MALICIOUS SITE DETECTING USING MACHINE LEARNING

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE OF

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

SUBMITTED BY

NAME	REGISTER NO
M.ARUNBALAJI	191009007
M.BHUVANESHWARAN	191009011
C.S. KIRUBANANDHAN	191009029
R.S. SREEBALAJE	191009068

UNDER THE GUIDANCE OF
Dr. D. SURESH M.E., Ph.D.,
Assistant Professor

DEPARTMENT OF INFORMATION TECHNOLOGY

ANNAMALAI UNIVERSITY

ANNAMALAI NAGAR-608002

MAY - 2023

ANNAMALAI UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this report titled "**MALICIOUS SITE DETECTING USING MACHINE LEARNING"** is the bonafide record of the work done by **M.ARUNBALAJI (Reg.No:191009007), M.BHUVANESHWARAN (Reg.No:191009011), C.S.KIRUBANANDHAN (Reg.No:191009029), R.S.SREEBALAJE (Reg.No:191009068)** for the degree of B.E (Information Technology) during the year 2022-2023

Dr. D. SURESH
Project Guide and
Assistant Professor of
Dept. of Information Technology

Dr. K.SELVA KUMAR
Professor and Head
Dept. of Information Technology
Annamalai University

INTERNAL EXAMINER

EXTERNAL EXAMINER

PLACE: ANNAMALAI NAGAR

DATE:

ACKNOWLEDGEMENT

We owe our Special Thanks to **Dr. K.SELVAKUMAR, Professor and Head, Department of Information Technology**, for his encouraging and valuable counsel throughout this Project.

We extend our heartily gratitude to **Dr.D.SURESH, Project Guide and Assistant Professor, Department of Information Technology**, for his canalized guidance, dedication, constant inspiration and encouragement.

We also express our sincere thanks to **Dr.V.TAMIZHAZHAGAN, Project Coordinator and Assistant Professor** , Department of Information Technology, for having taught us the value of discipline, which helped us to carry out the project in a systematic manner.

Last but not the least, I would like to thank my family and team mates for their support and encouragement for the successful completion of the thesis work.

M.ARUNBALAJI (reg.no: 191009007)

M.BHUVANESHWARAN (reg.no: 191009011)

C.S. KIRUBANANDHAN (reg.no:191009029)

R.S. SREEBALAJE (reg.no:191009068)

TABLE OF CONTENTS

ABSTRACT

The growing use of the internet resulted in emerging of new websites every day. Web surfing has become important for everyone regardless of their occupation, age or location. However, as the use of the internet is increasing so is the vulnerability to malware attacks through malicious websites. Identifying and dealing with such malicious website has been quite difficult in the past as it is quite challenging to separate good websites from bad websites. However, by using machine learning algorithm on large datasets it is now possible to detect such websites beforehand. Classifier trained using algorithm such as logistic regression can be used to detect malicious websites and the users can be warned about the risk before they visit such sites. This project focuses on using a classification algorithm to distinguish whether a website is malicious or not using the Malicious and Benign Website Dataset. We have showcased that it is possible to detect malicious websites with a reasonable amount of certainty (90% of the 75 malicious websites in the test set were identified) using machine learning models. We have also determined the features that were critical in predicting the likely hood of a website being malicious. Most of our key features are easily available (URL Length, TTL, TLD, Age of website).

Keyword: malicious websites, cyber-security, machine learning, logistic Regression

திட்ட சுருக்கம்

வளர்ந்து வரும் இணையப் பயன்பாடு நாளுக்கு நாள் புதிய இணையதளங்கள் தோன்றுவதற்கு வழிவகுத்தது. அவர்களின் தொழில், வயது அல்லது இருப்பிடத்தைப் பொருட்படுத்தாமல் அனைவருக்கும் வலை உலாவல் முக்கியமானது. இருப்பினும், இணையத்தின் பயன்பாடு அதிகரித்து வருவதால், தீங்கிழைக்கும் வலைத்தளங்கள் மூலம் தீம்பொருள் (Malware) தாக்குதல்களுக்கு பாதிப்பு ஏற்படுகிறது. இதுபோன்ற தீங்கிழைக்கும் இணையதளங்களை அடையாளம் கண்டு கையாள்வது கடந்த காலங்களில் மிகவும் கடினமாக இருந்தது, ஏனெனில் மோசமான வலைத்தளங்களிலிருந்து நல்ல வலைத்தளங்களை பிரிப்பது மிகவும் சவாலானது. எவ்வாறாயினும், பெரிய தரவுத்தொகுப்புகளில் இயந்திர கற்றல் வழிமுறையை பயன்படுத்துவதன் மூலம் இப்போது அத்தகைய வலைத்தளங்களை முன்பே கண்டறிய முடியும். தளவாட பின்னடைவு போன்ற வழிமுறையை பயன்படுத்தி பயிற்றுவிக்கப்பட்ட வகைப்படுத்தி தீங்கிழைக்கும் வலைத்தளங்களைக் கண்டறியப் பயன்படுகிறது மற்றும் பயனர்கள் அத்தகைய தளங்களைப் பார்வையிடும் முன் ஆபத்து குறித்து எச்சரிக்கலாம். இந்தத் திட்டமானது, ஒரு இணையதளம் தீங்கிழைக்கிறதா அல்லது தீங்கிழைக்கும் மற்றும் தீங்கற்ற இணையதளத் தரவுத்தொகுப்பைப் பயன்படுத்தவில்லையா என்பதைப் பிரித்தறிய, வகைப்படுத்தல் வழிமுறையை பயன்படுத்துவதில் கவனம் செலுத்துகிறது. இயந்திர கற்றல் மாதிரிகளைப் பயன்படுத்தி தீங்கிழைக்கும் இணையதளங்களை நியாயமான அளவு உறுதியுடன் (சோதனை தொகுப்பில் உள்ள 75 தீங்கிழைக்கும் இணையதளங்களில் 90% அடையாளம் காணப்பட்டது) கண்டறிய முடியும் என்பதை நாங்கள் வெளிப்படுத்தியுள்ளோம். ஒரு இணையதளம் தீங்கிழைக்கும் வாய்ப்பைக் கணிப்பதில் முக்கியமான அம்சங்களையும் நாங்கள் தீர்மானித்துள்ளோம். எங்களின் பெரும்பாலான முக்கிய அம்சங்கள் எளிதாகக் கிடைக்கின்றன (URL நீளம், TTL, TLD, இணையதளத்தின் வயது).

