

# **CHAPTER 1**

## **INTRODUCTION**

A cyber-attack is any offensive tactic that targets computer information systems, computer networks, infrastructures, or personal computer devices. They are unwelcome attempts to steal, expose, alter, disable or destroy information through unauthorized access to computer systems. A cyber-attack can be launched from anywhere by any individual or group using one or more various attack strategies. Most common types of cyberattacks are: malware, man-in-the-middle, phishing, distributed denial of service (DDoS), SQL injection, & ransomware attack.

### **1.1 What is Phishing?**

Phishing is one of the most widely used, effective, and destructive cyber-attacks, in which attackers try to trick users into revealing sensitive personal information, such as their passwords and credit card information. It is usually carried out by creating a counterfeit website that mimics a legitimate website. Hackers can also socially engineer email messages to manipulate users to open them. By opening an attached file or clicking on an embedded link, the recipients are deceived into downloading the virus contained within the email.

It is difficult for the average person to differentiate between phishing websites and normal websites because phishing websites appear similar to the websites they imitate. In many cases, users do not check the entire website URL, and, once they visit a phishing website, the attacker can access sensitive and personal information and cause unauthorized purchases, stealing of funds, or identity theft.

Furthermore, phishing is frequently used as part of a bigger attack, such as an advanced persistent threat (APT) event, to build a foothold in business or governmental networks. Employees are compromised in this scenario in order to circumvent security perimeters, disseminate malware inside a closed environment, or get privileged access to protected data.

An organization that falls victim to such an attack usually suffers significant financial losses as well as a loss of market share, reputation, and consumer trust. Depending on the scale, a

phishing attempt could turn into a security disaster from which a company will have a difficult time recovering.

## **1.2 Types of Phishing Attacks:**

Phishing attacks employ a variety of techniques such as link manipulation, filter evasion, website forgery, covert redirect, and social engineering.

- **Link manipulation:**

The phishing is mainly about links. There are some clever ways to manipulate a URL to make look like a legitimate URL. One method is to represent the malicious URLs as hyperlinks with name on websites. Another method is to use misspelled URLs which will look like a legitimate one.

- **Filter evasion:**

Phishers show the content of their website in pictures or they use Adobe-Flash making it difficult to be detected by some phishing detection methods.

- **Website forgery:**

In this type of attack, Phishing is happening at a legitimate website by manipulating the target website JavaScript code. These types of attacks which are also known as cross-site scripting are very hard to detect because the victim is using the legitimate website.

- **Covert redirect:**

This attack targets websites using OAuth 2.0 and OpenID protocol. While trying to grant token access to a legitimate website, users are giving their token to a malicious service. However, this method did not gain much attention due to its low significance.

- **Social engineering:**

This type of phishing is carried out through social interaction. It uses psychological tricks to deceive users to give away security information. At first, the phisher investigates the potential weak points of targets required for the attack. Then, the phisher tries to gain the target's trust and at last, provide a situation in which the target reveals important information. There are some social engineering phishing methods, namely, baiting, scareware, pretexting, and spear phishing.

### **1.3 Anti-phishing Solutions**

There has been great deal of research in the area of phishing prevention and detection. Training users on phishing-related activities, the use of anti-phishing software, browser extensions and toolbars, two-factor authentication, filtering phishing emails and websites, secure browser advancements, and so on are all part of the recommended solution. Anti- phishing solutions are often divided into three categories: phishing prevention, user training, and phishing detection.

#### **1.3.1 Phishing Prevention Solutions:**

Phishing prevention systems work to protect authentication schemes and user interaction platforms from phishing attempts by adding an extra layer of security. This minimizes the chances of a user being duped by a phishing website created by an attacker. Watermarking, RFID, external authentication, picture passwords, QR codes, and other standard protection tactics are used. These methods may result in complex user interfaces, additional costs for each login, and the need for users to keep additional authentication devices, making them difficult to adopt and use.

#### **1.3.2 User Training Solutions:**

User training solutions try to educate users via emails and other forms of communication so that they can recognize phishing efforts directed at them. Although the idea is sound, it does not give a foolproof solution because a huge percentage of users are inexperienced and may not grasp how SSL, certificates, and website URLs can be verified for validity, even after training. As a result, relying solely on such ineffective solutions can be disastrous.

#### **1.3.3 Phishing Detection Solutions:**

Phishing detection solutions, which detect a phishing website via a web browser on the clientside or specialized software on the server side, are better than phishing prevention and user training solutions. This is due to their low reliance on inexperienced internet users. When a website is identified as a phishing or likely phishing website, the user's access is prevented or the user is warned that the website may not be legitimate. This solution needs little user training and does not necessitate any changes to a website's existing authentication mechanisms.

#### **1.4 A Machine Learning Approach for Phishing Detection**

Machine Learning is a way of teaching computers to learn and act like humans, and to enhance their learning over time in an autonomous manner, using data and information fed to them in the form of observations and real-world interactions. Machine learning provides simplified and efficient methods for data analysis. It has indicated promising outcomes in real-time classification problems recently. The key advantage of machine learning is the ability to create flexible models for specific tasks like phishing detection. Since phishing is a classification problem, machine learning models can be used as a powerful tool. Machine learning models could adapt to changes quickly to identify patterns of fraudulent transactions that help to develop a learning-based identification system. There are many machine learning algorithms like logistic regression, random forest, support vector machine etc., which can be used for phishing detection.

Using data and information provided to them in the form of observations and real-world interactions, machine learning is a technique for teaching computers to learn and act like people and to refine their learning over time in an independent manner. Data analysis can be done more quickly and effectively thanks to machine learning. Recently, it has shown promising results in real-time categorization issues. The main benefit of machine learning is its capacity to design adaptable models for certain tasks, such as phishing detection. Machine learning models can be an effective tool because phishing is a categorization issue. Machine learning models could swiftly adapt to new situations in order to spot fraudulent transaction patterns and aid in the creation of learning-based identification systems. Numerous machine learning methods, such as logistic

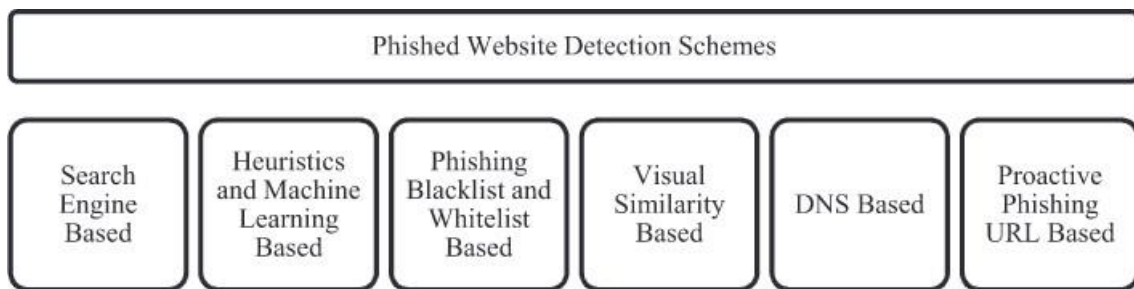
Better than phishing prevention and user education solutions are phishing detection solutions, which can identify a phishing website using a web browser on the client side or specialised software on the server side. Their limited reliance on unskilled internet users is the reason for this. The user's access to a website is restricted or the user is alerted that the website might not be reliable when it is detected as a phishing or likely phishing website. Little user education is required for this method, and the authentication techniques already used by a website can remain unchanged.

## CHAPTER 2

### LITERATURE SURVEY

#### 2.1 Phished Website Detection Schemes

Researchers have applied phishing website characteristics to undertake substantial anti- phishing research. A broad classification of phishing detection schemes based on the underlying technique utilized for phished website identification is shown in Figure 2.1.



***Fig-2.1 A classification of phished website detection schemes***

##### 2.1.1 Search Engine Based:

Search engine-based techniques extract features such as text, images, and URLs from websites, then search for them using single or multiple search engines and collect the findings.

Varshney *et al.* [10] Focused on the need of lightweight phishing detection approach using search engines. Authors identified the lightest possible features (page title and domain name) that can be extracted from a webpage without a complete webpage loading. Based on this, authors developed an intelligent anti-phishing chrome extension named lightweight phish detector (LFD). LPD not only detects but also suggests the authentic webpage to the user when a user reaches a deceptive or phishing page on the browser. Ramesh et al. Proposed a technique that collects and matches a group of domains having direct and indirect association with the domain of the suspicious webpage to detect phishing.

### **2.1.2 Heuristics and Machine Learning Based:**

All techniques in this category extract a set of features such as webpage content, URL, and/or network features and a set of machine learning or classification techniques, which are used to create a model for classification.

In 2007, Zhang *et al.* [4] proposed CANTINA, a content-based phishing detection system. In 2015, Rao and Ali [3] presented PhishShield, a desktop application that focuses on phishing detection using the URL and website content of phishing websites, as a heuristic approach to phishing detection.

In 2011 Xiang *et al.* [11] proposed modification of CANTINA called CANTINA+ with eight new features. Mohammad, Thabtah, and McCluskey [5] proposed adaptive self-constructing neural network for classification in 2014.

### **2.1.3 Phishing Blacklist and Whitelist Based:**

The methods in this category utilize the whitelist of normal websites and the blacklist containing anomalous websites to detect phishing. The blacklist is obtained either by user feedback or via reporting by the third parties who perform phishing URL detection using one of the other phishing detection schemes.

In 2016, Jain and Gupta [1] suggested an automatic whitelist update mechanism to combat phishing attacks. In 2014, Lung-Hao and Kuei-Ching *et al.* [12] presented PhishTrack, a framework for automatically updating the blacklist of phishing websites. This framework searches existing blacklists to find suspicious URLs.

### **2.1.4 Visual Similarity Based:**

The visual similarity-based approach compares the visual content of suspect websites and the visual content of trusted domains. All techniques in this category use visual similarity between authentic and phishing webpages and their visual features to detect phishing.

Zhang *et al.* [14] presented visual similarity-based phishing website detection using spatial features in 2014. Yun Lin *et al.* [15] suggested a hybrid deep learning system using visual similarity in 2021. Chiew *et al.* [13] Proposed a logo extraction and image search-based

phishing detection scheme.

