

MACHINE LEARNING BASED PHISHING WEBSITE DETECTOR

BY

Sayed Abdus Sobur
ID: 221-25-094

This Report Presented in Partial Fulfillment of the Requirements for the Degree of
Masters of Science in Computer Science and Engineering

Supervised By

Professor Dr. Touhid Bhuiyan
Head
Dept. of Computer Science & Engineering
Daffodil International University

Co-Supervised By

Mr. Abdus Sattar
Assistant Professor & Coordinator M.Sc
Dept. of Computer Science & Engineering
Daffodil International University



DAFFODIL INTERNATIONAL UNIVERSITY

DHAKA, BANGLADESH

17th JANUARY 2023

APPROVAL

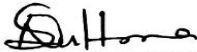
This Thesis titled “**Machine Learning Based Phishing Website Detector**”, submitted by Sayed Abdus Sobur, ID No: 221-25-094 to the Department of Computer Science and Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of M.Sc. in Computer Science and Engineering and approved as to its style and contents. The presentation has been held on 17-01-2023.

BOARD OF EXAMINERS



Dr. S M Aminul Haque, PhD
Associate Professor & Associate Head
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Chairman



Ms. Naznin Sultana
Associate Professor
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Internal Examiner



Mr. Md. Sadekur Rahman
Assistant Professor
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Internal Examiner



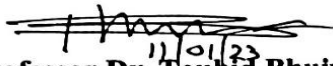
Dr. Mohammad Shorif Uddin, PhD
Professor
Department of Computer Science and Engineering
Jahangirnagar University

External Examiner

DECLARATION

We hereby declare that, this thesis has been done by us under the supervision of **Professor Dr. Touhid Bhuiyan, Head, Department of CSE Daffodil International University**. We also declare that neither this thesis nor any part of this thesis has been submitted elsewhere for award of any degree or diploma.

Supervised by:



Professor Dr. Touhid Bhuiyan
Head
Dept. of Computer Science & Engineering
Daffodil International University

Co-Supervised by:



Mr. Abdus Sattar
Assistant Professor & Coordinator M.Sc.
Dept. Of Computer Science & Engineering
Daffodil International University

Submitted by:



Sayed Abdus Sobur
ID: 221-25-094
Department of CSE
Daffodil International University

ACKNOWLEDGEMENT

First I express my heartiest thanks and gratefulness to almighty God for His divine blessing make me possible to complete the final year thesis successfully.

I really grateful and wish my profound my indebtedness to **Professor Dr. Touhid Bhuiyan, Head,** Department of CSE, Daffodil International University, Dhaka. Deep Knowledge & keen interest of my supervisor in the field of “*Machine Learning*” to carry out this thesis. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism , valuable advice, reading many inferior draft and correcting them at all stage have made it possible to complete this thesis.

I would like to express my heartiest gratitude to Head, Department of CSE, for his kind help to finish my thesis and also to other faculty member and the staff of CSE department of Daffodil International University.

ABSTRACT

Cyber-attacks have increased in number along with the growth of online services. Phishing, in which attempts are made to steal confidential information by pretending to be a legitimate source, is one of the most prevalent and successful attacks. The success of phishing sites is based on manipulating human emotions, which causes worries and creates an urgent situation by warning that failure to act promptly could result in significant losses in data and money. Because of this, we cannot rely solely on humans to identify phishing; instead, we need more efficient and automatic phishing detection systems. Although many detectors have been suggested, there needs to be more work done due to a large number of phishing websites. In order to increase the accuracy of phishing detection, we propose a phishing site classifier model in this study that uses multinomial naive bayes, logistic regression, and natural language processing over a url text. This study demonstrated the success of the algorithm in increasing the precision of phishing site detection, and the literature will demonstrate this algorithm's success in url text classification. The classifier will be put to the test using supervised learning. The classifier will become efficient in identifying phishing sites using url text among the current detection methods, and it will do so quickly and with a high degree of accuracy, according to experimental tests.

TABLE OF CONTENTS

CONTENTS	PAGE
Approval Page	i
Declaration	ii
Acknowledgement	iii
Abstract	iv
List of Figures	vii-viii
List of Table	ix
CHAPTER	1-5
CHAPTER 1: INTRODUCTION	
1.1 Introduction	1-3
1.2 Motivation	3-4
1.3 Rationale of the Study	4
1.4 Research Questions	4
1.5 Expected output	4-5
1.7 Report Layout	5
CHAPTER 2: BACKGROUND	6-11
2.1 Preliminaries	6
2.2 Related works	6-9
2.3 Comparative Analysis & Summary	9
2.4 Scope of the Problem	10
2.5 Challenges	10-11

CHAPTER 3: RESEARCH METHODOLOGY	12-24
3.1 Research Subject & Instrumentation	12
3.2 Data Collection Procedure	12-13
3.3 Applied Mechanism	13-23
3.4 Implementation Requirements	23-24
CHAPTER 4: EXPERIMENTAL RESULTS & DISCUSSION	25-35
4.1 Experimental Setup	25
4.2 Experimental Result & Analysis	25-35
4.3 Discussion	35
CHAPTER 5: IMPACT ON SOCIETY, ENVIRONMENT & SUSTAINABILITY	36-37
5.1 Impact on Society	36
5.2 Impact on Environment	36
5.3 Ethical Aspects	36
5.4 Sustainability	37
CHAPTER 6: SUMMARY, CONCLUSION & IMPLICATION OF FUTURE RESEARCH	38
6.1 Summary of the Study	38
6.2 Conclusions	38
6.3 Implication for Further Study	38
REFERENCES	39-41
APPENDIX	42-50

LIST OF FIGURES

TITLE	PAGE
Figure 1.1: Current Phishing Situation	1
Figure 1.2 : PayPal signup page URL.	3
Figure 3.1: Bad URL from dataset	12
Figure 3.2: Good URL's from dataset	13
Figure 3.3: Working Mechanism	13
Figure 3.4: CountVectorizer Working System	15
Figure 3.5: Stemming VS Lemmatization	17
Figure 3.6: Logistic Regression Working Procedure	18
Figure 3.7: Naive Bayes Classifier	19
Figure 3.8: MNB Classification System	21
Figure 3.9: Confusion matrix	22
Figure 4.1: Dataset Ratio	25
Figure 4.2: RegexpTokenizer	26
Figure 4.2.1: RegexpTokenizer Output	26
Figure 4.3: Before using SnowballStemmer	27
Figure 4.3.1: After using SnowballStemmer	28
Figure 4.4: URLs that are not attempts at phishing	29
Figure 4.5: URLs that are attempts at phishing	30
Figure 4.6: Confusion Matrix of Logistic Regression	31

Figure 4.7: Confusion Matrix of Multinomial Naive Bayes	32
Figure 4.7: Accuracy difference between Logistic Regression and Multinomial Naive Bayes	33
Figure 4.8: Confusion Matrix of Best Fit Model	34
Figure 4.9: Model Testing Result	35

LIST OF TABLES

NAME	PAGE
Table 4.1: Logistic Regression Accuracy	31
Table 4.2: Logistic Regression Classification Result	31
Table 4.3: Multinomial Naive Bayes Accuracy	32
Table 4.4: Multinomial Naive Bayes Classification Result	32
Table 4.5: Best Fit Model Accuracy	33
Table 4.6: Best Fit Model Classification Result	34

CHAPTER 1

INTRODUCTION

1.1 Introduction

In this age of technology, people are able to communicate with one another through the use of the internet and electronic devices such as desktop computers, laptops, and personal digital assistants (PDA). E-banking as well as e-commerce shopping have been chosen by most of its users due to their comfort ability, availability, and convenience of use. This preference was brought about by the revolution brought about by the internet. Owing to the fact that all of these dealings and conversations are conducted over a channel that is exposed to the public and is therefore inherently insecure. The malicious actor is attempting to seize control of the vulnerable system in order to carry out a variety of attacks on the users' transactions while they are in the process of completing them. An example of this kind of attack is known as a "phishing attack." In this kind of attack, the intruder attempts to obtain sensitive and personal information from the user by imitating reliable websites in order to redirect the link toward the intruder. The intruder tries to force the genuine user into believing that they are accessing a fraudulent website during this Phishing attack by creating a fake version of the reliable webpage that is under their control. When a genuine user provides their personal information to a fraudulent website, the information that the user provides is immediately and covertly captured by the attacker in the background. The perpetrator of the phishing attempt will have access to all of the sensitive information if they are successful in obtaining this information.

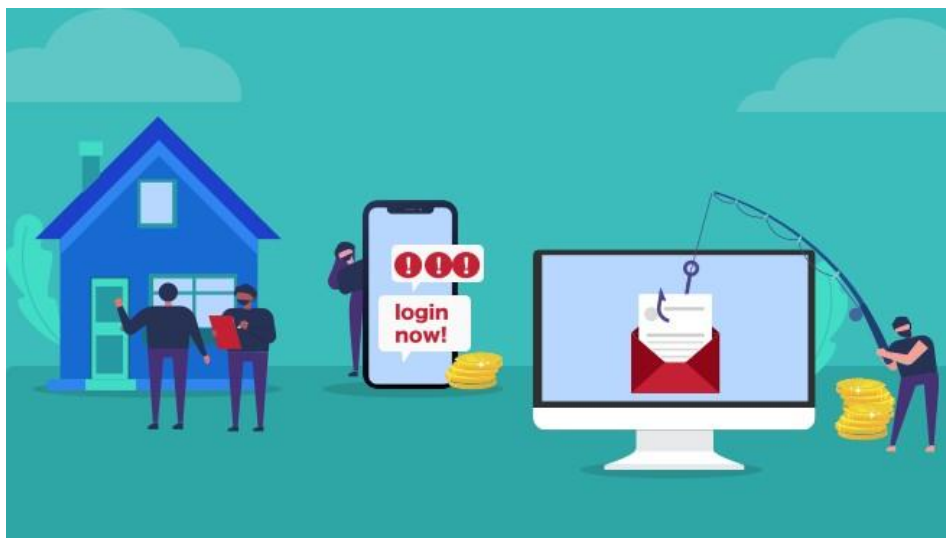


Figure 1.1: Current Phishing Situation

It is common practice to refer to resources found on the Internet by their Uniform Resource Locator (URL) addresses. Sahoo et al. presented information in [22] regarding the characteristics of the URL as well as its two fundamental components, which are as follows: the protocol identifier, which specifies the protocol that will be used, and the resource name, which indicates the IP address or the domain name where the resource can be found. It is clear that each URL has its own unique structure and arrangement of components. Attackers frequently attempt to modify one or more components of the structure of the URL in order to fool users into propagating their malicious URL. Links that are known to have a negative impact on users are referred to as malicious URLs. Users who visit these URLs will be redirected to resources or pages on which cybercriminals can run codes on them' computers, send users to undesirable websites, harmful websites, or other phishing sites, or download malware. In addition, malicious URLs can be concealed within download links that are believed to be safe, and they can quickly propagate through the sharing of files and messages on shared networks. [23, 24, 25] Drive-by Download, Phishing as well as Social Engineering, and Spam are all examples of attack tactics that make use of malicious URLs. According to the data that is published in [26], the assaults that make use of the technique of propagating malicious URLs are now placed first among the top 10 most popular attack strategies that are being used in 2019. The use of harmful URLs, botnet URLs, and phishing URLs are the three primary methods of distributing URLs, and according to this data, all three of these approaches are seeing an increase in the amount of assaults they are causing as well as the risk level they provide. It is abundantly obvious, based on the statistics that demonstrate a growth in the number of harmful URL distributions over the course of successive years, that there is a need to investigate and put into practice strategies or processes that can identify and prohibit the use of dangerous URLs. Concerning the issue of recognizing dangerous URLs, there are now two primary trends: the first is the detection of malicious URLs based on indicators or sets of rules, and the second is the detection of malicious URLs based on approaches that include behavior analysis [22, 23]. The technique of identifying harmful URLs based on a set of markers or criteria enables for the detection of malicious URLs to occur both rapidly and correctly. On the other hand, this approach is not able to identify newly created malicious URLs since it relies on a set of predetermined signs and rules and cannot detect new rules or signs. The approach of detecting dangerous URLs based on behavior analysis techniques use machine learning or deep learning algorithms to classify URLs based on the behaviors they

exhibit. This method was developed to detect URLs that lead to malicious websites. In this article, machine learning methods are used to classify URLs according to the properties of those URLs. In addition to that, a novel approach for extracting URL attributes is presented in the study.

