**PREDICTION OF DIABETES EMPOWERED WITH FUSED MACHINE LEARNING**

*Report submitted to the SASTRA Deemed to be University as the requirement for the course*

**ICT300: MINI PROJECT**

*Submitted by*

**ANAMIKA TIWARI S**

**(Reg. No.: 125015010, B.TECH INFORMATION TECHNOLOGY)**

**THAMBI VISHNU AKSHAY**

**(Reg. No.: 125014080, B.TECH INFORMATION AND COMMUNICATION TECHNOLOGY)**

**AERUVAKA MANIKANTA**

**(Reg. No.: 125003173, B.TECH COMPUTER SCIENCE AND ENGINEERING)**

**MAY 2024**



## SCHOOL OF COMPUTING THANJAVUR, TAMIL NADU, INDIA– 613 401

i



**SCHOOL OF COMPUTING**

**THANJAVUR – 613 401**

# Bonafide Certificate

This is to certify that the report titled **A novel approach for phishing URLs detection using lexical-based machine learning in a real-time environment** submitted as a requirement for the course, **ICT300: MINI PROJECT** for B.Tech. is a bonafide record of the work done by **KANISHKA.K(Reg. No.: 125015010, B.TECH INFORMATION TECHNOLOGY), THAMBI.VISHNU AKSHAY(Reg. No.: 125014080, B.TECH INFORMATION AND**

**COMMUNICATION TECHNOLOGY), AERUVAKA MANIKANTA (Reg No:125003173, B.TECH COMPUTER SCIENCE AND ENGINEERING**) during the academic year 2023-24, in the School of Computing, under my supervision.

**Signature of Project Supervisor** :

**Name with Affiliation** :

**Date** :

Mini Project *Viva voc*e held on

**Examiner 1 Examiner 2**

ii

# ACKNOWLEDGEMENTS

We would like to thank our Honorable Chancellor **Prof. R. Sethuraman** for providing us with an opportunity and the necessary infrastructure for carrying out this project as a part of our curriculum.

We would like to thank our Honorable Vice-Chancellor **Dr. S. Vaidhyasubramaniam** and **Dr. S. Swaminathan,** Dean, Planning & Development, for the encouragement and strategic support at every step of our college life.

We extend our sincere thanks to **Dr. R. Chandramouli**, Registrar, SASTRA Deemed to be University for providing the opportunity to pursue this project.

We extend our heartfelt thanks to **Dr. V. S. Shankar Sriram**, Dean, School of

Computing, **Dr. R. Muthaiah**, Associate Dean, Research, **Dr. K.Ramkumar**, Associate Dean, Academics, **Dr. D. Manivannan**, Associate Dean, Infrastructure, **Dr. R. Algeswaran**, Associate Dean, Students Welfare for their motivation and support offered in materializing this project.

Our guide **Lakshmi.B**, Asst. Professor, School of Computing was the driving force behind this whole idea from the start. Her deep insight in the field and invaluable suggestions helped me in making progress throughout our project work. We also thank the project review panel members for their valuable comments and insights which made this project better.

We would like to extend our gratitude to all the teaching and non-teaching faculties of the School of Computing who have either directly or indirectly helped us in the completion of the project.

We gratefully acknowledge all the contributions and encouragement from my family and friends resulting in the successful completion of this project. We thank you all for providing me an opportunity to showcase my skills through project.

iii

# LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **Figure No** | **Title** | **Page No.** |
| 1.4 | Methodology | 2 |
| 2.1 | Existing Technique | 3 |
| 2.1.1 | Blacklisting phishing | 3 |
| 2.1.2 | Visual similarity based technique | 3 |
| 2.1.3 | Hybrid features based technique | 3 |
| 2.2 | Proposed Technique | 4 |
| 2.2.1 | Machine learning based technique | 4 |
| 4.1 | Legitimate Dataset | 23 |
| 4.2 | Online valid Dataset | 23 |
| 4.16 | Model Evaluation | 27 |
| 4.17 | Web application | 27 |

# ABBREVIATIONS

SSL Secure Socket Layer

HTML HyperText Markup Language

CSS Cascading Style Sheets

DNS Domain Name System.

iv

# ABSTRACT

Diabetes is a common health problem, and it is significant to find efficient ways to predict it in early stages and take action. Because of modern life style and due to the dietary habits the risk of diabetes and cholesterol have increased. To predict the disease, it is extremely important to understand its symptoms. By using machine learning (ML), this article introduces a novel approach to diabetes prediction by integrating Support Vector Machine (SVM) and Artificial Neural Network (ANN) models. The conceptual framework hinges on the integration of these two robust algorithms, which meticulously examine a dataset to find out the presence or absence of diabetes. The dataset employed in this research is meticulously partitioned into training and testing data, maintaining a ratio of 70:30. This segregation enables the models to learn from a substantial portion of the data while preserving an independent subset for rigorous evaluation. The outputs generated by the SVM and ANN models serve as input membership functions for a sophisticated fuzzy logic model. This fuzzy model then synthesizes the information gleaned from both algorithms, culminating in a decisive determination of whether a diabetes diagnosis is positive or negative. One of the important aspects of the proposed methodology is the utilization of cloud storage to archive the fused models. This forward-thinking approach ensures the accessibility and scalability of the models for future deployments, thereby enhancing the efficiency and adaptability of the system. In the context of real-time healthcare, the fused ML model exhibits remarkable predictive expertise. By assimilating a patient's ongoing medical records, the model offers instantaneous assessments regarding their diabetic status. The predictive accuracy of the proposed fused ML model impressively stands at 94.87%, outperforming previously published methods. This heightened precision signifies a substantial advancement in diabetes prediction methodologies, holding promising implications for timely diagnosis and intervention. In conclusion, the fusion of SVM and ANN models, coupled with a fuzzy logic system, represents a cutting-edge approach to diabetes prediction. The integration of cloud storage further fortifies the model's utility for future applications. With its superior predictive accuracy, this model stands as a beacon in the realm of disease detection, offering a potent tool for improving healthcare outcomes in the face of the global diabetes epidemic.

**Keywords:** Phishing, Legitimate, Precision, Recall, F1 score, Classifiers, URL.

v

## TABLE OF CONTENTS

### TITLE PAGE NO

[Bonafide Certificate ii](#_Toc42366)

[Acknowledgements iii](#_Toc42367)

[List of Figures iv](#_Toc42368)

[Abbreviations iv](#_Toc42369)

[Abstract v](#_Toc42370)

Summary of the Base Paper 1

1.1 Base Paper Details 1

1.2 Introduction 1

1.3 Solution Proposed 2

1.4 Methodology 2

Merits and Demerits of the Base Paper 3

2.1 Existing Techniques 3

2.2 Proposed Techniques 4

2.3 Merits 4

2.4 Demerits 4

Source code 5

3.1 Legitimate URLs Feature Extraction Code 5

3.2 Phishing URL Feature Extraction 7

3.3 Code Model Training code 11

1. Snapshots 23
2. Conclusion and Future Plans 28
3. References 29

## CHAPTER 1 SUMMARY OF THE BASE PAPER

### 1.1 BASE PAPER DETAILS

**Title**: A novel approach for phishing URLs detection using lexical-based machine learning in a real-time environment

**Journal Name**: Machine learning

**Publisher**: Daehee Jang

**Year**: 14th October, 2021

### 1.2 INTRODUCTION

The rise of new technologies and businesses has led to a surge in website creation, especially those requiring logins to gather user data. This abundance of websites, however, makes it difficult to tell real ones from fakes. As more people use the internet, attackers have a larger pool of potential victims. Phishing scams, traditionally launched through deceptive emails, have evolved to include fake links spread through online ads, social media posts, and messaging platforms. These cleverly designed websites mimic legitimate ones, tricking users into surrendering personal information like passwords and credit card details. Phishing can also be used to install malware on victims' devices. Modern phishing tactics go beyond appearance. Attackers may use convincing messages to lure users into entering their credentials on fake websites. Once submitted, this information falls into the hands of the attackers who control the site. Phishing schemes can even be used to trick victims into downloading malware, potentially adding their devices to a network controlled by the attacker. Studies show that many users have trouble distinguishing real websites from fraudulent ones, often falling prey to convincing content and design.