#### **2.1.5 DNS Based:**

All techniques use DNS information to identify the authenticity of domain names and IP addresses associated with it to detect phishing. Chen *et al.* [16] Proposed a scheme where a page signature extractor module on the client side obtains the signature of the current webpage. This is sent as a DNS query to a remote server for comparison with signatures of phishing webpages. After receiving the response, policy enforcer module on the client side, takes responsive actions. Frevost *et al.* [17] Proposed a scheme in which the domain name of the URL visited by the user is sent to two DNS servers: the default and third-party DNS servers. If the default IP address is in the IP addresses returned by the third party DNS server, then the current website is considered legitimate. If not, then a webpage content similarity analysis of the visited webpage and reference webpage (obtained from the IP addresses returned by the third-party DNS server) is performed to detect phishing.

#### **2.1.6 Proactive Phishing URL Based:**

All techniques in this category use a mechanism to generate or identify a set of probable phishing URLs. These URLs are mined or reported over web to proactively detect them before they are detected or reported by users or other phishing detection systems.

He *et al.* [18] proposed an approach to proactively identify newly registered malicious domain names. It is based on the observation that legitimate domain names are made up of meaningful English words. The second order Markov model identifies useful features, and random forest classification is applied for detection. Ferolin *et al.* [19] Proposed a scheme in which phishing emails are classified, and web links inside the email are reverse looked up using Who is query to find the location, IP address, and host information of the server hosting the phishing web links. Then, a proactive warning notification is sent to the administrator of the server hosting the web links to take proper action.

Typically, phishing crimes go unreported. This method comprises tricking the user into downloading malicious files, rerouting traffic to malicious websites, compromising user passwords, and gathering private data like bank account numbers and other details.

## **CHAPTER 3**

### **PROBLEM IDENTIFICATION**

With the growth in the field of e-commerce, phishing attack and cybercrimes are rapidly growing. Attackers use websites, emails, and malware to conduct phishing attacks. According to the Anti-Phishing Working Group (APWG) Q4 2020 report, in 2020, there was an average of 225,759 phishing attacks per month, an increase of 220% compared to 2016. The country most affected by phishing sites is China, with 47.9% of machines infected. Phishing has become one of the biggest threats in cyber security. According to the FBI Internet Crime Centredata records, the economic loss due to phishing crimes can reach \$3.5 billion in 2019.

Phishing crimes are usually underreported. This technique involves manipulating the user to download malicious files, redirecting to malicious websites, compromising user credentials as well as collecting sensitive information like bank account details etc.

Phishing website detection helps the user to determine whether the website is suspicious or not by extracting and analysing relevant URL features using Blacklist and Machine Learning approach.

All methods rely on DNS information to determine the legitimacy of domain names and the IP addresses connected to them in order to spot phishing. Chen and others. a plan was put up in which a client-side page signature extractor module would retrieve the webpage's signature. This is transmitted as a DNS request to a distant server for signature comparison with phishing websites. The client-side policy enforcer module responds to the response and takes appropriate action. Frevost and others [17] The domain name of the URL the user visited is sent to two DNS servers: the default and third-party DNS servers, according to the proposed scheme. If the third-party DNS server's response includes the default IP address,



## **CHAPTER 4**

### **OBJECTIVES**

The main objective of this project is to improve the existing phishing detection techniques. Through this project we aim to:

- Observe the existing and proposed applications for phishing detection.
- Establish the need for incorporating machine learning approach as a solution for the problem statement.
- Collect the datasets of URL-based features for phishing and legitimate websites from reliable open source platforms.
- Identify significant predictors from all available aspects of URL dataset.
- Eliminate duplicate, low-variance data and identify missing values to obtain a valid dataset.
- Choose machine learning algorithms to compare for best classification performance.
- Train classifier using the obtained dataset and consider predictor importance and various evaluation metrics.
- Create a MongoDB blacklist consisting of suspicious URLs.
- Develop feature extraction code to acquire feature values corresponding to the dataset used.
- Develop a UI for user input and display of results.
- Integrate the various components of the project – classifier, blacklist and UI and deploy the final application.
- Help in improvising existing techniques for phishing website detection.
- If the URL is not found, then the attribute of the URL is extracted and sent to the ML model for prediction.
- If URL is a phishy website then it will be stored in the blacklist. So that whenever someone enters the same phishing website, the system will directly fetch from the Blacklist database.

## **CHAPTER 5**

### **REQUIREMENT ANALYSIS**

#### **5.2 Functional Requirements**

- The user should enter the URL in a correct format.
- The system will make sure that the URL is formatted correctly. For instance, the URL must begin with http:// or https://.
- If the URL format is Incorrect then, the user will be asked to enter the URL again.
- The system looks for the URL in blacklist database and then displayed to the user if found.
- If the URL is not found, then the attribute of the URL is extracted and sent to the ML model for prediction.
- If URL is a phishy website then it will be stored in the blacklist. So that whenever someone enters the same phishing website, the system will directly fetch from the Blacklist database.
- The web application's home page provides information to the user about the application.
- The web application has two ways to access the URL search bar.

#### **5.3 Non-Functional Requirements**

- The web application should be available anywhere and anytime.
- The MongoDB Database is used for blacklisting so that it can be retrieved anywhere and anytime on any device.
- Security and privacy are maintained here because users' private information is not fetched.
- Simple design so the normal user can also easily use it.
- The system will verify that the URL is properly formatted. The URL, for instance, must start with http:// or https://.
- The user will be prompted to enter the URL again if the format is incorrect.
- If the system locates the URL in the blacklist database, it displays it to the user.

## **5.4 Software Requirements**

### **5.1.1 Kaggle**

Kaggle is used to obtain the required dataset. **Kaggle** a subsidiary of Google LLC, is an online community of data scientists and machine learning practitioners. Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges.

### **5.1.2 Python 3**

**Python** is an interpreter, high-level, general purpose programming language created by Guido Van Rossum and first released in 1991, Python's design philosophy emphasizes code Readability with its notable use of significant Whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

### **5.1.3 MongoDB**

MongoDB is used to maintain the URL blacklist. **MongoDB** is a source-available cross- platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server Side Public License (SSPL) which is deemed non-free by several distributions.

### **5.1.4 Flask**

Flask is a micro web framework written in Python. It is classified as a micro framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

### **5.1.5 Visual Studio Code**

Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactor ring, and

embedded Git.

## **5.1 Hardware Requirements**

- Processor: Intel core i5 or above.
- 64-bit, quad-core, 2.5 GHz minimum per core
- Ram: 4 GB or more
- Hard disk: 10 GB of available space or more.
- Display: Dual XGA (1024 x 768) or higher resolution monitors
- Operating system: Windows

With the growth in the field of e-commerce, phishing attack and cybercrimes are rapidly growing. Attackers use websites, emails, and malware to conduct phishing attacks. According to the Anti-Phishing Working Group (APWG) Q4 2020 report, in 2020, there was an average of 225,759 phishing attacks per month, an increase of 220% compared to 2016. The country most affected by phishing sites is China, with 47.9% of machines infected. Phishing has become one of the biggest threats in cyber security. According to the FBI Internet Crime Centre data records, the economic loss due to phishing crimes can reach \$3.5 billion in 2019.

He *et al.* [18] proposed an approach to proactively identify newly registered malicious domain names. It is based on the observation that legitimate domain names are made up of meaningful English words. The second order Markov model identifies useful features, and random forest classification is applied for detection. Ferolin *et al.* [19] Proposed a scheme in which phishing emails are classified, and web links inside the email are reverse looked up using Who is query to find the location, IP address, and host information of the server hosting the phishing web links. Then, a proactive warning notification is sent to the administrator of the server hosting the web links to take proper action.

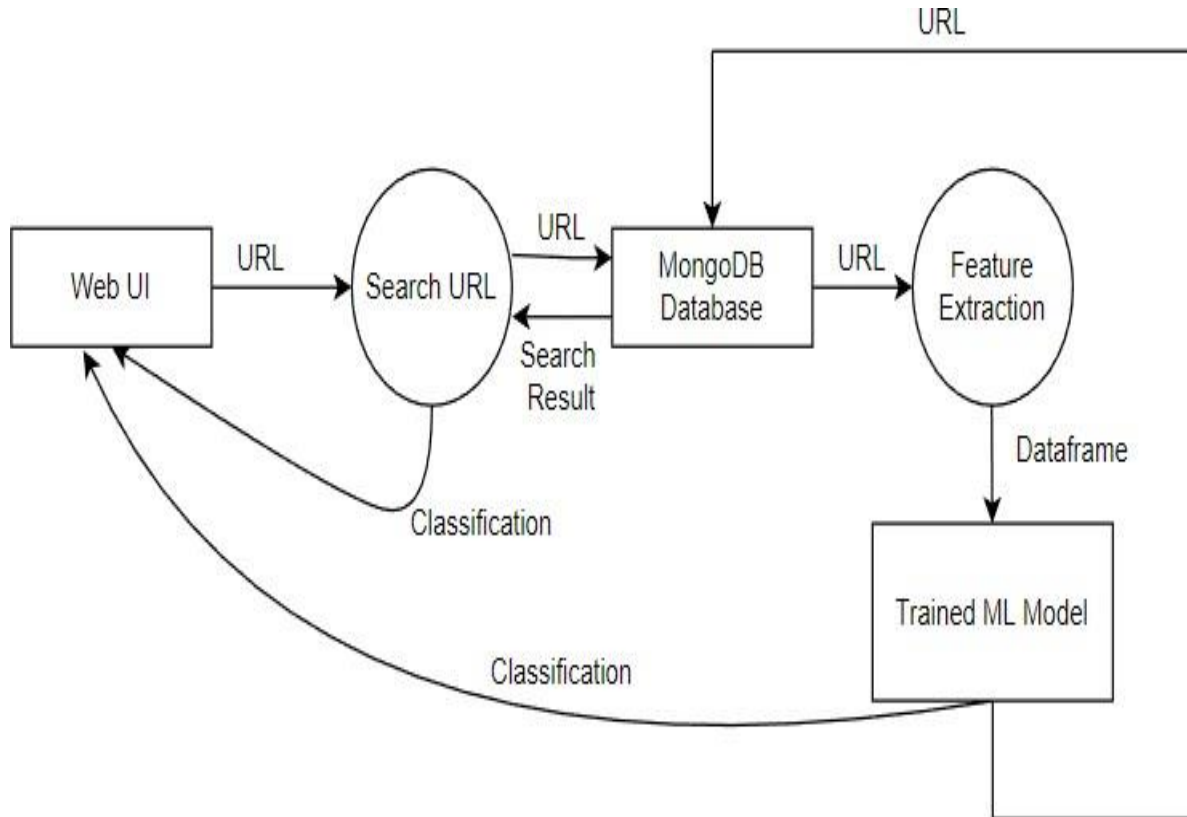
Typically, phishing crimes go unreported. This method comprises tricking the user into downloading malicious files, rerouting traffic to malicious websites, compromising user passwords, and gathering private data like bank account numbers and other details.