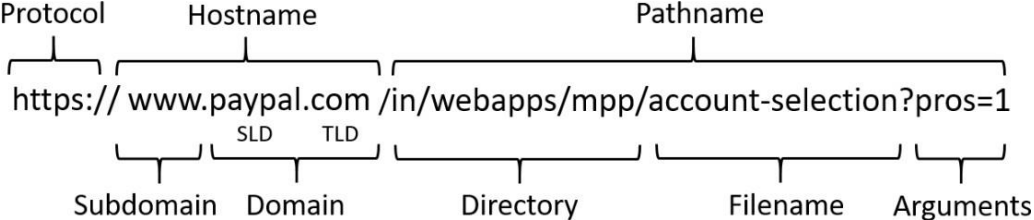


Figure 1.2 : PayPal signup page URL.

1.2 Motivation

There are several distinct varieties of phishing assaults, each of which may be employed by cybercriminals in a variety of contexts and for a variety of goals. The financial industry is consistently targeted by phishing scams more than any other sector. When a person authenticates themselves to their online banking utilizing their username and password, a phishing attack is most likely to occur in this sector of the internet. At this point in time, the attackers are creating the replication of both the URL and the webpage in order to trick users into entering their credentials on bogus websites that have been copied. The credentials of the users will be recorded in this manner, and those users will be able to get access control towards the user account without the user needing to worry about it. The next type of phishing assault typically targets websites that are involved in online commerce. Intruders develop copies of authentic websites in order to trick people into conducting business on their phony websites and steal their personal information. After the transaction has been completed, the adversaries make a note of their credentials, such as their login and password, as well as the transaction parameters, such as the ATM card information, pin number, as well as CVV number. As a result of the attacker caring about these actions, the attacker gains access to the system and is able to carry out the operation on behalf of the genuine user without the genuine user's knowledge or consent. These are the kinds of situations in which a phishing attempt might endanger the users who are really

Authorized to access the system. In order to keep track of the various phishing attacks that take place in different parts of the world, a non-profit organization known as the Anti-Phishing Working Group was established. This organization is responsible for conducting in-depth investigations of the various phishing attacks that take place and publishing their findings so that users can identify malicious websites. In most cases, cybercriminals will construct phony websites, which they will then distribute to victims in the form of links via social networking sites such as Facebook, Instagram, WhatsApp, and LinkedIn. When a user clicks on the link, they are immediately sent to one of the phony websites, where their personal information may be recorded. The most recent Phishing assaults are quite powerful, and they are even capable of breaching the security services provided by various protocols like HTTPS and SSL. As a result, the already-in-place security mechanisms are no longer reliable. In order to circumvent the restrictions imposed by previously implemented systems, the unique solutions presented in this work are intended to: On top of that, the proposed method uses the URL-based attributes as that of the input for the machine learning based classification algorithm. By doing so, the proposed method is able to successfully detect normal websites from fake sites and can control online phishing attacks for users on the internet.

1.3 Rationale of the Study

High detection efficiency: To achieve high detection efficiency, benign sites should be classified as phishing as infrequently as possible (false-positive), while phishing sites should be classified correctly (true-positive). To assess the efficacy of the proposed features, I conducted extensive experiments using various machine learning algorithms. The evaluation results show that the proposed approach accurately identifies legitimate websites, with a high true negative rate and a very low false positive rate.

1.4 Research Questions

Following research questions were formulated as part of the phishing website study:

- What are the general ways people use to phishing websites?
- What are the currently available phishing detection technologies?
- Can we make a website that can trick machine learning training algorithms

1.5 Expected output

Through this work we will do the following:

- Detect fake website
- More Accuracy

- Lightweight Classifier
- Find out the Efficient Algorithm
- Reduce the number of false positives,

1.6 Report Layout

This article is broken up into sections, and Section II examines the same efforts that have been done by other researchers. The dataset is broken out in detail in Section III. The pre-processing pipeline is described and the technique is discussed in the same section. Section IV presents the findings, along with several representations of those results. The study is brought to a close with the discussion of its potential future scope in the last part.

CHAPTER 2 BACKGROUND

2.1 Preliminaries

On the Internet, phishing scams remain one of the most serious dangers that users face. The URL of a website can provide light on whether or not it is a phishing site, as previous research has shown this to be a viable method for making this determination. Previous work that is relevant to the issue is presented in this section. To begin, one of the easiest ways to recognize phishing websites is to utilize an outdated method such as blacklisting. However, this method cannot be used to locate newly created phishing websites. Utilizing this type of method also requires a significant amount of time. The research that is being done currently on the identification of phishing links may be broken down into some distinct categories. These categories are indeed the Visual Similarity-based method, the Heuristic Based approach, the Fuzzy rule-based approach, and the Machine Learning approach.

2.2 Related Works

2.2.1 Visual Similarity

In this method, the visual look of a phishing website is evaluated to that of a legitimate website. This comparison involves examining the HTML tags, images, and versions of JavaScript that are present on the page that is under suspicion, as well as other aspects of the website. A method quite similar to this one was utilized by Eric et al [1], who retrieved the unique signature of a website that was suspected of being phishing and compared it to the signature of a website that was real. For the purpose of gathering information about the pictures contained on a particular website, a "signature" of the website in question was utilized. To be more exact, it describes a set of traits that reflect distinct features of a website. These attributes are referred to as "attributes." They employ three elements to identify phishing websites: text fragments, graphics placed in the site, and the general visual look of the web page as it is created by the browser. Even though they have a very low rate of false positives on the dataset, they used to only have 41 phishing pages, which is much too few. This makes their sample size much too tiny. Blacklists are, in their most basic form, a library of URLs that have already been established as being associated with phishing. Blacklist-based solutions are efficient, but they are ineffective against zero-hour phishing attempts [2], and it is simple for attackers to circumvent them [3]. Researchers have

developed a proactive technique called PhishNet [4], which makes use of simple algorithms to create additional candidate phishing URLs from existing blacklisted URLs. The goal of this strategy is to circumvent the difficulties that are associated with blacklists. You can produce new candidate URLs by, for example, replacing a brand name inside the blacklisted URLs with a different brand name or replacing a TLD with a different TLD. This is one method. PhishNet may also be employed to identify new phishing URLs by doing an approximate matching operation (based on regular expressions) also with URLs that are already in the blacklist. URL blacklisting has proven successful to some extent; however, it is rather simple for an adversary to trick the system by subtly altering one or more parts of such URL string [5].

2.2.2 Heuristics-Based Approach

The heuristic-based strategy is the second option available to you. This technique takes into account a variety of characteristics gleaned as from target pages in order to assess whether or not the page in question is a phishing scam or an authentic website. In this technique, the heuristic design of phishing sites matches the feature set that also is typically seen in websites that are used for phishing. CANTINA is a framework that was provided by Zhang et al [6], which has suggested a technique known as CANTINA that detects phishing pages by evaluating text content to use the TF-IDF algorithm. CANTINA was developed by Zhang et al. However, the TF-IDF algorithm, as well as the language of the website both, have a role in determining the scope of the scheme's limits.

2.2.3 Fuzzy Rule-based Approach

The primary phishing characteristic indications can be more accurately represented using fuzzy logic techniques, which make use of linguistic factors. Phishing websites were identified by Maher et al. [7] by the application of fuzzy logic, which was based on six distinct criteria. In their system, they have created separate levels, with each layer including one or more criteria; in addition, they use a formula to compute the phishing rate. The fact that this technology is unable to identify any phishing websites is one of its many drawbacks.

2.2.4 Search Engine

Searching the complete URL of the website in question on well-known search engines like Google, Bing, or Yahoo is yet another uncomplicated method that might be effective in determining whether or not the website in question is a phishing scam [8], [9]. URLs of reputable websites typically retrieve a high quantity of search outcomes and are ranked first, but URLs of fake sites usually return zero outcomes or are not ranked at all. This is because legitimate websites are more trustworthy than phishing websites. However, searching search engines for each URL results in additional expense and may not be feasible due to the rate-limiting that is enforced by providers of search engines. The search engine may provide paid services with higher rate limits; nevertheless, scanning each URL would still expose the user's private information even if the rate restrictions were increased.

2.2.5 Machine Learning Approach

Researchers have turned to strategies that utilize machine learning (ML) in order to enhance the detection of new emerging phishing websites. According to the research carried out by McGrath and colleagues [10], the length, as well as character patterns of phishing URLs, seem strikingly dissimilar to those of non-phishing URLs. As a direct result of this, subsequent studies trained a variety of machine learning models by utilizing URL-based attributes, and these models were able to accurately identify phishing URLs. The method of phishing detection that was demonstrated by Ma et al. [11] uses two different kinds of URL features: lexical features extracted from URL names, including such URL length as well as bag-of-words (BoW), but also external features acquired by querying remote servers, such as whois lookup. Both of these kinds of features can be found in URLs. PhishDef was proposed by Le et al. [12], who also demonstrated that the use of lexical features is just as successful as the utilization of complete features. Sahingoz et al. [13] investigated the usage of natural language processing (NLP) features, word vectors, and hybrid features using various machine learning algorithms and found that all three types of features were successful. Verma et al. [14] investigated the effectiveness of unigrams, bigrams, and trigrams features to learn the nature as well as the construction of phishing URLs as compared to benign URLs. They discovered that classifiers trained on n-gram features accomplish a higher classification accuracy than those trained on the other types of features.

The acquisition of representative URL samples for the purpose of training a classifier presents a significant obstacle to the application of machine learning to the detection of phishing. PhishTank [15], which consists of human-verified phishing URLs, and DMOZ [18], which consists of human-confirmed benign URLs, are the URL sources that are utilized the most frequently in the literature [13], [14], [16], [17]. Nevertheless, the URLs that are reported by PhishTank are not objective (submitted by a few users). In addition, it only contains phishing data for a few hundred brands, the majority of which are based in the United States [19]. On the other side, DMOZ has not been updated since it ceased operations in 2017 and is no longer operational. In addition, the DMOZ dataset consists primarily of the hostnames of innocuous websites, but the PhishTank dataset includes the entire URLs of domains that are used for phishing. Whittaker et al. [20] provided a description of the construction of the proprietary machine-learning model that is used by Google to identify phishing websites. This model makes use of URL data, Google Page Rank, and page features. A strategy for detecting phishing assaults was proposed by Ardi et al. [21], and it involves storing a cryptographic hash to whitelisted web pages as well as comparing this hash with the cryptographic hash of newly accessed web pages.

However, the primary advantage of using models that have been trained solely on URL-based features is that they are able to determine the label of a new URL the moment the page is loaded by the web browser. This eliminates the risk of other potential threats, such as cryptojacking and drive-by download attacks. In addition, the majority of strategies that are page based only become effective once the entirety of a web page has been generated in the browser. Because of this, there is a chance that visitors will reveal private information on the website before it is identified as a phishing scam.