முக்கிய வார்த்தை: தீங்கிழைக்கும் இணையதளங்கள், இணைய பாதுகாப்பு, இயந்திர கற்றல், தளவாட பின்னடைவு.

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVATIONS	DESCRIPTION
URL	Uniformed Resources Locator
TTL	Time to live
TLD	Top Level Domain
SVM	Support Vector Machine
ANN	Artificial Neural Network
NLP	Natural Language Processing
IDE	Integrated Development Environment
CSV	Comma Separated Value
JSON	JavaScript Object Notation
NaN	Not a Numbers
HTTPS	Hypertext Transfer Protocol Secure
TF	Term Frequency
IDF	Inverse Document Frequency
ML	Machine Learning

CHAPTER 1
INTRODUCTION

1.1 OVERVIEW

Online services have become an irreplaceable part of today's businesses, schools, banking, or personal lives. With their increasing popularity, the number of malicious websites is growing. A malicious website contains some unsolicited content with a purpose to gather sensitive data or install malware into a user's machine. Usually, some interaction from the user part is needed, but in the case of a drive by download, malware is installed automatically without asking for permission. Prevention from such attacks is complicated, as being cautious is sometimes not enough. Attackers can able to exploit vulnerabilities in web applications to insert malicious code without knowledge of the owner. On the first look, legitimate websites can become a potential threat for users. Browser and antiviruses re expected to prevent a user from accessing such websites. The URL is usually the first and cheapest information we have about a website. Therefore it is a natural choice to develop techniques that would recognize a malicious URL from benign. Moreover, accessing and downloading the content of the website can be time consuming and brings risks associated with downloading potentially harmful content. The most common technique used to aim this problem is blacklisting. The list of 'bad URLs' is compiled, and the browser prevents the user from accessing them. The major problem with this technique is blacklist incompleteness, URL only appear there if it was already reported before. To solve this issue, we need a more proactive solution that can spot and match patterns in malicious URLs.

1.2 OBJECTIVE

The objective of the project is develop a model which can be used to detect malicious URLs. The goal is to determine if chosen algorithm can efficiently decide if the given URL on input is malicious or not.

1.3 PROBLEM STATEMENT

Web Security has become very important in recent years as internet connectivity has penetrated more and more regions across the world. While this penetration is great for global connectivity, it also means that more people have access to websites that can potentially attack them using malwares, viruses, and other malicious agents. Thus, it becomes more important than ever to identify and deal with such websites before a normal user has access to them. The

aim of this project is to detect malicious websites using a machine learning algorithm called classifier, we will try to detect a malicious website. This will help in safe web surfing and better user experience. By timely reporting malicious websites, the users will be able to avoid any violation and serious privacy breach. The users will also be able to avoid any illegal activities that they can get involved in. Labelling malicious websites will also help to eliminating fraud, as users become victim of attacks that use blackmailing and false information to get monetary advantage of their victim. For example, ransomware attacks are getting quite common. Systems get infected by such viruses through surfing malicious websites.

1.4 PROJECT DESCRIPTION

The goal of this project is to develop a machine learning-based approach for detecting malicious websites. The approach will use a logistic regression, classifier algorithm to determine whether a given URL is good or bad. The project will involve collecting a large dataset of URLs and labeling them as either good or bad based on their characteristics. Features such as the domain name, length of the URL, presence of certain keywords, and other relevant information will be used to train the logistic regression model. Once the model is trained, it will be tested on a separate set of URLs to evaluate its accuracy and performance. The results of the testing phase will be used to fine-tune the model and improve its accuracy. The final outcome of this project will be a reliable and efficient tool for detecting malicious websites that can be used to improve internet security and protect users from potential cyber threats.

1.5 ORGANIZATION OF THE THESIS

The rest of the report is organized as follows. Chapter 2 gives the survey of various related work to this project. Chapter 3 describes the architecture design and gives the algorithm used in the module design. Chapter 4 describes the data exploration and sanitization Chapter 5 describes the tokenization and vectorization Chapter 6 describes the model training and prediction Chapter 7 describes the deploy and update related work of the project Chapter 8 gives the results and discussion of project Chapter 9 discuss about conclusion and future work.

CHAPTER 2

LITERATURE SURVEY

Many scholars have done some sort of analysis on the statistics of phishing URLs. Our technique incorporates key concepts from past research. We review past work in the detection of phishing sites using URL features, which inspired our current approach.

Happy et al. [4] describe phishing as "one of the most dangerous ways for hackers to obtain users' accounts such as usernames, account numbers and passwords, without their awareness." Users are ignorant of this type of trap and will ultimately, they fall into Phishing scam. This could be due to a lack of a combination of financial aid and personal experience, as well as a lack of market awareness or brand trust [5].

In this article, Mehmet et al. [6] suggested a method for phishing detection based on URLs. To compare the results, the researchers utilized eight different algorithms to evaluate the URLs of three separate datasets using various sorts of machine learning methods and hierarchical architectures. The first method evaluates various features of the URL; the second method investigates the website's authenticity by determining where it is hosted and who operates it; and the third method investigates the website's graphic presence. We employ Machine Learning techniques and algorithms to analyse these many properties of URLs and websites.

Garera et al. [7] classify phishing URLs using logistic regression over hand-selected variables. The inclusion of red flag keywords in the URL, as well as features based on Google's Web page and Google's Page Rank quality recommendations, are among the features. Without access to the same URLs and features as our approach, it's difficult to conduct a direct comparison.

In this research, Yong et al. [8] created a novel approach for detecting phishing websites that focuses on detecting a URL which has been demonstrated to be an accurate and efficient way of detection. To offer you a better idea, our new capsule-based neural network is divided into 4 several parallel components. One method involves removing shallow characteristics from URLs. The other two, on the other hand, construct accurate feature representations of URLs and use shallow features to evaluate URL legitimacy. The final output of our system is calculated by adding the outputs of all divisions. Extensive testing on a dataset collected from the Internet

indicate that our system can compete with other cutting-edge detection methods while consuming a fair amount of time.

For phishing detection, Vahid Shahrivari et al. [9] used machine learning approaches. They used the logistic regression classification method, KNN, Adaboost algorithm, SVM, ANN and random forest. They found random forest algorithm provided good accuracy. Dr.G. Ravi Kumar [10] used a variety of machine learning methods to detect phishing assaults. For improved results, they used NLP tools. They were able to achieve high accuracy using a Support Vector Machine and data that had been pre-processed using NLP approaches. Amani Alswailem [11] et al. tried different machine learning model for phishing detection but was able to achieve more accuracy in random forest.