### 1.3 SOLUTION PROPOSED

Several phishing prevention software programs, including McAfee, Spoof Guard, Google Safe Browsing, utilise blacklisting techniques. Even though there areas many such programs designed to stop phishing attacks, scammers are still able to trick people and steal their personal information. Blacklisting methods used to be good at catching phishing attacks, but attackers are getting more smarter and finding many ways to trick the blacklist systems. .Attackers are able to easily escape from the detection systems by modifying the URLs of malicious links to match entries on the blacklisting database. To bypass detection, attackers can tinker with different parts of the web address with the suffix such as .com and .org , folders within the website, the company name shown in the address, or the search query part at the end. As a result, the old way of blacklisting technique is not enough. Phishing attacks are getting more wiser for these outdated techniques. This is the reason why we urgently need new anti-phishing tools which can spot fake websites on their own, without having to rely on outdated blacklists. To combat this issue, some researchers have suggested using artificial intelligence, specifically machine learning, is the best way to fight against the phishing attacks. While

1

machine learning seems like a good solution, some approaches have downsides. For instance, they might mistakenly flag safe sites as phishing attempts, or take too long to analyse websites. This can be caused by needing a lot of processing time or getting confused by unnecessary information. In this study, a new anti-phishing system is designed to identify phishing URLs the moment you try to visit them. Unlike other tools, this system works independently, without relying on information from the external sources. Our strategy for detecting phishing attacks focuses more on high accuracy with less and minimal features. To do this work, we examined what makes phishing websites tick according to the previous research. Based on this, we have made a powerful method to identify these deceptive sites. We developed fifteen lexical-based features and have evaluated them using various machine learning classifiers, achieving the highest accuracy rate.

### 1.4 METHODOLOGY

The suggested system design is illustrated in Figure 1.4.1. When a user tries to visit a new website,the system breaks down the address into its different parts. We have created a method, coded in Python, that extracts specific details that is lexical information from the URL. This extracted information is then transformed into a format suitable for analysis by machine learning algorithms. Finally, the machine learning algorithms analyze this data and decide if the website is real or a phishing attempt. If it's legitimate, you can access it, otherwise, the system blocks it.

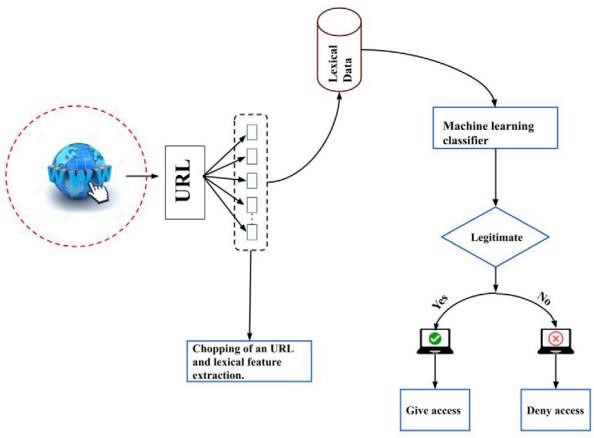


Fig 1.4.1 Proposed Phishing detection architecture.

2

**CHAPTER 2**

## MERITS AND DEMERITS OF THE BASE PAPER

**2.1 EXISTING TECHNIQUES:**

### 2.1.1 Blacklisting phishing

Traditionally, catching phishing websites involves keeping a list of bad ones. This list includes website addresses, entire domains, and even internet addresses .

Whenever someone tries to visit a new website, the system checks this list to see if it matches anything bad. If there's a match, it gets blocked. But here's the problem: this list needs constant updates, ideally every 12 hours according to the study. Even then, attackers are sneaky. They can slightly change the website address, like using a different domain name, path, or adding something to the search part of the URL. These tweaks trick the system and keep the phishing website operational.

Blacklisting is like playing catch-up. It only works against dangers we already know about. It can't stop all phishing attacks, leaving other ways for attackers to exploit people. Making a new website address is easy, and if it's not on the list yet, it can slip through the cracks. This leaves users vulnerable to entirely new phishing attempts and websites that haven't been flagged as bad yet.

### 2.1.2 Visual similarity-based technique

People are getting tricked by fake websites because they focus too much on how the address looks, which can be easily copied to look real. They forget to check things like the URL itself carefully and security certificates SSL.

One way to catch these phishers is to look at how the website is built, like its code in HTML and CSS and images. If a suspicious website looks very similar to a real one in these aspects, it might be a fake. There are even tools that use this approach and can be somewhat accurate around 80% success.

Another method involves keeping a list of good websites and comparing them to suspicious ones. The idea is that phishers often copy how real websites look. This method can work, but again, it only works if the fake site copies a real one. If the phisher comes up with a completely new design, or tricks people in other ways like social engineering, this method won't help.

3

### 2.1.3 Hybrid features based technique

Instead of relying on just one method, some experts suggest a more well-rounded approach that is the hybrid technique that combines different ways to spot phishing attacks. This includes examining the website address, its content, and the domain it's on. By looking at all these things together, they believe they can catch more phishing attempts. One idea is to combine machine learning, which can analyze data patterns, with image analysis to get a better picture of whether a website is fake. However, this approach isn't without its drawbacks. It can be more complex to set up and maintain compared to simpler detection methods, requiring more technical knowledge and resources. Additionally, while it might catch fewer false positives mistakenly blocking safe sites, it might also miss some phishing attempts false negatives if they're particularly sneaky.

### 2.2 PROPOSED TECHNIQUE

#### 2.2.1Machine learning based techniques

Recent research on fighting phishing attacks with machine learning suggests building a big database of both real and fake websites. This involves collecting information like websites address, content on the page, and technical details that is the DNS record for both types of sites. This data is then cleaned up and fed into machine learning algorithms to learn how to spot phishing attempts. The accuracy of this approach depends on what kind of data is used that is the feature sets and which machine learning methods are chosen. The authors of this study have come up with a new method that avoids limitations like needing to understand the language used on the website or relying on external sources for information. Their approach allows for identifying phishing websites in real-time, making it very efficient and dependable.

### 2.3 MERITS

Machine learning is a superstar in the fight against phishing websites because it can handle enormous amounts of data. By analyzing this data, it can learn to recognize patterns that give away phishing attempts. This lets machine learning identify these threats in real-time, warning users right away so they can avoid them. Machine learning is also great at scaling up, meaning it can handle the huge number of phishing URLs out there by processing tons of data quickly. Plus, it automates the process, reducing the need for human involvement and making phishing detection faster and more efficient.

### 2.4 DEMERITS

Machine learning isn't perfect. Sometimes it might mistake a safe website for a phishing attempt that is false positives. This can be annoying for users and make them lose faith in the system.

Additionally, attackers can try to trick machine learning algorithms adversarial attacks. To fight back, these algorithms need regular updates to stay on top of new attacker tactics and keep catching phishing attempts effectively**.**

4

## CHAPTER 3 SOURCE CODE

### 3.1 SOURCE CODE

#### 3.1.1 Legitimate URLs Feature Extraction Code

import ipaddress import re

#1. Using the IP Address def having\_ip\_address(url): try:

ipaddress.ip\_address(url) ip = 1 except:

ip = 0

return ip

legiturldata['having\_ip\_address'] = legitdataset['URLs'].apply(having\_ip\_address)

#2. Long URL def long\_url(l):

if len(l) < 54:

return 0

elif len(l) >= 54 and len(l) <= 75: return 2

return 1 legiturldata['long\_url'] = legitdataset['URLs'].apply(long\_url)

#3. Using URL Shortening Services “TinyURL” def shortening\_service(url):

match=re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|'

'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|'

'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|'

'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|'

'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|'

'q\.gs|is\.gd|po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|'

'x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.me|v\.gd|tr\.im|l ink\.zip\.net',url) if match: return 1

else: return 0 legiturldata['shortening\_service'] = legitdataset['URLs'].apply(shortening\_service)

#4. URL’s having “@” Symbol

5

def have\_at\_symbol(l): if "@" in l:

return 1

return 0 legiturldata['having\_@\_symbol'] = legitdataset['URLs'].apply(have\_at\_symbol)

#5. Redirecting using “//” def redirection(l):

if "//" in l:

return 1

return 0 legiturldata['redirection\_//\_symbol'] = seperation\_of\_protocol[1].apply(redirection)

#6. Adding Prefix or Suffix Separated by (-) to the Domain def prefix\_suffix\_seperation(l): if '-' in l:

return 1

return 0

legiturldata['prefix\_suffix\_seperation'] =

seperation\_domain\_name['domain\_name'].apply(prefix\_suffix\_seperation)

#7. Sub Domain and Multi Sub Domains def sub\_domains(l): if l.count('.') < 3: return 0

elif l.count('.') == 3:

return 2

return 1 legiturldata['sub\_domains'] = legiturldata['domain\_name'].apply(sub\_domains)

#8. The Existence of “HTTPS” Token in the Domain Part of the URL def https\_token(url): match=re.search('https://|http://',url) if match.start(0)==0:

url=url[match.end(0):]

match=re.search('http|https',url) if match:

return 1

else:

return 0

legiturldata['https\_token'] = legitdataset['URLs'].apply(https\_token) """### \*\*2. Domain based Features\*\*