## CHAPTER 6

### SYSTEM DESIGN

#### Software Module

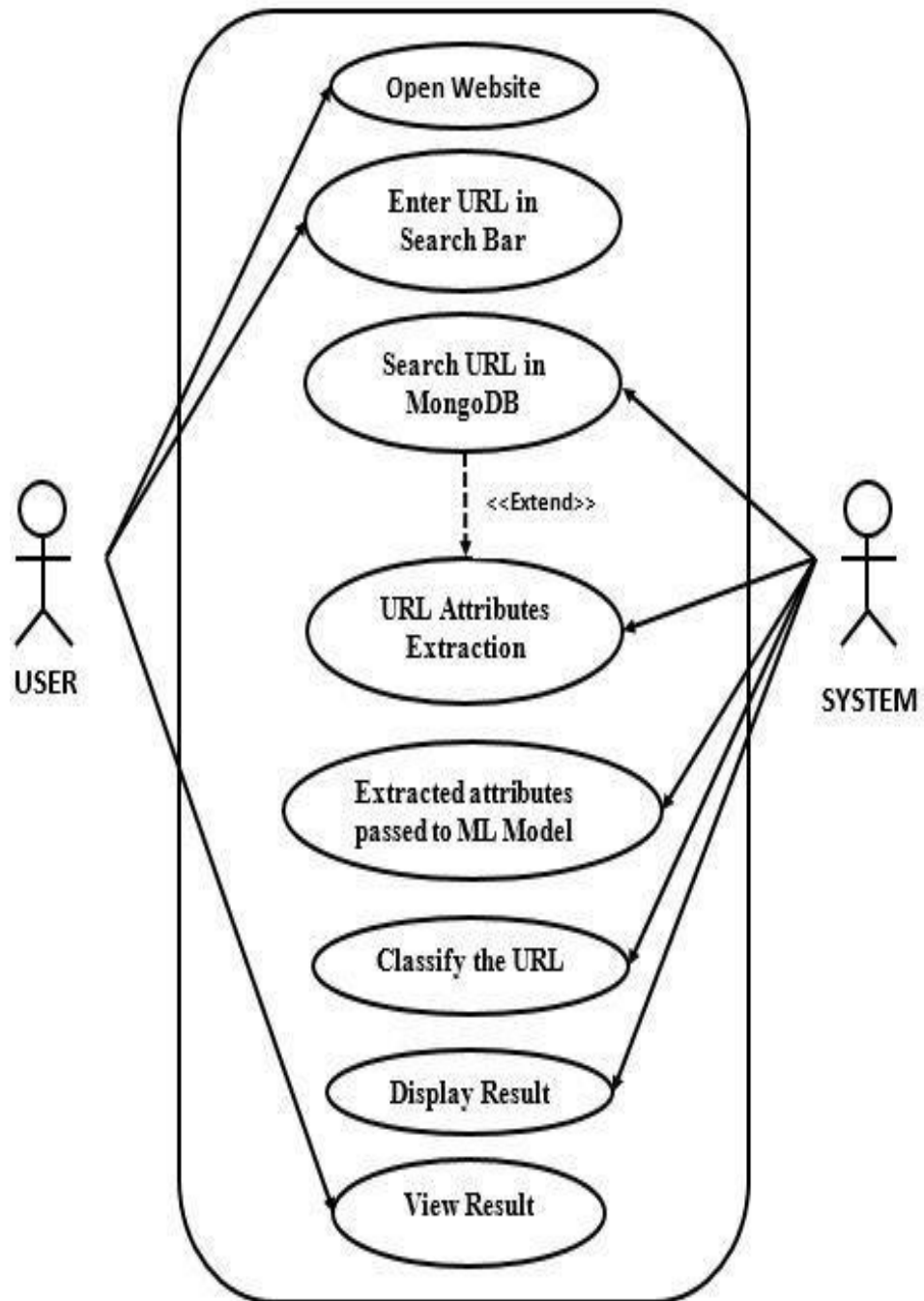
#### 6.1 Data Flow Diagram



**Fig-6.1 Data Flow Diagram of Application**

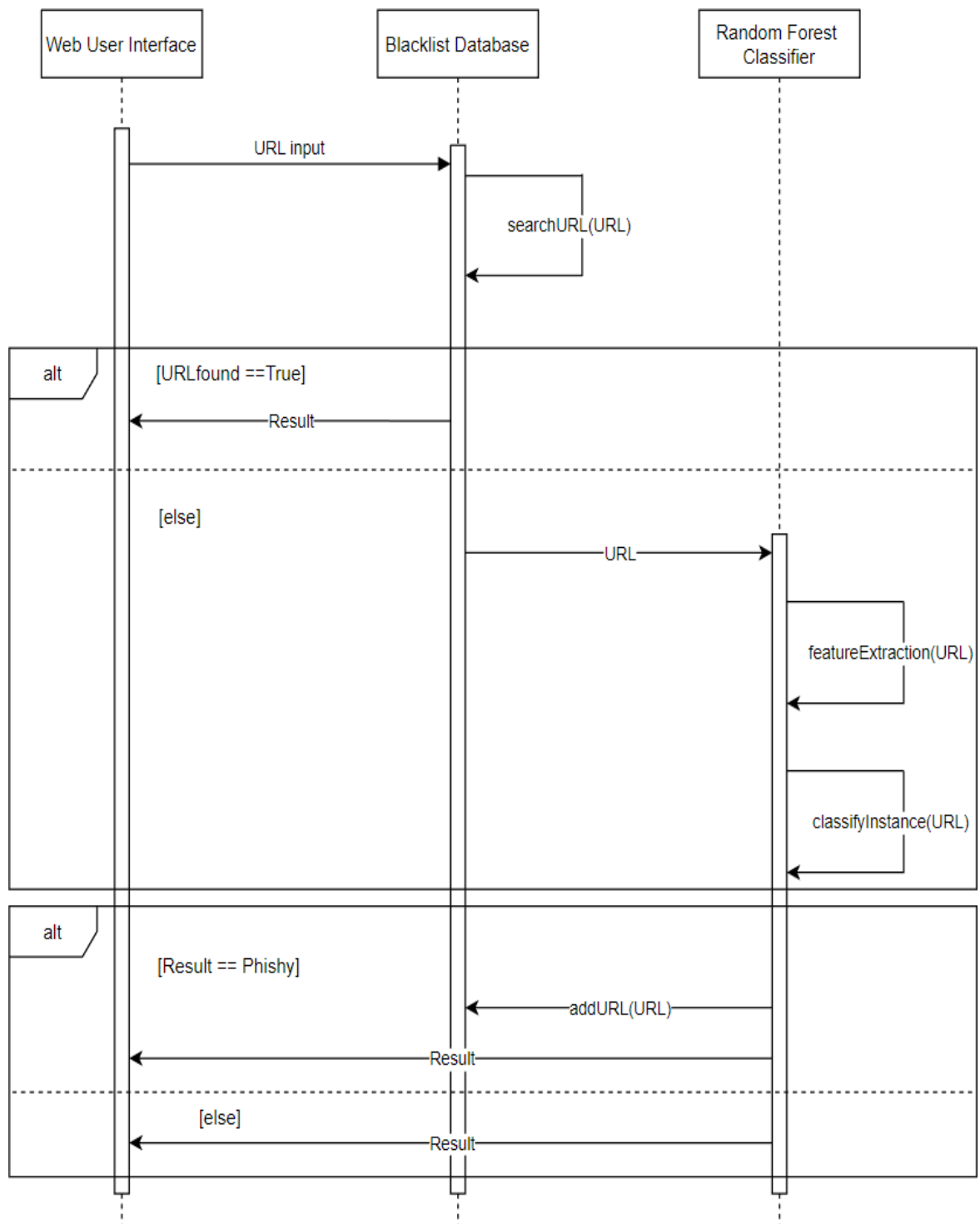
A cyber-attack is any offensive tactic that targets computer information systems, computer networks, infrastructures, or personal computer devices. They are unwelcome attempts to steal, expose, alter, disable or destroy information through unauthorized access to computer systems. A cyber-attack can be launched from anywhere by any individual or group using one or more various attack strategies. Most common types of cyberattacks are: malware, man-in-the-middle, phishing, distributed denial of service (DDoS), SQL injection, & ransomware attack.

## 6.2 Use Case Diagram



*Fig-6.2 Use Case Diagram for Application*

### 6.3 Sequence Diagram



*Fig-6.3 Sequence Diagram of Application*

## CHAPTER 7

# METHODOLOGY

This project proposes a framework that integrates two different approaches for phishing website detection: 1) Blacklist Approach 2) Machine Learning Approach. The Blacklist method is used for faster detection of illegitimate URLs through quick access into the database. Machine Learning is used for predicting new phishing URLs. The proposed framework comprises of the following steps:

### 1) Data Acquisition:

This step involves collection of datasets consisting of both phishing and legitimate URLs from open-source platforms like Kaggle. The balanced dataset consists of 11430 URLs with exactly 50% phishing and 50% legitimate URLs.