Hossein et al. [12] created the "Fresh-Phish" open-source framework. This system can be used to build machine-learning data for phishing websites. They used a smaller feature set and built the query in Python. They create a big, labelled dataset and test several machine-learning classifiers on it. Using machine-learning classifiers, this analysis yields very high accuracy. These studies look at how long it takes to train a model.

X. Zhang [13] suggested a phishing detection model based on mining the semantic characteristics of word embedding, semantic feature, and multi-scale statistical features [13] in Chinese web pages to detect phishing performance successfully. To obtain statistical aspects of web pages, eleven features were retrieved and divided into five classes. To obtain statistical aspects of web pages, eleven features were retrieved and divided into five classes. To learn and evaluate the model, AdaBoost, Bagging, Random Forest, and SMO [13] are utilized. The legitimate URLs dataset came from DirectIndustry online guides, and the phishing data came from China's Anti-Phishing Alliance.

With novel methodologies, M. Aydin [14] approaches a framework for extracting characteristics that is versatile and straightforward. Phish Tank [14] provides data, and Google [14] provides authentic URLs. C# programming and R programming were utilized to obtain the text attributes. The dataset and third-party service providers yielded a total of 133 features. The feature selection approaches of CFS subset based and Consistency subset-based feature selection [14] were employed and examined with the WEKA tool. The performance of the Nave

Bayes and Sequential Minimal Optimization (SMO) algorithms [14] was evaluated, and the author prefers SMO to NB for phishing detection

2.1 SUMMARY OF THE SURVEY

In the rise of internet all over the world, The above cited paper were useful in visualization and understanding the flow of the entire project and also designing of its architecture, input and its format to be given for each module along with expected output were identified. The above cited papers are focused at one point in detecting malicious URL. With reference to all these papers, our project is also focused on detecting malicious URL using logistic regression Classifier.

CHAPTER 3

SYSTEM DESIGN

Software design is a way through which the requirements can be converted into a proper representation. This representation can be used to create architecture and detailed design diagrams through which further process of the project can be carried out.

3.1 PROPOSED SYSTEM

The main scope of this Project is to analyze the effectiveness of chosen machine learning classification technologies within the problem of malicious URL detection. Specifically, we are interested in results using only URL address itself without the need to download potentially risky content of the page. We consider the problem of detecting malicious URLs as a binary classification problem, i.e., URL can be benign or malicious. Determining the type of malicious URL may be possible, but it is usually not the main goal of detection algorithms. In the beginning, we analyzed and selected the set of features that may have a positive impact on the results. In the next step, we extracted and preprocessed these features. We decided to use logistic Regression classifier The reason behind this choice is the success of using logistic Regression, and extremely easy and user-friendly. At the end, we analyzed the results from the use of various settings, features sets, amounts of data, or data from different days.

3.2 HARDWARE REQUIREMENTS

The hardware requirements for executing this model are:

- RAM - 8 GB
- Operating System - Windows 10
- Processor - Intel(R) Core (TM) i3
- Processor speed – 2.30 GHz-3.60 GHz

3.3 SOFTWARE REQUIREMENTS

The programming language used to develop this application Python and the IDE used is jupyter

- Programming Language - Python
- IDE – Jupyter
- Streamlit Framework

3.4 SYSTEM METHODOLOGIES

This research used the linear-sequential model, often known as the waterfall model. Although the waterfall approach is considered conventional, it works best in instances where there are few requirements. The application was built hand-written code. Below figure for waterfall model

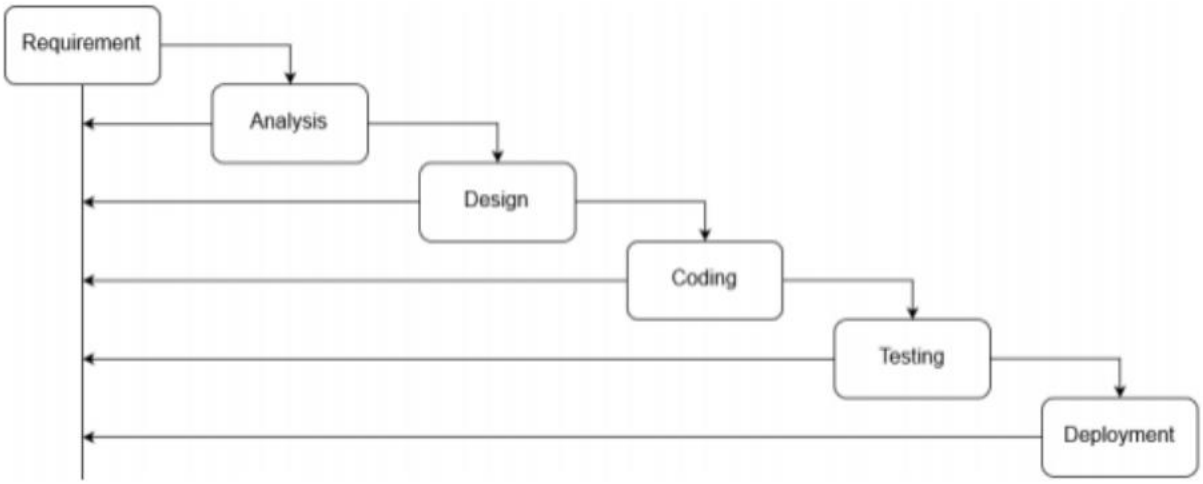

Figure 3.1: System Methodologies

3.5 SYSTEM ARCHITECTURE

The architecture diagram shows the different processes that will be executed in the project and detailed design shows the complete working of each module that can be used for better understanding and executing the project to get the desired output.