2.3 Comparative Analysis & Summary

According to the study and assessment of malicious URL detection systems that were presented earlier, the vast majority of the currently available malicious URL detection techniques are signature-based URL detection processes. This was discovered as a result of the investigation and evaluation of malicious URL detection methods. As a consequence of this, the efficacy of these technologies is constrained.

2.3 Scope of the Problem

Phishing is a well-known act in which cybercriminals steal sensitive information from users by impersonating websites or by attracting visitors to click other false webpages where their personal information is exposed and made available to the cybercriminals, even though the users have done so unintentionally and innocently. In recent years, the proliferation of numerous online resources, such as financial services, entertainment, educational opportunities, app uploading, and social networking, has contributed significantly to the spectacular growth of the internet. In recent years, intelligent ways for phishing websites based on machine learning techniques have grown more prevalent, smarter, and web-based in comparison to previous methods for phishing websites. These approaches are founded on the idea that computer programs can learn from experience. A detection method for phishing pages that contain a variety of malicious traits, based on NLP and Random Forest.

2.4 Challenges

This subsection provides examples of comparable prior work. Blacklisting is one of the simplest methods for identifying phishing websites, however, it cannot be used to identify new phishing websites. This procedure is also quite time-consuming. In this method, a phishing website's visual look is compared to that of a real website. This involves examining the HTML elements, images, and JavaScript version used on the suspicious page. Eric et al. [1] employed a similar strategy in which they collected the signature of a suspected phishing website and compared it to the signature of a valid website. A web page's signature was employed to gather data comprising the graphics and textual content on this webpage. To be more precise, it is a collection of characteristics representing many features of a website. They employ three elements to detect phishing sites: text fragments, embedded pictures, and the overall visual look of the web page as generated by the browser. Although they have a minuscule incidence of false positives on the dataset, they previously consisted of only 41 phishing pages, which is much too tiny. The heuristic technique is the second option available. This approach combines distinct characteristics gathered from the target web page to identify whether or not it is a phishing site. In this technique, the heuristic design of suspicious websites matches the feature set often seen on phishing websites. Zhang et al. [2] introduced CANTINA, a system that identifies phishing pages by evaluating text content with the TF-IDF algorithm. However, the scheme's restrictions are defined by the website's language and the TF-

IDF algorithm. Fuzzy logic approaches utilize language variables to express the primary phishing indicator characteristics. Based on six distinct criteria, Maher et al. [3] employed fuzzy logic to identify phishing websites. In their system, they have created distinct levels, with each layer including one or more criteria; they also use a formula to compute the phishing rate. This approach has the problem of being unable of detecting any phishing pages. One of the most recent methods used by researchers to determine whether a website is a phishing site is machine learning. Ankit et al. [4] established a system for anti-phishing that employs hyperlink-specific features of multiple machine learning methods. As the website's source code is susceptible to modification for harmful intentions, the fact that the features are based on the source code is a downside of this technique. This might lead to a rise in incorrect predictions. In spite of all these works many challenges needs to be addressed. Therefore, we proposed an efficient model to detect phishing attacks using various machine learning algorithms.

Chapter 3

Research Methodology

3.1 Research Subject

The primary purpose of our study is to identify potentially harmful URLs through the use of three distinct approaches and evaluate the accuracy of each in doing so. First, we take phishing and benign URLs and extract eight attributes from each of them. Then, we generate dataset with those features as that of the column names as well as a label that indicates 0 for genuine as well as 1 for phishing. To ensure that the new dataset we produced is representative across the board, we utilized 5,49,346 samples from each dataset. The address bar is the basis for the characteristics that were extracted.

3.2 Data Collection

In the course of our investigation, we made use of a dataset that included both negative and positive URLs in their entirety. The URLs were harvested from publicly available data sources; however, a significant number of these sources did not contain the protocol. Because of this, the protocol (http:// or https://) and the sub domain (such as www) were omitted from the URL string in cases where they were present. Every URL in the dataset has a label associated with it, either the number 1 or the number 0. A label of 1 indicates that the URL is harmful, whereas a label of 0 suggests that it is harmless. Figure 2 only includes a small selection of the data samples we have. The dataset was divided in two for the purposes of training and validation; 70% of the data was utilized for training, while the remaining 30% was used for validation.

```
Out[5]:
```

	URL	Label
0	nobell.it/70ffb52d079109dca5664cce6f317373782/...	bad
1	www.dghjdgf.com/paypal.co.uk/cycgi-bin/websrcr...	bad
2	serviciosbys.com/paypal.cgi.bin.get-into.herf....	bad
3	mail.printakid.com/www.online.americanexpress....	bad
4	thewhiskeydregs.com/wp-content/themes/widescr...	bad

Figure 3.1: Bad URL from dataset

Out[6]:

	URL	Label
549841	cam.ac.uk	good
549842	over-blog-kiwi.com	good
549843	merriam-webster.com	good
549844	bp3.blogger.com	good
549845	kinja.com	good

Figure 3.2: Good URL's from dataset

3.3 Applied Mechanism

Our model has to be trained on the dataset before it can be used to differentiate between URLs that are harmful and those that are safe to visit. On the other hand, our algorithm is unable to decipher raw URLs, along with any other kind of alphanumeric text including symbols. Therefore, in order to make the URLs worthy of training, they need to go through a pre-processing pipeline. Within this pipeline, the URLs will go through three stages: tokenization, character-to-index mapping, and sequence padding.

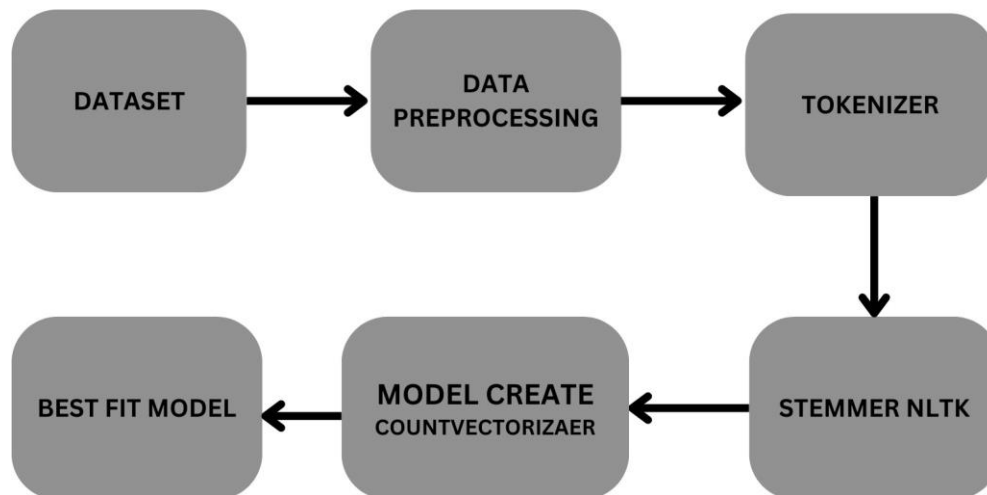


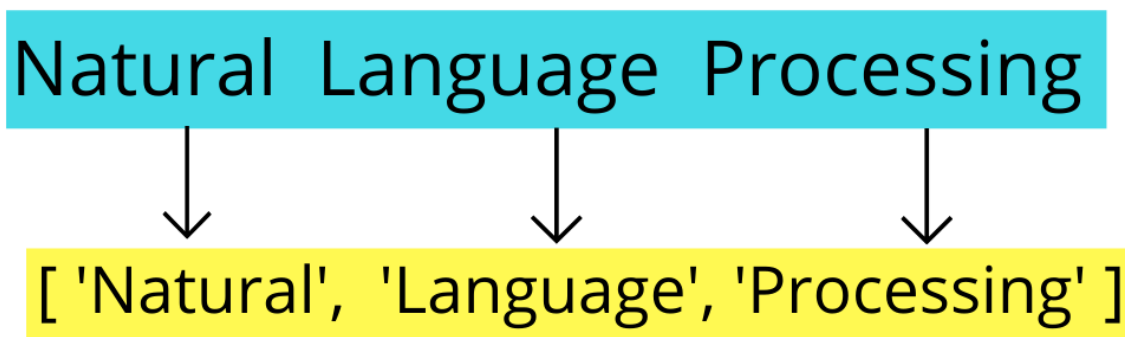
Figure 3.3: Working Mechanism

In our proposed model at first, it will process the data then tokenize and stem it with the help of countvectorizer we will make a model and select our best model.

331 Tokenizar

Tokenization is an essential step in the pre-processing phase of virtually every NLP-related activity. Tokenization refers to the technique of slicing up a string of text into more manageable linguistic components called tokens (words, punctuations, numbers, special symbols, etc.). According to the findings of our study, each and every character (including letters, numbers, and special symbols) that makes up the URL is regarded as a token.

Tokenization



332 CountVectorizer

By doing preprocessing activities such as converting all words to lowercase and deleting special characters, CountVectorizer implies breaking down a sentence or any other text into words. This may be done with any sentence or any text. Because NLP models only understand numbers and cannot comprehend textual data, the textual data must be vectorized before the models can use it. The models that are used for natural language processing can only comprehend the numerical value. So we need to transform textual input to a numerical number. Additionally, the BOW model is a quite straightforward one. It turns whole thoughts into a random collection of phrases that have no context. Nevertheless, it changes the text into a vector of integers with a predetermined length. According to this model, each word is given a one-of-a-kind number that corresponds to a count of the number of times that word appears in the model.

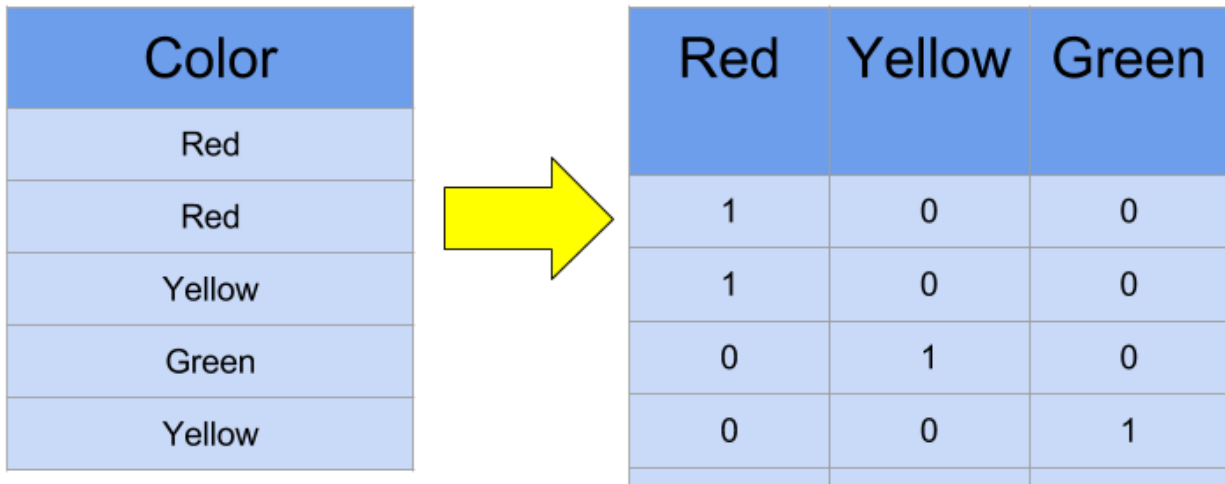


Figure 3.4: CountVectorizer Working System

The arrangement of the words is not our primary concern; rather, we are concentrating on how they are represented. Tokenization is the process of breaking down a sentence into its component words. Tokenization is accomplished by doing several preprocessing activities, such as converting all words to lowercase and deleting special characters, among other things. Therefore, an encoding vector is returned with the length of the full vocabulary, which includes all of the words, as well as an integer count of the number of times that each word appears in the phrase.

