

# Phishing Website Detection

Mini Project Report  
Bachelor in Computer Science

**Submitted By**

M.Jahnavi : S190310

Pisini Bhavani : S190308

Jammu Sunitha : S190269

Shaik Meharuneesha : S190489

**Under the Esteemed Guidance of**

Miss J.Vishnu Priyanka

CSE Department



Rajiv Gandhi University Of Knowledge And Technologies

S.M. Puram , Srikakulam -532410

Andhra Pradesh, India

# Abstract

Phishing is an internet scam in which an attacker sends out fake messages that look to come from a trusted source. A URL or file will be included in the mail, which when clicked will steal personal information or infect a computer with a virus. Traditionally, phishing attempts were carried out through wide-scale spam campaigns that targeted broad groups of people indiscriminately. The goal was to get as many people to click on a link or open an infected file as possible. There are various approaches to detect this type of attack. One of the approaches is machine learning. The URL's received by the user will be given input to the machine learning model then the algorithm will process the input and display the output whether it is phishing or legitimate. There are various ML algorithms like SVM, Neural Networks, Random Forest, Decision Tree, XG boost etc. that can be used to classify these URLs. The proposed approach deals with the Random Forest, Decision Tree classifiers. The proposed approach effectively classified the Phishing and Legitimate URLs with an accuracy of 87.0% and 82.4% for Random Forest and decision tree classifiers respectively.

# Contents

<b>Abstract</b>	<b>1</b>
<b>1 Introduction</b>	<b>4</b>
1.1 Introduction to Your Project . . . . .	4
1.2 Applications . . . . .	4
1.3 Motivation Towards our Project . . . . .	5
1.3.1 Increasing Prevalence of Phishing Attacks . . . . .	5
1.3.2 Protecting User Data: . . . . .	5
1.4 Problem Statement . . . . .	6
1.4.1 What is Phishing? . . . . .	6
1.4.2 What is legitimate? . . . . .	6
1.4.3 What is Detection? . . . . .	6
<b>2 Approach To Your Project</b>	<b>7</b>
2.1 Explain About Your Project . . . . .	7
2.2 Data Set . . . . .	7
2.3 Prediction technique . . . . .	15
2.4 Visualizations - Graphs . . . . .	17
<b>3 Code</b>	<b>20</b>
3.1 Installing required libraries . . . . .	20
3.2 Importing required modules . . . . .	20
3.3 Reading Dataset . . . . .	21
3.4 Data Analysis . . . . .	23
3.4.1 Shape of DataFrame . . . . .	24

3.4.2	Information about Dataframe . . . . .	24
3.4.3	Description of DataFrame . . . . .	26
3.4.4	Finding maximum value . . . . .	26
3.5	Data Cleaning . . . . .	27
3.5.1	Checking and Finding Null values . . . . .	27
3.5.2	Removing Null values . . . . .	27
3.5.3	Shuffling the rows . . . . .	28
3.6	Machine Learning Model Selection . . . . .	29
3.6.1	Importing required modules . . . . .	29
3.6.2	Cleaned DataSet . . . . .	30
3.6.3	Splitting Data into Train Test Data . . . . .	30
3.6.4	XGBOOST Model . . . . .	31
3.7	USER INTERFACE . . . . .	34
3.7.1	Legitimate Website . . . . .	34
3.7.2	Phishing Website . . . . .	34
<b>4</b>	<b>Conclusion and Future Work</b>	<b>35</b>
4.1	Conclusion . . . . .	35

# Chapter 1

## Introduction

### 1.1 Introduction to Your Project

Phishing has become the most serious problem, harming individuals, corporations, and even entire countries. The availability of multiple services such as online banking, entertainment, education, software downloading, and social networking has accelerated the Web's evolution in recent years. As a result, a massive amount of data is constantly downloaded and transferred to the Internet. Spoofed emails pretending to be from reputable businesses and agencies are used in social engineering techniques to direct consumers to fake websites that deceive users into giving financial information such as usernames and passwords. Technical tricks involve the installation of malicious software on computers to steal credentials directly, with systems frequently used to intercept users' online account usernames and passwords.

### 1.2 Applications

Web Browsers, Email Clients, Enterprise Security Systems, Financial Institutions, E-commerce Platforms, Social Media Platforms

1. Web Browsers

2. Email Clients

3. Enterprise Security Systems

4. E-commerce Platforms

5.Social Media Platforms

6.Mobile Applications

7.Content Management Systems (CMS) and Web Hosting Services

## **1.3 Motivation Towards our Project**

The motivation behind a phishing website detection project is driven by several critical factors aimed at enhancing cybersecurity and protecting users from online threats. Here are some key motivational aspects:

### **1.3.1 Increasing Prevalence of Phishing Attacks**

#### **Rising Threat:**

Phishing attacks are becoming more sophisticated and prevalent, posing significant risks to individuals and organizations.

#### **High Impact:**

Successful phishing attacks can lead to severe financial losses, identity theft, and unauthorized access to sensitive information.

### **1.3.2 Protecting User Data:**

#### **Personal Information Security:**

Safeguarding personal information, such as passwords, credit card details, and social security numbers, from being compromised.

#### **Trust and Confidence:**

Enhancing user trust in online interactions by providing a secure browsing experience.

## 1.4 Problem Statement

Detection the websites whether they are phishing or legitimate.