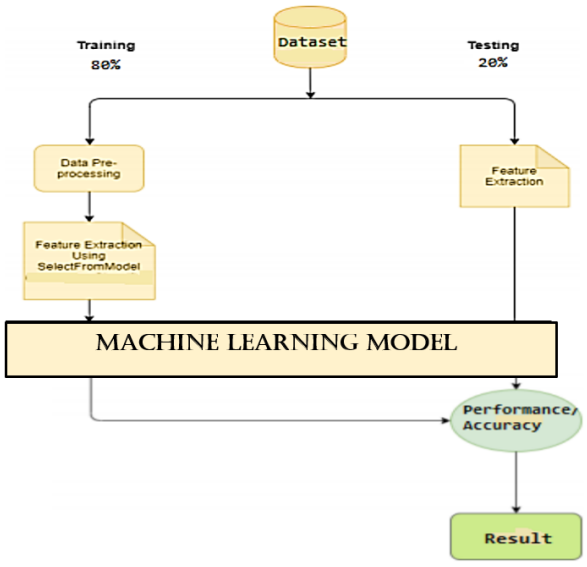

Figure 3.2: Architecture of Malicious Site Detection

CHAPTER 4
DATA EXPLORATION AND SANITIZATION

4.1 DATA COLLECTION AND CLEANING

The phishing URLs were gathered using the opensource tool Phish Tank website Kaggle and whois This site provides a set of phishing URLs in a variety of forms, including csv, json, and others, which are updated hourly. This dataset is used to train machine learning models. Fill in missing numbers, smooth out creaking data, detect and delete outliers, and repair anomalies to clean up the data.

4.2 DATA PRE-PROCESSING

Data pre-processing is a cleaning operation that converts unstructured raw data into a neat, well-structured dataset that may be used for further research. Data pre-processing is a cleaning operation that transforms unstructured raw data into well-structured and neat dataset which can be used for further modules.

4.3 META INFORMATION OF DATAFRAME

In the context of a dataframe, "meta information" refers to data that provides additional information about the structure and content of the dataframe. Some common examples of meta information that may be associated with a dataframe include:

Column names: The names of the columns in the dataframe, which can provide context for the type of data contained in each column.

Data types: The data types of the values in each column, such as numeric, string, or boolean.

Missing values: Information about whether any values in the dataframe are missing, and how those missing values are represented.

4.4 CHECKING FOR NAN VALUES

NaN (Not a Number) is a numeric data type that means an undefined value or value that cannot be represented, especially results of floating-point calculations. For example, NaNs can represent infinity, result of division by zero, missing value, or the square root of a negative (which is imaginary, whereas a floating-point number is real). NaN values can arise in data for a variety of reasons, such as missing values or invalid calculations. Checking for NaN values in data is an important step in data preprocessing and cleaning. Here are some common methods to check for NaN values

4.5 FEATURES EXTRACTION

In the literature and commercial products, there are numerous algorithms and data formats for phishing URL detection. A phishing URL and its accompanying website have various characteristics that distinguish them from harmful URLs. For example, to mask the true domain name, an attacker can create a long and complicated domain name. Different types of features that are used in machine learning algorithms in the academic study detection process are used.

URL length: Malicious URLs are often longer than legitimate URLs, and can contain many random characters or numbers.

Domain age: Newly registered domains are more likely to be used for malicious purposes than older, well-established domains.

Domain reputation: Domains with a history of hosting malicious content are more likely to be used for future malicious activity.

IP reputation: The IP address associated with a domain can also have a reputation for being associated with malicious activity.

Use of subdomains: Malicious actors often use subdomains to obfuscate the true location of a URL.

Use of uncommon or misspelled words: Malicious actors may use uncommon or misspelled words in URLs to try to evade detection.

Presence of certain keywords: Some keywords, such as "phishing" or "malware," may indicate that a URL is malicious.

HTTPS vs HTTP: Malicious URLs are often served over HTTP, while legitimate sites are more likely to use HTTPS.

Redirects: Malicious URLs may use multiple redirects to evade detection or to obfuscate the true location of the final destination.

Presence of certain characters: Some characters, such as "@" or "-", may be indicative of a malicious URL.

CHAPTER 5

TOKENIZATION AND VECTORIZATION

5.1 TOKENIZATION

Tokenization is the process of converting a piece of text into individual tokens or units. These tokens are then used as input features for machine learning algorithms. Assigning unique identifiers to tokens is an important step in the tokenization process as it helps to ensure that each token is represented consistently across the dataset. Tokenization replaces a sensitive data element for example, a bank account number, with a non-sensitive substitute, known as a token. The token is a randomized data string that has no essential or exploitable value or meaning.

There are five steps:

- Cleaning Text,
- Lower Casing,
- Splitting the Text,
- Assigning Unique Identifiers,
- Creating Token Dictionary.

5.1.1 Cleaning Text

The first step in tokenization is to clean the text by removing any unwanted characters such as punctuation marks, special characters, and numbers. This is done to ensure that the resulting tokens only contain valid words.

5.1.2 Lower Casing

In many cases, it is useful to convert all the text to lowercase to avoid having different tokens for the same word with different cases.

5.1.3 Splitting the text

The text is then split into words, phrases, or sentences. This can be done in a number of ways, such as using spaces, punctuation, or specific delimiters.

```
import re
    re.split(r"[\./-:]","https://facebook.com")
```

Output:

```
['https', 'facebook', 'com']
```

5.1.4 Assigning unique identifiers

Each token is assigned a unique identifier, or token, that can be used to represent it in the further analysis. This allows the text to be analyzed, searched, and manipulated in a variety of ways, depending on the application.

5.1.5 Creating a token dictionary:

A token dictionary is created that maps each token to its corresponding identifier. This dictionary is used to identify and count the frequency of each token in the text.

Make Tokens("https://facebook.com").

Output: ['facebook',’com’]

There are different methods of tokenization, such as rule-based tokenization, statistical tokenization, and neural network-based tokenization. Rule-based tokenization involves using a set of predefined rules to split the text into tokens. Statistical tokenization involves using statistical models to learn patterns in the text and splitting it accordingly. Neural network-based tokenization involves using deep learning algorithms to automatically learn how to tokenize text.

5.2 VECTORIZATION

Vectorization, on the other hand, is the process of converting these tokens into numerical vectors that can be processed by machine learning algorithms. The most common approach to vectorization is the Bag-of-Words model, which represents each token as a unique feature and counts the frequency of each token in a document. This results in a high-dimensional vector where each dimension corresponds to a unique token.

There are three major methods for performing vectorization on text data:

1.CountVectorizer

2.TF-IDF

3. Word2Vec

5.2.1 Term Frequency — Inverse Document Frequency (TFIDF)

TF-IDF or Term Frequency–Inverse Document Frequency, is a statistical measure that tells how relevant a word is to a document. It combines two metrics - term frequency and inverse document frequency - to produce a relevance score. The Term Frequency is the frequency of a word in a document. It is calculated by dividing the occurrence of a word inside a document by

the total number of words in that document. The Inverse Document frequency is a measure of how much information a word provides. It is the product of TF and IDF.