333 Character-to-index mapping

Simply alphabetic, numeric, and special characters make up the tokens that were formed in the stage before this one. However, because our model can only comprehend numeric values, the tokens that we use need to be converted. As a consequence of this, we give every character in our lexicon its own distinct integer id, and at this stage of the pre-processing procedure, every list of tokens is mapped to the list of ids that corresponds to it.

334 Sequence-padding

This procedure is carried out for the most part in order to assist us in automatically training the model using the numerical data. To do sequence padding, additional zeros are appended to the token vector before it is used. This is done to ensure that all of the vectors have the same structure and to facilitate the process of feeding various data sets into our model in batches.

335 Embedding

Following the completion of the aforementioned three stages of preprocessing, we now have a list of numbers that correspond to each URL in our dataset; more specifically, each integer represents a different token in our lexicon. On the other hand, this representation of tokens as integers does not include any information that relates to our URLs. It is thus vital to transform these numbers to some kind of encoding, since this will assist our model in training on our dataset in a more effective manner. One such representation that is used frequently is called one-hot encoding, in which each integer is expressed as an array with the same length as the size of the vocabulary. In this representation, a "1" is placed where the vector index is equal to the integer, and "0"s are placed in all other positions. However, these encodings have a number of drawbacks, the first of which is that the size of each encoding may be rather substantial, depending on the vocabulary. As a consequence, the resultant matrices are quite huge and sparse, which makes the process of training ineffective. The second and more significant flaw with these representations is that they do not take into account the relationship between the tokens. This means that when two tokens that are comparable are plotted in vector space, they are placed a significant distance apart from one another. As a result, we have made use of embeddings. Embeddings are dense representations in the form of vectors that take into account the similarities between the tokens.

3.3.6 Snowball Stemmer – NLP

It is a stemming algorithm that is also known as such Porter2 stemming method since it is an improved version of the Porter Stemmer due to the fact that various problems with the Porter Stemmer have been solved in this stemmer.

3.3.6.1 Stemming

The term "lemma" refers to the result of the process of reducing a word to its word stem, which can then be attached to suffixes, prefixes, or the roots of other words. Stemming is, in layman's terms, the process of reducing a word to its base word or stem in such a manner that words of a similar sort lie under a common stem. Stemming may also be thought of as derivation. For instance, the terms "care," "cared," and "caring" are all derived from the same root word: "care." Processing natural language requires careful attention to stemming (NLP).

Stemming vs Lemmatization

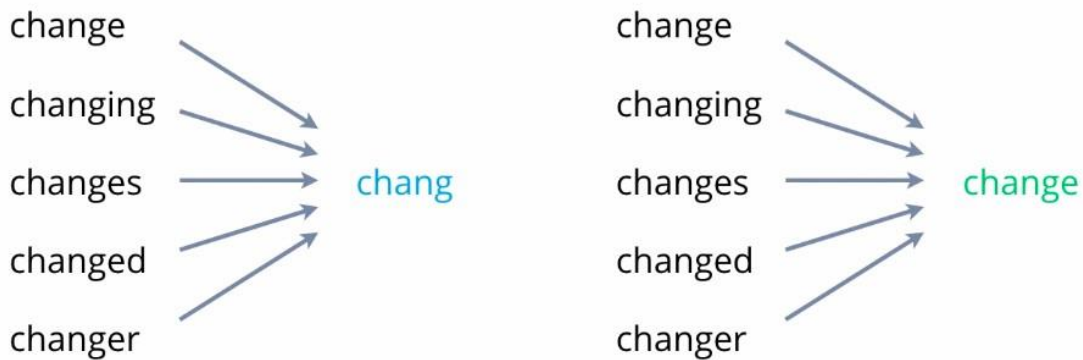


Figure 3.5: Stemming VS Lemmatization

33.7 Logistic Regression

An example of supervised learning is something like logistic regression. Calculating or forecasting the likelihood of a binary (yes/no) event happening is one of its primary applications. Using machine learning to detect whether or not a person has COVID-19 infection is an application of logistic regression. Another application of logistic regression is determining whether or not a person has the COVID-19 virus. Binary classification is the term used to describe a situation in which there are only two viable answers to a question: either yes, they are infected, or no, they are not infected. In this hypothetical scenario, the likelihood of a patient being contaminated with COVID-19 might be determined by looking at factors such as the viral load, the symptoms, the existence of antibodies, and other factors. Our variables (Independent Variables), that would impact our outcome, would include viral load, symptoms, as well as antibodies (Dependent Variable).

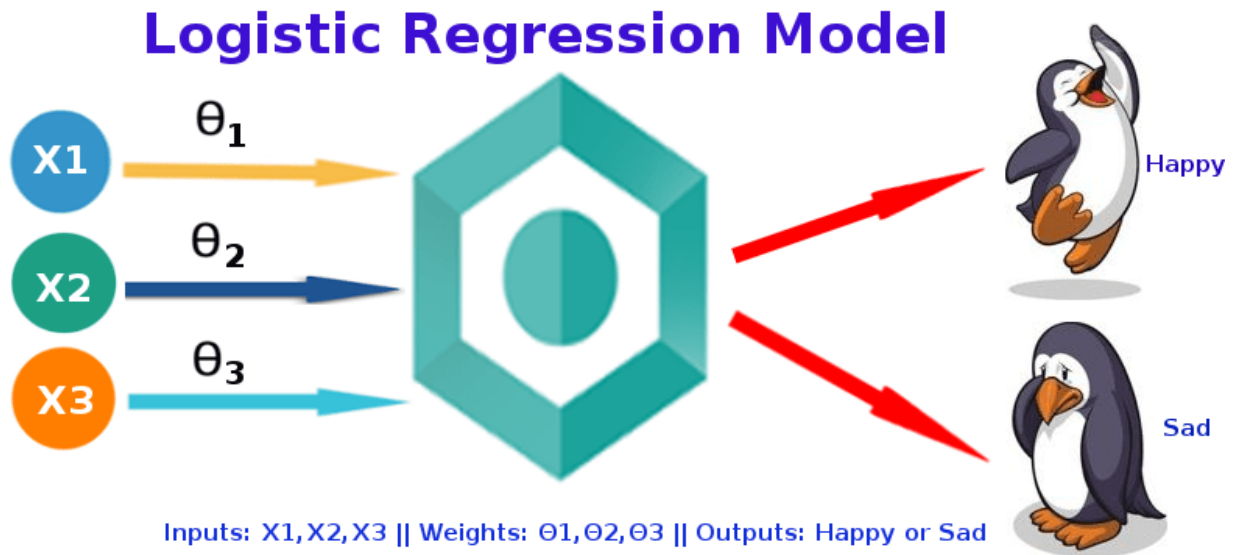


Figure 3.6: Logistic Regression Working Procedure

Logistic regression is a technique that may be utilized to handle classification issues; the most popular application of this technique is binary logistic regression, in which the result is also binary (yes or no). In the actual world, logistic regression was used to a wide variety of different domains and industries.

338 Logistic Regression and the Underlying Mathematics

Probability is always in the range of 0 (does not occur) to 1 (always occurs) (happens). Using the example of Covid-19, the likelihood of testing positive and the probability of not testing positive will both add up to one in the case of binary categorization. For the purpose of calculating probability in logistic regression, we make use of the logistic function or the sigmoid function. The logistic function is a straightforward S-shaped curve that is applied to data in order to transform it into a number that falls between 0 and 1.

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

' $h_{\theta}(x)$ ' is output of logistic function , where $0 \leq h_{\theta}(x) \leq 1$

' β_1 ' is the slope

' β_0 ' is the y-intercept

' X ' is the independent variable

$(\beta_0 + \beta_1 * x)$ - derived from equation of a line $Y(\text{predicted}) = (\beta_0 + \beta_1 * x) + \text{Error value}$

Logistic regression, in a nutshell, is used to classify issues where the output or dependent variable in question is binary or categorical. When putting in place logistic regressions, it is important to keep certain assumptions in mind. Some of these assumptions include the many forms of logistic regression, the various types of independent variables, and the amount of training data that is currently accessible.

339 Multinomial Naive Bayes

While there are dozens of programs and tools available for the evaluation of numerical data, only a handful of options are available for the study of textual data. One of the most common supervised learning classifications that are employed in the analysis of categorical text data is called multinomial Naive Bayes. This method was developed by George Naive.

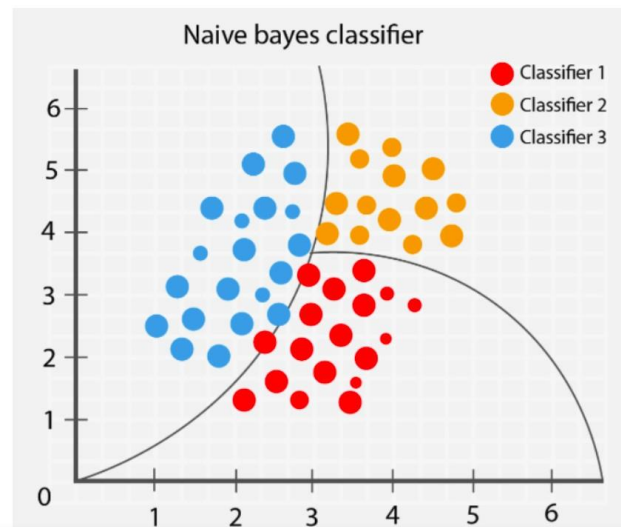


Figure 3.7: Naive Bayes Classifier

Text data categorization is becoming increasingly popular for the simple reason that there is an overwhelming quantity of data that has to be examined. This data may be found through email, documents, websites, and other places. Finding out how people who are going to use a program or product are going to perceive it is made easier by having a better understanding of the context around a certain form of text.

You will emerge from reading this article with a comprehensive knowledge of the multinomial Naive Bayes method as well as all of the ideas that are associated with it. In this section, we will

go through an overview of the algorithm, including how it operates, the benefits it offers, and the applications it may be used for.

The Multinomial Naive Bayes algorithm is indeed a probabilistic learning approach that is utilized in Natural Language Processing the majority of the time (NLP). The Bayes theorem serves as the foundation for the computer program that can determine the tag associated with a piece of text such as an email or a newspaper article. After determining the probabilities of having each tag for a specific sample, it selects the tag with the highest likelihood as the one to output.

The Naive Bayes classifier is indeed a collection of several different algorithms, all of which adhere to the same fundamental concept, which states that the features that are being categorized are not connected to any of the other features in any way. There is no correlation between the presence or absence of one trait and the presence and absence of the other feature.

33.10 How Multinomial Naive Bayes Works

The Naive Bayes method is a useful tool that may be utilized for the analysis of text data as well as challenges involving numerous classes. It is essential to have a fundamental understanding of the Bayes theorem notion before attempting to comprehend the operation of the Naive Bayes theorem because the former is founded on the latter.

The Bayes theorem, which was developed by Thomas Bayes, is a mathematical formula that determines the likelihood of an event taking place given the previous knowledge of conditions that are associated with the occurrence. The following equation serves as its foundation:

$$P(A|B) = P(A) * P(B|A)/P(B)$$

Here, predictor B is already available, so we only need to figure out how likely class A is.

