215.html									
19	http://allegro.pl/sporty-strzeleckie-i-myslistwo-13495?ref=simplified-category-tree									
20	http://allegro.pl/sporty-towarzyskie-i-rekreacja-13408?ref=simplified-category-tree									
21	http://allegro.pl/triumph-miss-sexy-allday-string-stringi-36-s-bez-i4855636396.html									
22	http://allegro.pl/triumph-stringi-exquisite-essence-string-stal-38-i5073330901.html									
23	http://allegro.pl/wyszczuplajace-body-string-pod-biust-jony-srebro-i5124700240.html									

Figure 4.1 Dataset of Phishing URLs

Source: The Dataset is collected from an open-source service called Phish-Tank. This dataset consists of 5,000 random phishing URLs which are collected to train the ML models.

	A	B	C	D	E	F	G	H	I	J
1	phish_id	url	phish_det	submissio	verified	verificatio	online	target		
2	6911546	https://ja	http://ww	2021-01-0	yes	2021-01-0	yes	Other		
3	6911545	http://po	http://ww	2021-01-0	yes	2021-01-0	yes	Other		
4	6911536	https://sp	http://ww	2021-01-0	yes	2021-01-0	yes	Other		
5	6911494	https://hy	http://ww	2021-01-0	yes	2021-01-0	yes	Other		
6	6911483	http://sto	http://ww	2021-01-0	yes	2021-01-0	yes	Other		
7	6911482	https://oc	http://ww	2021-01-0	yes	2021-01-0	yes	Other		
8	6911481	https://tr	http://ww	2021-01-0	yes	2021-01-0	yes	Other		
9	6911480	https://jo	http://ww	2021-01-0	yes	2021-01-0	yes	Other		
10	6911467	https://su	http://ww	2021-01-0	yes	2021-01-0	yes	Other		
11	6911466	https://cr	http://ww	2021-01-0	yes	2021-01-0	yes	Other		
12	6911465	http://est	http://ww	2021-01-0	yes	2021-01-0	yes	Other		
13	6911424	https://ek	http://ww	2021-01-0	yes	2021-01-0	yes	eBay, Inc.		
14	6911414	https://is	http://ww	2021-01-0	yes	2021-01-0	yes	Development Bank of Singa		
15	6911408	http://wir	http://ww	2021-01-0	yes	2021-01-0	yes	Other		
16	6911403	http://ma	http://ww	2021-01-0	yes	2021-01-0	yes	ABSA Bank		
17	6911400	http://ww	http://ww	2021-01-0	yes	2021-01-0	yes	Other		
18	6911398	https://bi	http://ww	2021-01-0	yes	2021-01-0	yes	Other		
19	6911395	https://fg	http://ww	2021-01-0	yes	2021-01-0	yes	Other		
20	6911394	http://fgh	http://ww	2021-01-0	yes	2021-01-0	yes	Other		
21	6911389	https://w	http://ww	2021-01-0	yes	2021-01-0	yes	Other		
22	6911388	https://w	http://ww	2021-01-0	yes	2021-01-0	yes	Other		
23	6911382	http://clfi	http://ww	2021-01-0	yes	2021-01-0	yes	Other		

Figure 4.2 Dataset of Legitimate URLs

Source: The Dataset were obtained from the open datasets of the University of New Brunswick, The dataset consists of collections of benign, spam, phishing, malware & defacement URLs. Out of all these types, the benign URL dataset is considered for this project. This dataset consists of 5,000 random legitimate URLs which are collected to train the ML models.

4.2.2 Feature extraction on the datasets

The features extraction used on the dataset are categorized into

- i. Address bar based features
- ii. Domain-based features
- iii. Html & java-script based features

In figure 4.3, figure 4.4, and figure 4.5 the images show the list of code feature extraction done on the dataset while figure 4.6 shows the code computation for all the feature extraction used on the dataset.

3.1. Address Bar Based Features:

Many features can be extracted that can be considered as address bar base features. Out of them, below mentioned were considered for this project.

- Domain of URL
- IP Address in URL
- "@" Symbol in URL
- Length of URL
- Depth of URL
- Redirection "/" in URL
- "http/https" in Domain name
- Using URL Shortening Services "TinyURL"
- Prefix or Suffix "." in Domain

Each of these features are explained and the coded below:

```
In [12]: # importing required packages for this section
from urllib.parse import urlparse,urllencode
import ipaddress
import re
```

3.1.1. Domain of the URL

Here, we are just extracting the domain present in the URL. This feature doesn't have much significance in the training. May even be dropped while training the model.

```
In [13]: # 1.Domain of the URL (Domain)
def getDomain(url):
    domain = urlparse(url).netloc
    if re.match(r"~www.",domain):
        domain = domain.replace("www.", "")
    return domain
```

```
In [14]: # 2.Checks for IP address in URL (Have_IP)
def havingIP(url):
    try:
        ipaddress.ip_address(url)
        ip = 1
    except:
        ip = 0
    return ip
```

3.1.3. "@" Symbol in URL

Checks for the presence of '@' symbol in the URL. Using '@' symbol in the URL leads the browser to ignore everything preceding the '@' symbol and the real address often follows the '@' symbol.

If the URL has '@' symbol, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

```
In [15]: # 3.Checks the presence of @ in URL (Have_At)
def haveAtSign(url):
    if "@" in url:
        at = 1
    else:
        at = 0
    return at
```

3.1.4. Length of URL

Computes the length of the URL. Phishers can use long URL to hide the doubtful part in the address bar. In this project, if the length of the URL is greater than or equal 54 characters then the URL classified as phishing otherwise legitimate.

If the length of URL ≥ 54 , the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

```
In [16]: # 4.Finding the Length of URL and categorizing (URL_Length)
def getLength(url):
    if len(url) < 54:
        length = 0
    else:
        length = 1
    return length
```

Figure 4.3: Code for Address bar based feature extraction

3.2. Domain Based Features:

Many features can be extracted that come under this category. Out of them, below mentioned were considered for this project.

- DNS Record
- Website Traffic
- Age of Domain
- End Period of Domain

Each of these features are explained and the coded below.

```
In [23]: !pip install python-whois
Requirement already satisfied: python-whois in c:\users\goodness\anaconda3\lib\site-packages (0.7.3)
Requirement already satisfied: future in c:\users\goodness\anaconda3\lib\site-packages (from python-whois) (0.18.2)
```

```
In [24]: # importing required packages for this section
import re
from bs4 import BeautifulSoup
import whois
import urllib
import urllib.request
from datetime import datetime
```

```
In [26]: # 12.Web traffic (Web_Traffic)
def web_traffic(url):
    try:
        #Filling the whitespaces in the URL if any
        url = urllib.parse.quote(url)
        rank = BeautifulSoup(urllib.request.urlopen("http://data.alex.com/data?cli=10&dat=s&url=" + url).read(), "xml").find(
            "REACH")['RANK']
        rank = int(rank)
    except TypeError:
        return 1
    if rank < 100000:
        return 1
    else:
        return 0
```

3.2.3. Age of Domain

This feature can be extracted from WHOIS database. Most phishing websites live for a short period of time. The minimum age of the legitimate domain is considered to be 12 months for this project. Age here is nothing but different between creation and expiration time.

If age of domain > 12 months, the value of this feature is 1 (phishing) else 0 (legitimate).

```
In [27]: # 13.Survival time of domain: The difference between termination time and creation time (Domain_Age)
def domainAge(domain_name):
    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 1
    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 1
    else:
        ageofdomain = abs((expiration_date - creation_date).days)
        if ((ageofdomain/30) < 6):
            age = 1
        else:
            age = 0
    return age
```

Figure 4.4: Code for domain-based features extraction

3.3. HTML and JavaScript based Features

Many features can be extracted that come under this category. Out of them, below mentioned were considered for this project.

- IFrame Redirection
- Status Bar Customization
- Disabling Right Click
- Website Forwarding

Each of these features are explained and the coded below.

```
In [29]: # importing required packages for this section
import requests
```

3.3.1. IFrame Redirection

IFrame is an HTML tag used to display an additional webpage into one that is currently shown. Phishers can make use of the "iframe" tag and make it invisible i.e. without frame borders. In this regard, phishers make use of the "frameBorder" attribute which causes the browser to render a visual delineation.

If the iframe is empty or response is not found then, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

```
In [30]: # 15. IFrame Redirection (iFrame)
def iframe(response):
    if response == "":
        return 1
    else:
        if re.findall(r"<iframe>|<frameBorder>", response.text):
            return 0
        else:
            return 1
```

3.3.2. Status Bar Customization

Phishers may use JavaScript to show a fake URL in the status bar to users. To extract this feature, we must dig-out the webpage source code, particularly the "onMouseOver" event, and check if it makes any changes on the status bar

If the response is empty or onmouseover is found then, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

```
In [31]: # 16. Checks the effect of mouse over on status bar (Mouse_Over)
def mouseOver(response):
    if response == "":
        return 1
    else:
        if re.findall("<script>.+onmouseover.+</script>", response.text):
            return 1
        else:
            return 0
```

3.3.3. Disabling Right Click

Phishers use JavaScript to disable the right-click function, so that users cannot view and save the webpage source code. This feature is treated exactly as "Using onmouseover to hide the Link". Nonetheless, for this feature, we will search for event "event.button==2" in the webpage source code and check if the right click is disabled.

If the response is empty or onmouseover is not found then, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

```
In [32]: # 17. Checks the status of the right click attribute (Right_Click)
def rightClick(response):
    if response == "":
        return 1
    else:
        if re.findall(r"event.button ?== ?2", response.text):
            return 0
        else:
            return 1
```

Figure 4.5: Code for Html & java-script based features extraction

```

In [40]: #Function to extract features
# There are 17 features extracted from the dataset
def featureExtractions(url):

    features = []
    #Address bar based features (9)
    features.append(getDomain(url))
    features.append(havingIP(url))
    features.append(haveAtSign(url))
    features.append(getLength(url))
    features.append(getDepth(url))
    features.append(redirection(url))
    features.append(httpDomain(url))
    features.append(prefixSuffix(url))
    features.append(tinyURL(url))

    #Domain based features (4)
    dns = 0
    try:
        domain_name = whois.whois(urlparse(url).netloc)
    except:
        dns = 1

    features.append(dns)
    features.append(web_traffic(url))
    features.append(1 if dns == 1 else domainAge(domain_name))
    features.append(1 if dns == 1 else domainEnd(domain_name))

    # HTML & Javascript based features (4)
    try:
        response = requests.get(url)
    except:
        response = ""
    features.append(iframe(response))
    features.append(mouseOver(response))
    features.append(rightClick(response))
    features.append(forwarding(response))
    # features.append(Label)

    return features

featureExtractions('http://www.facebook.com/home/service')

Out[40]: ['facebook.com', 0, 0, 0, 2, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0]

```

Figure 4.6: Code computation for all the feature extraction used dataset.