* Age of Domain \* DNS Record
* Website Traffic
* Domain Registration Length
* Statistical-Reports Based Feature """

!pip install python-whois

import whois from bs4 import BeautifulSoup import urllib.request from urllib.parse import quote from datetime import datetime import time import socket import re

#9. Age of Domain def age\_of\_domain\_sub(domain\_name):

creation\_date = domain\_name.creation\_date expiration\_date = domain\_name.expiration\_date if (isinstance(creation\_date,str) or isinstance(expiration\_date,str)): try:

creation\_date = datetime.strptime(creation\_date,'%Y-%m-%d')

expiration\_date = datetime.strptime(expiration\_date,"%Y-%m-%d") except:

return 1

if ((expiration\_date is None) or (creation\_date is None)): return 1

elif ((type(expiration\_date) is list) or (type(creation\_date) is list)):

return 2

else:

ageofdomain = abs((expiration\_date - creation\_date).days) if ((ageofdomain/30) < 6):

age = 1

else:

age = 0

return age

def age\_of\_domain\_main(domain):

dns = 0 try:

domain\_name = whois.whois(domain)

except: dns = 1

if dns == 1: return 1

else:

return age\_of\_domain\_sub(domain\_name)

7

legiturldata['age\_of\_domain'] = legitdataset['URLs'].apply(age\_of\_domain\_main)

#10.DNS Record def dns\_record(domain): dns = 0 try:

domain\_name = whois.whois(domain) print(domain\_name)

except:

dns = 1

if dns == 1:

return 1

else: return dns legiturldata['dns\_record'] = legiturldata['domain\_name'].apply(dns\_record)

# 11. Web traffic def web\_traffic(url):

try:

url = urllib.parse.quote(url)

rank = [BeautifulSoup(urllib.request.urlopen("http://data.alexa.com/data?cli=10&dat=s&url="](http://data.alexa.com/data?cli=10&dat=s&url) +

url).read(), "xml").find(

"REACH")['RANK']

rank = int(rank)

except TypeError: return 1

if rank <100000:

return 1

else: return 2 legiturldata['web\_traffic'] = legitdataset['URLs'].apply(web\_traffic)

#12. Domain Registration Length def domain\_registration\_length\_sub(domain): expiration\_date = domain.expiration\_date today = time.strftime('%Y-%m-%d') today = datetime.strptime(today, '%Y-%m-%d') if expiration\_date is None:

return 1

elif type(expiration\_date) is list or type(today) is list :

return 2 #If it is a type of list then we can't select a single value from list. So,it is

regarded as suspected website else:

registration\_length = abs((expiration\_date - today).days) if registration\_length / 365 <= 1:

return 1 else:

return 0 def domain\_registration\_length\_main(domain): dns = 0 try:

domain\_name = whois.whois(domain)

except:

dns = 1

if dns == 1:

return 1

else: return domain\_registration\_length\_sub(domain\_name)

legiturldata['domain\_registration\_length'] =

legitdataset['URLs'].apply(domain\_registration\_length\_main)

#13. Statistical-Reports Based Feature def statistical\_report(url):

hostname = url h = [(x.start(0), x.end(0)) for x in [re.finditer('https://|http://|www.|https://www.](http://www/)|http://www.',

hostname)] z = int(len(h)) if z != 0:

y = h[0][1]

hostname = hostname[y:]

h = [(x.start(0), x.end(0)) for x in re.finditer('/', hostname)] z = int(len(h)) if z != 0: hostname = hostname[:h[0][0]]

url\_match=re.search('at\.ua|usa\.cc|baltazarpresentes\.com\.br|pe\.hu|esy\.es|hol\.es|sweddy\.com|myji no\.ru|96\.lt|ow\.ly',url) try:

ip\_address = socket.gethostbyname(hostname)

ip\_match=re.search('146\.112\.61\.108|213\.174\.157\.151|121\.50\.168\.88|192\.185\.217\.116|78\.46 \.211\.158|181\.174\.165\.13|46\.242\.145\.103|121\.50\.168\.40|83\.125\.22\.219|46\.242\.145\.98|10

7\.151\.148\.44|107\.151\.148\.107|64\.70\.19\.203|199\.184\.144\.27|107\.151\.148\.108|107\.151\.14 8\.109|119\.28\.52\.61|54\.83\.43\.69|52\.69\.166\.231|216\.58\.192\.225|118\.184\.25\.86|67\.208\.74\

.71|23\.253\.126\.58|104\.239\.157\.210|175\.126\.123\.219|141\.8\.224\.221|10\.10\.10\.10|43\.229\.1 08\.32|103\.232\.215\.140|69\.172\.201\.153|216\.218\.185\.162|54\.225\.104\.146|103\.243\.24\.98|1 99\.59\.243\.120|31\.170\.160\.61|213\.19\.128\.77|62\.113\.226\.131|208\.100\.26\.234|195\.16\.127\.

102|195\.16\.127\.157|34\.196\.13\.28|103\.224\.212\.222|172\.217\.4\.225|54\.72\.9\.51|192\.64\.147\ .141|198\.200\.56\.183|23\.253\.164\.103|52\.48\.191\.26|52\.214\.197\.72|87\.98\.255\.18|209\.99\.17 \.27|216\.38\.62\.18|104\.130\.124\.96|47\.89\.58\.141|78\.46\.211\.158|54\.86\.225\.156|54\.82\.156\.

19|37\.157\.192\.102|204\.11\.56\.48|110\.34\.231\.42',ip\_address) except:

return 1

if url\_match: return 1

else: return 0 legiturldata['statistical\_report'] = legiturldata['domain\_name'].apply(statistical\_report)

"""### \*\*3.HTML and JavaScript based Features\*\*

* IFrame Redirection
* Status Bar Customization

"""

import requests import re

#14.iFrame Redirection def iframe\_sub(response):

if response == "":

return 1

else:

if re.findall(r"[<iframe>|<frameBorder>]", response.text): return 0

else: return 1

def iframe\_main(url): try: response = requests.get(url)

except: response = '' return iframe\_sub(response) legiturldata['iframe'] = legitdataset['URLs'].apply(iframe\_main)

#15. Status Bar Customization def mouse\_over\_sub(response):

if response == "" : return 1

else:

if re.findall("<script>.+onmouseover.+</script>", response.text):

return 1

else:

return 0

def mouse\_over\_main(url): try: response = requests.get(url)

except:

response = ''

return mouse\_over\_sub(response)

legiturldata['mouse\_over'] = legitdataset['URLs'].apply(mouse\_over\_main) legiturldata['label'] = 0

"""## Storing the extracted legitimate URLs fatures to csv file"""

legiturldata.to\_csv('/content/drive/MyDrive/extracted\_dataset/extracted\_legitmate\_dataset.csv',index =False)

**3.1.2 Phishing URL Feature Extraction Code** #1. Using the IP Address def having\_ip\_address(url): try:

ipaddress.ip\_address(url) ip = 1 except:

ip = 0

return ip phishurldata['having\_ip\_address'] = phishdataset['url'].apply(having\_ip\_address)

#2. Long URL

def long\_url(l):

if len(l) < 54:

return 0

elif len(l) >= 54 and len(l) <= 75: return 2

return 1 phishurldata['long\_url'] = phishdataset['url'].apply(long\_url)

#3. Using URL Shortening Services “TinyURL” def shortening\_service(url):

match=re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|'

'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|'

'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|'

'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|'

'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|'

'q\.gs|is\.gd|po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|'

'x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.me|v\.gd|tr\.im|l ink\.zip\.net',url) if match:

return 1

else:

return 0

phishurldata['shortening\_service'] = phishdataset['url'].apply(shortening\_service)

#4. URL’s having “@” Symbol def have\_at\_symbol(l):

if "@" in l:

return 1

return 0 phishurldata['having\_@\_symbol'] = phishdataset['url'].apply(have\_at\_symbol)

#5. Redirecting using “//” def redirection(l):

if "//" in l:

return 1

return 0 phishurldata['redirection\_//\_symbol'] = seperation\_of\_protocol[1].apply(redirection)

#6. Adding Prefix or Suffix Separated by (-) to the Domain def prefix\_suffix\_seperation(l): if '-' in l:

return 1

return 0

phishurldata['prefix\_suffix\_seperation'] =

seperation\_domain\_name['domain\_name'].apply(prefix\_suffix\_seperation)

#7. Sub Domain and Multi Sub Domains def sub\_domains(l): if l.count('.') < 3: return 0

elif l.count('.') == 3:

return 2

return 1 phishurldata['sub\_domains'] = phishurldata['domain\_name'].apply(sub\_domains)