### 2) Data Cleaning and Preprocessing:

It involves removing the unnecessary rows and columns from the dataset, selecting only the required features, and also converting the categorical variables into a form that can be provided to the Machine Learning models for prediction. The target label 'status' has two unique values 'legitimate' which is replaced by 0, and 'phishing' replaced by 1. The important features like URL length, number of various symbols in URL, domain age, shortening services, web traffic etc., are considered.

url	length_url	ip	nb_dots	nb_hyphens	nb_at	nb_qm	nb_and	nb_eq	nb_percent	nb_slash	nb_colon
http://www.crestonwood.com/route	37	0	3	0	0	0	0	0	0	3	1
http://shadetretechnology.com/V4	77	1	1	0	0	0	0	0	0	5	1
https://support-appleid.com.secure	126	1	4	1	0	1	2	3	0	5	1
http://rgipt.ac.in	18	0	2	0	0	0	0	0	0	2	1
http://www.iracing.com/tracks/gate	55	0	2	2	0	0	0	0	0	5	1
http://appleid.apple.com-app.es/	32	0	3	1	0	0	0	0	0	3	1
http://www.mutuo.it	19	0	2	0	0	0	0	0	0	2	1
http://www.shadetretechnology.co	81	1	2	0	0	0	0	0	0	5	1
http://vamo aestudiarmedicina.blog	42	0	2	0	0	0	0	0	0	3	1

**Fig-7.1 URL Features Dataset**



nb_semicolumn	nb_www	nb_com	nb_dslash	https_token	prefix_suffix	phish_hints	shortening_service	whois_registered_domain	domain_age	web_traffic	status
0	1	0	0	1	0	0	0	0	-1	0	0
0	0	0	0	1	0	0	0	0	5767	0	1
0	0	1	0	0	1	0	0	0	4004	5828815	1
0	0	0	0	1	0	0	0	0	-1	107721	0
0	1	0	0	1	0	0	0	0	8175	8725	0
0	0	1	0	1	1	0	0	1	-1	0	1
0	1	0	0	1	0	0	0	0	7529	0	0
0	1	0	0	1	0	0	0	0	5767	0	1
0	0	0	0	1	0	0	1	0	7298	0	0

Fig-7.2 URL Features Dataset (Continued)

### 3) Data Visualization:

Data visualization is the process of analyzing the data and observing how different factors affect the data and how the various columns in the dataset are related with the help of charts, graphs and other visuals. The resulting visual representation of data makes it easier to identify new insights about the information represented in the data. Through this process we are able to observe how each URL feature affects the possibility of the website being phishing.

### 4) Database Creation:

This step involves creation of blacklist consisting of only phishing URLs. A new csv file is created to store only the illegitimate URLs. This list is then pushed into a mongoDB collection. MongoDB enables the quick access of already stored illegal websites.

### 5) Training of Machine Learning Model and Prediction:

This process includes selecting the suitable ML algorithm and training it using URL features collected from dataset. 75% of the dataset is used for training. In this project we have used Random Forest Classifier for training the model.

#### Random Forest Classifier:

Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. Each tree in a random forest specifies the class prediction, and the result will be the most predicted class among the decision of trees. Random Forests achieve a reduction

in overfitting by combining many weak learners that underfit because they only utilize a subset of all training samples. Random Forests can handle a large number of variables in a data set. Also, during the forest construction process, they make an unbiased estimate of the generalization error. Besides, they can estimate the lost data well.

We found that Random Forest is highly accurate, relatively robust against noise and outliers, it is fast, simple to implement and understand, and can do feature selection implicitly. Being unaffected by noise is the main advantage of Random Forest. According to Central Limit Theorem, Random Forest reduces variance by increasing the number of trees.

### **Prediction:**

About 20% of data is used testing purpose. After training the Random Forest Classifier, the trained model is tested with various data inputs.

### **6) Extraction of URL Features:**

The user provides URL of the required website as input. Relevant features are extracted from this URL and a dataframe is created. This dataframe is then passed to the trained Random Forest Classifier for prediction. Some of the URL features are:

- URL Length: Phishers can use a long URL to hide the doubtful part in the address bar.
- Having IP Address: If an IP address is used instead of the domain name in the URL.
- Shortening Service: Links to the webpage that has a long URL.
- Having @ Symbol: Using the @ symbol in the URL leads the browser to ignore everything preceding the @ symbol and the real address often follows the @ symbol.
- Prefix Suffix: 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.
- Age of Domain: If the age of the domain is less than a month.
- Web Traffic: This feature measures the popularity of the website by determining the number of visitors.

### **7) Performance Evaluation and Comparison:**

In this phase the performance of the model is evaluated using various metrics. They are:

- **Accuracy:** It is a metric that generally describes how the model performs across all classes. It is useful when all classes are of equal importance. It is calculated as the ratio between the number of correct predictions to the total number of predictions. In this

project we have achieved the accuracy of about 92.75%.

- **Precision:** It is calculated as the ratio between the number of *Positive* samples correctly classified to the total number of samples classified as *Positive* (either correctly or incorrectly). The precision measures the model's accuracy in classifying a sample as positive. We have achieved the precision of about 94%.
- **Recall:** It is calculated as the ratio between the numbers of *Positive* samples correctly classified as *Positive* to the total number of *Positive* samples. The recall measures the model's ability to detect *Positive* samples. The obtained recall score is 91%.

This model is then compared with the performance of other ML models like Support Vector Machine and Logistic Regression. The Random Forest Classifier is found to be more accurate.

- **Support Vector Machine:**

It is one of the most popular classifiers. The idea behind SVM is to get the closest point between two classes by using the maximum distance between classes. It needs high calculations to train data. Also, it is sensitive to noisy data and is therefore prone to overfitting.

- **Logistic Regression:**

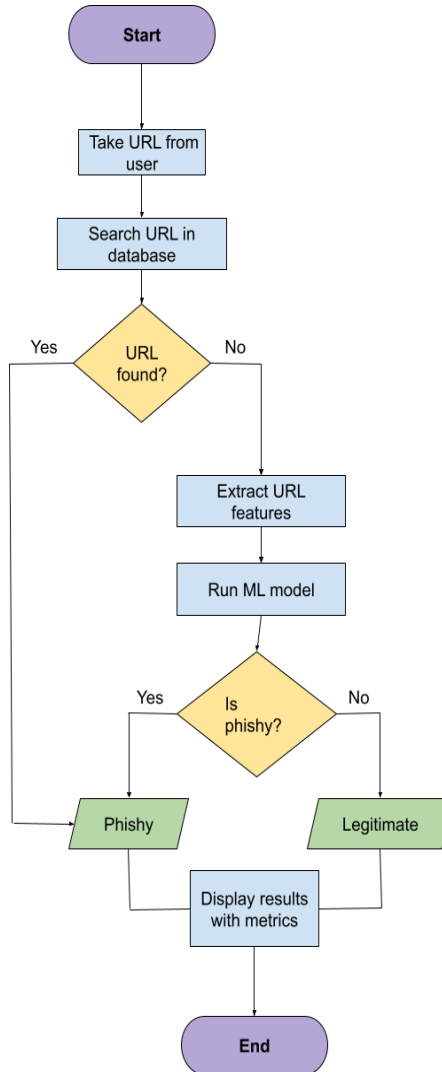
It is a classification algorithm used to assign observations to a discrete set of classes. It transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes. It works well when the relationship in the data is almost linear despite if there are complex nonlinear relationships between variables, it has poor performance.

## 8) Automatic Modification of Blacklist:

When the URL entered by the user is classified as phishing, it is automatically added to the blacklist database so that in future if the same URL is searched then the system can directly fetch it from database and classify it as phishing

Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. Each tree in a random forest specifies the class prediction, and the result will be the most predicted class among the decision of trees.

The below flowchart demonstrates the phishing website detection process:



**Fig-7.3 Application Flowchart**

The user opens the application and enters the complete URL of the website he wishes to visit. After that, the system retrieves the URL and searches the mongoDB blacklist database. If the URL is found, then it directly warns the user to not visit the webpage since it is found to be a phishing website. If the URL is not found in the database, then the required features of the URL are extracted and a dataframe is created. It is then passed to the trained ML model, which predicts the legitimacy of the URL.

## **CHAPTER 8**

### **IMPLEMENTATION**

#### **8.1 Importing Python libraries**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import datetime
import ipaddress
import time
from typing import List
from urllib.parse import urlparse
import whois
import re
from bs4 import BeautifulSoup
import requests
import pickle
import pymongo
from flask import Flask, flash, redirect, url_for, request, render_template
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn import svm
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.model_selection import train_test_split
from urllib.parse import urlencode
```

## 8.2 Creation of clean dataset

```
warnings.filterwarnings('ignore')
```

```
#Reading the csv file and storing it in a dataframe
```

```
data = pd.read_csv("dataset_phishing.csv")
```

```
# Creating a new dataframe with selected columns
```

```
df =
```

```
pd.DataFrame(data,columns=['url','length_url','ip','nb_dots','nb_hyphens','nb_at','nb_qm','nb_and','nb_eq','nb_percent','nb_slash','nb_colon','nb_semicolumn','nb_www','nb_com','nb_dslash','https_token','prefix_suffix','phish_hints','shortening_service','submit_email','whois_registered_domain','status'])
```

```
#Replacing the categorical values
```

```
df["status"].replace({ "legitimate":0, "phishing":1 },inplace=True)
```

```
#Convert dataframe into csv file
```

```
df.to_csv('phishing_dataset_final.csv',index=False)
```

## 8.3 Creation of Blacklist Database

We extracted all the phishing URLs from the final dataset into a separate CSV file.

```
#Reading the csv file and storing it in a dataframe
```

```
data = pd.read_csv("phishing_dataset.csv")
```

```
#Taking status and url in another dataframe
```

```
df = pd.DataFrame(data,columns=['url','status'])
```

```
#Taking only phishy urls
```

```
df = df.query('status==1')
```

```
#Dropping status column
```

```
df.drop(['status'],axis=1,inplace=True)
```

```
#Saving to another csv file
```

```
df.to_csv('blacklist.csv',index=False)
```

We pushed all the URLs from blacklist.csv to a 'URL\_Blacklist' collection in the MongoDB database.

### **[PUSH]**

We implemented two functions: 1) search for URL in database and 2) to add new URL that gets classified as 'Phishy'.