### 1.4.1 What is Phishing?

Phishing is a cyber attack where attackers impersonate legitimate entities to deceive victims. They often use emails, messages, or fake websites to trick individuals. The goal is to obtain sensitive information such as passwords, credit card numbers, or personal details. Phishing exploits trust and fear, often presenting urgent or enticing scenarios

### 1.4.2 What is legitimate?

”Legitimate” refers to something that is lawful, valid, or recognized as correct according to established rules or standards. It implies authenticity and adherence to legal or accepted principles. In a broader sense, it can describe actions, claims, or entities that are justifiable and conform to societal norms. For example, a legitimate business operates in compliance with legal regulations. Similarly, a legitimate argument is one that is well-founded and logically sound.

### 1.4.3 What is Detection?

Detection is the process of identifying the presence or occurrence of objects, events, or patterns within a dataset or environment. It involves analyzing data to recognize specific elements or signals that indicate the existence of something of interest. Detection can be applied in various fields, such as detecting anomalies in network traffic, identifying objects in images, or recognizing speech. Effective detection systems often use machine learning algorithms to improve accuracy and efficiency. It is a critical component in applications like surveillance, diagnostics, and automated decision-making systems.

# Chapter 2

## Approach To Your Project

### 2.1 Explain About Your Project

Our project aim is to detect phishing websites using Machine Learning Techniques.

### 2.2 Data Set

We have the columns to detect phishing websites.

Here, We are explain each column which are present in dataset.....

- **URL Length** - The length of a URL can be an important feature in phishing detection. Phishing URLs often have excessive lengths to include various parameters and obfuscate the true destination of the link. By analyzing the length of a URL, the model can identify unusually long URLs that are more likely to be phishing attempts.

*# 4.Finding the length of URL and categorizing (URL\_Length)*

```
def getLength(url):  
    if len(url) < 54:  
        length = 0  
    else:  
        length = 1
```



```
return length
```

- **IP Address** - Checks for the presence of IP address in the URL. URLs may have IP address instead of domain name. If an IP address is used as an alternative of the domain name in the URL, we can be sure that someone is trying to steal personal information with this URL.

If the domain part of URL has IP address, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

```
# Checks for IP address in URL (Have_IP)
```

```
def havingIP(url):  
    try:  
        ipaddress.ip_address(url)  
        ip = 1  
    except:  
        ip = 0  
    return ip
```

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

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

```
# Checks the presence of @ in URL (Have_At)
```

```
def haveAtSign(url):  
    if "@" in url:  
        at = 1
```

```

else:
    at = 0
return at

```

- **Redirection “//” in URL** - Checks the presence of “//” in the URL. The existence of “//” within the URL path means that the user will be redirected to another website. The location of the “//” in URL is computed. We find that if the URL starts with “HTTP”, that means the “//” should appear in the sixth position. However, if the URL employs “HTTPS” then the “//” should appear in seventh position.

If the “//” is anywhere in the URL apart from after the protocol, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

*# Checking for redirection '//' in the url (Redirection)*

```

def redirection(url):
    pos = url.rfind('//')
    if pos > 6:
        if pos > 7:
            return 1
        else:
            return 0
    else:
        return 0

```

- **Prefix or Suffix “-” in Domain** - checking the presence of ‘-’ in the domain part of URL. The dash symbol is rarely used in legitimate URLs. Phishers tend to add prefixes or suffixes separated by (-) to the domain name so that users feel that they are dealing with a legitimate webpage.

If the URL has '-' symbol in the domain part of the URL, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

*#Checking for Prefix or Suffix Separated by (-) in the Domain*

*↪ (Prefix/Suffix)*

```
def prefixSuffix(url):  
    if '-' in urlparse(url).netloc:  
        return 1          # phishing  
    else:  
        return 0          # legitimate
```

- **URL Depth** -URL depth refers to the number of subdirectories within a URL path. Phishing URLs often have greater depth as they try to mimic legitimate websites while embedding their phishing content deeper within the structure. Tracking the depth can help in distinguishing between genuine and phishing URLs.

*# Gives number of '/' in URL (URL\_Depth)*

```
def getDepth(url):  
    s = urlparse(url).path.split('/')  
    depth = 0  
    for j in range(len(s)):  
        if len(s[j]) != 0:  
            depth = depth+1  
    return depth
```

- **HTTP Domain** - The presence of "http" instead of "https" in the URL can be a red flag, as secure websites typically use HTTPS. Phishing sites might use HTTP to avoid the security layers that HTTPS provides, making it easier to intercept data. This feature assesses the protocol used and flags non-secure domains.

*# Existence of \HTTPS" Token in the Domain Part of the URL*

*↪ (https\_Domain)*

```
def httpDomain(url):  
    domain = urlparse(url).netloc  
  
    if 'https' in domain:  
        return 1  
    else:  
        return 0
```

- **Tiny URL** -Tiny URLs are shortened links, often used to hide the true destination of a URL. While legitimate uses exist, they can also be exploited by phishers to obscure malicious links. Analyzing whether a URL is shortened can help detect phishing attempts masked by tiny URLs.

*# Checking for Shortening Services in URL (Tiny\_URL)*

```
def tinyURL(url):  
    match=re.search(shortening_services,url)  
  
    if match:  
        return 1  
    else:  
        return 0
```

- **DNS Record**-A phishing site may not have a valid DNS record or may have anomalies in its DNS setup. This feature involves checking the DNS records for validity and consistency, helping to identify sites that do not meet the typical standards of legitimate websites.