4.2.2 Data Analysis & Visualization

The image as shown in figure 4.7 shows the distribution plot of how legitimate and phishing datasets are distributed base on the features selected and how they are related to each other.

In figure 4.8 shows the plot of a correlation heat-map of the dataset. The plot shows correlation between different variables in the dataset.

In figure 4.9 and figure 4.10, it shows the feature importance in the model for Decision tree classifier and Random forest classifier respectively.

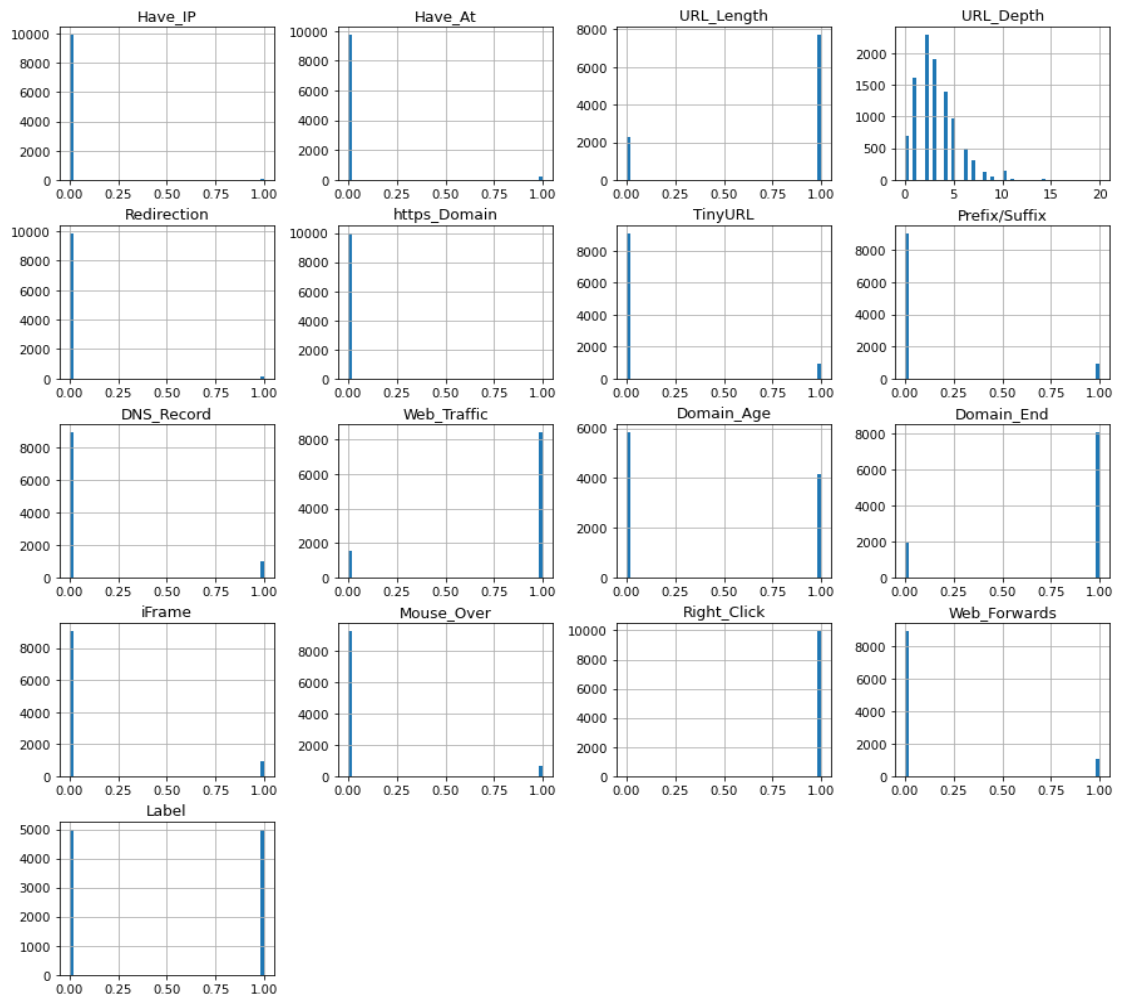


Figure 4.7: Distribution plot of dataset base on the features selected

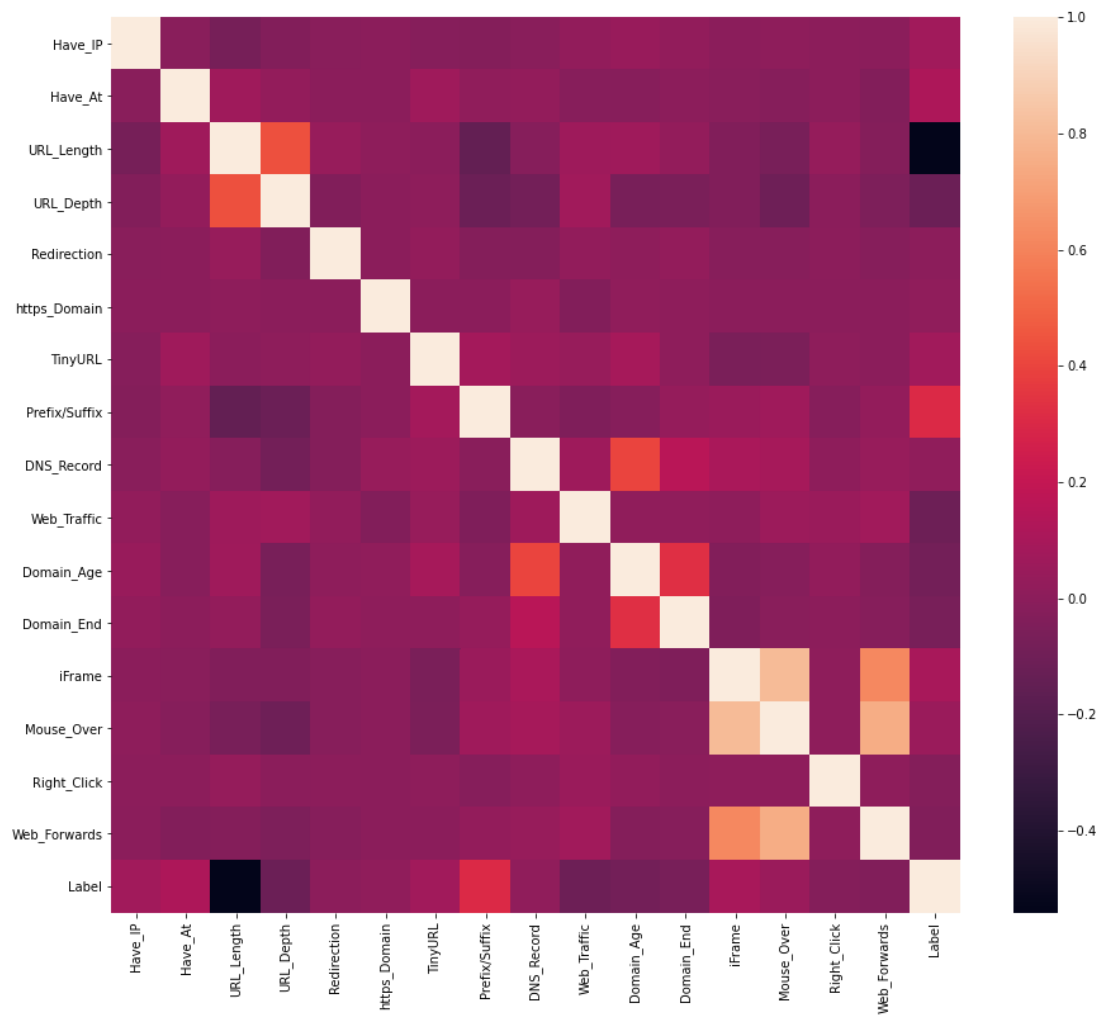


Figure 4.8: Correlation heat map of the dataset

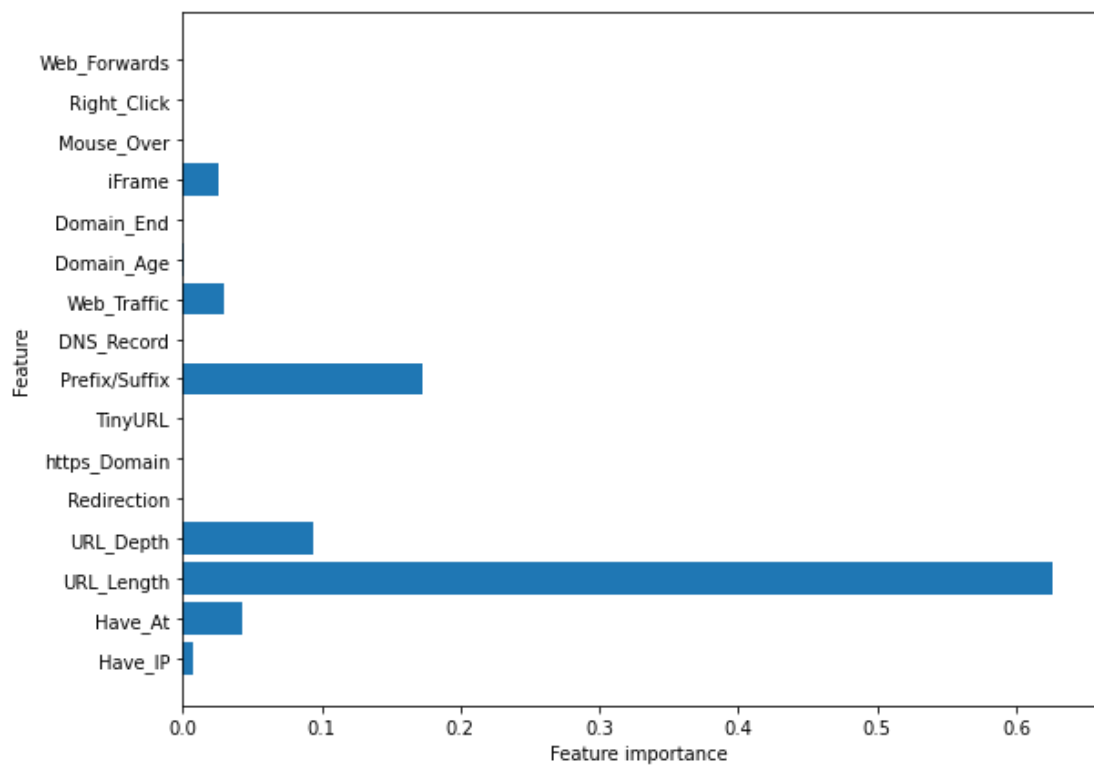


Figure 4.9: Feature importance for Decision Tree classifier

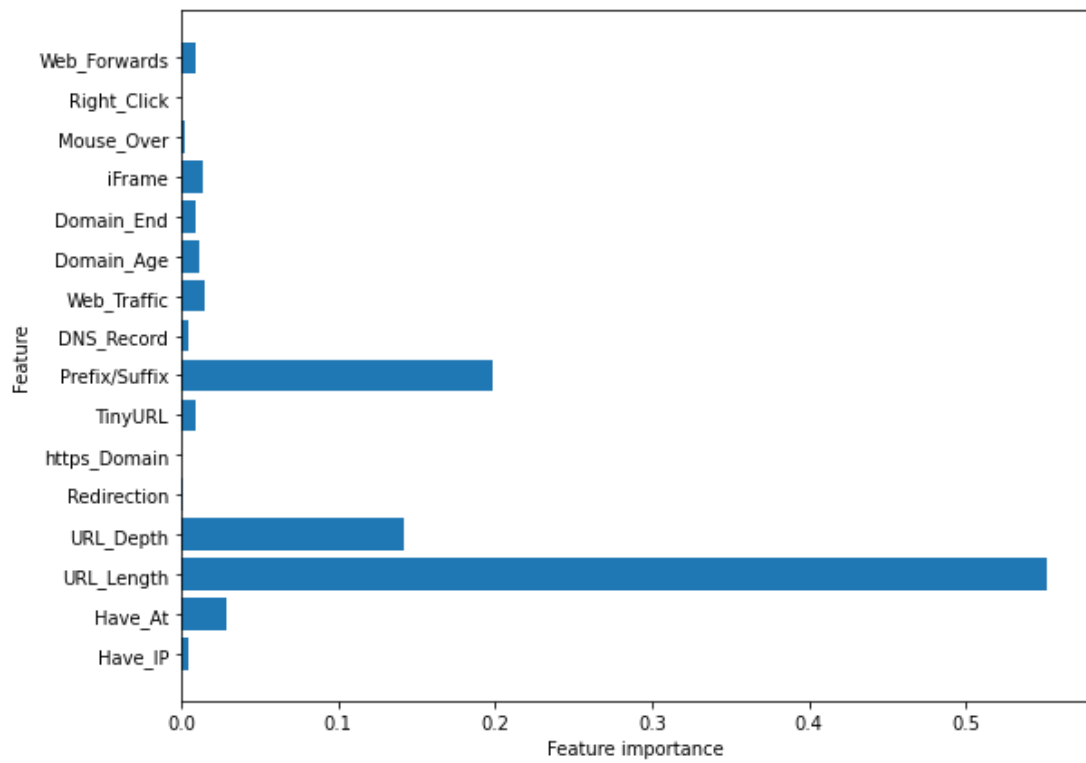


Figure 4.10: Feature importance for Random forest classifier

4.2.3 Data pre-processing

The datasets were first cleaned to remove empty entries and fill some entries by applying data pre-processing techniques and transform the data to use in the models. Figure 4.11 shows the summary of the dataset while figure 4.12 shows the number of missing values in the dataset which all appear to be zero.

In [8]: `tuna.describe()`

Out[8]:

	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	TinyURL	Prefix/Suffix	DNS_Record	Web_Traffic	Do
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	100
mean	0.005500	0.022600	0.773400	3.072000	0.013500	0.000200	0.090300	0.093200	0.100800	0.845700	
std	0.073961	0.148632	0.418653	2.128631	0.115408	0.014141	0.286625	0.290727	0.301079	0.361254	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	1.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	
50%	0.000000	0.000000	1.000000	3.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	
75%	0.000000	0.000000	1.000000	4.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	
max	1.000000	1.000000	1.000000	20.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	

Figure 4.11: Summary of the dataset

```
In [10]: #checking the data for null or missing values  
dfsa.isnull().sum()
```

```
Out[10]: Have_IP      0  
Have_At      0  
URL_Length   0  
URL_Depth    0  
Redirection  0  
https_Domain 0  
TinyURL      0  
Prefix/Suffix 0  
DNS_Record   0  
Web_Traffic  0  
Domain_Age   0  
Domain_End   0  
iFrame       0  
Mouse_Over   0  
Right_Click  0  
Web_Forwards 0  
Label        0  
dtype: int64
```

Figure 4.12: Number of missing values in the dataset

4.2.4 Phishing detection model

The based methodology stated that the proposed system utilizes machine learning models and deep neural networks. These models consist of Decision Tree, Support Vector Machine, XGBooster, Multilayer Perceptions, Auto Encoder Neural Network, and Random Forest.

The models determine whether a website URL is phishing or legitimate. The models help give a 2-class prediction (legitimate (0) and phishing (1)). In the model development process, over six (6) machine learning models and deep neural network algorithms all together were used to detect phishing URLs using Jupyter notebook IDE with packages such as pandas, Beautiful Soup, who-is, urllib, etc.