```
client =
```

```
pymongo.MongoClient('mongodb+srv://majorproject:project12345@cluster0.e90mk.mongod  
b.net/URL_Blacklist?retryWrites=true&w=majority')
```

```
# connect to DB
```

```
db = client['URL_Blacklist']
```

```
# connect to collection in DB
```

```
collection = db['URL_Blacklist']
```

```
def addURL_MongoDB(url):
```

```
    # add new URL to DB
```

```
    data = {'url': url}
```

```
    collection.insert_one(data)
```

```
def search_URL(url):
```

```
    # search user input URL in database
```

```
    found = collection.find({'url': url})
```

```
    exists = True
```

```
    if found:
```

```
        for result in found:
```

```
            return exists
```

```
        exists = False
```

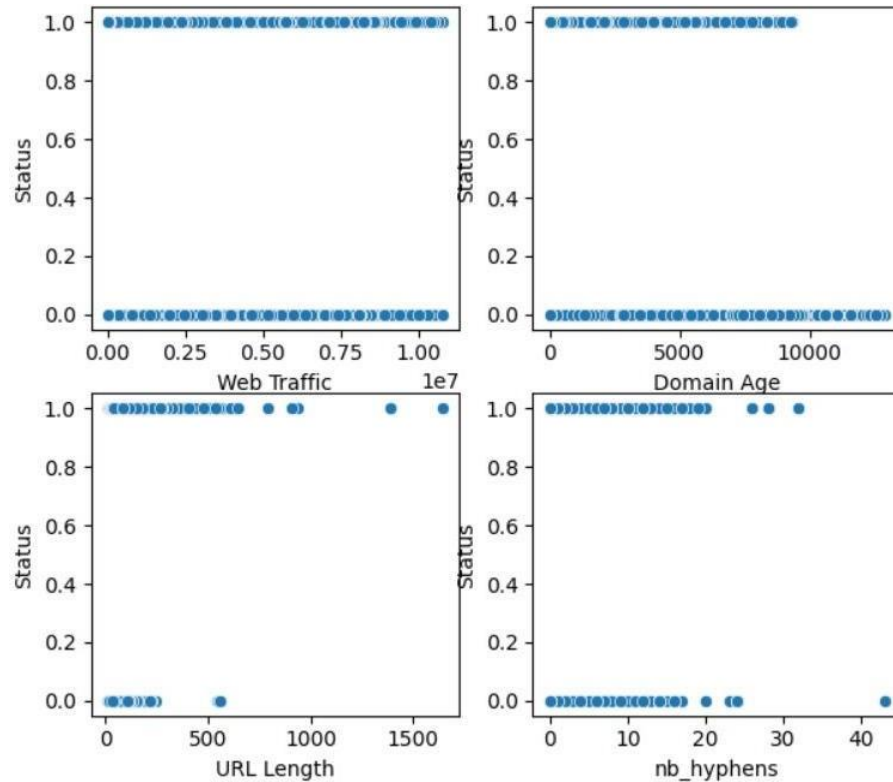
```
    return exists
```

#### **8.4 Data Visualization**

We plotted graph of various features and analysed them.

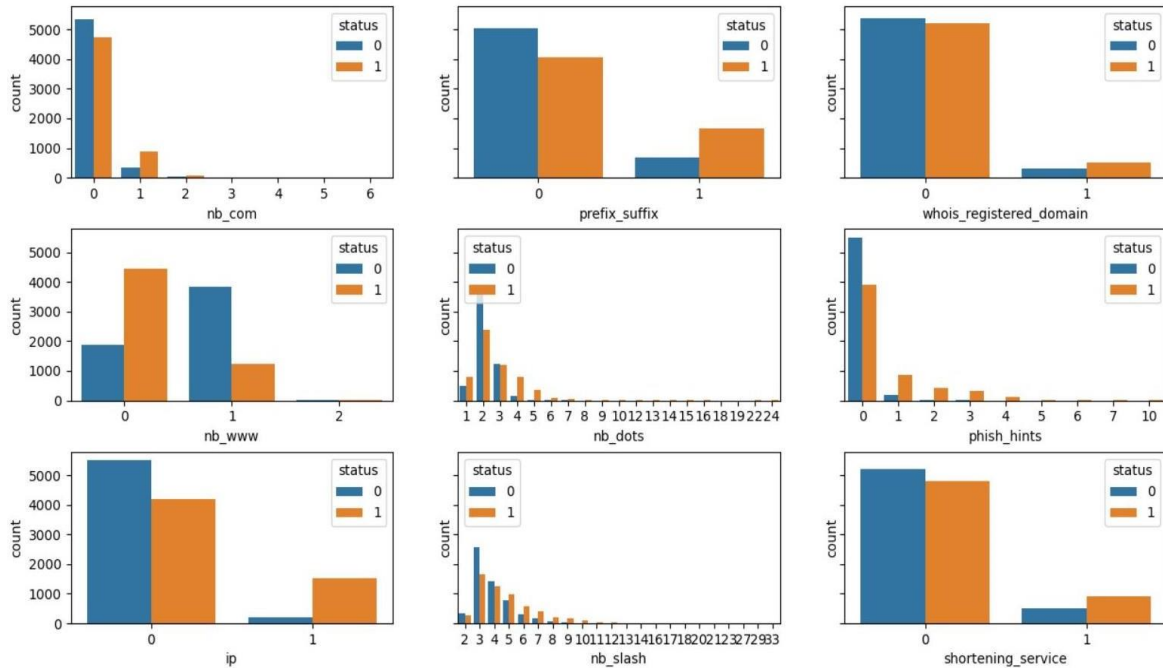
```
plt.figure(figsize = (5 , 5))
plt.subplot(221)
sns.scatterplot(x = d["web_traffic"] , y = d["status"])
plt.xlabel("Web Traffic")
plt.ylabel("Status")
plt.subplot(222)
sns.scatterplot(x = d["domain_age"] , y = d["status"])
plt.xlabel("Domain Age")
plt.ylabel("Status")
plt.subplot(223)
sns.scatterplot(x = d["length_url"] , y = d["status"])
plt.xlabel("URL Length")
plt.ylabel("Status")
plt.subplot(224)
sns.scatterplot(x = d["nb_hyphens"] , y = d["status"])
plt.xlabel("nb_hyphens")
plt.ylabel("Status")
plt.show()
```





*Fig-8.1. Scatter plot of status vs URL features*

```
fig,ax=plt.subplots(3,3,sharey=True,figsize=(15,8))
plt.subplots_adjust(hspace=0.3)
sns.countplot(data=d,x='nb_com',hue='status',ax=ax[0,0])
sns.countplot(data=d,x='prefix_suffix',hue='status',ax=ax[0,1])
sns.countplot(data=d,x='whois_registered_domain',hue='status',ax=ax[0,2])
sns.countplot(data=d,x='nb_www',hue='status',ax=ax[1,0])
sns.countplot(data=d,x='nb_dots',hue='status',ax=ax[1,1])
sns.countplot(data=d,x='phish_hints',hue='status',ax=ax[1,2])
sns.countplot(data=d,x='ip',hue='status',ax=ax[2,0])
sns.countplot(data=d,x='nb_slash',hue='status',ax=ax[2,1])
sns.countplot(data=d,x='shortening_service',hue='status',ax=ax[2,2])
plt.show()
```



**Fig-8.2 Bar graph for visualization of URL features**

### 8.5 Creation and Training of Random Forest Classifier

```
d=pd.read_csv("phishing_dataset_final.csv")
x=d.iloc[:, :-1]
y=d['status']
x.drop(['url'],axis=1,inplace=True)
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.25,random_state=0)
rfc = RandomForestClassifier(n_estimators=10)
rfc.fit(x_train,y_train)
y_pred = rfc.predict(x_test)
```

### 8.6 Obtaining the Feature Importance

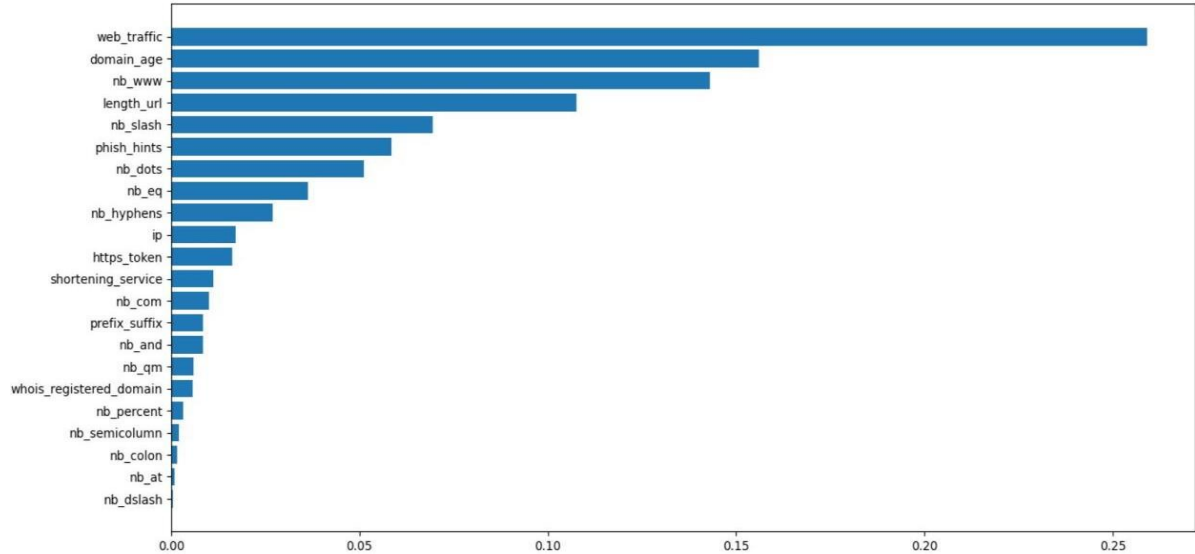
The feature importance is used to realize how effective each feature is at predicting target variable. The ‘web traffic’ feature is found to be the most effective in phishing website detection.

```
rfc.fit(x,y)
importances = rfc.feature_importances_
sorted_indices = np.argsort(importances)
fig,ax = plt.subplots()
```

```

ax.barh(range(len(importances)),importances[sorted_indices])
ax.set_yticks(range(len(importances)))
ax.set_yticklabels(np.array(x.columns)[sorted_indices])
plt.show()

```



**Fig-8.3 Feature importance graph**

## 8.7 Performance Evaluation of Trained Classifier

We have determined the accuracy, precision and recall score of the trained model. We have also created a confusion matrix to visualize and summarize the performance of the classifier.