- TFIDF gives more weightage to the word that is rare in the corpus.
- TFIDF provides more importance to the word that is more frequent in the document.

$$TFIDF\ (w, d, D) = TF(w, d) * IDF(w, D)$$

Sample function for vectorization:

vectorizer = TfidfVectorizer(tokenizer=makeTokens)

```
X = vectorizer.transform(df["URL"])
y = df["Type"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
print(X)
```

```
  (0, 11497)    0.25139743402664766
  (0, 8375)     0.9678839445743572
  (1, 37062)    0.6702329103333574
  (1, 30113)    0.45135935245786685
  (1, 29826)    0.5438555278586417
  (1, 22054)    0.20388458199193188
  (1, 11497)    0.09856380115674397
  (2, 32787)    0.3000433105271359
  (2, 21995)    0.35548702665952575
  (2, 12375)    0.8852135254754563
  (3, 48333)    0.07712003142149557
  (3, 34554)    0.26002514879359806
```

Figure 5.1: Sample Function for vectorization

CHAPTER 6

MODEL TRAINING AND PREDICTION

6.1 MODEL TRAINING

Machine learning is a branch of Artificial Intelligence which deals with teaching computers the ability to learn and improve from experience. The primary aim is for the machine to be able to access data and use it to learn and discover patterns in it which can then be used to make predictions, perform categorization and clustering. ML is broadly classified into two types – supervised and unsupervised machine learning:

• Supervised Machine learning uses a labelled set of examples to learn patterns and relationships between the data and the outcome. It then uses these learnings to make predictions for new data

• Unsupervised Machine Learning tries to uncover the hidden structure from data that is unlabeled. Here it is not about figuring out the right output but instead the focus is on drawing inferences from the datasets.

In this project we will use Supervised Machine Learning to learn patterns that will help us in predicting whether a website is malicious or benign. We will do this using the labelled dataset available. Below I describe a Logistic Regression supervised learning algorithms that we will use to train our machine learning model.

6.1.2 Logistic Regression

LG is a machine learning algorithm used to train classifiers. It is basically the logistic/sigmoid function layered on top a linear regression model. Mathematically it is represented as below:

$$z = wx + b$$

$$y = sigmoid(z)$$

$$sigmoid(z) = 1\ 1 + e\ -t$$

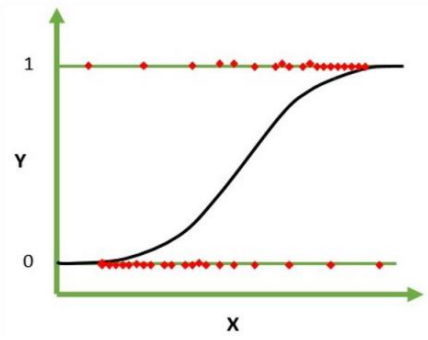

Figure 6.1: The Sigmoid Function

Figure 6.1 shows that as the value for z gets larger and larger the value of Y tends to 1. On the other hand, as the value of z gets smaller and smaller the value of Y tends to 0. This means that the output of this model is always in the range of 0 to 1 which basically gives us the probability of an observation being 1 or 0.

Logistic regression calculates the probability of a binary outcome and by setting a threshold, it classifies the data points into either outcome. Our data will have a binary outcome as we must decide whether a website is malicious or not.

6.1.3 Training Model Logistic Regression

```
X = vectorizer.transform(df["URL"])
y = df["Type"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# model training
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
with open("logistic_regression_model.pkl","wb") as f:
    pickle.dump(model,f)
```

```
with open("logistic_regression_model.pkl","rb") as f:
    model = pickle.load(f)
```

```
print("Train Score: ",model.score(X_train,y_train))
print("Test Score: ",model.score(X_test,y_test))
```

```
Train Score:  0.9278581930292282
Test Score:  0.923975301707263
```

Figure 6.2: Sample Training Model

6.1.4 Libraries Used

PANDAS: It's a Python-based machine learning library. Pandas is a free and open-source programming language. Pandas is a programming language that is commonly used for dataset loading and data analytics. Pandas is used for machine learning in a variety of domains, including economics, finance, and others. It is extremely user-friendly and can display datasets in a tabular style for easier comprehension.

SciKit LEARN: Scikit learn is one of the most essential Python libraries for machine learning. Scikit learn includes several tools for statistical classification, modelling, regression, dimensionality reduction and clustering.

NUMPY: Numpy is a Python-based machine learning package. In Python, Numpy is used to deal with arrays. NumPy is used for all calculations using 1-d or 2-d arrays. Numpy also has routines for working with linear algebra and the Fourier transform.

MAPTLOTLIB: MAPTlotlib is a library for data visualization. It's a Python open-source module for plotting graphs from model results. These diagrams can aid in comprehending the circumstance of the outcomes. For easier comprehension, several components of the results can be graphically formatted.

WHOIS : Whois is a library that provides a way to retrieve and parse WHOIS records for domain names, IP addresses, and other network entities. WHOIS is a protocol used for querying databases that contain information about domain names and IP addresses, including the name and contact information of the owner, the registrar, and the server administrator.

SEABORN: Seaborn is a popular Python data visualization library built on top of Matplotlib. It provides a high-level interface for creating attractive and informative statistical graphics. Seaborn is designed to work well with data frames from Pandas, making it an ideal tool for data analysis tasks.

REQUESTS: Requests library in Python provides a simple and powerful way to make HTTP requests from Python code. It is used to communicate with web services, APIs, and other HTTP-based services.

DATETIME: The datetime library in Python provides functionality for working with dates and times.