*# DNS Record availability (DNS\_Record)*

*# obtained in the featureExtraction function itself*

- **Domain Age** -The age of a domain can be a significant indicator of its legitimacy. Phishing domains are often recently created and have a short lifespan. By analyzing the domain age, the model can flag recently created domains that might be used for phishing.

```

    # Survival time of domain: The difference between termination
    ↪ time and creation time (Domain_Age)
def domainAge(domain_name):
    creation_date = domain_name.creation_date
    expiration_date = domain_name.expiration_date
    if (isinstance(creation_date,str) or
    ↪ isinstance(expiration_date,str)):
        try:
            creation_date = datetime.strptime(creation_date,'%Y-%m-%d')
            expiration_date = datetime.strptime(expiration_date,"%Y-%m-%d")
        except:
            return 1
    if ((expiration_date is None) or (creation_date is None)):
        return 1
    elif ((type(expiration_date) is list) or (type(creation_date) is
    ↪ list)):
        return 1
    else:
        ageofdomain = abs((expiration_date - creation_date).days)
        if ((ageofdomain/30) < 6):
            age = 1
        else:
            age = 0
    return age

```

- **Domain Expiry** -Similarly, the expiry date of a domain can be an indicator of phishing. Phishing sites often have short registration periods. Analyzing how soon a domain is set to expire can help in identifying potentially malicious domains that are not intended for long-term use.

```

# End time of domain: The difference between termination time
↪ and current time (Domain_End)
def domainEnd(domain_name):
    expiration_date = domain_name.expiration_date
    if isinstance(expiration_date, str):
        try:
            expiration_date = datetime.strptime(expiration_date, "%Y-%m-%d")
        except:
            return 1
    if (expiration_date is None):
        return 1
    elif (type(expiration_date) is list):
        return 1
    else:
        today = datetime.now()
        end = abs((expiration_date - today).days)
        if ((end/30) < 6):
            end = 0
        else:
            end = 1
    return end

```

- **iFrame Usage** -iFrames can be used maliciously to embed a phishing page within a legitimate-looking site. By detecting the presence and use of iFrames, especially

those that obscure content or capture user input, the model can identify potentially harmful websites.

```
# IFrame Redirection (iFrame)

def iframe(response):
    if response == "":
        return 1
    else:
        if re.findall(r"<iframe>|<frameBorder>", response.text):
            return 0
        else:
            return 1
```

- **Forwarding Count** -Phishing websites might use multiple URL redirections to confuse users and evade detection. By counting the number of times a URL is forwarded before reaching its final destination, the model can identify suspicious behavior typical of phishing attacks.

```
# Checks the number of forwardings (Web_Forwards)

def forwarding(response):
    if response == "":
        return 1
    else:
        if len(response.history) <= 2:
            return 0
        else:
            return 1
```

- **Label** -The label is the target variable in the dataset, indicating whether a website is phishing (often labeled as "1") or legitimate (labeled as "0"). This binary classi-

fication is what the machine learning model aims to predict, based on the features described above. The accuracy of the model in predicting this label is a measure of its effectiveness in detecting phishing websites.

## 2.3 Prediction technique

We are using already predefined machine learning techniques like Random Forest,pytorch and some other sklearn libraries based on the accuracy measures we pick up the best one.Actually XG Boost classifier works well on our data.

- What is XG Boost classifier?

The XGBoost classifier is a high-performance implementation of the gradient boosting framework, designed to optimize both speed and accuracy. It builds an ensemble of decision trees in a sequential manner, where each new tree corrects the errors made by the previous ones. Key features of XGBoost include regularization to prevent overfitting, support for handling missing values, and the ability to work with sparse data efficiently. Additionally, it leverages parallel and distributed computing for faster training. XGBoost is widely used in machine learning competitions and practical applications for its superior predictive performance and scalability.

XGBoost is one of the most popular machine learning algorithms these days. XGBoost stands for eXtreme Gradient Boosting. Regardless of the type of prediction task at hand; regression or classification. XGBoost is an implementation of gradient boosted decision trees designed for speed and performance.