Here are the models, their accuracy was tested using sklearn matrices with an accuracy score and their matrices are shown in figure 4.13. The XGBooster model had the highest performance score of 86.6%, the Multilayer Perceptions model had an accuracy of 86.5%, the Decision Tree model had an accuracy of 81.4%, the Random Forest model had an accuracy of 81.8%, the Support Vector Machine model had an accuracy of 80.4%, and the Auto Encoder Neural Network model had an accuracy of 16.1%.

```

49]: #Sorting the dataframe on accuracy
      results.sort_values(by=['Test Accuracy', 'Train Accuracy'], ascending=False)

```

	ML Model	Train Accuracy	Test Accuracy
3	XGBoost	0.866	0.864
2	Multilayer Perceptrons	0.865	0.864
0	Decision Tree	0.814	0.812
1	Random Forest	0.818	0.811
5	SVM	0.804	0.794
4	AutoEncoder	0.161	0.177

For the above comparison, it is clear that the XGBoost Classifier works well with this dataset.

So, saving the model for future use.

Figure 4.13: Accuracy performance of models

4.3 General Working of The System

A one-page phishing detection web application called “Phish-Buster” has been developed to run on any browser. The application was developed using programming languages such as HTML, CSS, PHP, and JavaScript.

The phishing detection web application has the following pages:

4.3.1 The home page

The home page contains a session for a user to enter a URL and predict if it is phishing or legitimate. It predicts the state of the URL base on the feature selection as shown in figure 4.14.

The purpose of this page is to help its users validate a URL link and also provide various resources on phishing attacks. The User can also take a google phishing test to help understand how to detect phishing messages and URLs. Also, users can download a book that contains information and other resources on phishing.

4.3.2 The about page

The about page contains details about the application and also information about the author of the project as shown in figure 4.15

4.3.3 The use case page

The use case page contains different case scenarios of phishing attacks that happened to various companies in previous years back. Also, it contains images of phishing attacks messages sent by phishers as shown in figure 4.16

4.3.4 Resource page

The resource page it contains different resources regarding phishing such as definition, types, and techniques of a phishing attack as well as reference links to the source in which the content where retrieved from Also, it contains two (2) sub-session link: the first session reports phishing case and the second session consists of a phishing website.

The First Session: The Report phishing Case link redirects users to submit any case of phishing attack be it URL or a phishing email to Google Support and Google safe browsing.

The Second Session: Phishing website link redirects users to two (2) websites that contain resourceful information and phishing test. These sites consist of phishing box and intradyn as shown in Figure 4.17.

4.3.5 Web application Source Code

As shown from figure 4.18, consists of pages of source code of the web application running on visual studio code, other source codes are shown in Appendix A