6.2 PREDICTION

Since this is a classification problem, we will use the confusion matrix to evaluate the model we train. A confusion matrix is a 2-dimensional table that puts the actual value of an observation on the rows while the predicted values of an observation are on the columns. An example of a confusion matrix for binary classification (classification problem with only two possible outcomes)

```
In [20]: import matplotlib.pyplot as plt

In [37]: cf_matrix=confusion_matrix(y_test,y_pred)

         plot_ = sns.heatmap(cf_matrix,annot = True,fmt = "d")
         plt.xlabel ("Predicted Condition")
         plt.ylabel("True Condition")
         plt.show()
```

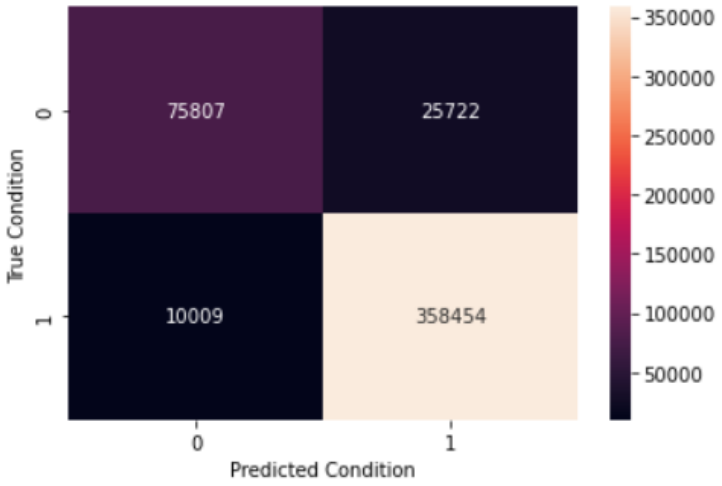

Figure 6.3: Confusion Matrix of the model

Below we define key terminology and metrics associated with a confusion matrix

True Positive: These are observations that we predicted as 'YES' and were actually 'YES' as well. These are present on the top left cell of the matrix in Figure 6.

True Negative: These are observations that we predicted as 'NO' and were actually 'NO' as well. These are present on the bottom right cell of the matrix in Figure 6.

False Positive: These are observations that we predicted as 'YES' and were actually 'NO'. These are present on the bottom left cell of the matrix in Figure 6. These are also known as Type I error for a model.

False Negative: These are observations that we predicted as 'NO' and were actually 'YES'. These are present on the top right cell of the matrix in Figure 6. These are also known as Type II error for a model.

Recall: This tells us how many of the actual positive observations in our database were we able to predict correctly. This is also known as Sensitivity of a model.

$$\text{Recall} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalseNegative}}$$

Precision: This tells us how many of the observations that we have predicted to be positive are actual positives.

$$\text{Precision} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalsePositive}}$$

Specificity: This tells us how many of the actual negative observations in our dataset we were able to predict correctly.

$$\text{Specificity} = \frac{\text{TrueNegative}}{\text{TrueNegative} + \text{FalsePositive}}$$

Accuracy: This tells us how many observations we predicted correctly regardless of whether they are negative or positive.

$$\text{Accuracy} = \frac{\text{TruePositive} + \text{TrueNegative}}{\text{TruePositive} + \text{TrueNegative} + \text{FalsePositive} + \text{FalseNegative}}$$

F1-Score is a measure that measure the balance between recall and precision of a model simultaneously. It is an important measure as it can be difficult to compare models with high recall and low precision to models with low recall and high precision. We will be using the F1 Score to evaluate the models we train.

$$\text{F1} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

```
# print all the metrics
print("accuracy:",accuracy)
print("precision:",precision)
print("recall:",recall)
print("f1 score:",f1)
#print("confusion matrix",confusion_matrix(y_test,y_pred))
```

```
accuracy: 0.923975301707263
precision: 0.933046312106951
recall: 0.9728358071231032
f1 score: 0.9525257128583556
```

Figure 6.4: Evaluation Metrics Of The Model

For this project, we will need to ensure that our model is able to detect as many of the malicious websites as possible, even if it comes at the cost of predicting some benign websites as malicious. This is because classifying a malicious website as benign is very costly while classifying a benign website as malicious is not that costly.

We set a threshold of 20% such that any website whose probability of being malicious was more than 20% was classified as malicious. This low threshold was chosen keeping in mind that it is more important to correctly classify malicious websites as compared to correctly classifying benign websites.

The results and predictions generated by model trained using algorithm were tabulated as a confusion matrix. The models were compared based on F1-score on the test data, which strikes a balance between precision and recall values.

CHAPTER 7
DEPLOY AND UPDATE

7.1 DEPLOY

Model deployment using a pickle file involves serializing the trained model into a byte stream using the pickle module in Python. This serialized object can then be stored as a file, which can be easily transported and loaded into a production environment. The pickle file contains all the information necessary to reconstruct the trained model, including the model architecture, trained weights, and any other configuration settings. During deployment, the pickle file is loaded into the production environment, and the trained model is reconstructed from the serialized object. This allows for a fast and efficient deployment process, as the serialized model can be easily transported and loaded without the need for retraining or additional configuration.

7.1.1 Pickle

Pickle is a Python module used for serializing and deserializing Python objects. Serialization is the process of converting an object into a byte stream, while deserialization is the process of reconstructing the object from the byte stream. Pickle can be used to serialize a wide range of Python objects, including lists, dictionaries, functions, classes, and even custom objects.One of the main advantages of pickle is that it allows for the easy transport and storage of Python objects. For example, trained machine learning models can be serialized using pickle and then stored as a file, which can be easily transported and loaded into another environment. This makes it a popular choice for model deployment and sharing.

```
# model training
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
with open("logistic_regression_model.pkl","wb") as f:
    pickle.dump(model,f)
```

```
with open("logistic_regression_model.pkl","rb") as f:
    model = pickle.load(f)
```

```
print("Train Score: ",model.score(X_train,y_train))
print("Test Score: ",model.score(X_test,y_test))
```

```
Train Score:  0.9278581930292282
Test Score:  0.923975301707263
```

Figure 7.1: Sample Model Fit

7.1.2 Streamlit

Streamlit is an open-source Python library that allows you to create web applications in minutes. It's designed to help data scientists and machine learning engineers easily create and deploy interactive data applications, without needing to have expertise in web development.