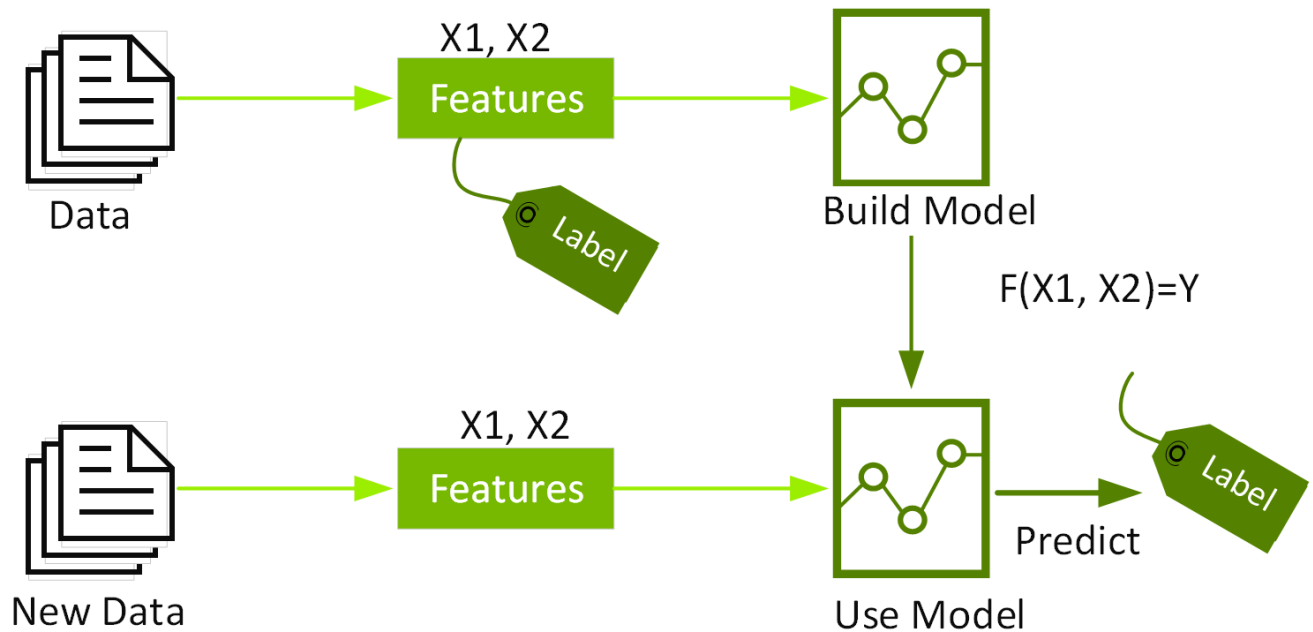


Figure 2.1

- What is Random Forest classifier?

A Random Forest is an ensemble learning method used for classification, regression, and other tasks that operates by constructing a multitude of decision trees during training. For classification tasks, the output of the Random Forest is the mode of the classes (the majority vote) from all the individual trees, while for regression, it is the mean prediction of the individual trees. This method improves predictive accuracy and controls overfitting by averaging multiple decision trees, each built on different subsets of the training data. Random Forests are robust, handle large datasets with higher dimensionality well, and can manage missing values and maintain accuracy when a significant portion of the data is missing.

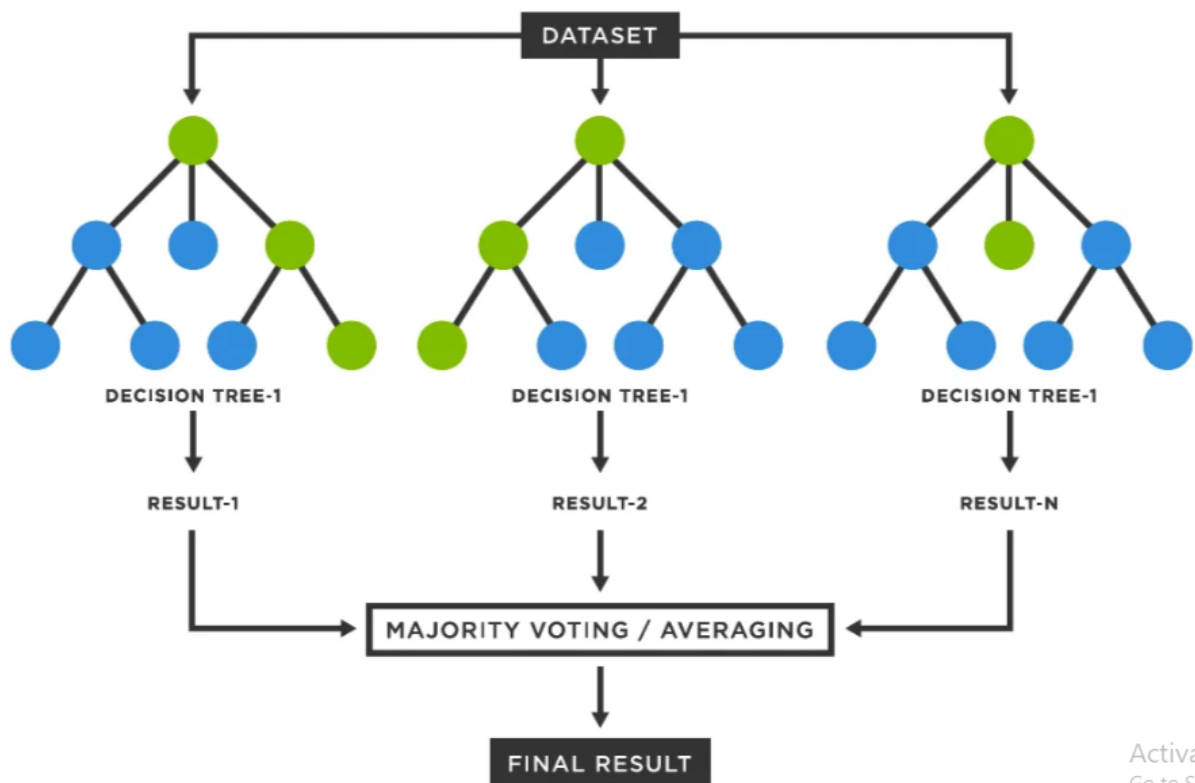
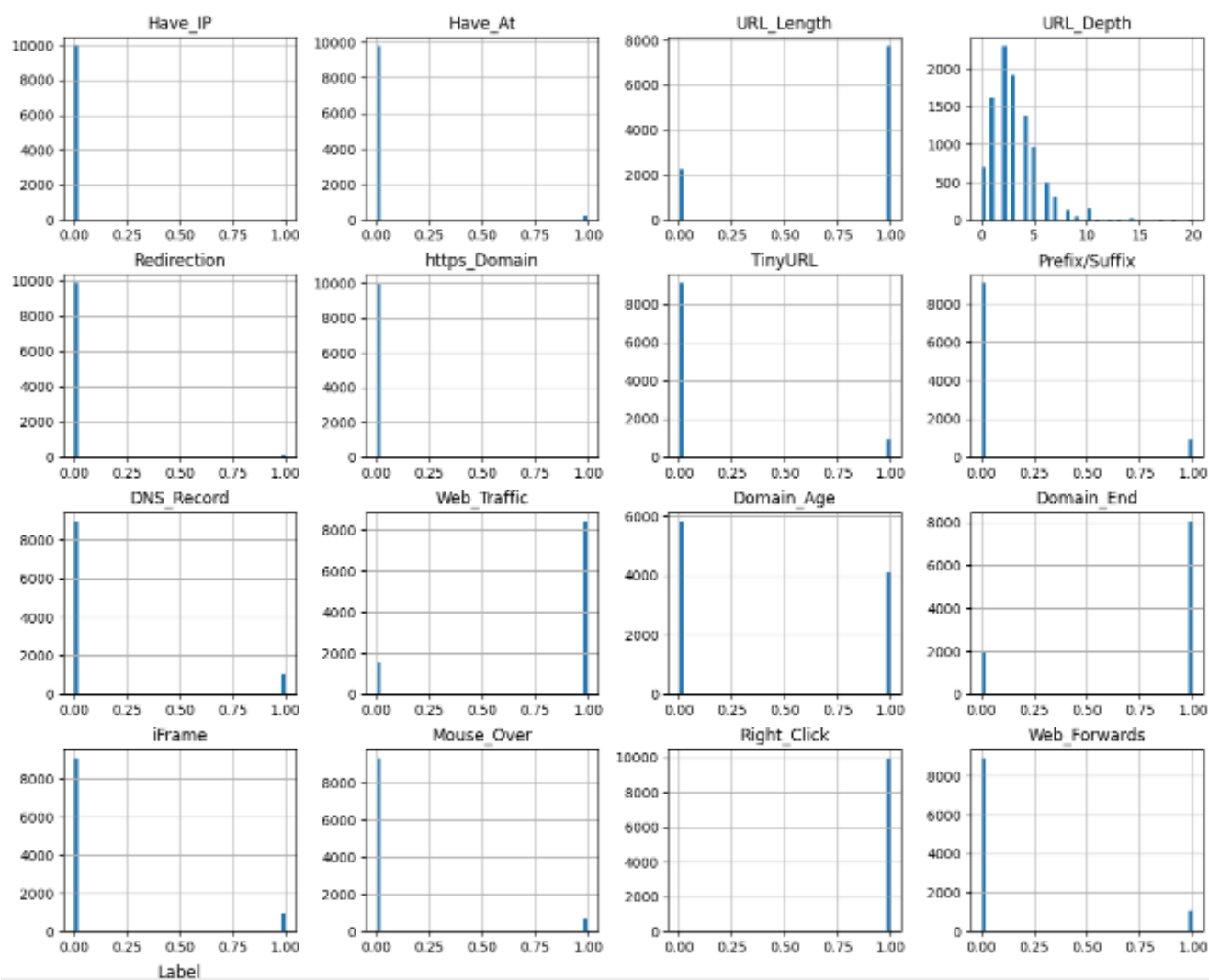