#8. The Existence of “HTTPS” Token in the Domain Part of the URL def https\_token(url): match=re.search('https://|http://',url) if match.start(0)==0:

url=url[match.end(0):]

match=re.search('http|https',url) if match:

return 1

else:

return 0

phishurldata['https\_token'] = phishdataset['url'].apply(https\_token)

"""### \*\*2. Domain based Features\*\*

12

* Age of Domain \* DNS Record
* Website Traffic
* Domain Registration Length
* Statical-Report Based Feature """

!pip install python-whois

import whois

from bs4 import BeautifulSoup import urllib.request from urllib.parse import quote from datetime import datetime import time import socket import re

#9. Age of Domain def age\_of\_domain\_sub(domain):

creation\_date = domain.creation\_date expiration\_date = domain.expiration\_date if ((expiration\_date is None) or (creation\_date is None)): return 1

elif ((type(expiration\_date) is list) or (type(creation\_date) is list)): return 2

else:

ageofdomain = abs((expiration\_date - creation\_date).days) if ((ageofdomain/30) < 6): return 1

else: return 0

def age\_of\_domain\_main(domain):

dns = 0 try:

domain\_name = whois.whois(domain)

except: dns = 1

if dns == 1:

return 1

else:

return age\_of\_domain\_sub(domain\_name) phishurldata['age\_of\_domain'] = phishurldata['domain\_name'].apply(age\_of\_domain\_main)

#10.DNS Record def dns\_record(domain): dns = 0

try:

domain\_name = whois.whois(domain) print(domain\_name)

except:

dns = 1

if dns == 1:

return 1

else: return dns phishurldata['dns\_record'] = phishurldata['domain\_name'].apply(dns\_record)

# 11. Web traffic def web\_traffic(url):

try:

url = urllib.parse.quote(url)

rank = [BeautifulSoup(urllib.request.urlopen("http://data.alexa.com/data?cli=10&dat=s&url="](http://data.alexa.com/data?cli=10&dat=s&url) +

url).read(), "xml").find(

"REACH")['RANK']

rank = int(rank)

except TypeError: return 1

if rank <100000:

return 1

else: return 2 phishurldata['web\_traffic'] = phishdataset['url'].apply(web\_traffic)

#12. Domain Registration Length def domain\_registration\_length\_sub(domain): expiration\_date = domain.expiration\_date today = time.strftime('%Y-%m-%d') today = datetime.strptime(today, '%Y-%m-%d') if expiration\_date is None:

return 1

elif type(expiration\_date) is list or type(today) is list :

return 2 #If it is a type of list then we can't select a single value from list. So,it is

regarded as suspected website else:

registration\_length = abs((expiration\_date - today).days) if registration\_length / 365 <= 1: return 1

else: return 0

def domain\_registration\_length\_main(domain):

dns = 0 try:

domain\_name = whois.whois(domain)

except:

14

dns = 1

if dns == 1:

return 1

else:

return domain\_registration\_length\_sub(domain\_name)

phishurldata['domain\_registration\_length'] =

phishurldata['domain\_name'].apply(domain\_registration\_length\_main) phishurldata.shape

#13.Statical-Report Based Feature def statistical\_report(url):

hostname = url h = [(x.start(0), x.end(0)) for x in [re.finditer('https://|http://|www.|https://www.](http://www/)|http://www.',

hostname)] z = int(len(h)) if z != 0:

y = h[0][1]

hostname = hostname[y:]

h = [(x.start(0), x.end(0)) for x in re.finditer('/', hostname)] z = int(len(h)) if z != 0: hostname = hostname[:h[0][0]]

url\_match=re.search('at\.ua|usa\.cc|baltazarpresentes\.com\.br|pe\.hu|esy\.es|hol\.es|sweddy\.com|myji no\.ru|96\.lt|ow\.ly',url) try:

ip\_address = socket.gethostbyname(hostname)

ip\_match=re.search('146\.112\.61\.108|213\.174\.157\.151|121\.50\.168\.88|192\.185\.217\.116|78\.46 \.211\.158|181\.174\.165\.13|46\.242\.145\.103|121\.50\.168\.40|83\.125\.22\.219|46\.242\.145\.98|10

7\.151\.148\.44|107\.151\.148\.107|64\.70\.19\.203|199\.184\.144\.27|107\.151\.148\.108|107\.151\.14 8\.109|119\.28\.52\.61|54\.83\.43\.69|52\.69\.166\.231|216\.58\.192\.225|118\.184\.25\.86|67\.208\.74\

.71|23\.253\.126\.58|104\.239\.157\.210|175\.126\.123\.219|141\.8\.224\.221|10\.10\.10\.10|43\.229\.1 08\.32|103\.232\.215\.140|69\.172\.201\.153|216\.218\.185\.162|54\.225\.104\.146|103\.243\.24\.98|1 99\.59\.243\.120|31\.170\.160\.61|213\.19\.128\.77|62\.113\.226\.131|208\.100\.26\.234|195\.16\.127\. 102|195\.16\.127\.157|34\.196\.13\.28|103\.224\.212\.222|172\.217\.4\.225|54\.72\.9\.51|192\.64\.147\ .141|198\.200\.56\.183|23\.253\.164\.103|52\.48\.191\.26|52\.214\.197\.72|87\.98\.255\.18|209\.99\.17 \.27|216\.38\.62\.18|104\.130\.124\.96|47\.89\.58\.141|78\.46\.211\.158|54\.86\.225\.156|54\.82\.156\.

19|37\.157\.192\.102|204\.11\.56\.48|110\.34\.231\.42',ip\_address) except:

return 1

if url\_match:

return 1

else:

return 0

phishurldata['statistical\_report'] = phishdataset['url'].apply(statistical\_report)

"""### \*\*3.HTML and JavaScript based Features\*\*

* IFrame Redirection
* Status Bar Customization

"""

#14.iFrame Redirection def iframe\_sub(response):

if response == "":

return 1

else:

if re.findall(r"[<iframe>|<frameBorder>]", response.text): return 0

else: return 1

def iframe\_main(url): try: response = requests.get(url)

except: response = '' return iframe\_sub(response) phishurldata['iframe'] = phishdataset['url'].apply(iframe\_main)

#15. Status Bar Customization def mouse\_over\_sub(response):

if response == "" :

return 1

else: if re.findall("<script>.+onmouseover.+</script>", response.text): return 1

else: return 0

def mouse\_over\_main(url): try: response = requests.get(url)

except: response = ''

return mouse\_over\_sub(response)

phishurldata['mouse\_over'] = phishdataset['url'].apply(mouse\_over\_main) phishurldata['label'] = 1

phishurldata.shape

16

phishurldata.head()

"""## Storing the extracted legitimate URLs fatures to csv file""" phishurldata.to\_csv('extracted\_phishing\_dataset.csv', index= False)

#### 3.1.3 Model Training code

# Splitting the dataset into train and test sets: 80-20 split from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 12)

X\_train.shape, X\_test.shape y\_train.value\_counts() y\_test.value\_counts()

"""## \*\*Machine Learning Models & Training\*\*

This data set comes under classification problem, as the input URL is classified as phishing (1) or legitimate (0). The supervised machine learning models (classification) considered to train the dataset in this notebook are:

"""

from sklearn.metrics import precision\_score from sklearn.metrics import recall\_score from sklearn.metrics import f1\_score from sklearn.metrics import accuracy\_score

# Creating holders to store the model performance results

ML\_Model = [] precision=[] recall=[] f1=[] accuracy = []

#function to call for storing the results def storeResults(model, a,b,c,d): ML\_Model.append(model) precision.append(round(a, 3))

recall.append(round(b, 3)) f1.append(round(c, 3)) accuracy.append(round(d, 3))

import matplotlib.pyplot as plt from sklearn.metrics import roc\_curve, auc """### \*\*Support Vector Machines\*\*"""

#Support vector machine model from sklearn.svm import SVC

svm = SVC(kernel='linear', C=1.0, random\_state=12) svm.fit(X\_train, y\_train)

#predicting the target value from the model for the samples y\_pred\_svm = svm.predict(X\_test)

#computing the accuracy of the model performance precision\_svm = precision\_score(y\_test,y\_pred\_svm) recall\_svm = recall\_score(y\_test,y\_pred\_svm) f1\_svm = f1\_score(y\_test,y\_pred\_svm) accuracy\_svm = accuracy\_score(y\_test,y\_pred\_svm)

print("SVM: Precision on test Data: {:.3f}".format(precision\_svm)) print("SVM: Recall on test Data: {:.3f}".format(recall\_svm)) print("SVM: F-Measure on test Data: {:.3f}".format(f1\_svm)) print("SVM: Accuracy on test Data: {:.3f}".format(accuracy\_svm))

fpr1, tpr1, thresholds = roc\_curve(y\_test, y\_pred\_svm) roc\_auc1 = auc(fpr1, tpr1)

plt.figure()

plt.plot(fpr1, tpr1, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc\_auc1) plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--') plt.xlim([0.0, 1.0]) plt.ylim([0.0, 1.05]) plt.xlabel('False Positive Rate') plt.ylabel('True Positive Rate') plt.title('Receiver operating characteristic')

plt.legend(loc="lower right") plt.show() storeResults('SVM', precision\_svm,recall\_svm,f1\_svm,accuracy\_svm)

"""### \*\*Logistic Regression\*\*"""

#logistic regression model from sklearn.linear\_model import LogisticRegression lr\_regression = LogisticRegression(random\_state = 0) lr\_regression.fit(X\_train, y\_train)

#predicting the target value from the model for the samples y\_pred\_lr = lr\_regression.predict(X\_test)

#computing the accuracy of the model performance precision\_lr = precision\_score(y\_test,y\_pred\_lr) recall\_lr = recall\_score(y\_test,y\_pred\_lr) f1\_lr = f1\_score(y\_test,y\_pred\_lr) accuracy\_lr = accuracy\_score(y\_test,y\_pred\_lr) print("Logistic Regression: Precision on test Data: {:.3f}".format(precision\_lr)) print("Logistic Regression: Recall on test Data: {:.3f}".format(recall\_lr)) print("Logistic Regression: F-Measure on test Data: {:.3f}".format(f1\_lr)) print("Logistic Regression: Accuracy on test Data: {:.3f}".format(accuracy\_lr))

fpr2, tpr2, thresholds = roc\_curve(y\_test, y\_pred\_lr)

roc\_auc2 = auc(fpr2, tpr2)

plt.figure()

plt.plot(fpr2, tpr2, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc\_auc2) plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--') plt.xlim([0.0, 1.0]) plt.ylim([0.0, 1.05]) plt.xlabel('False Positive Rate') plt.ylabel('True Positive Rate') plt.title('Receiver operating characteristic')

plt.legend(loc="lower right") plt.show() storeResults('Logistic Regression', precision\_svm,recall\_lr,f1\_svm,accuracy\_lr)

"""### \*\*DECISION TREE\*\*"""

# Decision Tree model from sklearn.tree import DecisionTreeClassifier tree = DecisionTreeClassifier(max\_depth = 5) tree.fit(X\_train, y\_train)

#predicting the target value from the model for the samples y\_pred\_tree = tree.predict(X\_test) #y\_train\_tree = tree.predict(X\_train)

#computing the accuracy of the model performance #acc\_train\_tree = accuracy\_score(y\_train,y\_train\_tree) precision\_tree = precision\_score(y\_test,y\_pred\_tree) recall\_tree = recall\_score(y\_test,y\_pred\_tree) f1\_tree = f1\_score(y\_test,y\_pred\_tree) accuracy\_tree = accuracy\_score(y\_test,y\_pred\_tree)

#print("Decision Tree: Accuracy on training Data: {:.3f}".format(acc\_train\_tree)) print("Decision Tree: Precision on test Data: {:.3f}".format(precision\_tree)) print("Decision Tree: Recall on test Data: {:.3f}".format(recall\_tree)) print("Decision Tree: F-Measure on test Data: {:.3f}".format(f1\_tree)) print("Decision Tree: Accuracy on test Data: {:.3f}".format(accuracy\_tree))

fpr3, tpr3, thresholds = roc\_curve(y\_test, y\_pred\_tree) roc\_auc3 = auc(fpr3, tpr3)

plt.figure()

plt.plot(fpr3, tpr3, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc\_auc3) plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlim([0.0, 1.0]) plt.ylim([0.0, 1.05]) plt.xlabel('False Positive Rate') plt.ylabel('True Positive Rate') plt.title('Receiver operating characteristic')

plt.legend(loc="lower right") plt.show() storeResults('Decision Tree', precision\_tree,recall\_tree,f1\_tree,accuracy\_tree)

"""### \*\*RANDOM FOREST\*\*"""

# Random Forest model from sklearn.ensemble import RandomForestClassifier forest = RandomForestClassifier(max\_depth=5) forest.fit(X\_train, y\_train)

#predicting the target value from the model for the samples y\_pred\_forest = forest.predict(X\_test) #y\_train\_forest = forest.predict(X\_train)

#computing the accuracy of the model performance precision\_forest = precision\_score(y\_test,y\_pred\_forest) recall\_forest = recall\_score(y\_test,y\_pred\_forest) f1\_forest = f1\_score(y\_test,y\_pred\_forest) accuracy\_forest = accuracy\_score(y\_test,y\_pred\_forest)

print("Random Tree: Precision on test Data: {:.3f}".format(precision\_forest)) print("Random forest: Recall on test Data: {:.3f}".format(recall\_forest)) print("Random forest: F-Measure on test Data: {:.3f}".format(f1\_forest)) print("Random forest: Accuracy on test Data: {:.3f}".format(accuracy\_forest))

fpr4, tpr4, thresholds = roc\_curve(y\_test, y\_pred\_forest) roc\_auc4 = auc(fpr4, tpr4)

plt.figure()

plt.plot(fpr4, tpr4, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc\_auc4) plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--') plt.xlim([0.0, 1.0]) plt.ylim([0.0, 1.05]) plt.xlabel('False Positive Rate') plt.ylabel('True Positive Rate') plt.title('Receiver operating characteristic')

plt.legend(loc="lower right") plt.show() storeResults('Random Forest', precision\_forest,recall\_forest,f1\_forest,accuracy\_forest)

"""### \*\*k-NEAREST NEIGHBOR\*\*"""

#k nearest neighbour model

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n\_neighbors = 5, metric = 'minkowski', p = 2) classifier.fit(X\_train, y\_train)

#predicting the target value from the model for the samples y\_pred\_knnc = classifier.predict(X\_test)

#computing the accuracy of the model performance precision\_knnc = precision\_score(y\_test,y\_pred\_knnc) recall\_knnc = recall\_score(y\_test,y\_pred\_knnc) f1\_knnc = f1\_score(y\_test,y\_pred\_knnc) accuracy\_knnc = accuracy\_score(y\_test,y\_pred\_knnc)

print("K Nearest Neighbours: Precision on test Data: {:.3f}".format(precision\_knnc)) print("K Nearest Neighbours: Recall on test Data: {:.3f}".format(recall\_knnc)) print("K Nearest Neighbours: F-Measure on test Data: {:.3f}".format(f1\_knnc)) print("K Nearest Neighbours: Accuracy on test Data: {:.3f}".format(accuracy\_knnc))

fpr5, tpr5, thresholds = roc\_curve(y\_test, y\_pred\_knnc) roc\_auc5 = auc(fpr5, tpr5)

plt.figure()

plt.plot(fpr5, tpr5, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc\_auc5) plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--') plt.xlim([0.0, 1.0]) plt.ylim([0.0, 1.05]) plt.xlabel('False Positive Rate') plt.ylabel('True Positive Rate') plt.title('Receiver operating characteristic')

plt.legend(loc="lower right") plt.show() storeResults('K Nearest Neighbor', precision\_svm,recall\_knnc,f1\_svm,accuracy\_knnc)

"""### \*\*XGBoost\*\*"""

#XGBoost Classification model from xgboost import XGBClassifier

xgb = XGBClassifier(learning\_rate=0.4,max\_depth=7) xgb.fit(X\_train, y\_train)

#predicting the target value from the model for the samples y\_pred\_xgb = xgb.predict(X\_test)

#computing the accuracy of the model performance precision\_xgb = precision\_score(y\_test,y\_pred\_xgb) recall\_xgb = recall\_score(y\_test,y\_pred\_xgb) f1\_xgb = f1\_score(y\_test,y\_pred\_xgb) accuracy\_xgb = accuracy\_score(y\_test,y\_pred\_xgb)

print("XGBoost: Precision on test Data: {:.3f}".format(precision\_xgb)) print("XGBoost: Recall on test Data: {:.3f}".format(recall\_xgb)) print("XGBoost: F-Measure on test Data: {:.3f}".format(f1\_xgb)) print("XGBoost: Accuracy on test Data: {:.3f}".format(accuracy\_xgb))

fpr6, tpr6, thresholds = roc\_curve(y\_test, y\_pred\_xgb) roc\_auc6 = auc(fpr6, tpr6) plt.figure()

plt.plot(fpr6, tpr6, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc\_auc6) plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--') plt.xlim([0.0, 1.0]) plt.ylim([0.0, 1.05]) plt.xlabel('False Positive Rate') plt.ylabel('True Positive Rate') plt.title('Receiver operating characteristic')

plt.legend(loc="lower right") plt.show() storeResults('XGBoost', precision\_xgb,recall\_xgb,f1\_xgb,accuracy\_xgb)

"""## \*\*AdaBoost\*\*"""

#ada boost model

from sklearn.ensemble import AdaBoostClassifier

ab\_classifier = AdaBoostClassifier(n\_estimators=50,random\_state=0) ab\_classifier.fit(X\_train, y\_train)

#predicting the target value from the model for the samples y\_pred\_ada = ab\_classifier.predict(X\_test)

#computing the accuracy of the model performance precision\_ada = precision\_score(y\_test,y\_pred\_ada) recall\_ada = recall\_score(y\_test,y\_pred\_ada) f1\_ada = f1\_score(y\_test,y\_pred\_ada) accuracy\_ada = accuracy\_score(y\_test,y\_pred\_ada)

print("AdaBoost : Precision on test Data: {:.3f}".format(precision\_ada)) print("AdaBoost : Recall on test Data: {:.3f}".format(recall\_ada)) print("AdaBoost : F-Measure on test Data: {:.3f}".format(f1\_ada)) print("AdaBoost : Accuracy on test Data: {:.3f}".format(accuracy\_ada))

fpr7, tpr7, thresholds = roc\_curve(y\_test, y\_pred\_xgb) roc\_auc7 = auc(fpr7, tpr7)

22

**CHAPTER 4**

**SNAPSHOTS**

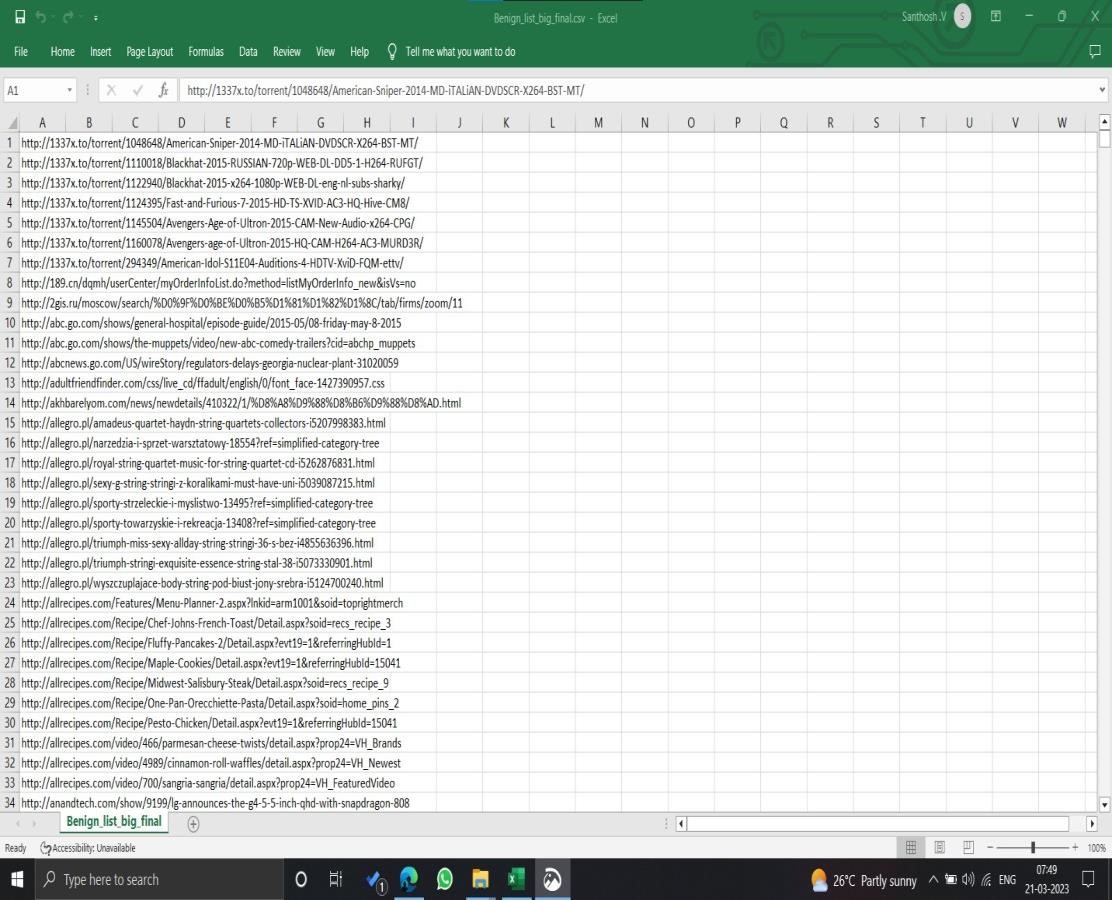


Fig 4.1 Legitimate Dataset

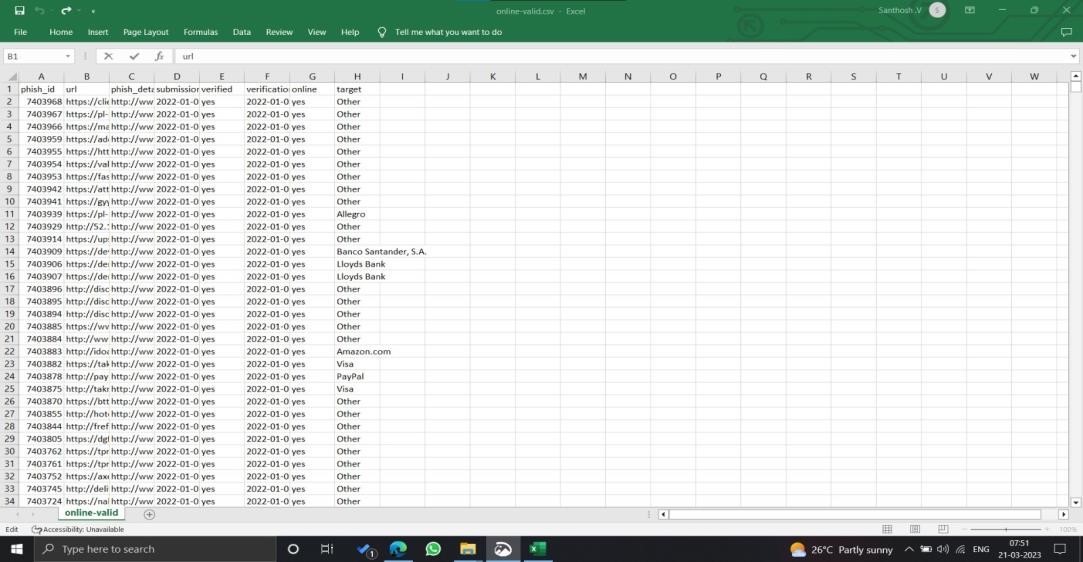


Fig 4.2 Online Valid Dataset

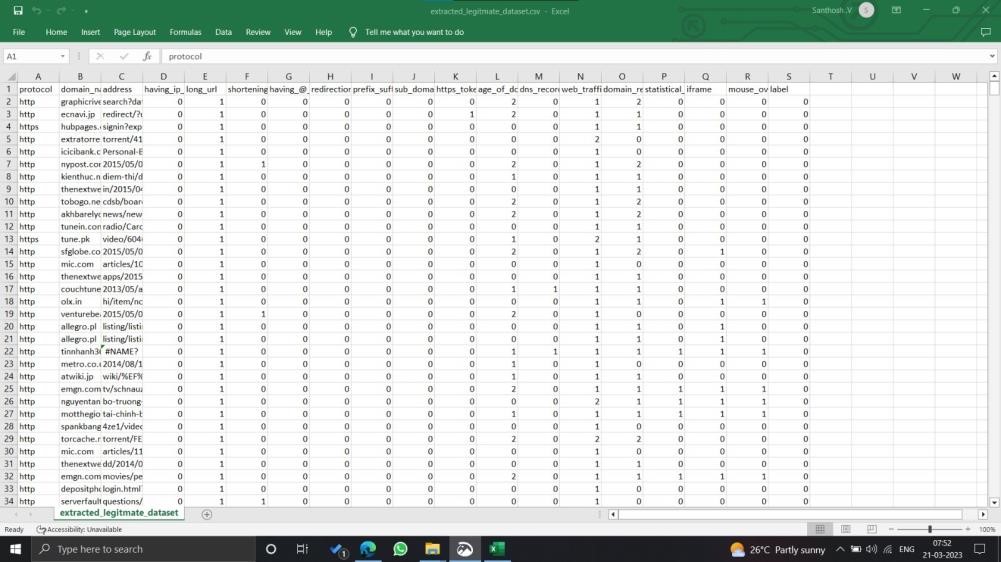


Fig 4.3 Legitimate Dataset After Feature Selection

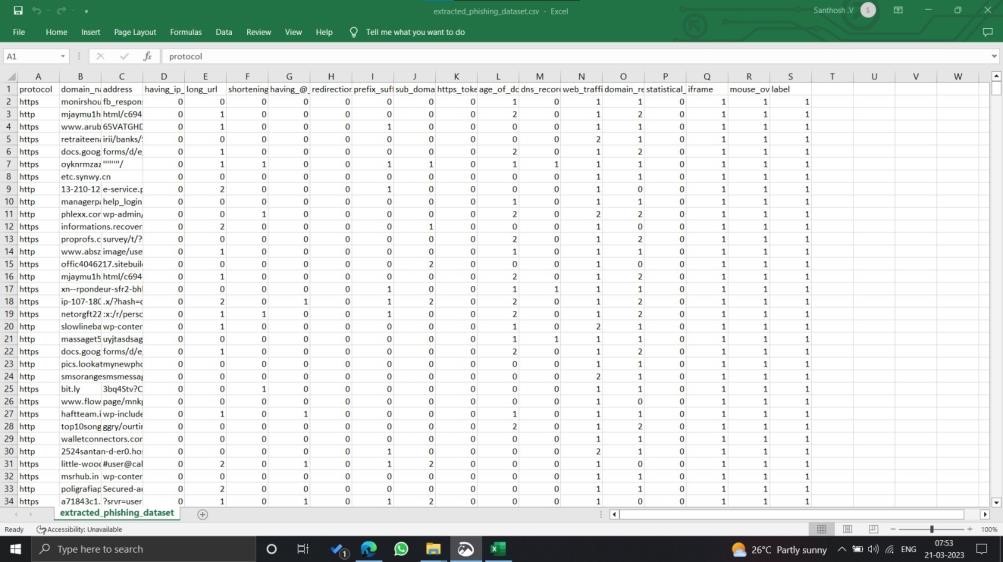


Fig 4.4 Online valid Dataset After Feature Selection

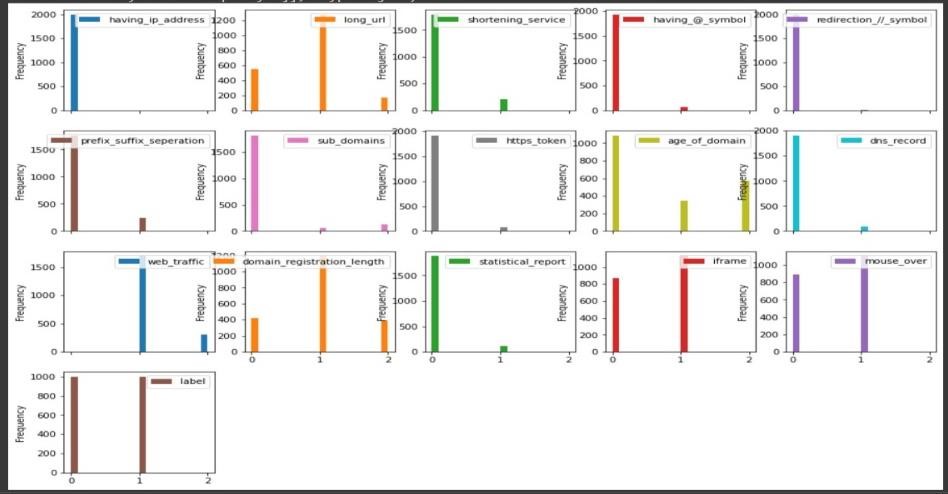


Fig 4.5 Feature Distribution

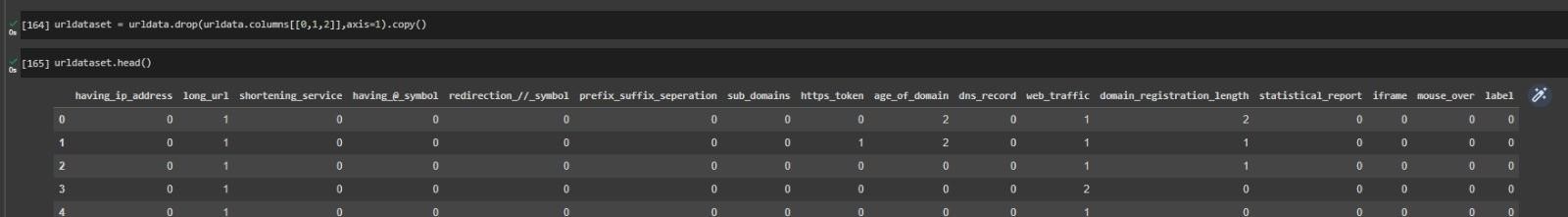


Fig 4.6 Drop the column(Data preprocessing)

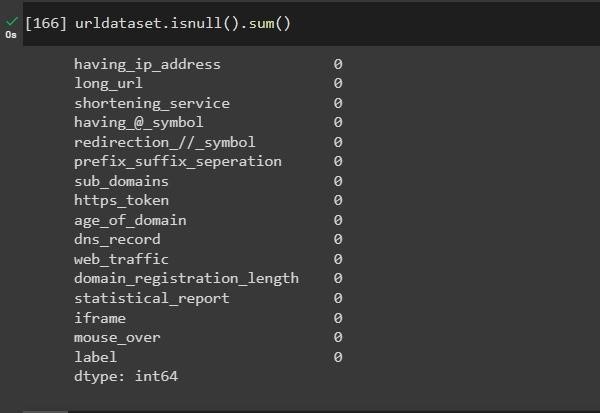


Fig 4.7 Checking data for null or missing values(Data preprocessing)



Fig 4.8 Shuffling the rows in the dataset so that when splitting the train and test set are equally distributed and evades the case of overfitting while model training (Data preprocessing)

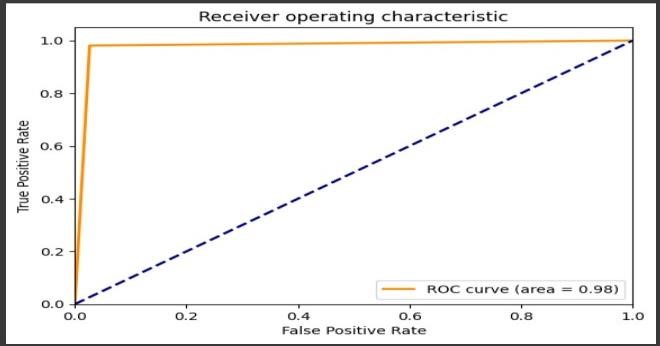
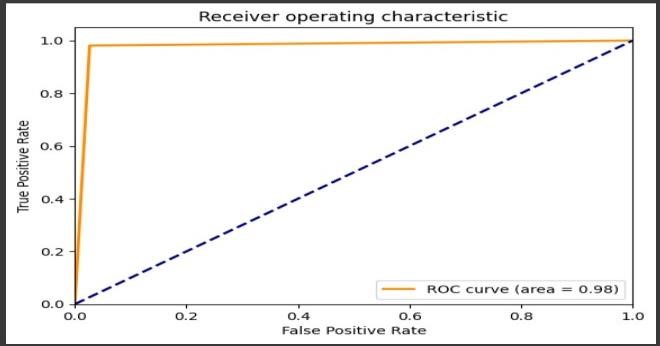


Fig 4.9 ROC Curve for SVM Fig 4.10 ROC Curve for KNN

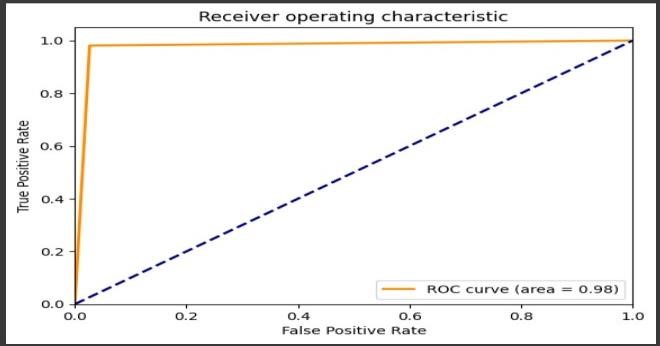
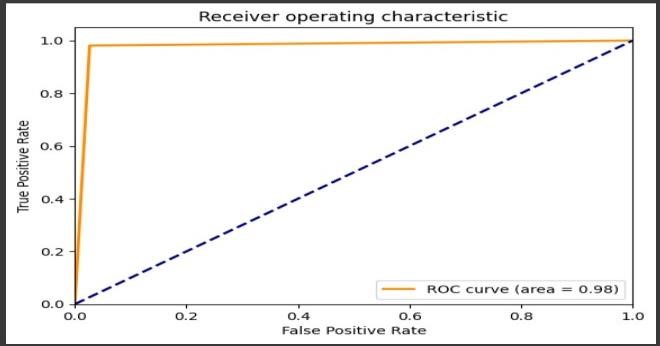


Fig 4.11 ROC Curve for Decision tree Fig 4.12 ROC Curve for Adaboost

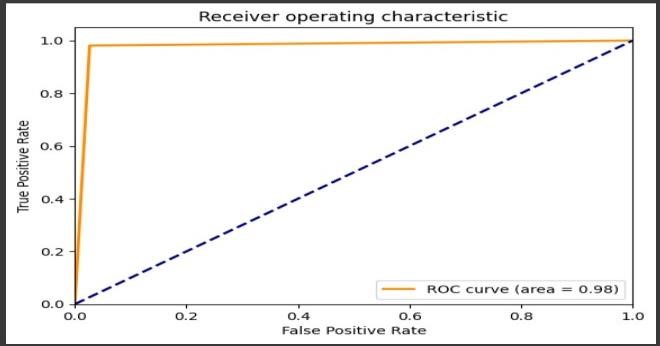
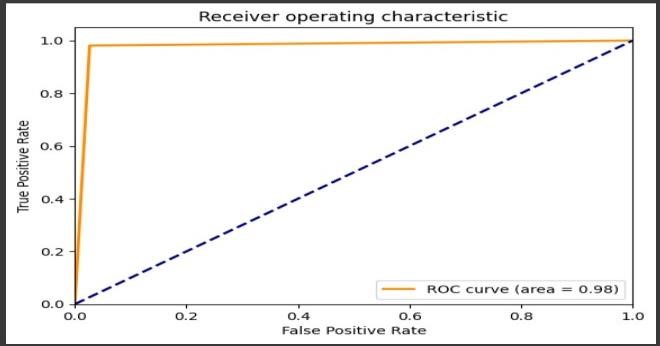


Fig 4.13 ROC Curve for Decision tree Fig 4.14 ROC Curve for LR

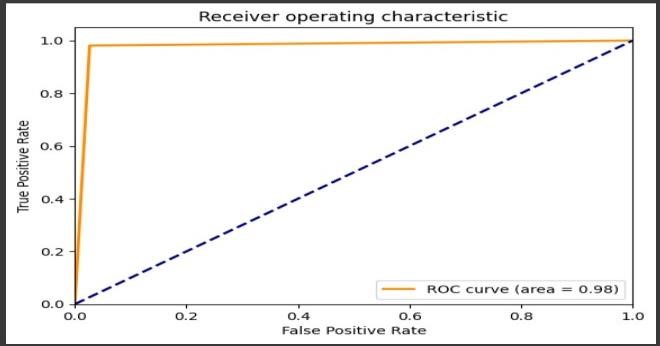


Fig 4.15 ROC Curve for Random Forest

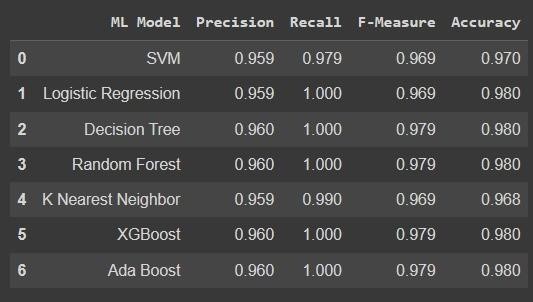


Fig 4.16 Model Evaluation

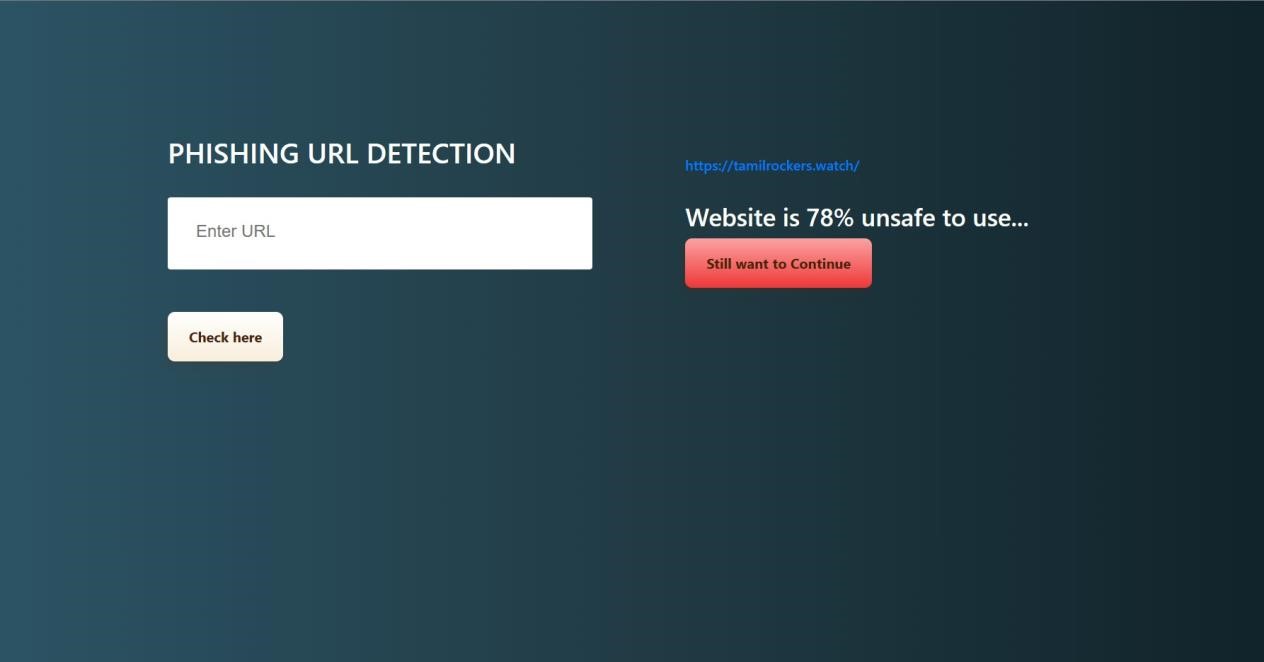


Fig 4.17 WebApplication

27

## CHAPTER 5 CONCLUSION AND FUTURE PLAN

### 5.1 CONCLUSION

New tech using smarts (machine learning) is getting good at spotting fake website addresses (phishing URLs). This is important because it helps protect people's information from being stolen in these online scams. However, this tech isn't a magic bullet. To be truly secure, it needs to be combined with other safety measures, like teaching people how to identify these scams themselves. Since this tech is constantly improving, it's important to keep it up-to-date to stay protected from new threats.

### 5.1 FUTURE WORKS

* Delve deeper to uncover even more revealing fingerprints that is the features of phishing URLs. This could involve exploring aspects like hidden code within the URL, past history of the domain linked to the URL, and even the way the URL is constructed grammatically. Additionally, leveraging social media trends and real-time threat intelligence to identify emerging phishing tactics can further enrich the feature set.

* Developing models that can not only handle a wider range of languages but also adapt to the unique characteristics of different character sets. By incorporating cultural nuances and recognizing phishing attempts that use symbols or lettering specific to certain languages, these models can provide a more comprehensive defense.

28

## CHAPTER 6 REFERENCES

1. Domain registered report available at: https://dofo.com/blog/domain-industryreport-april-2020/ Last accessed on May 11, 2020.
2. Amrita Dahiya, Brij B. Gupta, A reputation score policy and Bayesian game theory based incentivized mechanism for DDoS attacks mitigation and cyber defense, Future Gener. Comput. Syst. 117 (2021) 193–204.
3. A. Al-Nawasrah, A.A. Almomani, S. Atawneh, M. Alauthman, A survey of fast flux botnet detection with fast flux cloud computing, Int. J. Cloud Appl. Comput. (IJCAC) 10 (3) (2020) 17–53.
4. C. Esposito, M. Ficco, et al., Blockchain-based authentication and authorization for smart city applications, Inf. Process. Manage. 58 (2) (2021) 102468. S. Kaushik, C. Gandhi, Ensure hierarchal identity based data security in cloud environment, Int. J. Cloud Appl. Comput. (IJCAC) 9 (4) (2019) 21– 36.

29