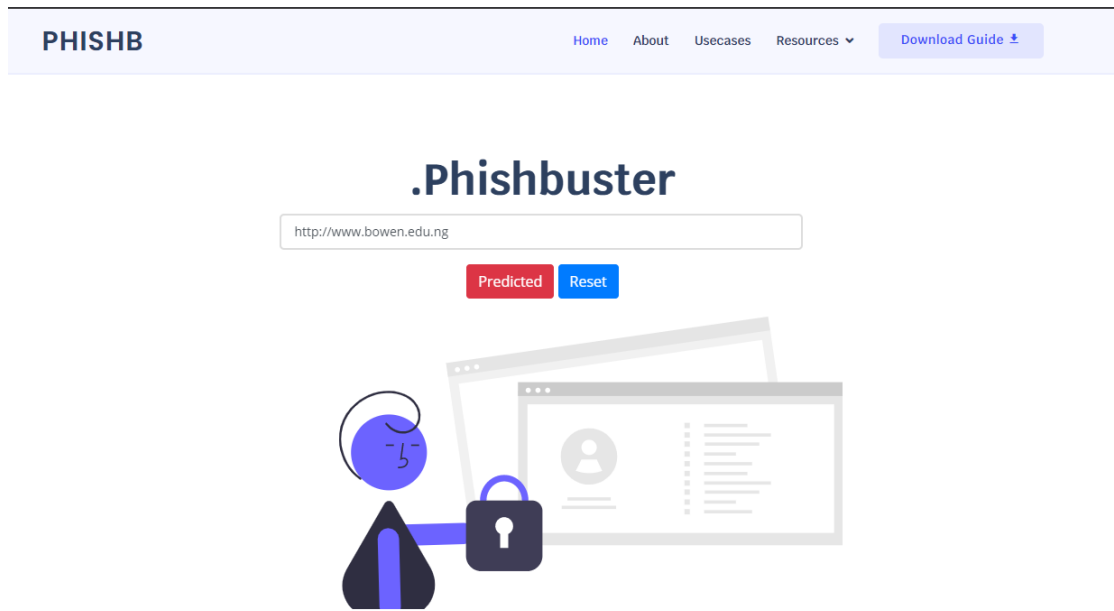


Figure 4.14 (a): The Home page

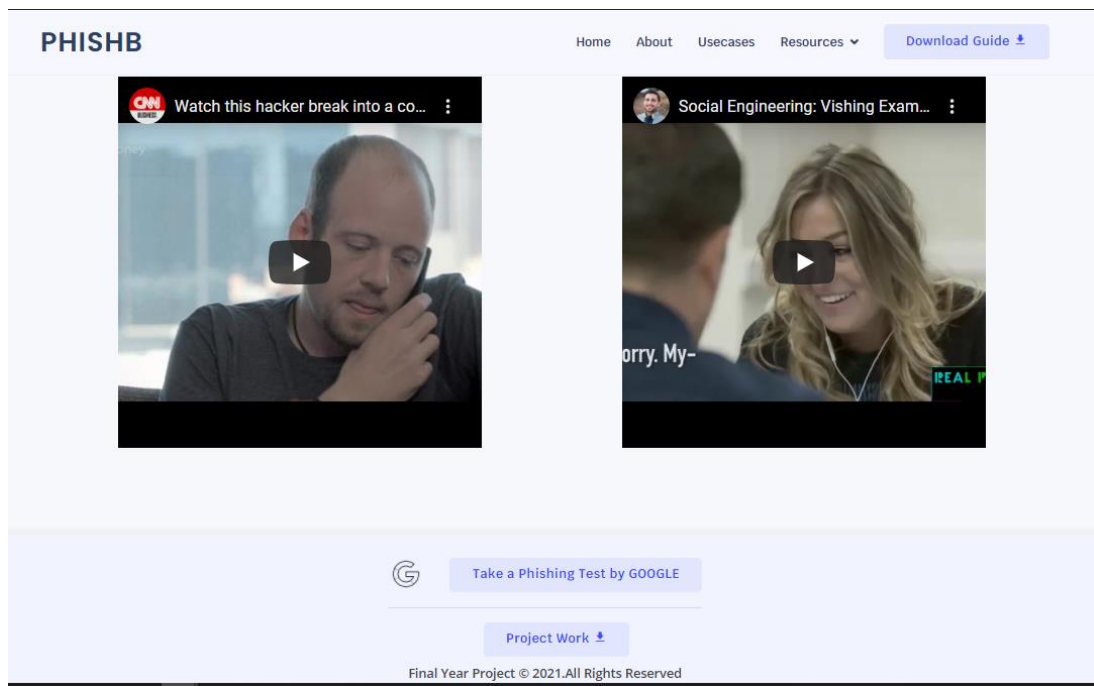


Figure 4.14 (b): The home page footer

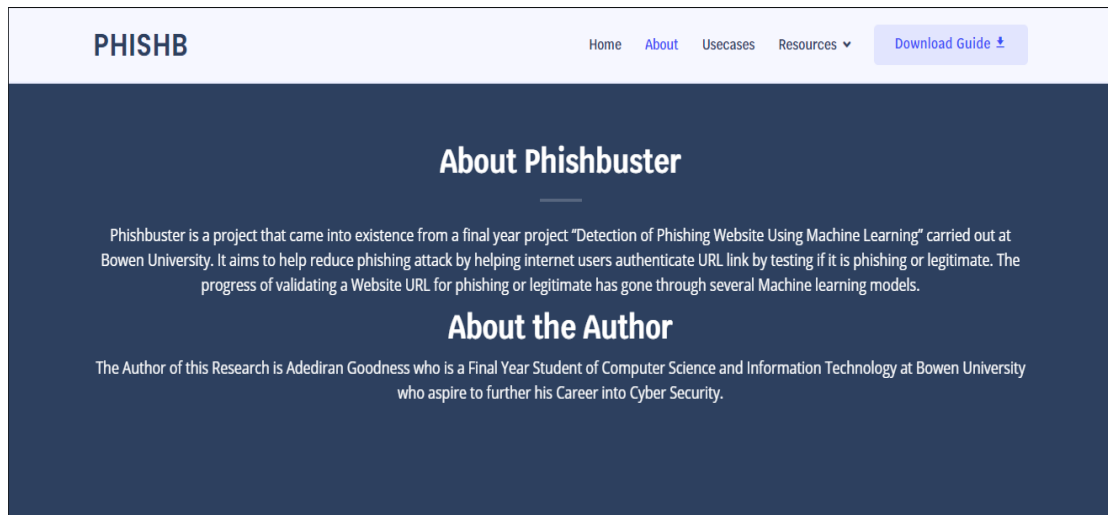


Figure 4.15: The About page



Figure 4.16 (a): The Use-case page

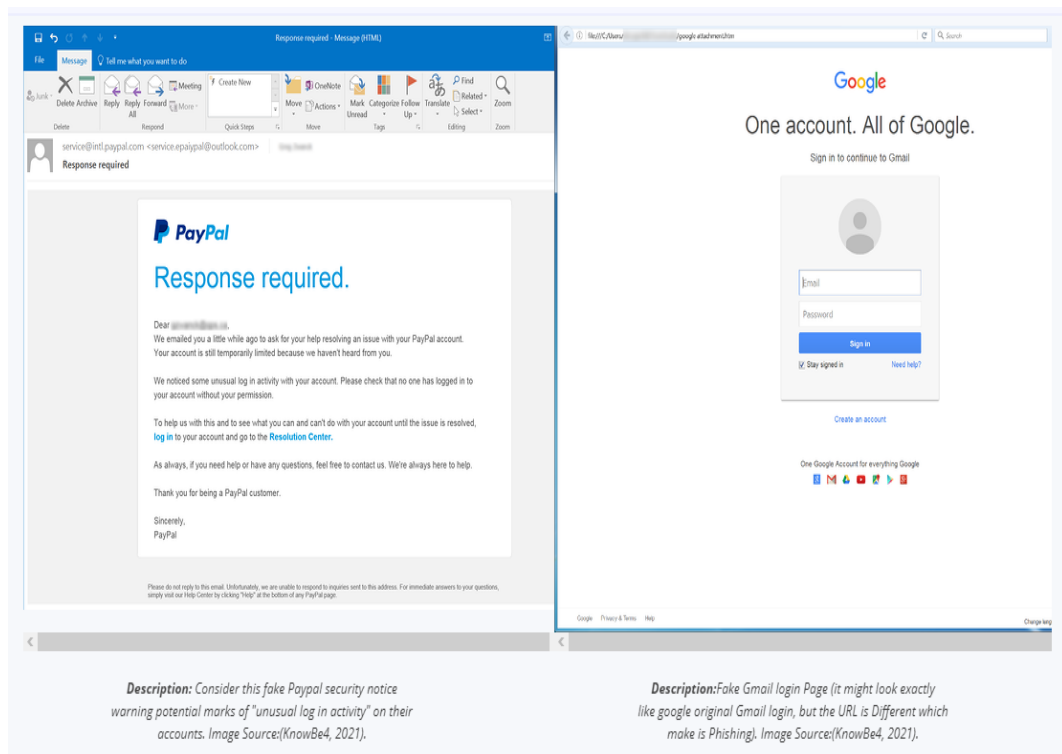


Figure 4.16 (b): The Use-case page

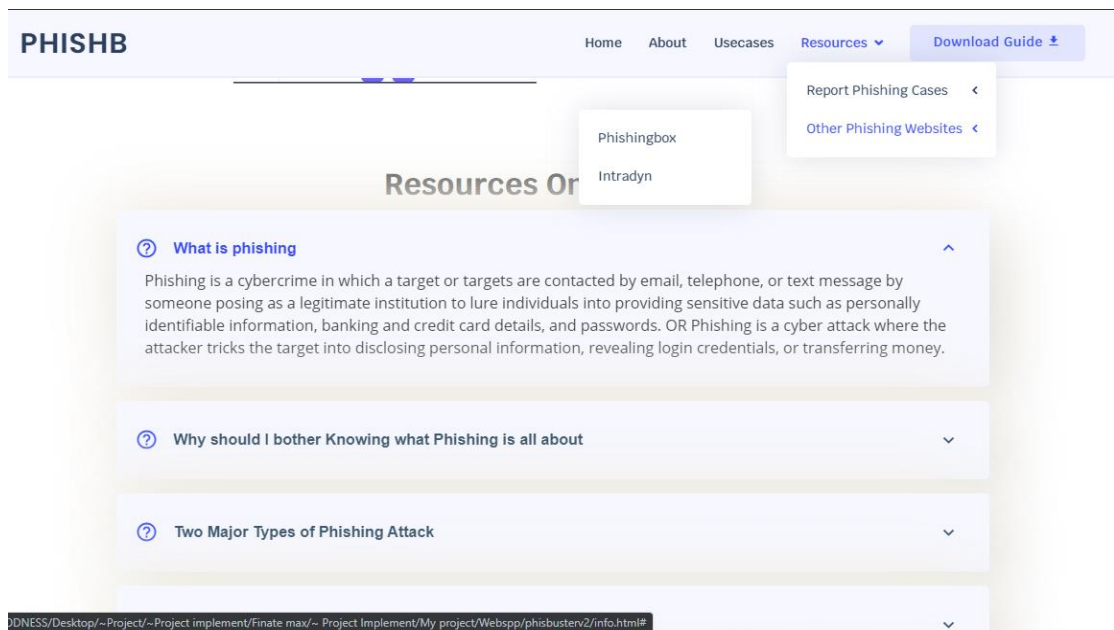


Figure 4.17: The Resource page

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="utf-8">
6   <meta content="width=device-width, initial-scale=1.0" name="viewport">
7
8   <title>Phishbuster</title>
9   <meta content="" name="description">
10  <meta content="" name="keywords">
11
12  <!-- Favicons -->
13  <link href="assets/img/mainlogo.jpeg" rel="icon">
14  <link href="assets/img/mainlogo.jpeg" rel="apple-touch-icon">
15
16  <!-- Google Fonts -->
17  <link
18    href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,600,600i,700,700i|Krub:300,300i,400,400i"
19    rel="stylesheet">
20
21  <!-- Vendor CSS Files -->
22  <link href="assets/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">
23  <link href="assets/vendor/icofont/icofont.min.css" rel="stylesheet">
24  <link href="assets/vendor/boxicons/css/boxicons.min.css" rel="stylesheet">
25  <link href="assets/vendor/owl.carousel/assets/owl.carousel.min.css" rel="stylesheet">
26  <link href="assets/vendor/venobox/venobox.css" rel="stylesheet">
27  <link href="assets/vendor/aos/aos.css" rel="stylesheet">
28
29  <!-- Template Main CSS File -->
30  <link href="assets/css/style.css" rel="stylesheet">
31 </head>
32
33 <body>
```

Figure 4.18: Code for the web application

4.3.6 API (Application Programming Interface)

The work of an API here is that it serves as an intermediary between the web server and web application. A python framework called Django was used on the model prediction on the web application. These two ends communicate using a JSON (Javascript Object Notation) to send a request and receive a response.

Figure 4.19 shows how the API was created by using Django python code for communication between the JSON dataset and feature extraction formula for detection phishing URL.

Figure 4.20 shows the creation of an API URL.

Figure 4.21 shows the execution of the Django local server for the API (Application Programming Interface)

Figure 4.22 shows testing the API Link created by executing a URL link on the postman, to test if it is phishing or legitimate.

Figure 4.23: shows the JavaScript code for linking the developed API to the web application other codes are shown in Appendix B.

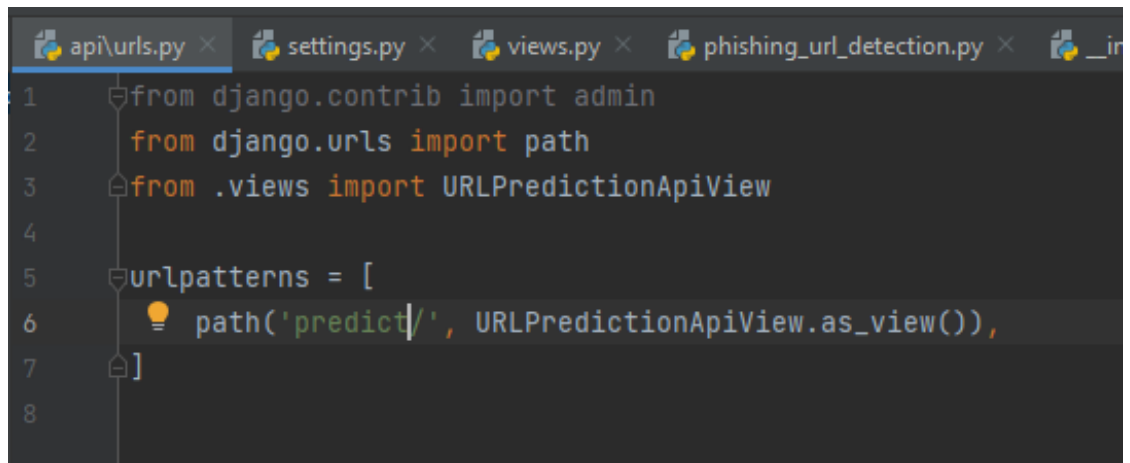
```

from rest_framework.views import APIView
from django.http import JsonResponse
import json
from .phishing_url_detection import DETECTION

class URLPredictionAPIView(APIView):
    def post(self, request):
        js = str(request.data).replace("'", '"')
        # GET THE URL FROM THE API
        url = (json.loads(js)['url'])
        detection = DETECTION()
        # CALL THE DETECTION METHOD HERE
        prediction = detection.featureExtractions(url)
        return JsonResponse({"success": True, "detection": prediction}, safe=False)

```

Figure 4.19: python code to send a request on JSON



The image shows a code editor with several tabs open: `api\urls.py`, `settings.py`, `views.py`, `phishing_url_detection.py`, and `_in`. The `api\urls.py` tab is active, displaying the following Python code:

```
1 from django.contrib import admin
2 from django.urls import path
3 from .views import URLPredictionAPIView
4
5 urlpatterns = [
6     path('predict/', URLPredictionAPIView.as_view()),
7 ]
8
```

Figure 4.20: Code for API URL

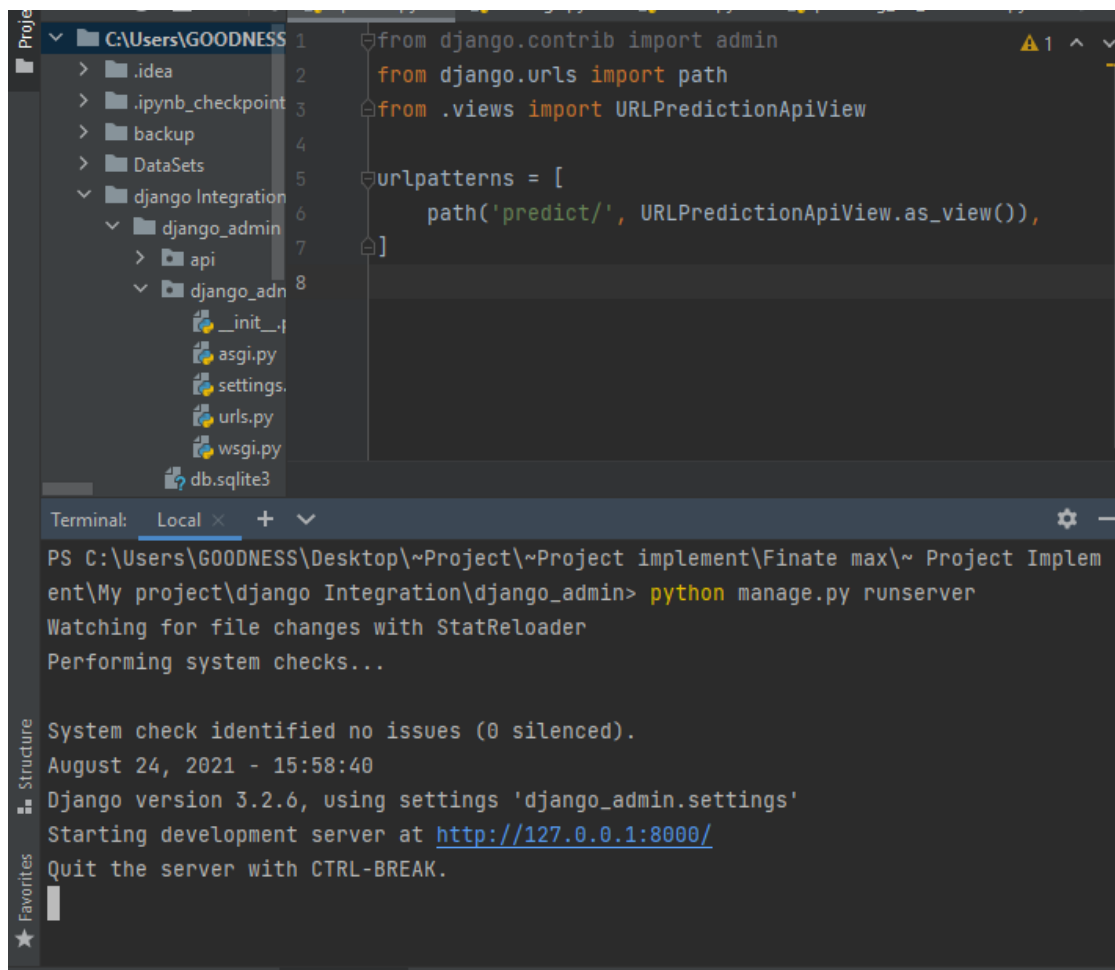


Figure 4.21: Executing web-app on Django local server

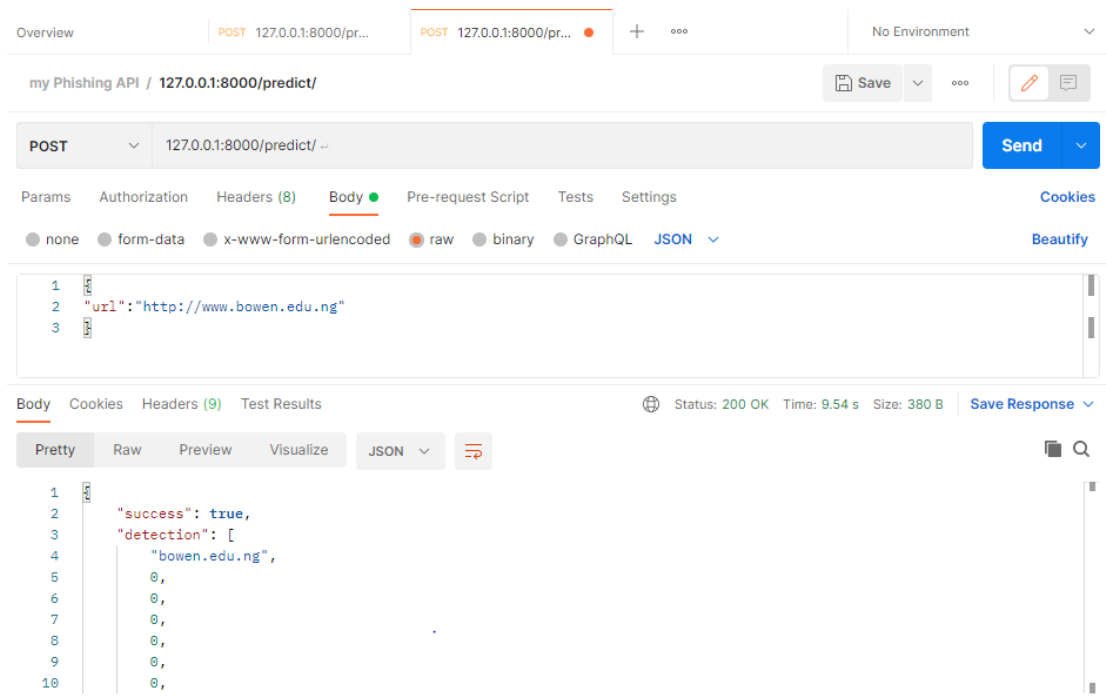


Figure 4.22: Testing the API link on postman