Figure 2.2: Random Forest Classification

## 2.4 Visualizations - Graphs

let us plot some graphs to find how the data is distributed and how the features are related to each other.

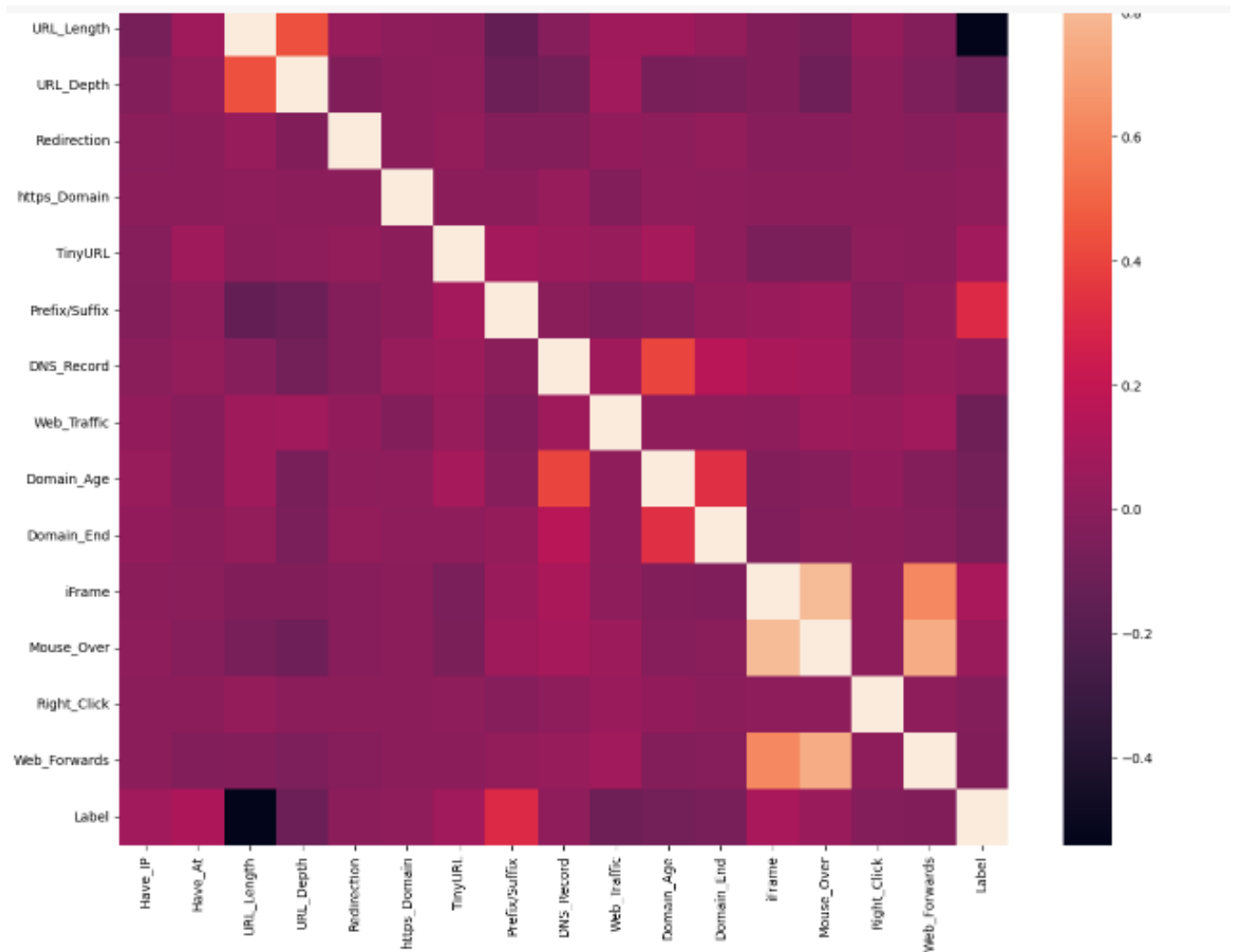
```
#Plotting the data distribution  
data0.hist(bins = 50,figsize = (15,15))  
plt.show()
```



Let us plot Correlation heat Map graph.

```
#Correlation heatmap
```

```
plt.figure(figsize=(15,13))  
sns.heatmap(data0.corr())  
plt.show()
```



# Chapter 3

## Code

### 3.1 Installing required libraries

we are installing transformers ,PyTorch and some additional libraries for text preprocessing.

```
pip install -q transformers
pip install PyTorch
pip install accelerate==0.20.1
#Installing additional libraries for text preprocessing
pip install -q preprocessor
pip install -q contractions
```

### 3.2 Importing required modules

we are importing pandas for data manipulating numpy for calculations , pyplot for graphs, seaborn for effective plots.

```
import pandas as pd
import numpy as np
import re
import seaborn as sns
```

```
import matplotlib.pyplot as plt

# Text processing libraries

import preprocessor

import contractions
```

### 3.3 Reading Dataset

for reading dataset we used readcsv() function.

```
#Loading the data

data0 = pd.read_csv("5.urldata.csv")

print(data0)
```

	Domain	Have_IP	Have_At	URL_Length	URL_Depth	\
0	graphicriver.net		0	0	1	1
1	ecnavi.jp		0	0	1	1
2	hubpages.com		0	0	1	1
3	extratorrent.cc		0	0	1	3
4	icicibank.com		0	0	1	3
...	...		...	...	...	...
9995	wvk12-my.sharepoint.com		0	0	1	5
9996	adplife.com		0	0	1	4
9997	kurortnoye.com.ua		0	1	1	3
9998	norcaltc-my.sharepoint.com		0	0	1	5
9999	sieck-kuehlssysteme.de		0	1	1	4

	Redirection	https_Domain	TinyURL	Prefix/Suffix	DNS_Record	\
0	0	0	0	0	0	
1	1	0	0	0	0	
2	0	0	0	0	0	

3	0	0	0	0	0
4	0	0	0	0	0
...	...	...	...	...	...
9995	0	0	1	1	0
9996	0	0	0	0	0
9997	0	0	1	0	0
9998	0	0	1	1	0
9999	0	0	1	1	0

	Web_Traffic	Domain_Age	Domain_End	iFrame	Mouse_Over	Right_Click
↪ \						
0	1	1	1	0	0	1
1	1	1	1	0	0	1
2	1	0	1	0	0	1
3	1	0	1	0	0	1
4	1	0	1	0	0	1
...	...	...	...	...	...	...
9995	1	1	1	0	0	1
9996	1	0	1	0	0	1
9997	0	1	1	1	0	1
9998	1	1	1	0	0	1
9999	1	1	1	0	0	1

	Web_Forwards	Label
0	0	0
1	0	0
2	0	0

```

3          0      0
4          0      0
...
9995       0      1
9996       0      1
9997       0      1
9998       0      1
9999       0      1

```

[10000 rows x 18 columns]


## 3.4 Data Analysis

Now I want to see some rows and columns

To see the first 5 rows, we use `df.head()` function.

*#code*

```
data0.head()
```

1 to 5 of 5 entries  

index	Domain	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	TinyURL	Prefix/Suffix	DNS_Record	Web_Traffic	Domain_Age	Domain_End	iFrame
0	graphicriver.net	0	0	1	1	0	0	0	0	0	1	1	1	0
1	ecnavi.jp	0	0	1	1	1	0	0	0	0	1	1	1	0
2	hubpages.com	0	0	1	1	0	0	0	0	0	1	0	1	0
3	extratorrent.cc	0	0	1	3	0	0	0	0	0	1	0	1	0
4	icicibank.com	0	0	1	3	0	0	0	0	0	1	0	1	0

To see the last 5 rows, we use `df.tail()` function.