```

cnf_matrix=metrics.confusion_matrix(y_test,y_pred)
sns.heatmap(pd.DataFrame(cnf_matrix),annot=True,fmt='g')

```

```
plt.show()
```

```
print("Accuracy : ",metrics.accuracy_score(y_test,y_pred))
```

```
print("Precision : ",metrics.precision_score(y_test,y_pred))
```

```
print("Recall : ",metrics.recall_score(y_test,y_pred))
```

```
rfc_score_train = rfc.score(x_train,y_train)
```

```
rfc_score_test = rfc.score(x_test,y_test)
```

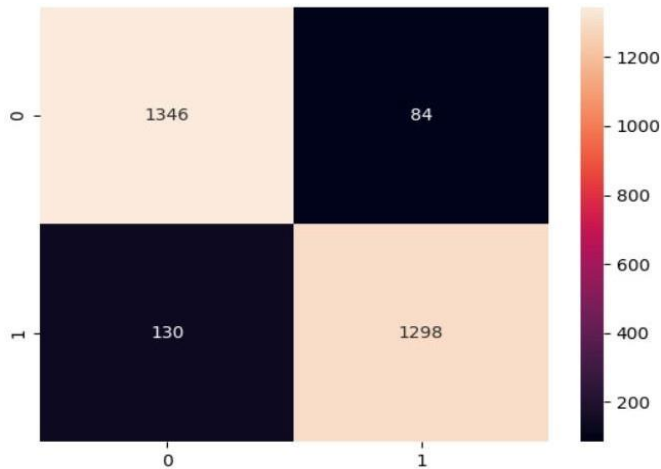


Fig-8.4 Confusion matrix

```
[[1346  84]
 [ 130 1298]]
Accuracy : 0.9251224632610217
Precision : 0.9392185238784371
Recall : 0.9089635854341737
```

Fig-8.5 Performance measurement

## 8.8 Comparison with other ML models

We used the same data to train Support Vector Machine and Logistic Regression Models. Then we tested the models and compared their accuracy with the accuracy of Random Forest Classifier. The Random Forest Classifier is found to be more accurate.

```
Scaler = StandardScaler()
Scaler.fit(x_train)
x_train=Scaler.transform(x_train)
x_test=Scaler.transform(x_test)
svm = svm.SVC()
svm.fit(x_train,y_train)
y_test_pred = svm.predict(x_test)
svm_score_train = svm.score(x_train,y_train)
svm_score_test = svm.score(x_test,y_test)
```

```
logmodel = LogisticRegression()
logmodel.fit(x_train,y_train)
```

```
prediction = logmodel.predict(x_test)
```

```
lr_score_train = logmodel.score(x_train,y_train)
```

```
lr_score_test = logmodel.score(x_test,y_test)
```

```
models = pd.DataFrame({'Model':['Random Forest', 'Support Vector Machine', 'Logistic  
Regression'],'Train_Score':[rfc_score_train,svm_score_train,lr_score_train],'Test_Score':[rfc_  
score_test,svm_score_test,lr_score_test]})
```

```
models.sort_values(by='Test_Score', ascending=False)
```

```
print(models)
```

```
Model = ['Random Forest', 'Support Vector Machine', 'Logistic Regression']
```

```
Train_Score = [rfc_score_train,svm_score_train,lr_score_train]
```

```
Test_Score = [rfc_score_test,svm_score_test,lr_score_test]
```

```
x = np.arange(len(Model))
```

```
width = 0.4
```

```
fig, ax = plt.subplots()
```

```
ax.bar(x-width/2,Train_Score,width,label="Train_Score")
```

```
ax.bar(x+width/2,Test_Score,width,label="Test_Score")
```

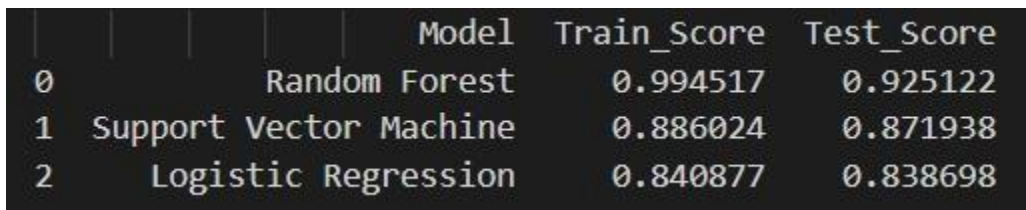
```
ax.set_ylabel('Scores')
```

```
ax.set_xticks(x)
```

```
ax.set_xticklabels(Model)
```

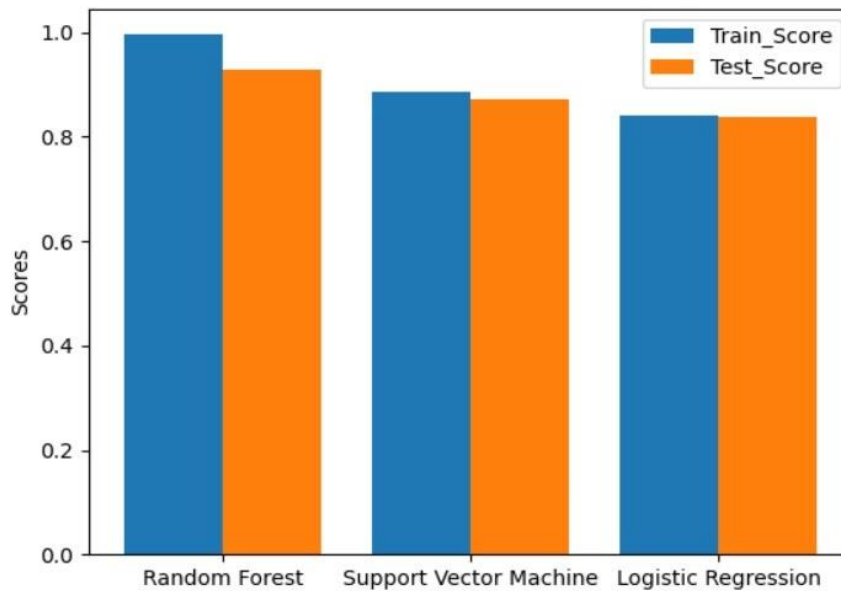
```
ax.legend()
```

```
plt.show()
```



	Model	Train_Score	Test_Score
0	Random Forest	0.994517	0.925122
1	Support Vector Machine	0.886024	0.871938
2	Logistic Regression	0.840877	0.838698

*Fig-8.6 Comparison with SVM and Logistic Regression*



*Fig-8.7 Graph for comparing performance of different classifiers*

## 8.9 Extraction of URL Features

In order to extract the features from the URL entered by the user, we have written some functions. This extractor function returns a list of features that can be passed to the ML model for prediction.

```
HINTS = ['wp', 'login', 'includes', 'admin', 'content', 'site', 'images', 'js', 'alibaba', 'css',
'myaccount', 'dropbox', 'themes', 'plugins', 'signin', 'view']
# URL hostname length
def url_length(url):
    return len(url)
# Having IP address in hostname
def having_ip_address(url):
    match = re.search(
        '(([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5]))|'
        '([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\/' # IPv4
        '((0x[0-9a-fA-F]{1,2})\\. (0x[0-9a-fA-F]{1,2})\\. (0x[0-9a-fA-F]{1,2})\\. (0x[0-9a-fA-F]{1,2}))\\/' # IPv4 in hexadecimal
```

```
'(?:[a-zA-F0-9]{1,4}:){7}[a-zA-F0-9]{1,4}'  
'[0-9a-zA-F]{7}', url) # Ipv6  
if match:  
    return 1  
else:  
    return 0  
# Count number of dots in hostname  
def count_dots(hostname):  
    return hostname.count('.')  
#Count dash (-) symbol at base url  
def count_hyphens(base_url):  
    return base_url.count('-')  
# Count at (@) symbol at base url  
def count_at(base_url):  
    return base_url.count('@')  
#Count slash (/) symbol at full url  
def count_slash(full_url):  
    return full_url.count('/')  
#count www in url words  
def check_www(words_raw):  
    count = 0  
    for word in words_raw:  
        if not word.find('www') == -1:  
            count += 1  
    return count  
#count com in url words  
def check_com(url):  
    return url.count('.com')  
# Count redirection (//) symbol at full url  
def count_double_slash(full_url):  
    list=[x.start(0) for x in re.finditer('//', full_url)]  
    if list[len(list)-1]>6:
```

```
        return 1
    else:
        return 0
    return full_url.count("//")
# Uses https protocol
def https_token(scheme):
    if scheme == 'https':
        return 0
    return 1
#prefix suffix
def prefix_suffix(url):
    if re.findall(r"https?:\/\/[^\-]+\-[^\-]+\/", url):
        return 1
    else:
        return 0
#number of phish-hints in url path
def phish_hints(url_path):
    count = 0
    for hint in HINTS:
        count += url_path.lower().count(hint)
    return count
# URL shortening
def shortening_service(full_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|lurl\.com|tweez\.me|v\.g
d|' 'tr\.im|link\.zip\.net',full_url)
```



```
    if match:
        return 1
    else:
        return 0
#whois_registered_domain
def whois_registered_domain(url):
    try:
        domain = urlparse(url).netloc
        domain_name = whois.whois(domain)
        domainResponse = domain_name.domain_name
        try:
            if type(domainResponse) == list:
                for host in domainResponse:
                    if re.search(host.lower(), domain):
                        return 0
                return 1
            else:
                if re.search(domainResponse.lower(),domain):
                    return 0
                else:
                    return 1
        except:
            return 1
    except:
        return 1
# Domain age of a url
import json
def domainAge(url):
    try:
        domain = urlparse(url).netloc
        #print(domain)
        domain_name = whois.whois(domain)
```

```
#print(type(domain_name.creation_date))
creation_date = domain_name.creation_date
expiration_date = domain_name.expiration_date
if ((expiration_date is None) or (creation_date is None)):
    return -1
else:
    if type(creation_date) == list:
        creation_date = creation_date[0]
    if type(expiration_date) == list:
        expiration_date = expiration_date[0]
    ageofdomain = abs((expiration_date - creation_date).days)
    return ageofdomain
except:
    return -1

# web traffic (Page Rank)
import urllib
def web_traffic(short_url):
    try:
        rank=
BeautifulSoup(urllib.request.urlopen("http://data.alexa.com/data?cli=10&dat=s&url=" +
short_url).read(), "xml").find("REACH")["RANK"]
    except:
        return 0
    return int(rank)

#Function to extract features
def featureExtraction(url):
    parsed = urlparse(url)
    scheme = parsed.scheme
    domain = urlparse(url).netloc
    words_raw = re.split("-|_|\\.|\\?|=|_|@|&|\\%|\\:|\\_", domain.lower())
    features = []
    #features.append(url)
```

```
features.append(url_length(url))
features.append(having_ip_address(url))
features.append(count_dots(url))
features.append(count_hyphens(url))
features.append(count_at(url))
features.append(count_qm(url))
features.append(count_and(url))
features.append(count_equal(url))
features.append(count_percentage(url))
features.append(count_slash(url))
features.append(count_colon(url))
features.append(count_semicolumn(url))
features.append(check_www(words_raw))
features.append(check_com(url))
features.append(count_double_slash(url))
features.append(https_token(scheme))
features.append(prefix_suffix(url))
features.append(phish_hints(url))
features.append(shortening_service(url))
features.append(whois_registered_domain(url))
features.append(domainAge(url))
features.append(web_traffic(url))
return features
```

### **8.10 Combining Both Blacklist and ML Approach**

We then wrote a program to combine both blacklist database and ML approach. The predictorcode first searches the URL in MongoDB blacklist. If it is found then directly the URL is classified as phishing, else the feature extractor function is called to extract the required features from URL and then it is passed to the ML model. If the model predicts the URL as phishing then it is added to the blacklist.