```

440
441 // OTHERS
442 if (data.detection[1] == 0) {
443     document.getElementById("have_ip").innerHTML += `<span class="badge badge-success" style="margin-left: 20px;"
444 } else {
445     document.getElementById("have_ip").innerHTML += `<span class="badge badge-danger" style="margin-left: 20px;">
446 }
447
448
449
450 if (data.detection[2] == 0) {
451     document.getElementById("have_at").innerHTML += `<span class="badge badge-success" style="margin-left: 20px;"
452 } else {
453     document.getElementById("have_at").innerHTML += `<span class="badge badge-danger" style="margin-left: 20px;">
454 }
455
456 }
457
458 if (data.detection[3] == 0) {
459     document.getElementById("url_length").innerHTML += `<span class="badge badge-success" style="margin-left: 20p
460 } else {
461     document.getElementById("url_length").innerHTML += `<span class="badge badge-danger" style="margin-left: 20px
462 }
463
464 }
465
466 if (data.detection[5] == 0) {
467     document.getElementById("redirection").innerHTML += `<span class="badge badge-success" style="margin-left: 20
468 } else {
469     document.getElementById("redirection").innerHTML += `<span class="badge badge-danger" style="margin-left: 20p
470 }
471

```

Figure 4.23: JavaScript code for linking API

CHAPTER FIVE

SUMMARY, CONCLUSION, AND RECOMMENDATIONS

5.1 Summary

Phishing attacks are a rapidly expanding threat in the cyber world, costing internet users billions of dollars each year. It involves the use of a variety of social engineering tactics to obtain sensitive information from users. Hence, Phishing techniques can be detected using a variety of types of communication, including email, instant chats, pop-up messages, and web pages.

This project was able to categorize and recognize how phishers carry out phishing attacks and the different ways in which researchers have helped to solve phishing detection. Hence, the proposed system of this project worked with different feature selection and machine learning and deep neural networks such as Decision Tree, Support Vector Machine, XGBooster, Multilayer Perceptions, Auto Encoder Neural Network, and Random Forest to identify patterns in which URL links can be detected easily.

The Model with the highest accuracy based on the feature extraction algorithm used to identify phishing URL from legitimate URL links was integrated to a web application where users can input website URL links to detect if it is legitimate or phishing.

5.2 Contribution to Knowledge

This project provides a new and faster way to help users detect if a URL link is phishing or legitimate and also provide them access to educational resources about phishing attacks.

5.3 Conclusion

The system developed detects if a URL link is phishing or legitimate by using machine learning models and deep neural network algorithms. The feature extraction and the models used on the dataset helped to uniquely identify phishing URLs and also the performance accuracy of the models used. It is also surprisingly accurate at detecting the genuineness of a URL link.

5.4 Recommendation

Through this project, one can know a lot about phishing attacks and how to prevent them. This project can be taken further by creating a browser extension that can be installed on any web browser to detect phishing URL Links.

REFERENCE

- Abdelhamid, N., Thabtah F., & Abdel-Jaber, H. Phishing detection: A recent intelligent machine learning comparison based on models' content and features," 2017 IEEE International Conference on Intelligence and Security Informatics (ISI), Beijing, 2017, pp. 72-77, DOI: 10.1109/ISI.2017.8004877.
- Anjum N. S., Antesar M. S., & Hossain M.A. (2016). A Literature Review on Phishing Crime, Prevention Review and Investigation of Gaps. Proceedings of the 10th International Conference on Software, Knowledge, Information Management & Applications (SKIMA), Chengdu, China, 2016, pp. 9-15, DOI: 10.1109/SKIMA.2016.7916190.
- Almomani, A., Gupta, B. B., Atawneh, S., Meulenberg, A., & Almomani, E. (2013). A survey of phishing email filtering techniques, Proceedings of IEEE Communications Surveys and Tutorials, vol. 15, no. 4, pp. 2070–2090.
- Ashritha, J. R., Chaithra, K., Mangala, K., & Deekshitha, S. (2019). A Review Paper on Detection of Phishing Websites using Machine Learning.Proceedings of International Journal of Engineering Research & Technology (IJERT), 7, 2. Retrieved from www.ijert.org.
- Anti-Phishing Working Group (APWG) Phishing activity trends report the first quarter. (2014) Retrieved from http://docs.apwg.org/reports/apwg_trends_report_q1_2014.pdf
- APWG report. (2014). Retrieved from [http://apwg.org/download/document/245/APWG Global Phishing Report 2H 2014.pdf](http://apwg.org/download/document/245/APWG_Global_Phishing_Report_2H_2014.pdf)

- Ayush, P. (2019). Workflow of a Machine Learning project. Retrieved from <https://towardsdatascience.com/workflow-of-a-machine-learning-project-ec1dba419b94>
- Camp W. (2001). Formulating and evaluating theoretical frameworks for career and technical education research. *Journal of Vocational Education Research*, 26(1), 4-25.
- DeepAI (n.d.). About clinical psychology. Retrieved from <https://deepai.org/machine-learning-glossary-and-terms/feature-extraction>
- Engine K., & Christopher K. (2005). Protecting Users Against Phishing Attacks. Proceedings of the Oxford University Press on behalf of The British Computer Society, Oxford University, 0, 2005, Retrieved from: https://sites.cs.ucsb.edu/~chris/research/doc/cj06_phish.pdf
- Gandhi, V. (2017). A Theoretical Study on Different ways to identify the Phishing URL and Its Prevention Approaches: presented at International Conference on Cyber Criminology, Digital Forensics and Information Security at DRBCCC Hindu College, Chennai. Retrieved from https://www.researchgate.net/publication/319006943_A_Theoretical_Study_on_Different_ways_to_Identify_the_Phishing_URL_and_Its_Prevention_Approaches
- Gupta, B. B., Tewari, A., Jain, A. K., & Agrawal, D. P. (2016). Fighting against phishing attacks: state of the art and future challenges, *Neural Computing and Applications*.
- Internet world stats usage and population statistics. (2014). Retrieved from <http://www.internetworldstats.com/stats.htm>.

- Imperva. (2021). Phishing attacks. Retrieved from <https://www.imperva.com/learn/application-security/phishing-attack-scam/>
- Kiruthiga, R., Akila, D. (2019, September). Phishing Websites Detection Using Machine Learning. Retrieved from [https://www.researchgate.net/publication/337049054 Phishing Websites Detection Using Machine Learning](https://www.researchgate.net/publication/337049054_Phishing_Websites_Detection_Using_Machine_Learning).
- KnowBe4 (2021). Phishing Techniques. Retrieved from <https://www.phishing.org/phishing-techniques>
- Kondeti, P. S., Konka, R. C., & Kavishree, S. (2021). Phishing Websites Detection using Machine Learning Techniques. International Research Journal of Engineering and Technology, 08(4), Page 1471-1473. Retrieved from <https://www.irjet.net/archives/V8/i4/IRJET-V8I4274.pdf>
- Noel, B. (2016). Support Vector Machines: A Simple Explanation. Retrieved from <https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html>
- Osanloo, A., & Grant, C. (2016). Understanding, selecting, and integrating a theoretical framework in dissertation research: creating the blueprint for your “house”. Administrative issues journal: connecting education, practice and research 4(2), 7.
- Peng, T., Harris, I., & Sawa, I. (2018). Detecting Phishing Attacks Using Natural Language Processing and Machine Learning. Proc. - 12th IEEE Int. Conf. Semant. Comput. ICSC 2018, vol. 2018–Janua, pp. 300–301.

- Pamela (2021). Phishing attacks. Retrieved from <https://www.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:online-data-security/xcae6f4a7ff015e7d:cyber-attacks/a/phishing-attacks>
- Rami, M. M., Fadi, T., & Lee, M. (2015). Phishing Websites Features. Retrieved from <https://eprints.hud.ac.uk/id/eprint/24330/6/MohammadPhishing14July2015.pdf>
- Rishikesh, M., & Irfan, S. (2018a). Phishing Website Detection using Machine Learning Algorithms. International Journal of Computer Applications, 23, 45. doi:10.5120/ijca2018918026
- Rishikesh, M., & Irfan, S. (2018b). Phishing Website Detection using Machine Learning Algorithms. International Journal of Computer Applications, 23, 45-46. doi:10.5120/ijca2018918026
- Rahul, S. (2017). How the decision tree algorithm works. Retrieved from <https://dataaspirant.com/how-decision-tree-algorithm-works/>
- Rishikesh, M., & Irfan, S. (2018c). Phishing Website Detection using Machine Learning Algorithms. International Journal of Computer Applications, 23, 46-47. doi:10.5120/ijca2018918026
- Saimadhu, P. (2017). How the random forest algorithm works in machine learning. Retrieved from <https://dataaspirant.com/random-forest-algorithm-machine-learning/>
- Shaikh, A.N., Shabut, A.M., Hossain, M.A. (2016, December 15-17). A literature review on phishing crime, prevention review, and investigation of gaps. Paper presented at the Tenth International Conference on Software, Knowledge,

- Information Management & Applications (SKIMA), Chengdu, China. Retrieved from <https://ieeexplore.ieee.org/document/7916190>
- Shreya, G. (2020). Phishing website detection by machine learning techniques. Retrieved from <https://github.com/shreyagopal/Phishing-Website-Detection-by-Machine-Learning-Techniques>
- Sönmez, Y., Tuncer, T., Gökal, H., & Avci, E. (2018). Phishing web sites features classification based on extreme learning machine. 6th Int. Symp. Digit. Forensics Secure. ISDFS 2018 - Proceeding, vol. 2018–Janua, pp. 1–5.
- RSA Anti-Fraud Command Center (n.d.) Retrieved from <https://www.emc.com/collateral/fraud-report/rsa-online-fraud-report-012014.pdf>.
- Shad, J., & Sharma, S. (2018). A Novel Machine Learning Approach to Detect Phishing Websites Jaypee Institute of Information Technology, pp. 425–430.
- The RSA Current State of Cybercrime. (n.d.). Retrieved from <https://www.rsa.com/en-us/perspectives/industry/online-fraud>.
- Tewari, A., Jain, A. K., & Gupta, B. B. (2016). A recent survey of various defense mechanisms against phishing attacks. Journal of Information Privacy and Security, vol. 12, no. 1, pp. 3–13.
- Joachim, S. (n.d). Missing Value Imputation (Statistics) – How to Impute Incomplete Data. Retrieved from <https://statisticsglobe.com/missing-data-imputation-statistics>
- Kartik, M. (2021). Everything You Need to Know About Feature Selection in Machine Learning. Retrieved from <https://www.simplilearn.com/tutorials/machine-learning-tutorial/feature-selection-in-machine-learning>

- Kiruthiga R., & Akila D. (2019). Phishing Website Detection using Machine Learning. International Journal of Recent Technology and Engineering, 8, 2S11. DOI: 10.35940/ijrte.B1018.0982S1119
- Kyaw, S. (2020). A Guide to KNN Imputation. Retrieved from <https://medium.com/@kyawsawhtoon/a-guide-to-knn-imputation-95e2dc496e>
- Kruegel, C., Kirda, E., Mutz, D., Robertson, W., & Vigna, G. (2005). Automating mimicry attacks using static binary analysis. Proceedings of the USENIX Security Symposium, pp. 161–176.
- Will, B. (2019). 6 Different Ways to Compensate for Missing Values In a Dataset (Data Imputation with examples). Retrieved from <https://towardsdatascience.com/6-different-ways-to-compensate-for-missing-values-data-imputation-with-examples-6022d9ca0779>
- Workflow of a Machine Learning project (2019). Retrieved from <https://towardsdatascience.com/workflow-of-a-machine-learning-project-ec1dba419b94>

APPENDIX A: Web application Source code

```
<? index.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="utf-8">
6      <meta content="width=device-width, initial-scale=1.0" name="viewport">
7
8      <title>Phishbuster</title>
9      <meta content="" name="description">
10     <meta content="" name="keywords">
11
12     <!-- Favicons -->
13     <link href="assets/img/mainlogo.jpeg" rel="icon">
14     <link href="assets/img/mainlogo.jpeg" rel="apple-touch-icon">
15
16     <!-- Google Fonts -->
17     <link
18         href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,600,600i,700,700i|Krub:300,300i,400,400i,600,600i,700,700i"
19         rel="stylesheet">
20
21     <!-- Vendor CSS Files -->
22     <link href="assets/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">
23     <link href="assets/vendor/icomfont/icomfont.min.css" rel="stylesheet">
24     <link href="assets/vendor/boxicons/css/boxicons.min.css" rel="stylesheet">
25     <link href="assets/vendor/owl.carousel/assets/owl.carousel.min.css" rel="stylesheet">
26     <link href="assets/vendor/venobox/venobox.css" rel="stylesheet">
27     <link href="assets/vendor/aos/aos.css" rel="stylesheet">
28
29     <!-- Template Main CSS File -->
30     <link href="assets/css/style.css" rel="stylesheet">
31 </head>
32
33 <body>
```