*#code*

```
data0.tail()
```

*All columns are not visible when we use head() and tail() functions. So, by using data.columns function we can see all columns*



	Domain	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	TinyURL	Prefix/Suffix	DNS_Record	Web_Traffic	Domain_Age	Domain_End	iFrame	Mouse_Over	Right_Click
9995	wvkt2-my.sharepoint.com	0	0	1	5	0	0	1	1	0	1	1	1	0	0	1
9996	adplife.com	0	0	1	4	0	0	0	0	0	1	0	1	0	0	1
9997	kurortnoye.com.ua	0	1	1	3	0	0	1	0	0	0	1	1	1	0	1
9998	norcallt-my.sharepoint.com	0	0	1	5	0	0	1	1	0	1	1	1	0	0	1
9999	sieck-kuehlssysteme.de	0	1	1	4	0	0	1	1	0	1	1	1	0	0	1

*#code*

```
data0.columns
```

```
Index(['Domain', 'Have_IP', 'Have_At', 'URL_Length', 'URL_Depth',
      'Redirection', 'https_Domain', 'TinyURL', 'Prefix/Suffix', 'DNS_Record',
      'Web_Traffic', 'Domain_Age', 'Domain_End', 'iFrame', 'Mouse_Over',
      'Right_Click', 'Web_Forwards', 'Label'],
      dtype='object')
```

### 3.4.1 Shape of DataFrame

*#code*

```
data0.shape
```

*#output (10000, 18)*

Our DataFrame contains 10000 rows and 18 columns

### 3.4.2 Information about Dataframe

Let us explain some information about our dataset using info() function

*#code*

```
data0.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

RangeIndex: 10000 entries, 0 to 9999

Data columns (total 18 columns):

#	Column	Non-Null Count	Dtype
---	-----	-----	-----
0	Domain	10000 non-null	object
1	Have_IP	10000 non-null	int64
2	Have_At	10000 non-null	int64
3	URL_Length	10000 non-null	int64
4	URL_Depth	10000 non-null	int64
5	Redirection	10000 non-null	int64
6	https_Domain	10000 non-null	int64
7	TinyURL	10000 non-null	int64
8	Prefix/Suffix	10000 non-null	int64
9	DNS_Record	10000 non-null	int64
10	Web_Traffic	10000 non-null	int64
11	Domain_Age	10000 non-null	int64
12	Domain_End	10000 non-null	int64
13	iFrame	10000 non-null	int64
14	Mouse_Over	10000 non-null	int64
15	Right_Click	10000 non-null	int64
16	Web_Forwards	10000 non-null	int64
17	Label	10000 non-null	int64

dtypes: int64(17), object(1)

memory usage: 1.4+ MB

Here, we clearly observed that..

- The column Domain object based datatype.

- Remaining all columns are belongs integer datatype.

### 3.4.3 Description of DataFrame

The describe() method returns description of the data in the DataFrame. If the dataframe contains numerical data, the description contains these information for each column.

```
#code
```

```
data0.describe()
```

	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	TinyURL	Prefix/Suffix	DNS_Record	Web_Traffic	Domain_Age	Domain_End	iFrame	Mouse_Over	Right_Click	Web_Forwards
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.0000	10000.000000	10000.00000	10000.00000	10000.000000
mean	0.005500	0.022600	0.773400	3.072000	0.013500	0.000200	0.090300	0.093200	0.100800	0.845700	0.413700	0.8099	0.090900	0.06660	0.99930	0.105300
std	0.073961	0.148632	0.418653	2.128631	0.115408	0.014141	0.286625	0.290727	0.301079	0.361254	0.492521	0.3924	0.287481	0.24934	0.02645	0.306955
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000	0.000000	0.00000	0.00000	0.000000
25%	0.000000	0.000000	1.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	1.0000	0.000000	0.00000	1.00000	0.000000
50%	0.000000	0.000000	1.000000	3.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	1.0000	0.000000	0.00000	1.00000	0.000000
75%	0.000000	0.000000	1.000000	4.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	1.000000	1.0000	0.000000	0.00000	1.00000	0.000000
max	1.000000	1.000000	1.000000	20.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.0000	1.000000	1.00000	1.00000	1.000000

### 3.4.4 Finding maximum value

for finding maximum value we use max() function.

```
#code
```

```
data0.max()
```

Domain	zxe32szeifr3.000webhostapp.com
Have_IP	1
Have_At	1
URL_Length	1
URL_Depth	20
Redirection	1

https_Domain	1
TinyURL	1
Prefix/Suffix	1
DNS_Record	1
Web_Traffic	1
Domain_Age	1
Domain_End	1
iFrame	1
Mouse_Over	1
Right_Click	1
Web_Forwards	1
Label	1
dtype: object	

## 3.5 Data Cleaning

### 3.5.1 Checking and Finding Null values

for checking null values we use `isnull()` and `notnull()` methods.

```
#code

data0.isnull()

data0.notnull()
```

### 3.5.2 Removing Null values

Let's us calculate is any null values in our dataset using `isnull()` and `sum()` functions

```
#code

data0.isnull().sum()
```

Counting Null values for each columns:

Domain	0
Have_IP	0
Have_At	0
URL_Length	0
URL_Depth	0
Redirection	0
https_Domain	0
TinyURL	0
Prefix/Suffix	0
DNS_Record	0
Web_Traffic	0
Domain_Age	0
Domain_End	0
iFrame	0
Mouse_Over	0
Right_Click	0
Web_Forwards	0
Label	0

dtype: int64

### 3.5.3 Shuffling the rows

shuffling the rows in the dataset so that when splitting the train and test set are equally distributed

*#code*