Streamlit makes it easy to create a user interface for your Python code, enabling you to quickly build interactive dashboards and visualizations. The library includes a set of pre-built widgets, such as sliders, buttons, and dropdown menus, which can be easily added to your app to allow for user input and interaction.

```python
import pickle
import subprocess
import re
import streamlit as st
import requests
import whois
from datetime import datetime
from PIL import Image
img = Image.open('iconlogo-removebg-preview.png')

# st.markdown('<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha3

# st.markdown('<img src="file:///C:/Users/ARUN/Desktop/Malicious/StreamLit/aranlogo.png" alt="" width="50" height="50" >',unsafe_allow_html

st.set_page_config(page_title="Aran",page_icon=img,layout="wide")


with open('style.css') as f:
    st.markdown(f'<style>{f.read()}</style>',unsafe_allow_html=True)

def makeTokens(f):
    total_tokens = re.split(r"[\./-:]",f)
    total_tokens = [token for token in total_tokens if token not in {"https","http",""}]  #remove redundant tokens
    return total_tokens

def get_certificate(long_url):
    try:
        response = requests.get(long_url)
        if response.ok:
            value = "The SSL certificate is valid."
```

Figure 7.2: Sample Code

7.1.3 Deployed Website Sample

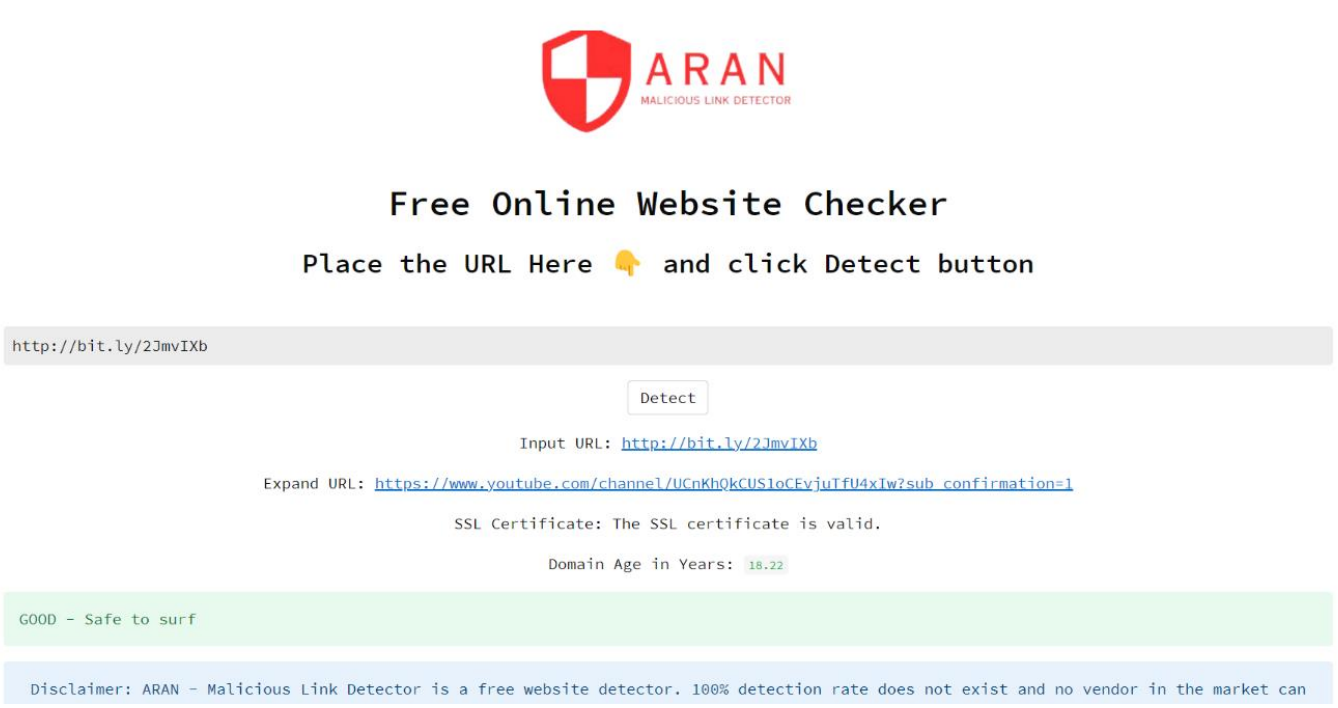

Figure 7.3: Deployed Sample for Good URL

Figure 7.4: Deployed Sample for Bad URL

7.2 UPDATE

Updating module in a project involves a series of steps to enhance the existing functionality and ensure the overall security of the application. The first step is to analyze the existing codebase and identify areas for improvement. This may involve reviewing the code to identify any bugs or inefficiencies, reviewing the list of known malicious URLs to ensure it is up-to-date, and assessing the current criteria used to identify suspicious URLs.

Once areas for improvement have been identified, the next step is to implement changes to improve the accuracy and efficiency of the detection algorithm. This may involve updating the list of known malicious URLs to include new threats, refining the criteria used to identify suspicious URLs, or incorporating machine learning techniques to improve the detection capabilities.

CHAPTER 8

RESULTS AND DISCUSSION

8.1 MODEL COMPARISON

Here we can see the performance comparison for all models on the testing set. It is easy to see that all models except SVM show reasonably high accuracy. When evaluated according to the F1- score, we see that the Logistic Regression -based model outperforms the rest. The other models all have a very similar performance.

Training Algorithm	Algorithm Sensitivity	Precision	Recall	F1	Accuracy
Logistic_Regression	0.97	0.93	0.97	0.95	0.92
Decision_Tree	0.83	0.56	0.83	0.67	0.87
Random_Forest	0.77	0.56	0.77	0.65	0.84
Support_Vector_Machine	0.55	0.82	0.55	0.66	0.77

Table 1. Comparison of Models on the testing set

8.2 DISCUSSION

Variables important in detecting malicious websites We used a variety of different variables to train classification models based on 4 different algorithms. Across the four different models, we observed a few variables that were considered critical by all. Since these variables are being considered important by each model, it is worth investigating this importance. Number of Special Characters Represented by the variable 'NUMBER_SPECIAL_CHARACTERS', this variable was considered important by each one of our classification models. This makes a lot of sense as special characters are not typically used in the websites we use often. One reason for this could be that the special characters make the URL stand out and this could maybe increase clicks. Another reason for this could be because a lot of malicious websites can contain currency symbols, which are again treated as special characters. Overall, a large number of special characters found in the URL of a website does serve as an indicator that the website is likely to be malicious.

8.2.1 Domain Age

It was seen quite clearly that the age of a website had an effect on the likelihood of a website being malicious or not. Older websites were predicted as more likely to be malicious than newer ones given all other information remained the same. This again makes sense as older website created using legacy code may possess many insecure bits of code that can leave it open to malicious attacks.

8.3 MACHINE LEARNING MODEL COMPARISON

We observed that the Logistic regression was the best performing model out of the 4 models we trained. This makes sense due to the following reasons:

• Logistic regression is a linear model as well as heavily sensitive to the distributions of predictor variables. It can be the case that the relationships between the predictors and the target are non-linear in nature have been shown to perform extremely robustly in a variety of classification problems.

• Decision trees are very simplistic and many decision trees come together to for a random forest.