```
index.html > ...
119 <i class="bx bx-help-circle icon-help"></i> <a id="have_ip" data-toggle="collapse" href="#faq-list-2"
120 class="collapsed">Have_IP <i class="bx bx-chevron-down icon-show"></i><i
121 class="bx bx-chevron-up icon-close"></i></a>
122 <div id="faq-list-2" class="collapse" data-parent=".faq-list">
123 <p>
124 If an IP address is used as an alternative of the domain name in the URL, such as
125 "http://125.98.3.123/fake.html"
126 </p>
127 </div>
128 </li>
129
130 <li data-aos="fade-up" data-aos-delay="200">
131 <i class="bx bx-help-circle icon-help"></i> <a id="have_at" data-toggle="collapse" href="#faq-list-3"
132 class="collapsed">Have_At<i class="bx bx-chevron-down icon-show"></i><i
133 class="bx bx-chevron-up icon-close"></i></a>
134 <div id="faq-list-3" class="collapse" data-parent=".faq-list">
135 <p>
136 Using "@" symbol in the URL leads the browser to ignore everything preceding the "@" symbol and
137 address often follows the "@" symbol
138 </p>
139 </div>
140 </li>
141
142 <li data-aos="fade-up" data-aos-delay="200">...
143 </li>
144
145 <li data-aos="fade-up" data-aos-delay="200">...
146 </li>
147
148 <li data-aos="fade-up" data-aos-delay="200">...
149 </li>
150 <li data-aos="fade-up" data-aos-delay="200">...
```

```

352         allow= accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture
353         allowfullscreen></iframe>
354     </p>
355 </div>
356
357 </div>
358
359 </div>
360 </section><!-- End Testimonials Section -->
361
362
363 </main><!-- End #main -->
364
365 <!-- ===== Footer ===== -->
366 <footer id="footer">
367     <div class="container d-md-flex py-4">
368
369         <div class="text-center text-md-center">
370             <div class="copyright">
371                 
372                 <a href="https://phishingquiz.withgoogle.com/" class="get-started-btn scrollto">Take a Phishing Test t
373                 GOOGLE</a> <br>
374                 <hr/> <a href="assets/resources/my_work.7Z" class="get-started-btn scrollto" download="my_work">Proj
375                 <p style="padding-top: 8px; font-weight: 600;">Final Year Project &copy; 2021.All Rights Reserved</p>
376             </div>
377         </div>
378     </div>
379 </footer><!-- End Footer -->
380
381 <a href="#" class="back-to-top"><i class="icofont-simple-up"></i></a>
382 <!-- <div id="preloader"></div> -->
383
384 <!-- Vendor JS Files -->
385 <script src="assets/vendor/jquery/jquery.min.js"></script>

```

```
info.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5    <meta charset="utf-8">
6    <meta content="width=device-width, initial-scale=1.0" name="viewport">
7
8    <title>Phishbuster | Info </title>
9    <meta content="" name="description">
10   <meta content="" name="keywords">
11
12   <!-- Favicons -->
13   <link href="assets/img/mainlogo.jpeg" rel="icon">
14   <link href="assets/img/mainlogo.jpeg" rel="apple-touch-icon">
15
16   <!-- Google Fonts -->
17   <link href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,600,600i,700,700i|Krub:300,300i,400,400i,600,600i,700,700i" rel="stylesheet">
18
19   <!-- Vendor CSS Files -->
20   <link href="assets/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">
21   <link href="assets/vendor/icomfont/icomfont.min.css" rel="stylesheet">
22   <link href="assets/vendor/boxicons/css/boxicons.min.css" rel="stylesheet">
23   <link href="assets/vendor/owl.carousel/assets/owl.carousel.min.css" rel="stylesheet">
24   <link href="assets/vendor/venobox/venobox.css" rel="stylesheet">
25   <link href="assets/vendor/aos/aos.css" rel="stylesheet">
26
27   <!-- Template Main CSS File -->
28   <link href="assets/css/style.css" rel="stylesheet">
29 </head>
30
31 <body>
32
33
```

```

info.html > html > head
90 <!--End Hero -->
91
92 ===== Frequently Asked Questions Section ===== -->
93 <div id="faq" class="faq">
94 <div class="text-center"><b>Resources On Phishing</b></h2>
95 <div class="container" data-aos="fade-up">
96
97 <div class="faq-list">
98 <ul>
99 <li data-aos="fade-up" data-aos-delay="100">
100 <i class="bx bx-help-circle icon-help"></i> <a data-toggle="collapse" class="collapse" href="#faq-list-1">
101 <div id="faq-list-1" class="collapse show" data-parent=".faq-list">
102 <p>
103 Phishing is a cybercrime in which a target or targets are contacted by email, telephone, or text message
104 OR
105 Phishing is a cyber attack where the attacker tricks the target into disclosing personal information,
106 </p>
107 </div>
108 </li>
109
110 <li data-aos="fade-up" data-aos-delay="200">
111 <i class="bx bx-help-circle icon-help"></i> <a data-toggle="collapse" href="#faq-list-2" class="collapsed">
112 <div id="faq-list-2" class="collapse" data-parent=".faq-list">
113 <p>
114 The purpose of phishing is to collect sensitive information with the intention of using that information
115 Successful Phishing attacks can: <br><br>
116 <i class="bx bxs-chevron-right-circle icon-circle"></i> Cause financial loss for victims <br><br>
117 <i class="bx bxs-chevron-right-circle icon-circle"></i> Put their personal information at risk <br><br>
118 <i class="bx bxs-chevron-right-circle icon-circle"></i> Put data and systems at risk
119 </p>
120 </div>
121 </li>
122

```

```

info.html > html > head
125 v id="faq-list-3" class="collapse" data-parent=".faq-list">
126 p>
127 <i class='bx bxs-chevron-right-circle icon-circle'></i> <b>Deceptive Phishing:</b> Deceptive phishing is by
128 <i class='bx bxs-chevron-right-circle icon-circle'></i> <b>Spear Phishing:</b> Spear Phishing targets specific
129 <i class='bx bxs-chevron-right-circle icon-circle'></i> <b>Other phishing Technique:</b> <i class='bx bxs-hand'
130 <i class='bx bx-right-arrow-alt icon-arrow'></i> <b>Angler Phishing:</b> This cyberattack comes by way of social media
131 <i class='bx bx-right-arrow-alt icon-arrow'></i> <b>Clone Phishing:</b> Clone phishing involves exact duplication of a legitimate website
132 <i class='bx bx-right-arrow-alt icon-arrow'></i> <b>Domain Spoofing:</b> In this category of phishing, the attacker uses a domain name that is very similar to the legitimate domain
133 <i class='bx bx-right-arrow-alt icon-arrow'></i> <b>Email Phishing:</b> Phishing emails are often the first step in a phishing attack
134 <i class='bx bx-right-arrow-alt icon-arrow'></i> <b>Search Engine Phishing:</b> Rather than sending correspondence to a specific email address, the attacker uses search engines to deliver the phishing message
135 <i class='bx bx-right-arrow-alt icon-arrow'></i> <b>Smishing:</b> Combine SMS with phishing and you have the smishing attack
136 <i class='bx bx-right-arrow-alt icon-arrow'></i> <b>Whaling:</b> A whaling attack targets the big fish, or executive, of a company
137 <i class='bx bx-right-arrow-alt icon-arrow'></i> <b>Vishing:</b> Combine VoIP with phishing and you get vishing
138 <i class='bx bx-right-arrow-alt icon-arrow'></i> <b>Malvertising:</b> Malvertising is malicious advertising
139 /p>
140 iv>
141
142
143 ata-aos="fade-up" data-aos-delay="200">
144 class="bx bx-help-circle icon-help"></i> <a data-toggle="collapse" href="#faq-list-4" class="collapsed">How to Prevent Phishing Attacks?</a>
145 v id="faq-list-4" class="collapse" data-parent=".faq-list">
146 p>
147 To help prevent phishing attacks, you should observe general best practices, similar to those you might understand from general cybersecurity training.
148
149 <i class='bx bxs-chevron-right-circle icon-circle'></i> Users are to choose strong passwords and be wary of phishing attempts
150 <i class='bx bxs-chevron-right-circle icon-circle'></i> If there are any suspicions about an email or social media message, do not click on links or download attachments
151 <i class='bx bxs-chevron-right-circle icon-circle'></i> Only open attachments from a trusted source. When in doubt, delete the attachment
152 <i class='bx bxs-chevron-right-circle icon-circle'></i> Note any language differences in messaging or emails
153 <i class='bx bxs-chevron-right-circle icon-circle'></i> Never give away personal information in an email or social media message
154 <i class='bx bxs-chevron-right-circle icon-circle'></i> Inspect emails for typos and inaccurate grammar. This is often a sign of a phishing attempt
155 <i class='bx bxs-chevron-right-circle icon-circle'></i> Don't supply personal information via email or text. Only provide information through secure channels
156 <i class='bx bxs-chevron-right-circle icon-circle'></i> Beware of urgent or time-sensitive warnings. Phishing attempts often create a sense of urgency
157 <i class='bx bxs-chevron-right-circle icon-circle'></i> Verify emails and other correspondence by contacting the sender through a different method

```

```

109 <!-- ===== testimonials section ===== -->
170 <section id="testimonials" class="testimonials section-bg">
171   <div class="container" data-aos="fade-up">
172
173     <div class="owl-carousel testimonials-carousel">
174       <div class="testimonial-item">
175         <p>
176           <iframe width="600" height="500" src="assets/img/pic1.png" title="YouTube video player" frameborder="
177             <p style="font-size: 12px;"> <b>Description:</b> Consider this fake Paypal security notice <br> wa
178           </p>
179         </div>
180
181       <div class="testimonial-item">
182         <p>
183           <iframe width="600" height="500" src="assets/img/pic2.png" title="YouTube video player" frameborder="
184             <p style="font-size: 12px;"><b>Description:</b>Fake Gmail login Page (it might look exactly <br> lik
185           </p>
186         </div>
187
188       <div class="testimonial-item">
189         <p>
190           <iframe width="600" height="500" src="assets/img/pic3.png" title="YouTube video player" frameborder="
191             <p style="font-size: 12px;"><b>Description:</b>This is a LinkedIn Phishing Attack which InMail <br>
192           </p>
193         </div>
194
195       <div class="testimonial-item">
196         <p>
197           <iframe width="600" height="500" src="assets/img/pic4.png" title="YouTube video player" frameborder="
198             <p style="font-size: 12px;"><b>Description:</b>The document above contain <b>Macros <br> Payloads</b>
199           </p>
200         </div>
201
202

```