$P(B)$ = prior probability of B

$P(A)$ = prior probability of class A

$P(B|A)$ = occurrence of predictor B given class A probability

The probability of the tags appearing in the text can be calculated with the help of this formula.

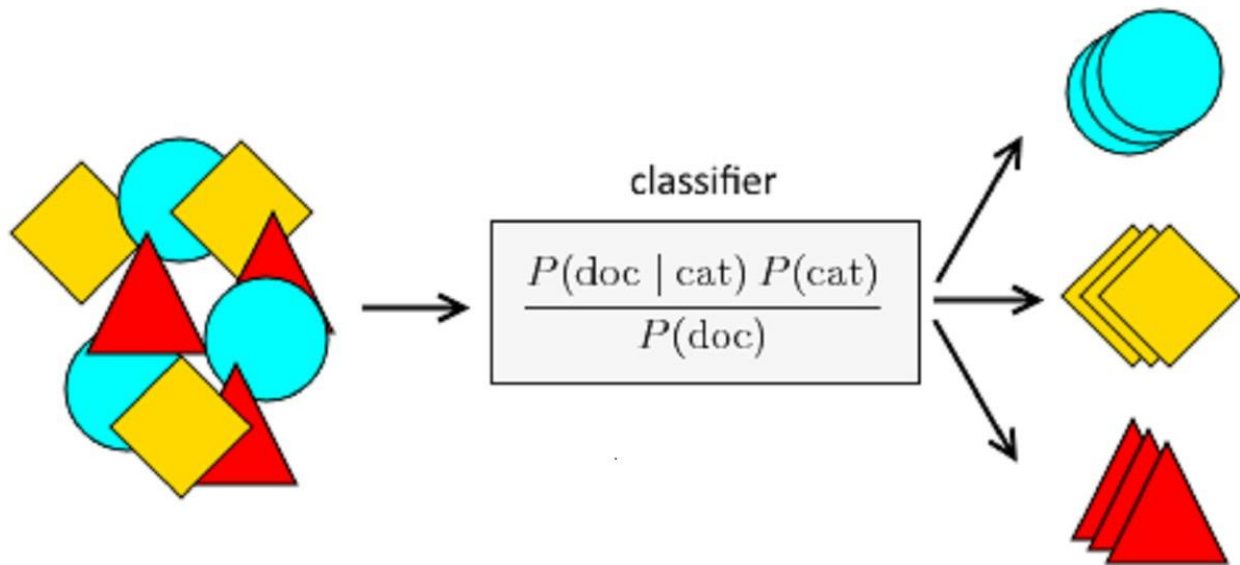


Figure 3.8: MNB Classification System

3.3.11 Evaluation Metrics

The concept of constructing learning models for machine learning is based on the notion of constructive feedback. Construct a model, monitor its performance using metrics, evaluate its efficacy, and keep going until you reach the level of accuracy you require. The performance of a model may be explained using evaluation measures. The capacity of assessment metrics to differentiate between different model outputs is an essential characteristic of these metrics. Developing a predictive model for the sake of developing a predictive model should not be a purpose. The process involves developing and choosing a model that provides a high level of accuracy on data that is not part of the sample set. As a result, it is essential to verify the correctness of the model before attempting to compute expected values.

3.3.12 Confusion Matrix

The number of classes that are being predicted is denoted by the N in a confusion matrix, which is a N x N matrix. Because N is equal to two in this particular situation, we will be working with a matrix that has those dimensions.

		ACTUAL VALUES	
		POSITIVE	NEGATIVE
PREDICTED VALUES	POSITIVE	TP	FP
	NEGATIVE	FN	TN

Figure 3.9: Confusion matrix

The following are some definitions pertaining to a confusion matrix that you will need to keep in mind:

- Accuracy: May be defined as the percentage of right predictions relative to the total number of guesses.
- Positive Predictive Value or Precision: The percentage of positive instances that were found to have been appropriately recognized.
- Negative Predictive Value: The percentage of instances that were found to be false positives that were accurately detected.
- Sensitivity or Recall: The proportion of actual positive cases which are correctly identified.
- Specificity: The percentage of true negative cases that can be identified without error.

3.3.14 F1 Score

In the last part of this article, we spoke about precision and recall for issues involving classification, and we also highlighted how important it is to choose precision and recall based on our use case. What if, for one of our use cases, we want to achieve the highest possible level of both precision and recall simultaneously? A classification problem's F1 score is calculated by finding the harmonic mean of the problem's accuracy and recall scores. The following is the formula used to get the F1-Score:

$$F_1 = \left(\frac{\text{recall}^{-1} + \text{precision}^{-1}}{2} \right)^{-1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Now, the issue that immediately springs to mind is why we are using a harmonic mean rather than an arithmetic mean. This is an obvious question. This is due to the fact that HM is more strict with punishing excessive values. Let's look at an example of this to better comprehend it. A binary classification model that we have yielded the outcomes of which are as follows:

If we were to calculate the arithmetic mean of these values, we would obtain 0.5. It is very evident that the aforementioned result was produced by a simple classifier that just ignores the input and makes a prediction regarding which of the classes should be output. Now, if we were to take HM, we would receive 0 as a correct result, which is appropriate seeing as how this model is completely pointless for any and all applications.

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

It appears to be quite easy. However, there are circumstances in which a data scientist would want to place a greater emphasis on or assign a greater amount of weight to, either precision or recall. By making a few adjustments to the formula above so that we may add an adaptable parameter named beta for the purpose of this discussion. The efficacy of a model is evaluated with respect to a user that places a weighting of twice as much value on recall as precision when using the fbeta metric.

3.3.15 Conclusion

In this chapter, we covered the preliminary work that was done on our research project, as well as the theoretical framework that underpinned it.

3.4 Implementation Requirements

Python (v3.xx) Python is an interpreted high-level language whose design philosophy calls for an emphasis on code readability and a syntax that allows programmers to express Concepts and their work require fewer lines of code than in languages such as Java & C++.

3.4.1 Python Libraries Python comes with an extensive standard library, an aspect of python referred to as "batteries included" python philosophy. Python's ability to provide tools for a wide

range of tasks is one of its greatest strengths. We can easily gain access by using the import keyword. Python provides a list of libraries used for data science, some of which are listed below and are used in this thesis:

3.4.2 Numpy

Numpy is a powerful base package for scientific computing.

- tools for integrating C/C++ and Fortran code
- routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

3.4.3 Matplotlib

Matplotlib is a Python 2D plotting library that produces publication-quality figures in a variety of hardcopy formats and in an interactive environment across platforms. Matplotlib is available for Python scripts, Python and IPython shells, Jupyter notebooks, web application servers, and four GUI toolkits. Using matplotlib, we can generate plots, histograms, power spectra, bar charts, error plots, scatterplots, and more with just a few lines of code.

3.4.4 Jupyter Notebook

Jupyter Notebook is an open source web application with an interactive environment for creating and sharing documents containing live code, equations, visualizations, and narrative text. Jupiter's code can generate rich interactive output such as HTML, images, and videos. It can be used for data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and many other applications.

CHAPTER 4

Experimental Result & Discussion

4.1 Experimental Setup

In my study, all data analysis, manipulation, training, and evaluation were performed in Jupyter notebooks. The purpose of the experiment is to obtain a logistic regression classifier model which has the best generalization performance and also achieves very good accuracy in its predictions. While most of the techniques and models deployed there have been criticized for over fitting the training data, so the model performs well on the training data with a very low error rate, but fails on unseen test data or real Poor performance in world data. To overcome this problem, a "hold-out" authentication technique is used. This is one of the standard procedures in the machine learning process. The training set is divided into training, validation and testing dataset.

4.2 Experimental Result & Analysis

The total sample size is 549846. Good and Bad URL quantity is given below:

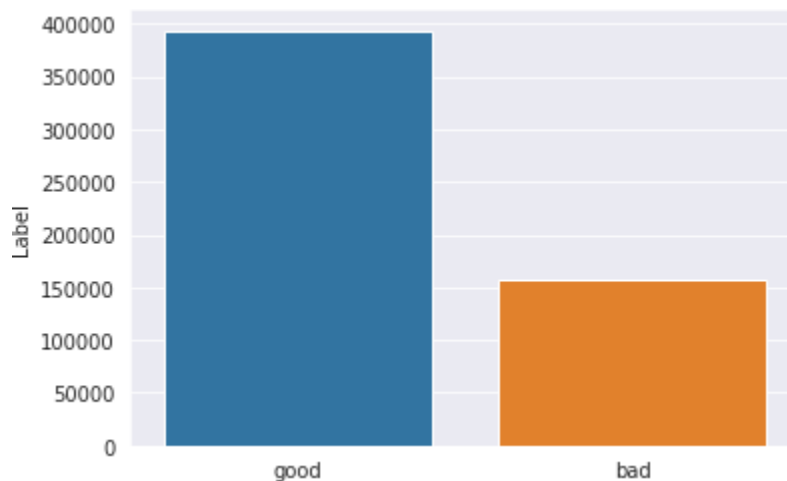


Figure 4.1: Dataset Ratio

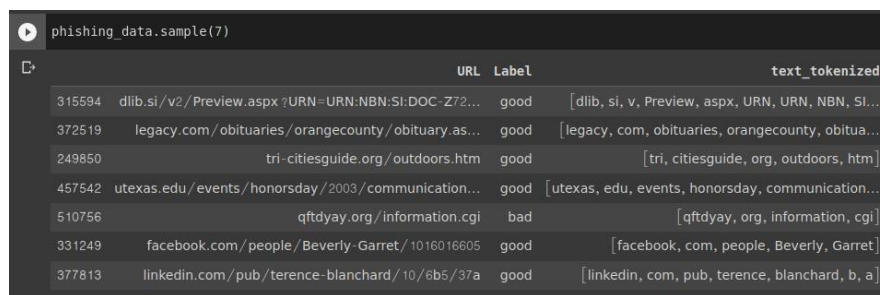
4.2.1 RegexpTokenizer

A regular expression is used by a RegexpTokenizer to divide a text into its component substrings. To give just one illustration, the following tokenizer creates tokens out of alphabetic sequences, monetary expressions, and any other sequences that do not contain whitespace:

```
>>> from nltk.tokenize import RegexpTokenizer
>>> s = "Good muffins cost $3.88\nin New York. Please buy me\ntwo of them.\n\nThanks
>>> tokenizer = RegexpTokenizer('\w+|\$[\d\.]+|\S+')
>>> tokenizer.tokenize(s)
['Good', 'muffins', 'cost', '$3.88', 'in', 'New', 'York', '.',
'Please', 'buy', 'me', 'two', 'of', 'them', '.', 'Thanks', '.']
```

Figure 4.2: RegexpTokenizer

If we want total control over how the text is tokenized, we utilize regular expressions to do this. In the figure below after using tokenizer, we get this result