```
data0 = data.sample(frac=1).reset_index(drop=True)
data0.head()
```

	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	TinyURL	Prefix/Suffix	DNS_Record	Web_Traffic	Domain_Age	Domain_End	iFrame	Mouse_Over	Right_Click
0	0	0	1	3	0	0	0	0	0	1	0	1	0	0	1
1	0	0	0	3	0	0	0	0	0	1	0	0	1	1	1
2	0	0	1	2	0	0	0	0	0	0	1	1	0	0	1
3	0	0	1	7	0	0	1	1	0	1	1	1	0	0	1
4	0	0	1	2	0	0	0	0	0	1	0	0	0	0	1

From the above execution, it is clear that the data doesnot have any missing values. By this, the data is throughly preprocessed is ready for training.

## 3.6 Machine Learning Model Selection

Here, we used some machine learning models from scikit-learn

**Models used :**

- RandomForestClassifier
- multiplayerperceptrons
- XGBoost
- DecisionTreeClassifier
- Autoencoder Neural Network
- Support Vector Machines

### 3.6.1 Importing required modules

First, let us import necessary libraries

```
#importing basic packages

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

### 3.6.2 Cleaned DataSet

```
data0.describe()

#Dropping the Domain column

data = data0.drop(['Domain'], axis = 1).copy()

#checking the data for null or missing values

data.isnull().sum()
```

### 3.6.3 Splitting Data into Train Test Data

we splitted the data columns into x and y i.e Dependent(Target Variable) and Independent(Predictor Variables)

Let us divide our data set into training set and training set.

```
#code

# Sepratating & assigning features and target columns to X & y

y = data['Label']

X = data.drop('Label',axis=1)

X.shape, y.shape

#output

((10000, 16), (10000,))

#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

#OUTPUT

((8000, 16), (2000, 16))
```

```
# Defining the number of samples in train, validation and test dataset

size_train = df_train.shape[0]
size_test = df_test.shape[0]
```

### 3.6.4 XGBOOST Model

```
# Creating holders to store the model performance results

ML_Model = []
acc_train = []
acc_test = []

#function to call for storing the results
def storeResults(model, a,b):
    ML_Model.append(model)
    acc_train.append(round(a, 3))
    acc_test.append(round(b, 3))

#XGBoost Classification model
from xgboost import XGBClassifier

# instantiate the model
xgb = XGBClassifier(learning_rate=0.4,max_depth=7)

#fit the model
xgb.fit(X_train, y_train)

#Output
XGBClassifier(base_score=None, booster=None, callbacks=None,
```



```

colsample_bylevel=None, colsample_bynode=None,
colsample_bytree=None, early_stopping_rounds=None,
enable_categorical=False, eval_metric=None,
↪ feature_types=None,
gamma=None, gpu_id=None, grow_policy=None,
↪ importance_type=None,
interaction_constraints=None, learning_rate=0.4,
↪ max_bin=None,
max_cat_threshold=None, max_cat_to_onehot=None,
max_delta_step=None, max_depth=7, max_leaves=None,
min_child_weight=None, missing=nan,
↪ monotone_constraints=None,
n_estimators=100, n_jobs=None, num_parallel_tree=None,
predictor=None, random_state=None, ...)

#predicting the target value from the model for the samples
y_test_xgb = xgb.predict(X_test)
y_train_xgb = xgb.predict(X_train)

#code

#computing the accuracy of the model performance
acc_train_xgb = accuracy_score(y_train,y_train_xgb)
acc_test_xgb = accuracy_score(y_test,y_test_xgb)

print("XGBoost: Accuracy on training Data: {:.3f}".format(acc_train_xgb))
print("XGBoost : Accuracy on test Data: {:.3f}".format(acc_test_xgb))

#output
XGBoost: Accuracy on training Data: 0.867
XGBoost : Accuracy on test Data: 0.858

#storing results

```

*#storing the results. The below mentioned order of parameter passing is  
→ important.*

*#Caution: Execute only once to avoid duplications.*

```
storeResults('XGBoost', acc_train_xgb, acc_test_xgb)
```

*# Comparision of Models*

*#creating dataframe*

```
results = pd.DataFrame({ 'ML Model': ML_Model,  
    'Train Accuracy': acc_train,  
    'Test Accuracy': acc_test})
```

*#results*

	ML Model	Train Accuracy	Test Accuracy
4	XGBoost	0.867	0.858
2	Multilayer Perceptrons	0.865	0.858
3	Multilayer Perceptrons	0.865	0.858
1	Random Forest	0.819	0.824
0	Decision Tree	0.812	0.820
6	SVM	0.800	0.806
5	AutoEncoder	0.002	0.001

*#For the above comparision, it is clear that the XGBoost Classifier works  
→ well #with this dataset.*

So, saving the model for future use.save XGBoost model to file

*#code*

```
import pickle
```

```
pickle.dump(xgb, open("XGBoostClassifier.pickle.dat", "wb"))
```

## 3.7 USER INTERFACE

### 3.7.1 Legitimate Website

#### Phishing URL Detector

Enter a URL to check if it's phishing or not.

Enter URL:

<https://www.youtube.com/>

Check

Extracting features...

Predicting...

Predicting made:

No Phishing Detected. This URL seems safe.

### 3.7.2 Phishing Website

#### Phishing URL Detector

Enter a URL to check if it's phishing or not.

Enter URL:

<http://secure-update.com/login>

Check

Extracting features...

Predicting...

Prediction made:

Phishing Alert! This URL is classified as phishing.

# Chapter 4

## Conclusion and Future Work

### 4.1 Conclusion

Detecting phishing websites using machine learning is a powerful and efficient approach to bolstering cybersecurity. Machine learning models, trained on extensive datasets of phishing and legitimate sites, can accurately identify malicious websites by analyzing features such as URL structure and domain age. This automated detection method enhances the speed and reliability of identifying phishing attempts. Additionally, the adaptability of machine learning algorithms ensures that detection systems remain effective against evolving phishing tactics, providing a dynamic and proactive defense against cyber threats.