```

usecases.html > ...
216     <div class="text-center text-md-center">
217         <div class="copyright">
218             
219             <a href="https://phishingquiz.withgoogle.com/" class="get-started-btn scrollto">Take a Phishing Test t
220             <hr/> <a href="assets/resources/my_work.72" class="get-started-btn scrollto" download="my_work">Proje
221             <p style="padding-top: 8px; font-weight: 600;">Final Year Project &copy; 2021.All Rights Reserved</p>
222         </div>
223     </div>
224 </div>
225 </footer><!-- End Footer -->
226
227 <a href="#" class="back-to-top"><i class="icofont-simple-up"></i></a>
228 <!-- <div id="preloader"></div> -->
229
230 <!-- Vendor JS Files -->
231 <script src="assets/vendor/jquery/jquery.min.js"></script>
232 <script src="assets/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>
233 <script src="assets/vendor/jquery.easing/jquery.easing.min.js"></script>
234 <script src="assets/vendor/php-email-form/validate.js"></script>
235 <script src="assets/vendor/owl.carousel/owl.carousel.min.js"></script>
236 <script src="assets/vendor/isotope-layout/isotope.pkgd.min.js"></script>
237 <script src="assets/vendor/venobox/venobox.min.js"></script>
238 <script src="assets/vendor/aos/aos.js"></script>
239

```

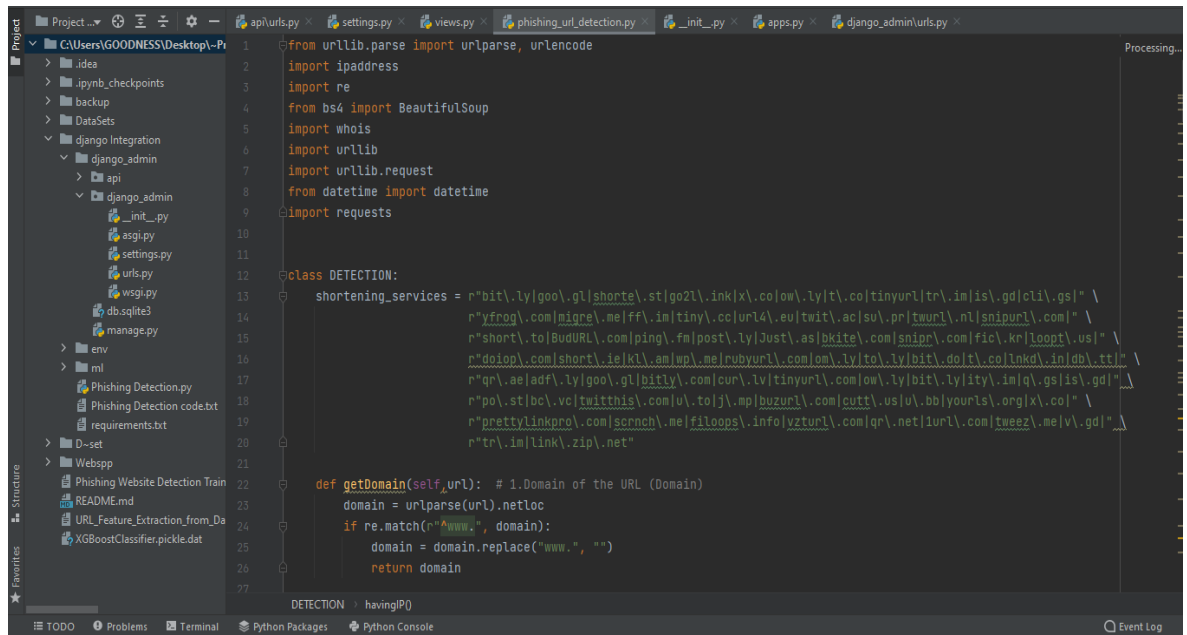


```

JS main.js > ...
1  const openModalButtons = document.querySelectorAll('[data-modal-target]')
2  const closeModalButtons = document.querySelectorAll('[data-close-button]')
3  const overlay = document.getElementById('overlay')
4
5  openModalButtons.forEach(button => {
6      button.addEventListener('click', () => {
7          const modal = document.querySelector(button.dataset.modalTarget)
8          openModal(modal)
9      })
10 })
11
12 overlay.addEventListener('click', () => {
13     const modals = document.querySelectorAll('.modal.active')
14     modals.forEach(modal => {
15         closeModal(modal)
16     })
17 })
18
19 closeModalButtons.forEach(button => {
20     button.addEventListener('click', () => {
21         const modal = button.closest('.modal')
22         closeModal(modal)
23     })
24 })
25
26 function openModal(modal){
27     if(modal == null) return
28     modal.classList.add('active')
29     overlay.classList.add('active')
30 }
31
32 function closeModal(modal){
33     if(modal == null) return

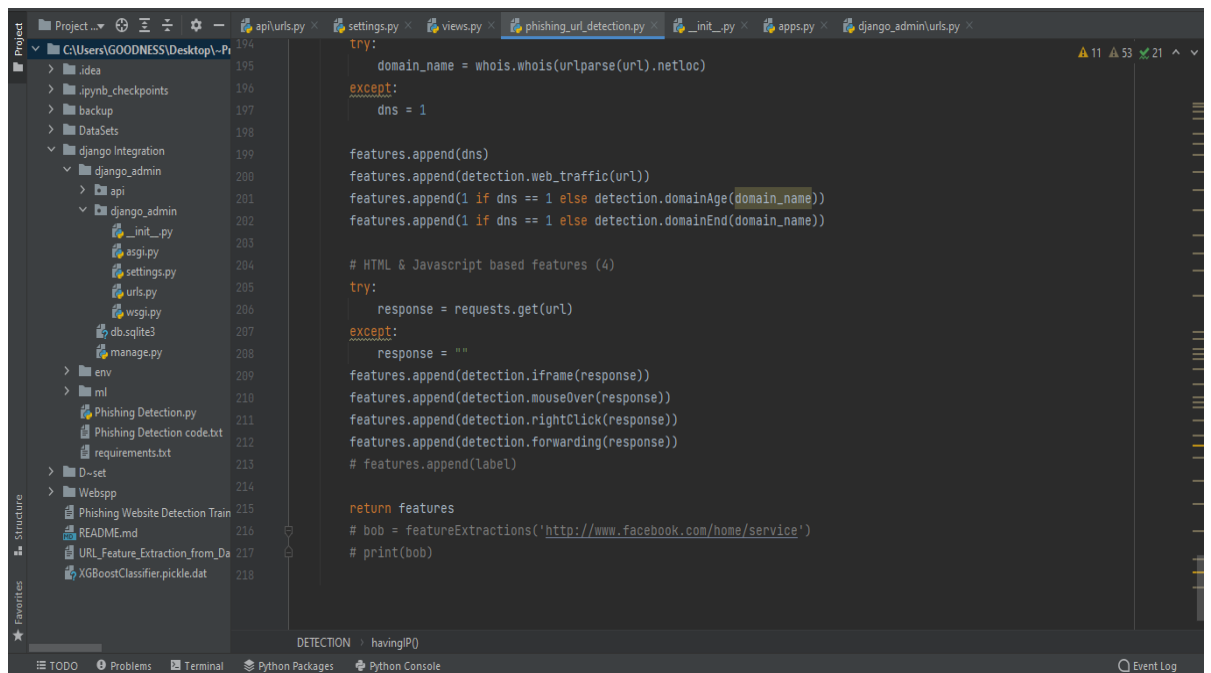
```

APPENDIX B: Source code for API integration to web app



The screenshot shows an IDE with a project structure on the left and a Python file named `phishing_url_detection.py` open in the editor. The project structure includes a `django Integration` folder with `django_admin` and `api` subfolders. The `api` folder contains `django_admin` with files like `__init__.py`, `asgi.py`, `settings.py`, `urls.py`, `wsgi.py`, `db.sqlite3`, and `manage.py`. The `api` folder also contains `env`, `ml`, `Phishing Detection.py`, `Phishing Detection code.bit`, `requirements.txt`, `D--set`, `Webapp`, `Phishing Website Detection Train`, `README.md`, `URL_Feature_Extraction_from_Da`, and `XGBoostClassifier.pickle.dat`.

```
1 from urllib.parse import urlparse, urlencode
2 import ipaddress
3 import re
4 from bs4 import BeautifulSoup
5 import whois
6 import urllib
7 import urllib.request
8 from datetime import datetime
9 import requests
10
11
12 class DETECTION:
13     shortening_services = r"bit\ly|goo\gl|shorte\st|go2l\ink|x\cow\ly|t\coltinyurl|tr\im|is\gd|cl|gs| " \
14         r"yfrog\com|migre\me|ff\im|tiny\cc|url4\eu|twit\ac|su\pr|twurl\nl|snipurl\com| " \
15         r"short\to|BudURL\com|ping\fm|post\ly|Just\as|bkite\com|snipr\com|fic\kr|loopt\us| " \
16         r"doio\com|short\ie|kl\am|wp\me|rubbyurl\com|ow\ly|to\ly|bit\dot\col|nkd\in|db\tt| " \
17         r"qr\ae|adf\ly|goo\gl|bitly\com|cur\lv|tinyurl\com|ow\ly|bit\ly|ity\im|q\gs|is\gd| " \
18         r"po\st|bc\vc|twitthis\com|u\to|j\mp|buzurl\com|cutt\us|u\bb|yourls\org|x\co| " \
19         r"prettylinkpro\com|scrnch\me|filoops\info|vztun\com|qr\net|lur\com|tweez\me|v\gd| " \
20         r"tr\im|link\zip\net"
21
22     def getDomain(self,url): # 1.Domain of the URL (Domain)
23         domain = urlparse(url).netloc
24         if re.match(r"^www.", domain):
25             domain = domain.replace("www.", "")
26         return domain
27
```



```
api\urls.py × settings.py × views.py × phishing_url_detection.py × _init_.py × apps.py × django_admin\urls.py ×
18
19 # Quick-start development settings - unsuitable for production
20 # See https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/
21
22 # SECURITY WARNING: keep the secret key used in production secret!
23 SECRET_KEY = 'django-insecure-op=eogvy#b7s03or*ze^zy@kh42_&3s)z2+2h8x0di0#s%u32p6'
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
28 ALLOWED_HOSTS = []
29
30 #####added code.....
31 # import os
32 # MODELS = os.path.join(BASEDIR, 'ml/models')
33
34
35
36 # Application definition
37
38 INSTALLED_APPS = [
39     'django.contrib.admin',
40     'django.contrib.auth',
41     'django.contrib.contenttypes',
42     'django.contrib.sessions',
43     'django.contrib.messages',
44     'django.contrib.staticfiles',
```