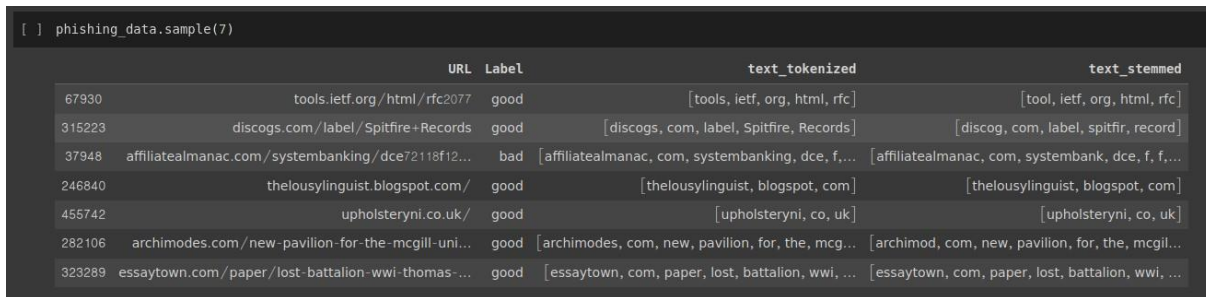


	URL	Label	text_tokenized
315594	dlib.si/v2/Preview.aspx?URN=URN:NBN:SI:DOC-Z72...	good	[dlib, si, v, Preview, aspx, URN, URN, NBN, SI...
372519	legacy.com/obituaries/orangecounty/obituary.as...	good	[legacy, com, obituaries, orangecounty, obitua...
249850	tri-citiesguide.org/outdoors.htm	good	[tri, citiesguide, org, outdoors, htm]
457542	utexas.edu/events/honorsday/2003/communication...	good	[utexas, edu, events, honorsday, communication...
510756	qftdyay.org/information.cgi	bad	[qftdyay, org, information, cgi]
331249	facebook.com/people/Beverly-Garret/1016016605	good	[facebook, com, people, Beverly, Garret]
377813	linkedin.com/pub/terence-blanchard/10/6b5/37a	good	[linkedin, com, pub, terence, blanchard, b, a]

Figure 4.2.1: RegexpTokenizer Output

4.2.2 SnowballStemmer

It is a stemming algorithm that is also recognized as the Porter2 stemming method since it is an improved version of the Porter Stemmer due to the fact that various problems with the Porter Stemmer have been solved in this stemmer. The term "lemma" refers to the result of the process of reducing a word to its word stem, which can then be attached to suffixes, prefixes, or the roots of other words. Stemming is, in layman's terms, the process of reducing a word to its base word or stem in such a manner that words of a similar sort lie under a common stem. Stemming may also be thought of as derivation. For instance, the terms "care," "cared," and "caring" are all derived from the same root word: "care." Processing natural language requires careful attention to stemming (NLP). In comparison to Porter Stemmer, Snowball Stemmer is a more vigorous cultivar. The shortcomings of the Porter Stemmer have been addressed in the Snowball Stemmer. The way in which these two things function is nearly identical to one another. When run through the snowball stemmer, words like "fairly" and "sportingly" are shortened to "fair" and "sport." However, when run through the porter stemmer, these words are shortened to "fairli" and "sportingli." The manner that each of these algorithms originate the word "Sportingly" reveals an important distinction between them, which can be observed quite clearly. Clearly Snowball Stemmer is able to provide a stem that is more precise.



```
[ ] phishing_data.sample(7)
```

	URL	Label	text_tokenized	text_stemmed
67930	tools.ietf.org/html/rfc2077	good	[tools, ietf, org, html, rfc]	[tool, ietf, org, html, rfc]
315223	discogs.com/label/Spitfire+Records	good	[discogs, com, label, Spitfire, Records]	[discog, com, label, spitfir, record]
37948	affiliatealmanac.com/systembanking/dce72118f12...	bad	[affiliatealmanac, com, systembanking, dce, f, ...]	[affiliatealmanac, com, systembank, dce, f, f, ...]
246840	thelouslylinguist.blogspot.com/	good	[thelouslylinguist, blogspot, com]	[thelouslylinguist, blogspot, com]
455742	upholsteryni.co.uk/	good	[upholsteryni, co, uk]	[upholsteryni, co, uk]
282106	archimodes.com/new-pavilion-for-the-mcgill-uni...	good	[archimodes, com, new, pavilion, for, the, mcgill, uni, ...]	[archimod, com, new, pavilion, for, the, mcgill, uni, ...]
323289	essaytown.com/paper/lost-battalion-wwi-thomas-...	good	[essaytown, com, paper, lost, battalion, wwi, ...]	[essaytown, com, paper, lost, battalion, wwi, ...]

Figure 4.3: Before using SnowballStemmer

After using SnowballStemmer got the above result. And we join the text in a sentence.

	URL	Label	text_tokenized	text_stemmed	text_to_sent
392226	motorcyclestoresusa.com/pennsylvaniamotorcycle...	good	[motorcyclestoresusa, com, pennsylvaniamotorcy...	[motorcyclestoresusa, com, pennsylvaniamotorcy...	motorcyclestoresusa com pennsylvaniamotorcycle...
70277	www.anotherbigidea.com/javaswf/	good	[www, anotherbigidea, com, javaswf]	[www, anotherbigidea, com, javaswf]	www anotherbigidea com javaswf
227354	peswiki.com/index.php/Electric_power_transmission	good	[peswiki, com, index, php, Electric, power, tr...	[peswiki, com, index, php, electr, power, tran...	peswiki com index php electr power transmiss
490065	hobby-hangar.net	bad	[hobby, hangar, net]	[hobbi, hangar, net]	hobbi hangar net
371761	lead411.com/Scott_Sandlin_1305615.html	good	[lead, com, Scott, Sandlin, html]	[lead, com, scott, sandlin, html]	lead com scott sandlin html
156687	castafile.com/	good	[castafile, com]	[castafil, com]	castafil com
395395	mylife.com/c-1065114468	good	[mylife, com, c]	[mylif, com, c]	mylif com c
424341	rollitup.org/hallucinatory-substances/256576-a...	good	[rollitup, org, hallucinatory, substances, aci...	[rollitup, org, hallucinatori, substanc, acid,...	rollitup org hallucinatori substanc acid vs mu...
237673	search.barnesandnoble.com/Fighting-for-Life/Wa...	good	[search, barnesandnoble, com, Fighting, for, L...	[search, barnesandnobl, com, fight, for, life,...	search barnesandnobl com fight for life walter...
255566	vimeo.com/nicolasgirard	good	[vimeo, com, nicolasgirard]	[vimeo, com, nicolasgirard]	vimeo com nicolasgirard

Figure 4.3.1: After using SnowballStemmer

4.2.3 WordCloud

A data visualization approach known as a "word cloud" is utilized for the purpose of showing text data within which the size of each word represents the frequency with which it is used or its relevance. With the use of a word cloud, important aspects of textual material may be brought to light. It is common practice to employ word clouds while conducting data analysis on social networking platforms. Python modules called matplotlib, pandas, and wordcloud are required in order to create a word cloud using Python. In recent years, text data has seen exponential growth, which has resulted in an ever-increasing demand for the analysis of extremely large volumes of such data. Word Cloud is a useful tool for analyzing text data through the process of data representation in the form of tags, or words; the significance of a word is determined by the frequency with which it appears in the text.

The most often used terms in URLs that are not attempts at phishing.

The most common words used in phishing sites are given below:



Figure 4.4: URLs that are not attempts at phishing



Figure 4.5: URLs that are attempts at phishing

4.2.4 Logistic Regression:

Training Accuracy:	0.9770602157212694
Testing Accuracy:	0.9629352111856367

Table 4.1: Logistic Regression Accuracy

4.2.5 Classification Result

	Precision	Recall	F1-score
Bad	0.90	0.97	0.93
Good	0.99	0.96	0.97
Accuracy			0.96
Macro avg	0.94	0.96	0.95
Weighted avg	0.96	0.96	0.97

Table 4.2: Logistic Regression Classification Result

4.2.6 Confusion Matrix

The confusion matrix is given below:

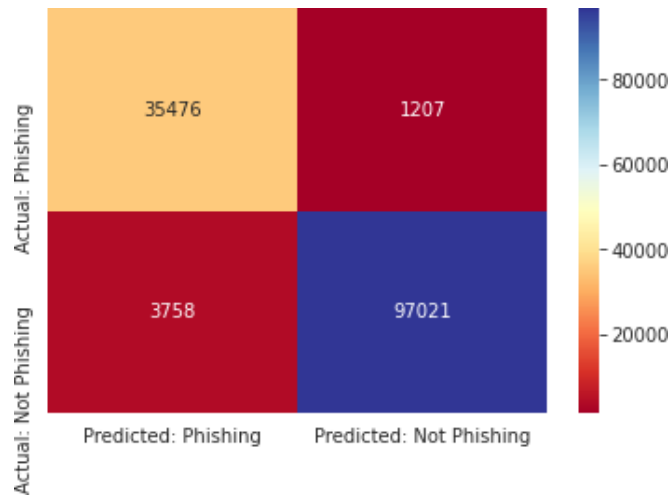


Figure 4.6: Confusion Matrix of Logistic Regression

4.2.7 Multinomial Naive Bayes

Training Accuracy:	0.9739660122604175
Testing Accuracy:	0.957690125270984

Table 4.3: Multinomial Naive Bayes Accuracy

4.2.8 Classification Result

	Precision	Recall	F1-score
Bad	0.90	0.97	0.93
Good	0.99	0.96	0.97
Accuracy			0.96
Macro avg	0.94	0.96	0.95
Weighted avg	0.96	0.96	0.97

Table 4.4: Multinomial Naive Bayes Classification Result

4.2.9 Confusion Matrix

The confusion matrix is given below:

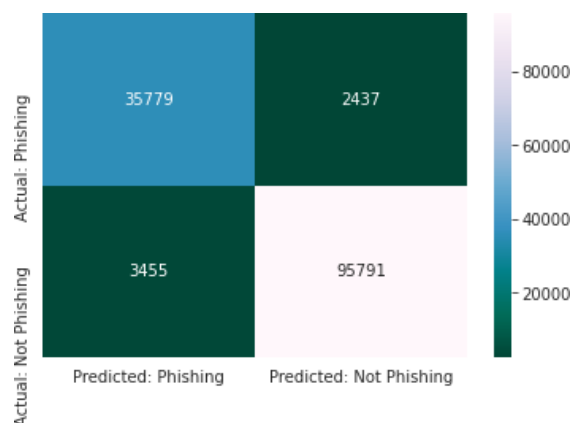


Figure 4.7: Confusion Matrix of Multinomial Naive Bayes

Actual Score

Actual Score of Logistic Regression: 0.9629352111856367

Actual Score of MultinomialNB: 0.957690125270984

Rounded Score

Final Rounded Score:	Accuracy
Logistic Regression	0.96
MultinomialNB	0.96



Figure 4.7: Accuracy difference between Logistic Regression and Multinomial Naive Bayes

4.2.10 Best Fit Model

The findings presented above make it quite evident that the Logistic Regression model provides the greatest fit, as indicated by its actual score of 96%.

Therefore, at this point we will create a sklearn pipeline utilizing logistic regression.

Pipeline Score: 0.9658014577574442

Training Accuracy:	0.9797979553037945
Testing Accuracy:	0.9658014578574442

Table 4.5: Best Fit Model Accuracy

4.2.11 Classification Result

	Precision	Recall	F1-score
Bad	0.91	0.97	0.94
Good	0.99	0.96	0.98
Accuracy			0.97
Macro avg	0.95	0.97	0.96
Weighted avg	0.97	0.96	0.97

Table 4.6: Best Fit Model Classification Result

4.2.12 Confusion Matrix

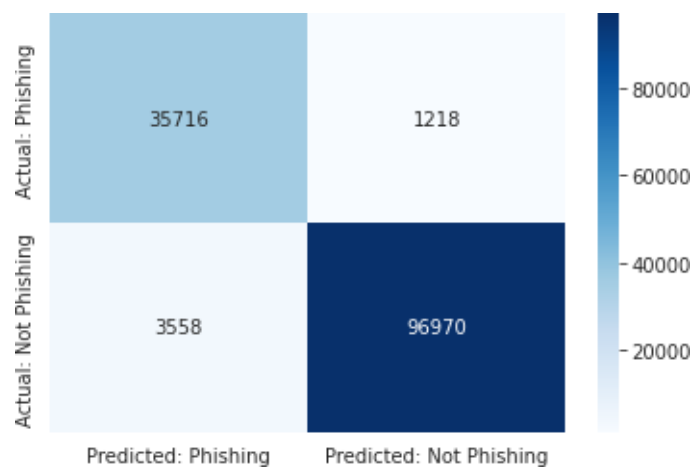
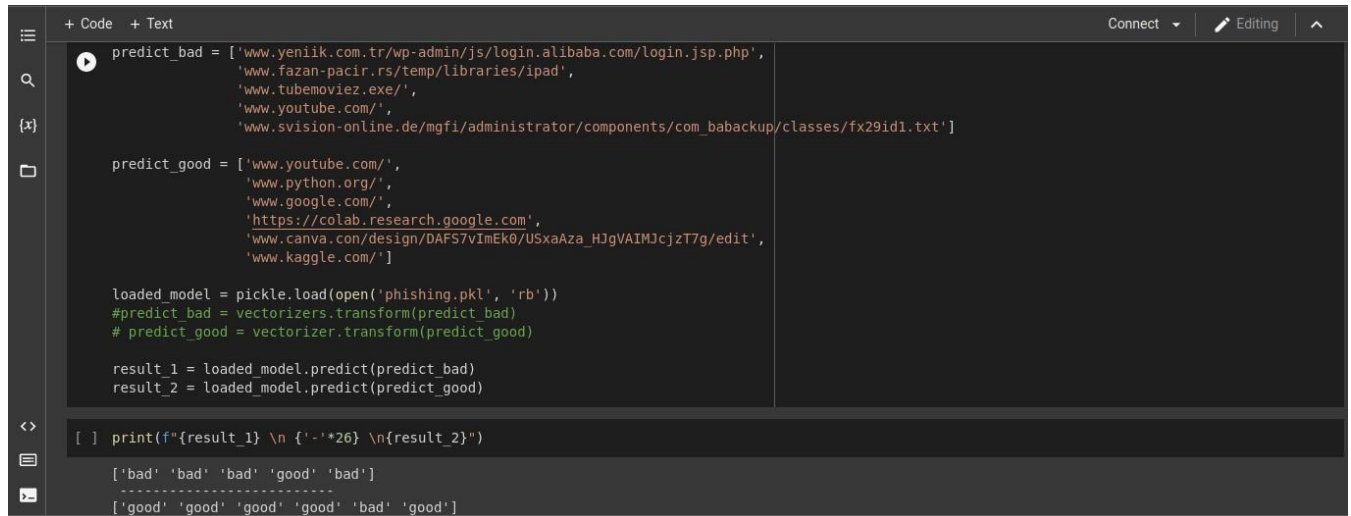


Figure 4.8: Confusion Matrix of Best Fit Model

4.2.13 Model Testing Result

I input some good and bad URL addresses and I got some satisfactory results. For cross-verification I insert a good URL in bad portion and also I insert a bad url in good portion. The outcome is given below.



```
+ Code + Text Connect Editing ^  
  
predict_bad = ['www.yeniik.com.tr/wp-admin/js/login.alibaba.com/login.jsp.php',  
              'www.fazan-pacir.rs/temp/libraries/ipad',  
              'www.tubemoviez.exe/',  
              'www.youtube.com/',  
              'www.svision-online.de/mgfi/administrator/components/com_babackup/classes/fix29id1.txt']  
  
predict_good = ['www.youtube.com/',  
               'www.python.org/',  
               'www.google.com/',  
               'https://colab.research.google.com',  
               'www.canva.com/design/DAFS7VimEk0/USxaAza_HJgVAIMJczT7g/edit',  
               'www.kaggle.com/']  
  
loaded_model = pickle.load(open('phishing.pkl', 'rb'))  
#predict_bad = vectorizers.transform(predict_bad)  
# predict_good = vectorizer.transform(predict_good)  
  
result_1 = loaded_model.predict(predict_bad)  
result_2 = loaded_model.predict(predict_good)  
  
[ ] print(f"{result_1} \n {'-'*26} \n{result_2}")  
  
['bad' 'bad' 'bad' 'good' 'bad']  
-----  
['good' 'good' 'good' 'good' 'bad' 'good']
```

Figure 4.9: Model Testing Result

4.3 Discussion

Python is utilized in order to carry out the evaluation of the suggested model. Accuracy, F1 score, Recall, and Precision are the four performance indicators that we have taken into consideration.

Chapter 5

Impact on Society, Environment & Sustainability

5.1 Impact on Society

. "Phishing" is one of the most common types of cybercrime that affects consumers, businesses, and individuals all over the world. So, exactly what is phishing? There is no formal definition, but it can be summarized as an attempt to obtain sensitive information, often for malicious purposes, by masquerading as a trustworthy entity or individual in electronic communication. While it is a relatively simple attack, it has emerged as the most common and dangerous threat posed by attackers

5.2 Impact on Environment

As I anticipate that it will be a convenience thesis with a significant impact on society, it must also have a significant impact on the environment. Because an environment is created by a society, and societies are created by people. If people have sufficient knowledge of phishing website detector, it will also have dominance in their environment. As a result, my thesis will contribute to the development of an environment in which everyone can use the phishing website detector model without error.

5.3 Ethical Aspects

It is my moral responsibility in most organizations, IT personnel are trusted with access to sensitive and personal data, so ethics are an important part of detect phishing website. What ethical standards exist often shapes their behavior and how they handle this responsibility. For example, The Institute of Electronics and Electrical Engineers is one organization that has an ethics code that many organizations and researchers try to follow. Furthermore, the Association of Computing Machinery's Code of Ethics and Professional Conduct emphasizes fundamental ethical considerations and provides organizations with guiding principles. These guidelines and code of ethics are intended primarily to serve as a foundation for ethical decision making in the course of everyday professional work.

5.4 Sustainability Plan

My plan is to introduce a detection method against phishing website attacks using logistic regression. It will be help to detect phishing website. This is the reason why I created the model and why I need to fix the properly so that people can easily check the phishing website. I should have known about this concept and I should learn and clear the concept. That's why I need to sustain this model properly. I also need to process how to sustain this model permanently.

CHAPTER 6

Summary, Conclusion, Implication for Future Research

5.1 Summary of the Study

Malicious actors are attempting to seize control of the vulnerable system in order to carry out a variety of attacks on the users' transactions. An example of this kind of attack is known as a "phishing attack" The intruder tries to force the genuine user into believing that they are accessing a fraudulent website.

5.2 Conclusions

An example of a common form of cybercrime is known as a phishing attack. In this type of attack, the perpetrators attempt to trick users into providing their personal information by disguising themselves as an official website. Logistic Regression and Multinomial Naive Bayes are the two varieties of machine learning classifiers that are utilized by the proposed method. Performance criteria like F1 score, Recall, and Precision are utilized in the implementation of these algorithms. It is abundantly obvious from the experimental findings that the LR algorithm has a better F1 score, as well as higher precision and recall. In addition, in comparison to other machine learning classifiers, the LR classifier boasts a phishing detection accuracy of 96%, which is significantly higher. Evaluation of these machine learning classifiers using larger datasets is the next step in the development of the system that has been suggested.

5.3 Implication for Further Study

Nevertheless, when the potential dangers posed by rogue URLs are taken into account, this figure is rather low, and there is room for future development. The rapid development of technology in recent years has led to the creation of a number of innovative and intricate architectural designs as well as preprocessing methods, some of which are said to be superior to the ones that have been described. Because of this, there is potential for future study, and it is anticipated that models built with these sophisticated algorithms would have higher performance

References

- [1] Medvet, Eric, Engin Kirda, and Christopher Kruegel. "Visual-similarity-based phishing detection." Proceedings of the 4th international conference on Security and privacy in communication networks. 2008
- [2] Steve Sheng, Brad Wardman, Gary Warner, Lorrie Cranor, Jason Hong, and Chengshan Zhang. An Empirical Analysis of Phishing Blacklists. In Sixth conference on email and anti-spam (CEAS). California, USA, 2009.
- [3] A Framework for Detection and Measurement of Phishing Attacks. In Proceedings of the 2007 ACM Workshop on Recurring Malcode, WORM '07, pages 1–8, New York, NY, USA, 2007. ACM.
- [4] P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta. Phish-Net: Predictive Blacklisting to Detect Phishing Attacks. In 2010 Proceedings IEEE INFOCOM, pages 1–5, March 2010
- [5] Frank Vanhoenshoven, Gonzalo Nápoles, Rafael Falcon, Koen Vanhoof, and Mario Köppen. Detecting malicious urls using machine learning techniques. In 2016 IEEE Symposium Series on Computational Intelligence (SSCI), pages 1–8, 2016.
- [6] Zhang, Yue, Jason I. Hong, and Lorrie F. Cranor. "Cantina: a content-based approach to detecting phishing websites." Proceedings of the 16th international conference on World Wide Web. 2007.
- [7] Aburrous, Maher, et al. "Intelligent phishing website detection system using fuzzy techniques." 2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications. IEEE, 2008.
- [8] Jun Ho Huh and Hyoungshick Kim. Phishing detection with popular search engines: Simple and effective. In Joaquin Garcia- Alfaro and Pascal Lafourcade, editors, Foundations and Practice of Security, pages 194–207, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
- [9] Routhu Srinivasa Rao and Alwyn Roshan Pais. Jail-phish: An improved search engine-based phishing detection system. Computers & Security, 83:246–267, 2019.

- [10] D. Kevin McGrath and Minaxi Gupta. Behind Phishing: An Examination of Phisher Modi Operandi. In Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats, LEET'08, pages 4:1–4:8, Berkeley, CA, USA, 2008. USENIX Association.
- [11] Justin Ma, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '09, pages 1245–1254, New York, NY, USA, 2009. ACM.
- [12] A. Le, A. Markopoulou, and M. Faloutsos. PhishDef: URL Names Say it All. In 2011 Proceedings IEEE INFOCOM, pages 191–195, April 2011.
- [13] Ozgur Koray Sahingoz, Ebubekir Buber, Onder Demir, and Banu Diri. Machine learning-based phishing detection from urls. *Expert Systems with Applications*, 117:345–357, 2019.
- [14] Rakesh Verma and Avisha Das. What's in a url: Fast feature extraction and malicious url detection. In Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics, IWSPA '17, pages 55–63, New York, NY, USA, 2017. ACM.
- [15] PhishTank. <https://www.antiphishing.org/resources/apwg-reports/>, 2022.
- [16] Harshal Tupsamudre, Ajeet Kumar Singh, and Sachin Lodha. Everything is in the name – a url based approach for phishing detection. In Shlomi Dolev, Danny Hendler, Sachin Lodha, and Moti Yung, editors, *Cyber Security Cryptography and Machine Learning*, pages 231–248, Cham, 2019. Springer International Publishing.
- [17] A. Le, A. Markopoulou, and M. Faloutsos. PhishDef: URL Names Say it All. In 2011 Proceedings IEEE INFOCOM, pages 191–195, April 2011.
- [18] DMOZ. <http://dmoz-odp.org/>, 2022.
- [19] Tyler Moore and Richard Clayton. *Financial Cryptography and Data Security*. chapter Evaluating the Wisdom of Crowds in Assessing Phishing Websites, pages 16–30. Springer-Verlag, Berlin, Heidelberg, 2008.
- [20] Colin Whittaker, Brian Ryner, and Marria Nazif. Large-scale classification of phishing pages. In NDSS '10, 2010.

- [21] Calvin Ardi and John Heidemann. AuntieTuna: Personalized Content-Based Phishing Detection. In Proceedings of the NDSS Workshop on Usable Security, San Diego, California, USA, February 2016. The Internet Society.
- [22] Sahoo, C. Liu, S.C.H. Hoi, “Malicious URL Detection using Machine Learning: A Survey”. CoRR, abs/1701.07179, 2017.
- [23] M. Khonji, Y. Iraqi, and A. Jones, “Phishing detection: a literature survey,” IEEE Communications Surveys & Tutorials, vol. 15, no. 4, pp. 2091–2121, 2013.
- [24] M. Cova, C. Kruegel, and G. Vigna, “Detection and analysis of driveby- download attacks and malicious javascript code,” in Proceedings of the 19th international conference on World wide web. ACM, 2010, pp. 281–290.
- [25] R. Heartfield and G. Loukas, “A taxonomy of attacks and a survey of defence mechanisms for semantic social engineering attacks,” ACM Computing Surveys (CSUR), vol. 48, no. 3, p. 37, 2015.
- [26] Internet Security Threat Report (ISTR) 2019–Symantec. <https://www.symantec.com/content/dam/symantec/docs/reports/istr-242019-en.pdf> [Last accessed 10/2019].

APPENDIX

Code

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import pandas as pd

from google.colab import drive
drive.mount('/content/gdrive/')

phishing_data = pd.read_csv('/content/gdrive/MyDrive/Sobur/phishing_site_urls.csv')
phishing_data.head()

phishing_data.tail()

phishing_data.info()

phishing_data.isnull().sum()

lbl_counts = pd.DataFrame(phishing_data.Label.value_counts())

import seaborn as sns

sns.set_style('darkgrid')
sns.barplot(lbl_counts.index, lbl_counts.Label)

from nltk.tokenize import RegexpTokenizer

tokenizer = RegexpTokenizer(r'[A-Za-z]+')
```

```

print(phishing_data.URL[0]) # This 0 is first row

clean_text = tokenizer.tokenize(phishing_data.URL[0])
print(clean_text)

import time
start = time.time()
phishing_data['text_tokenized'] = phishing_data.URL.map(lambda text: tokenizer.tokenize(text))
end = time.time()
time_req = end - start
formatted_time = "{:.2f}".format(time_req)
print(f"Time required to tokenize text is: \n{formatted_time} sec")

phishing_data.sample(7)

from nltk.stem.snowball import SnowballStemmer

sbs = SnowballStemmer("english")

# we will see the execution time to stem the tokenize text:
start = time.time()
phishing_data['text_stemmed'] = phishing_data['text_tokenized'].map(lambda text:
[sbs.stem(word) for word in text])
end = time.time()
time_req = end - start
formatted_time = "{:.2f}".format(time_req)
print(f"□ Time required for stemming all the tokenized text is: \n{formatted_time} sec")

phishing_data.sample(7)

start = time.time()

```

```
phishing_data['text_to_sent'] = phishing_data['text_stemmed'].map(lambda text: ' '.join(text))
end = time.time()
time_req = end - start
formatted_time = "{:.2f}".format(time_req)
print(f"Time required for joining text to sentence is: \n{formatted_time} sec")
```

```
phishing_data.sample(10)
```

```
phishing_sites = phishing_data[phishing_data.Label == 'bad']
not_phishing_sites = phishing_data[phishing_data.Label == 'good']
```

```
phishing_sites.head()
```

```
not_phishing_sites.head()
```

```
import matplotlib.pyplot as plt
```

```
import numpy as np
```

```
from wordcloud import WordCloud
```

```
from wordcloud import STOPWORDS
```

```
from PIL import Image
```

```
def my_wordcloud(text, mask=None, max_words=500, max_font_size=70, figure_size=(8.0,
10.0),
```

```
                title=None, title_size=70, image_color=False):
```

```
    stopwords = set(STOPWORDS)
```

```
    my_stopwords = {'com', 'http'}
```

```
    stopwords = stopwords.union(my_stopwords)
```

```
    wordcloud = WordCloud(background_color='#fff',
```

```
                          stopwords = stopwords,
```

```

        max_words = max_words,
        random_state = 42,
        mask = mask)

wordcloud.generate(text)

plt.figure(figsize=figure_size)

if image_color:
    image_color = ImageColorGenerator(mask);
    plt.imshow(wordcloud.recolor(color_func=image_color),
                interpolation='bilinear');

plt.title(title,
           fontdict={'size': title_size,
                    'verticalalignment': 'bottom'})

else:
    plt.imshow(wordcloud);
    plt.title(title,
              fontdict={'size': title_size,
                        'color': '#ff3333',
                        'verticalalignment': 'bottom'})

plt.axis('off');
plt.tight_layout()

d = '/content/gdrive/MyDrive/Sobur/'

my_data = not_phishing_sites.text_to_sent
my_data.reset_index(drop=True, inplace=True)

```

```
not_phishing_common_text = str(my_data)
#common_mask = np.array(Image.open(d+'idea.png'))
my_wordcloud(not_phishing_common_text,
             #common_mask,
             max_words=400,
             max_font_size=50,
             title = 'The Most common words use in not phishing URLs:',
             title_size=20)
```

```
my_data = phishing_sites.text_to_sent
my_data.reset_index(drop=True, inplace=True)
```

```
phishing_common_words = str(my_data)
#common_mask = np.array(Image.open(d+'target.png'))
my_wordcloud(phishing_common_words,
             #common_mask,
             max_words=500,
             max_font_size=20,
             title='The Most common words use in phishing URLs:',
             title_size=20)
```

```
from sklearn.feature_extraction.text import CountVectorizer
```

```
CV = CountVectorizer()
```

```
feature = CV.fit_transform(phishing_data.text_to_sent)
```

```
feature[:5].toarray()
```

```
from sklearn.model_selection import train_test_split
```



```

from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

train_X, test_X, train_Y, test_Y = train_test_split(feature, phishing_data.Label)

from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()

lr.fit(train_X, train_Y)

lr.score(test_X, test_Y)

Score_ml = {}
Score_ml['Logistic Regression'] = np.round(lr.score(test_X, test_Y), 2)

print('Training Accuracy: ',lr.score(train_X, train_Y))
print('Testing Accuracy: ',lr.score(test_X, test_Y))
# here we create confusion matrix:
conf_mat = pd.DataFrame(confusion_matrix(lr.predict(test_X), test_Y),
                        columns = ['Predicted: Phishing', 'Predicted: Not Phishing'],
                        index = ['Actual: Phishing', 'Actual: Not Phishing'])

print('\nClassification Report: \n')
print(classification_report(lr.predict(test_X), test_Y,
                            target_names = ['Bad', 'Good']))

print('\nconfusion Matrix: \n')
plt.figure(figsize = (6, 4))
sns.heatmap(conf_mat, annot = True, fmt='d', cmap="RdYlBu")
from sklearn.naive_bayes import MultinomialNB

```

```

mnb = MultinomialNB()
mnb.fit(train_X, train_Y)

mnb.score(test_X, test_Y)

Score_ml['MultinomialNB'] = np.round(mnb.score(test_X, test_Y), 2)

print("Training Accuracy: ',mnb.score(train_X, train_Y))
print("Testing Accuracy: ',mnb.score(test_X, test_Y))

conf_mat = pd.DataFrame(confusion_matrix(mnb.predict(test_X), test_Y),
                        columns = ['Predicted: Phishing', 'Predicted: Not Phishing'],
                        index = ['Actual: Phishing', 'Actual: Not Phishing'])

print("\nClassification Report\n')
print(classification_report(mnb.predict(test_X), test_Y,
                            target_names = ['Bad', 'Good']))

print("\nConfusion Matrix\n')
plt.figure(figsize = (6,4))
sns.heatmap(conf_mat, annot = True, fmt='d', cmap='PuBuGn_r')

results = pd.DataFrame.from_dict(Score_ml,
                                orient = 'index',
                                columns = ['Accuracy'])

print(f"Actual Score of Logistic Regression: \n{lr.score(test_X, test_Y)}\n")
print(f"Actual Score of MultinomialNB: \n{mnb.score(test_X, test_Y)}\n")
print(f"Final Rounded Score: \n{results}")

```

```

sns.set_style('darkgrid')
sns.barplot(results.index, results.Accuracy)

from sklearn.pipeline import make_pipeline

pipeline_ls = make_pipeline(CountVectorizer(tokenizer =
RegexTokenizer(r'[A-Za-z]+').tokenize, stop_words='english'), LogisticRegression())

train_X, test_X, train_Y, test_Y = train_test_split(phishing_data.URL, phishing_data.Label)

pipeline_ls.fit(train_X, train_Y)

pipeline_ls.score(test_X, test_Y)

print("Training Accuracy: ", pipeline_ls.score(train_X, train_Y))
print("Testing Accuracy: ", pipeline_ls.score(test_X, test_Y))

conf_mat = pd.DataFrame(confusion_matrix(pipeline_ls.predict(test_X), test_Y),
                          columns = ["Predicted: Phishing", "Predicted: Not Phishing"],
                          index = ["Actual: Phishing", "Actual: Not Phishing"])

print("\nClassification Report \n")
print(classification_report(pipeline_ls.predict(test_X), test_Y,
                             target_names = ['Bad', 'Good']))

print("\nConfusion Matrix \n")
plt.figure(figsize = (6,4))
sns.heatmap(conf_mat, annot = True, fmt = 'd', cmap="Blues")

import pickle
pickle.dump(pipeline_ls, open('phishing.pkl', 'wb'))

```

```

loaded_model = pickle.load(open('phishing.pkl', 'rb'))
result = loaded_model.score(test_X, test_Y)
print(result)

predict_bad = ['www.yeniik.com.tr/wp-admin/js/login.alibaba.com/login.jsp.php',
               'www.fazan-pacir.rs/temp/libraries/ipad',
               'www.tubemoviez.exe/',
               'www.youtube.com/',

               'www.svision-online.de/mgfi/administrator/components/com_babackup/classes/fx29id1.txt']

predict_good = ['www.youtube.com/',
                'www.python.org/',
                'www.google.com/',
                'https://colab.research.google.com',
                'www.canva.com/design/DAFS7vImEk0/USxaAza_HJgVAIMJczT7g/edit',
                'www.kaggle.com/']

loaded_model = pickle.load(open('phishing.pkl', 'rb'))
#predict_bad = vectorizers.transform(predict_bad)
# predict_good = vectorizer.transform(predict_good)

result_1 = loaded_model.predict(predict_bad)
result_2 = loaded_model.predict(predict_good)

print(f"{result_1} \n {'-'*26} \n{result_2}")

```

MACHINE LEARNING BASED PHISHING WEBSITE DETECTOR

Handwritten signature
11/01/23

ORIGINALITY REPORT

19%

SIMILARITY INDEX

10%

INTERNET SOURCES

8%

PUBLICATIONS

12%

STUDENT PAPERS

PRIMARY SOURCES

- | | | |
|----------|---|-----------|
| 1 | arxiv.org
Internet Source | 3% |
| 2 | Submitted to nith
Student Paper | 3% |
| 3 | Arijit Das, Ankita Das, Anisha Datta, Shukrity Si, Subhas Barman. "Deep Approaches on Malicious URL Classification", 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2020
Publication | 2% |
| 4 | Submitted to University of Hertfordshire
Student Paper | 2% |
| 5 | Submitted to Liverpool John Moores University
Student Paper | 2% |
| 6 | Cho Do Xuan, Hoa Dinh, Tisenko Victor. "Malicious URL Detection based on Machine Learning", International Journal of Advanced Computer Science and Applications, 2020
Publication | 2% |