```
import feature_extraction as fe
import database as db
```

```
import pickle

model_filename = "RandomForestClassifier.sav"

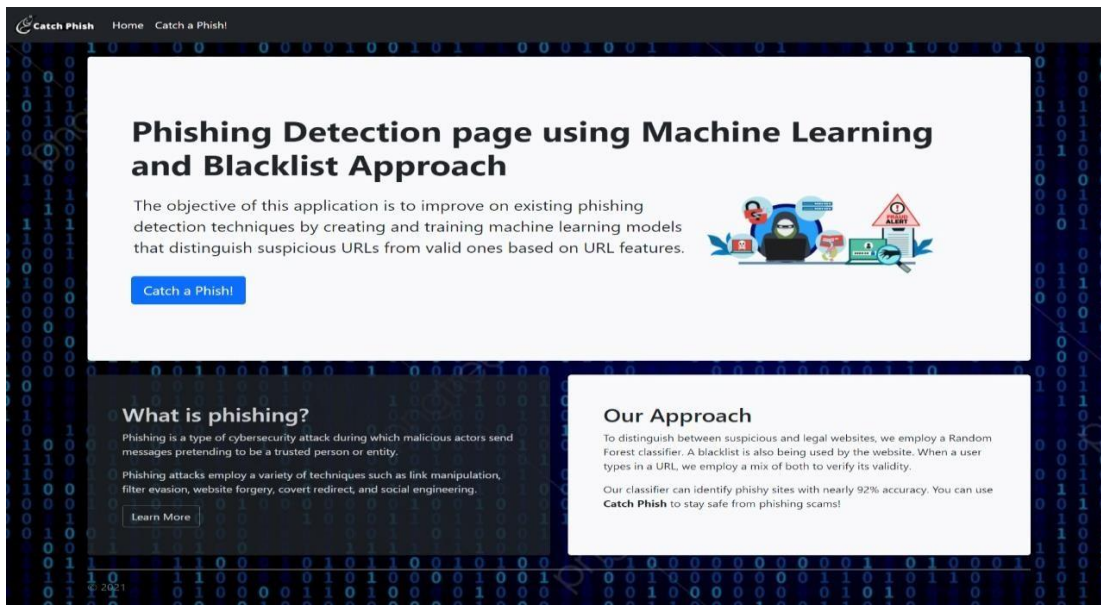
# Load model to predict new data
with open(model_filename, 'rb') as f:
    rfc = pickle.load(f)
# Classify URL input
def classifyURL(url):
    if db.search_URL(url):
        return "Phishy"
    else:
        df = fe.featureExtraction(url)
        res = int(rfc.predict([df]))
        if res:
            db.addURL_MongoDB(url)
            return "Phishy"
        return "Legitimate"
```

### **8.11 Catch Phish - A Webpage for Phishing Website Detection**

We have created a web application called Catch Phish to detect the phishing websites. The user interface is designed using HTML and CSS. Flask framework is used to build the web application.

#### **Homepage:**

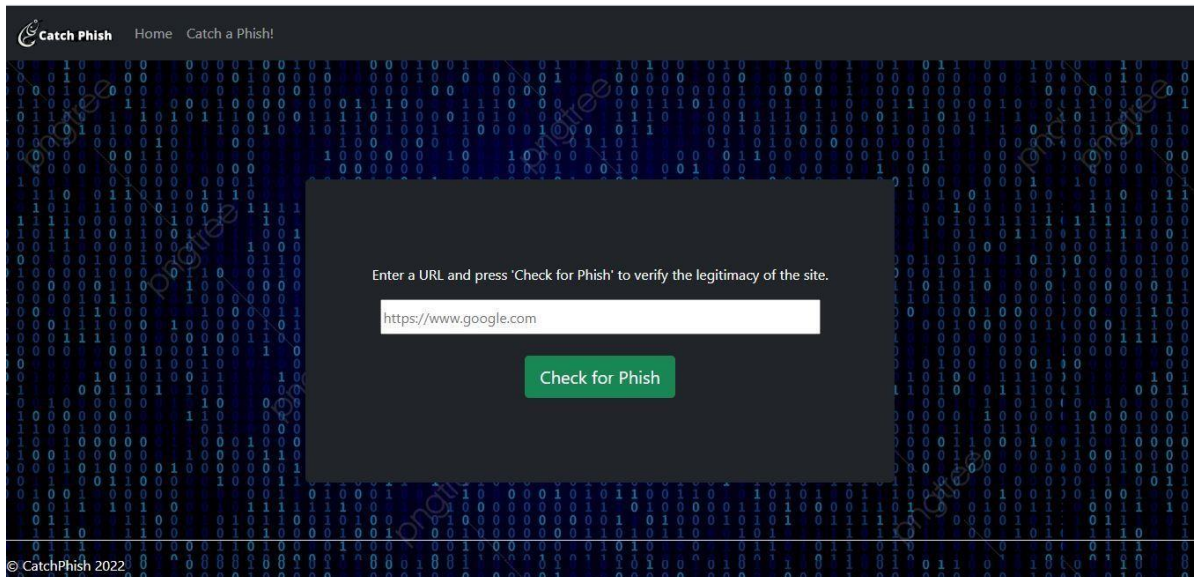
The homepage consists of details about the website. The user can click on ‘**Catch a Phish**’ button or can click on the ‘**Catch a Phish!**’ option on navbar to visit the page where he can enter the URL.



*Fig-8.8 Homepage of the web application*

### Catch a Phish Page:

In this page the user can enter the URL of the website he wants to check. After entering the complete URL, the user should click on the ‘**Check for Phish**’ button. Then the link is sent for verification.

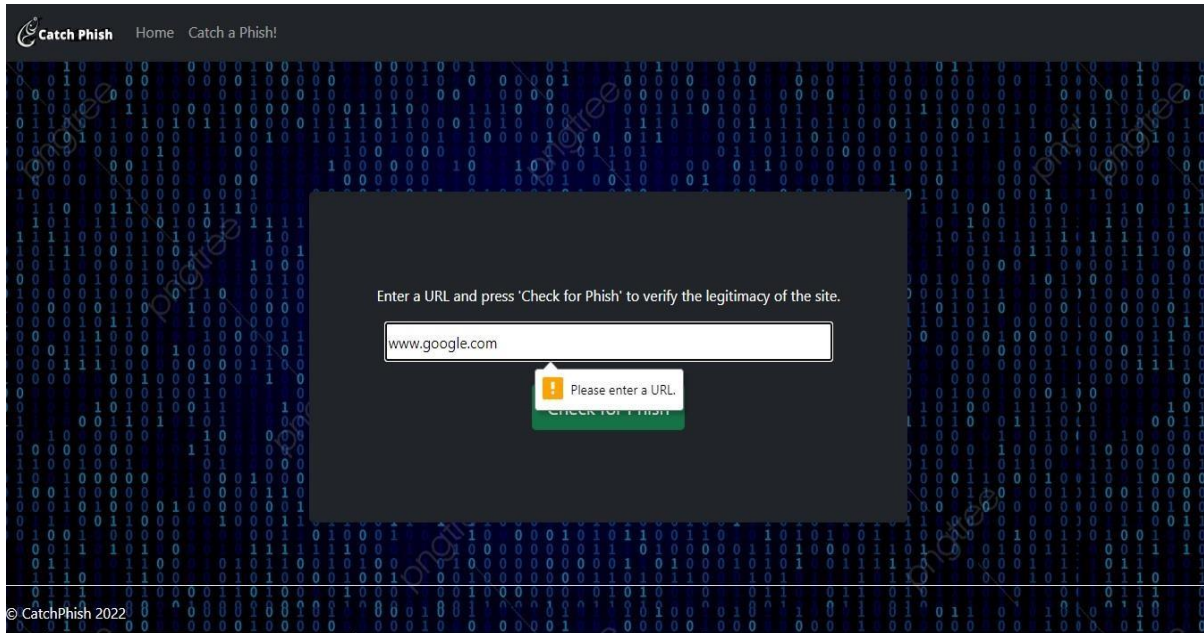


*Fig-8.9 Catch a phish page*

The form will only take complete URL. If the user enters the website name without the https or http token the form will ask the user to enter the complete URL. For example, if the user



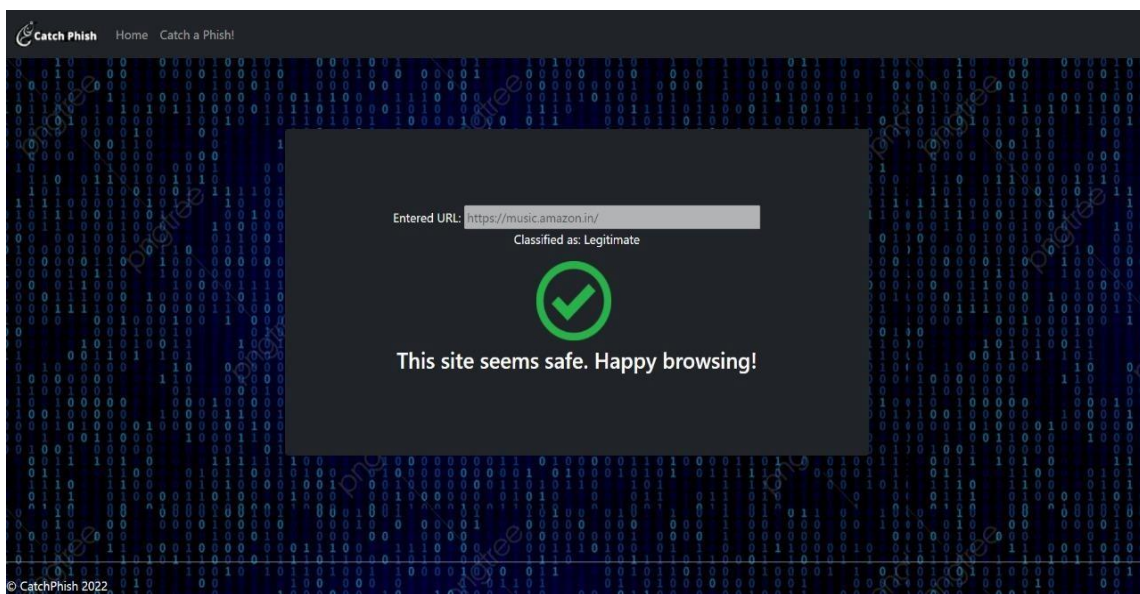
enters the URL as 'www.google.com' the form will ask the user to enter the complete URL including the http or https token.



*Fig-8.10 Invalid URL entry*

### Result Page:

If the user enters the valid URL, it will be verified and the user will be sent to the result page where the application will display whether the URL is fake or legitimate. If the entered URL is classified as legitimate then the webpage will notify the user about its legitimacy.



*Fig-8.11 Result for legitimate URL*

## CHAPTER 9

### TESTING

Testing is the process of evaluating and verifying the software application. It helps in detecting errors, reducing development costs and improving the performance of the application. It plays critical role in quality assurance and for assuring the reliability of the application.

#### 9.1 Importance of Testing:

- Testing and quality go hand in hand. Quality can be measured by the number of defects identified during testing and those will then be fixed within a software development lifecycle.
- When a critical defect is discovered in a live environment the impact and severity are both high because it affects the end user.
- Testing early not only enables the system to be better, but it also allows the project manager to have better control over the budget.
- Testing throughout the development lifecycle provides visibility as to how well and stable the software has been developed, therefore increasing the confidence in the customer.

#### 9.2 Levels of Testing:

There are four main stages of testing that need to be completed before an application can be cleared for use: unit testing, system testing and acceptance testing.

##### 1) Unit Testing:

In this type of testing, errors are detected individually from every component or unit by individually testing the components of software to ensure that if they are fit for use. It is the smallest testable part of the software. Our application includes 3 main modules: Database module, Machine Learning module and Feature Extraction module.

- **Database Module:** This module contains add and search URL functions that are used for testing. A URL is first added to the blacklist database and then searched in the database. If it is found then both functions are working correctly.

- **Machine Learning Module:**

This module deals with the training and prediction of classifier. It is tested by providing the URL features and checking if the classifier is predicting accurately.

- **Feature Extraction Module:**

This module includes extraction functions for all the required features. This can be tested by providing a URL and checking if the features are properly extracted.

## 2) Integration Testing:

In this testing, two or more modules which are unit tested are integrated to test if there are any interface errors present. Machine Learning and Feature Extraction modules were integrated to check if they are working as per the expectation.

## 3) System Testing:

In this testing, complete and integrated programs are tested i.e., all the system elements forming the system is tested as a whole to meet the requirements of the system. The Database, Machine Learning and Feature Extraction modules are combined and a user interface is provided to complete the application. The application is then subjected to testing. The URL is fetched from the user and checked in Database. If not found then URL Features are extracted and Machine Learning Classifier is used for prediction. The UI is also checked if it is fetching the URL and displaying the result properly.

Sl. No	URL	Expected Outcome	Predicted Outcome
1	<a href="https://pancakefinanceswap.app/">https://pancakefinanceswap.app/</a>	Phishy (2sec)	Phishy (2sec)
2	<a href="https://pancakefinanceswap.app/(Using Blacklist)">https://pancakefinanceswap.app/(Using Blacklist)</a>	Phishy (1sec)	Phishy (1sec)
3	<a href="https://www.meesho.com/">https://www.meesho.com/</a>	Legitimate (2sec)	Legitimate (2sec)
4	<a href="https://www.amazon.in/">https://www.amazon.in/</a>	Legitimate (2sec)	Legitimate (2sec)
5	<a href="https://phishtank.org/phish_search.php?valid=y&amp;active=All&amp;Search=Search">https://phishtank.org/phish_search.php?valid=y&amp;active=All&amp;Search=Search</a>	Legitimate (2sec)	Legitimate (2sec)

**Table-9.1 Testing of URLs**



## CONCLUSION

Through this project, we have improvised the current phishing detection techniques by combining both blacklist and machine learning approaches. The use of blacklist approach reduces the response time since the URL will be searched in the database before going through feature extraction and being passed to the classifier for prediction. We have considered 22 URL features for which we obtained the accuracy up to 92%. This application can be used by everyone to detect the phishing websites which are circulated nowadays through social medias. The simple design helps the user to understand the usage and easily access the application. This can be helpful in reducing the phishing attacks which are being carried out by sharing the malicious URLs.

The method of recognizing a URL type is generated by means of association rules which used different heuristics to get hidden knowledge. These rules are utilized for recognizing the type of URL any time a client accesses it. We have defined some heuristics mined from some URLs and more than 20 URLs from different sources are obtained. Our genuine URLs are gotten from five different sources and about 50 phishing URLs are dig out from the phishtank database (<http://www.phishtank.com>). The feature extraction is carried out with PHP, MySQL connector is used in order to fetch the data set in the data base of the system. This investigation was carried out using predictive apriori rules generation algorithms. The investigation is carried out to establish the rules centered on phishing URLs. Table 3.1 shows legitimate data source

The "blacklist" approach is a general method for detecting phishing websites by adding blacklisted URLs and IP addresses to the antivirus database. To get around blacklists, attackers use clever techniques like obfuscation and many other simple techniques like fast-flux, where proxies are automatically created to host the web page; algorithmic generation of new URLs; and so on. The method's biggest flaw is that it can't detect zero-hour phishing attacks.

## **FUTURE SCOPE**

As there are plenty of ideas and innovation that one could implement, there are also many innovation ideas that can be processed further or extended further in our project. Here we are concentrating on two approaches, i.e., blacklist and machine learning approach. One can also include the other approaches like the heuristic approach, visual similarity, whitelist approach, etc. Otherwise, the response time can be reduced by just including whitelist approaches in the project. For a better user experience, one may alternatively substitute a web application with a plugin or extension. Or can improve the web application created as front end for this project.

Phishing is described as the art of emulating a website of a creditable firm intending to grab user's private information such as usernames, passwords and social security number. Phishing websites comprise a variety of cues within its content-parts as well as browser-based security indicators. Several solutions have been proposed to tackle phishing. Nevertheless, there is no single magic bullet that can solve this threat radically. One of the promising techniques that can be used in predicting phishing attacks is based on data mining. Particularly the "induction of classification rules", since anti-phishing solutions aim to predict the website type accurately and these exactly fit the classification data mining. In this paper, we shed light on the important features that distinguish phishing websites from legitimate ones and assess how rule-based classification data mining techniques are applicable in predicting phishing websites

## **REFERENCES**

- [1] Jain, A.K.; Gupta, B. Comparative analysis of features based machine learning approaches for phishing detection; pp. 2125–2130
- [2] Lee, L.H.; Lee, K.C.; Chen, H.H.; Tseng, Y.H. Poster: Proactive blacklist update for anti-phishing; pp. 1448–1450.
- [3] Rao, R.S.; Ali, S.T. Phishshield: A desktop application to detect phishing webpages through heuristic approach. *Procedia Comput. Sci.* 2015, 54, 147–156.
- [4] Zhang, Y.; Hong, J.I.; Cranor, L.F. Cantina: A content-based approach to detecting phishing web sites; pp. 639–648.
- [5] Mohammad, R.M.; Thabtah, F.; McCluskey, L. Predicting phishing websites based on self-structuring neural network. *Neural Comput. Appl.* 2014, 25, 443–458.
- [6] Lin, Y.; Liu, R.; Divakaran, D.M.; Ng, J.Y.; Chan, Q.Z.; Lu, Y.; Si, Y.; Zhang, F.; Dong, J.S. Phishpedia: A Hybrid Deep Learning Based Approach to Visually Identify Phishing Webpages.
- [7] [www.kaggle.com](http://www.kaggle.com)
- [8] <https://archive.ics.uci.edu/ml/datasets/phishing+websites>
- [9] [https://www.researchgate.net/publication/328541785\\_Phishing\\_Website\\_Detection\\_using\\_Machine\\_Learning\\_Algorithms](https://www.researchgate.net/publication/328541785_Phishing_Website_Detection_using_Machine_Learning_Algorithms)
- [10] Varshney G, Misra M, Atrey PK. A phish detector using lightweight search features. *Computers & Security* 2016; 62:213-228

- [11] Xiang, G.; Hong, J.; Rose, C.P.; Cranor, L. Cantina+ a feature-rich machine learning framework for detecting phishing web sites. *ACM Trans. Inf. Syst. Secur.* 2011, 14, 1–28.
- [12] Lee, L.H.; Lee, K.C.; Chen, H.H.; Tseng, Y.H. Poster: Proactive blacklist update for anti-phishing. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, Scottsdale, AZ, USA, 3–7 November 2014; pp. 1448–1450.
- [13] Chiew KL, Chang HH, Sze SN, Tiong WK. Utilisation of website logo for phishing detection. *Computers & Security* 2015 ; 54:16-26.
- [14] Hadi, W.; Aburub, F.; Alhawari, S. A new fast associative classification algorithm for detecting phishing websites. *Appl. Soft Comput.* 2016, 48, 729–734
- [15] Lin, Y.; Liu, R.; Divakaran, D.M.; Ng, J.Y.; Chan, Q.Z.; Lu, Y.; Si, Y.; Zhang, F.; Dong,

J.S. Phishpedia: A Hybrid Deep Learning Based Approach to Visually Identify Phishing Webpages. In *Proceedings of the 30th {USENIX} Security Symposium ({USENIX} Security 21)*, Virtual Event, 11–13 August 2021.

- [16] Chen CS, Su SA, Hung YC. Protecting computer users from online frauds, to Google Patents, 2011.
- [17] Gastellier-Prevost S, Granadillo GG, Laurent M. A dual approach to detect pharming attacks at the client-side. In *New Technologies, Mobility and Security (NTMS)*, 2011 4th IFIP International Conference on, 2011; 1-5.
- [18] He Y, Zhenyu Z, Krasser S, Tang Y. Mining DNS for malicious domain registrations. In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, 2010 6th International Conference on, 2010; 1-6.
- [19] Ferolin RJ, Kang C-U. Phishing attack detection, classification and proactive prevention using fuzzy logic and data mining algorithm, *onlinepresent.org*, 12, 2012, 2012.

