• Support Vector machine is also a linear model. Finding one decision boundary with so many dimensions can be difficult.

The best F1-score we achieved on the test set was equal to 0.95. This is score while fairly decent can definitely be improved upon. One clear way of improving this will be to use sampling techniques to tackle the problem of class imbalance as only 30% of the websites in our data were malicious. Oversampling of the malicious websites can help in allowing the model to learns patterns that increase the likelihood of a website being malicious much better. Another way of improving the model performance could by using newer and more sophisticated training algorithms such as gradient boosters and neural networks.

CHAPTER 9
CONCLUSION AND FUTURE WORK

9.1 CONCLUSION

We cleaned and prepared a dataset containing malicious and benign websites and used it to train a classification model which predicts whether a website will be malicious or not based on a selection of features. We have showcased that it is possible to detect malicious websites with a reasonable amount of certainty (90% of the 100 malicious websites in the test set were identified) using machine learning models. We have also determined the features that were critical in predicting the likelihood of a website being malicious. Most of our key features are easily available (URL Length, number of Special characters, Age of website). Further improvements can be made on our results by using sampling techniques to deal with class imbalance and using more sophisticated models.

9.2 FUTURE WORK

Right now we are the using mere URL and its features like URL length, Domain age, Special Characters etc. to train a model and to detect whether the URL is malicious or benign. In Future we are planning to improving the predictions by reviewing its content and use the neural network to identify whether the URL is malicious or benign. We planning to deploy it as a chrome extension in the future to aid user instead of giving the URL manually to the website.

REFERENCES

[1] ‘APWG | Unifying The Global Response To Cybercrime’ (n.d.) available: https://apwg.org/

[2] 14 Types of Phishing Attacks That IT Administrators Should Watch For [online] (2021) https://www.blog.syscloud.com, available: https://www.blog.syscloud.comtypes-of-phishing/

[3] Lakshmanarao, A., Rao, P.S.P., Krishna, M.M.B. (2021) ‘Phishing website detection using

novel machine learning fusion approach’, in 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), Presented at the 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), 1164–1169

[4] H. Chapla, R. Kotak and M. Joiser, "A Machine Learning Approach for URL Based Web Phishing Using Fuzzy Logic as Classifier", 2019 International Conference on Communication and Electronics Systems (ICCES), pp. 383-388, 2019, July

[5] Vaishnavi, D., Suwetha, S., Jinila, Y.B., Subhashini, R., Shyry, S.P. (2021) ‘A Comparative

Analysis of Machine Learning Algorithms on Malicious URL Prediction’, in 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), Presented at the 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), 1398–1402.

[6] M. Korkmaz, O. K. Sahingoz and B. Diri, "Detection of Phishing Websites by Using Machine Learning-Based URL Analysis", 2020 11th International Conference on Computing Communication and Networking Technologies (ICCCNT), pp. 1-7, 2020, July

[7] Garera S., Provos N., Chew M., Rubin A. D., "A Framework for Detection and measurement of phishing attacks", In Proceedings of the ACM Workshop on Rapid Malcode (WORM), Alexandria, VA

[8] Y. Huang, J. Qin and W. Wen, "Phishing URL Detection Via Capsule-Based Neural Network", 2019 IEEE 13th International Conference on Anti-counterfeiting Security and Identification (ASID), pp. 22-26, 2019, October

[9] Vahid Shahrivari, Mohammad Mahdi Darabi and Mohammad Izadi, "Phishing Detection

Using Machine Learning Techniques", arXiv:2009.11116v1 [cs.CR], Sep 2020

[10] PhishTank > Developer Information [online] (2021) available: https://www.phishtank.com/developer_info.php

[11] G. Ravi Kumar, S. Gunasekaran and R Nivetha, "URL Phishing Data Analysis and Detecting Phishing Attacks Using Machine Learning In NLP", International Journal of Engineering Applied Sciences and Technology-2019, vol. 3, no. 10

[12] Amani Alswailem, Norah Alrumayh, Bashayr Alabdullah and Aram Alsedrani, "Detecting
Phishing Websites Using Machine Learning", International Conference on Computer Applications & Information Security (ICCAIS), vol. 97, 2019

[13] Hossein Shirazi, Kyle Haefner, Indrakshi Ray: Fresh-Phish: A Framework for AutoDetection of Phishing Websites: In (International Conference on Information Reuse and Integration (IRI)) IEEE,2017

[14] X. Zhang, Y. Zeng, X. Jin, Z. Yan, and G. Geng, “Boosting the Phishing Detection Performance by Semantic Analysis,” 2017

[15] M. Aydin and N. Baykal, “Feature extraction and classification phishing websites based on URL,” 2015 IEEE Conf. Commun. NetworkSecurity, CNS 2015, pp. 769–770, 2015

[16] Rami M. Mohammad, Fadi Thabtah and Lee McCluskey, Phishing Websites Features, 2014

[17] I. Tyagi, J. Shad, S. Sharma, S. Gaur and G. Kaur, "A Novel Machine Learning Approach
to Detect Phishing Websites", 5th International Conference on Signal Processing and Integrated Networks (SPIN), 2018

[18] El-Alfy, E.-S.M. (2017) ‘Detection of Phishing Websites Based on Probabilistic Neural Networks and K-Medoids Clustering’, The Computer Journal, 60(12), 1745–1759

[19] Kramer, M. (2018) Best Practices in Systems Development Lifecycle: An Analyses Based
on the Waterfall Model, SSRN Scholarly Paper ID 3131958, Social Science Research Network,

Rochester, NY, available: https://papers.ssrn.com/abstract=3131958

[20] Fumo, D., 2017. Pandas Library in a Nutshell — Intro To Machine Learning. [Online] Available at: https://medium.com/simple-ai/pandas-library-in-a-nutshell-intro-tomachinelearning-3-acbd39ec5c9c

[21] JAIN, K., 2015. Analytics VIdya. [Online] Available at: https://www.analyticsvidhya.com/blog/2015/01/scikit-learn-pythonmachine-learning-tool/

[22] Anon., n.d. NumPy Introduction. [Online] Available at: https://www.w3schools.com/python/numpy_intro.asp

[23] Anon., 2018. Python | Introduction to MAPTlotlib. [Online] Available at: https://www.geeksforgeeks.org/python-introduction-mAPTlotlib/

[24] Feature Selections for the Machine Learning Based Detection of Phishing Websites | IEEE

Conference Publication | IEEE Xplore [online] (2021) available: