WEB PHISHING DETECTION

Team ID : PNT2022TMID35745

Submitted By

NITHISH KUMAR G S

RAMANUJAN R

SRINIVASAN A

VISHNU B M

MADRAS INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY

INTRODUCTION

1.1 PROJECT OVERVIEW

The objective of phishing website URLs is to purloin the personal information like user name, passwords and online banking transactions. Phishers use websites which are visually and semantically similar to those real websites. As technology continues to grow, phishing techniques started to progress rapidly and this needs to be prevented by using anti-phishing mechanisms to detect phishing. Machine learning is a powerful tool used to strive against phishing attacks.

This paper surveys the features used for detection and detection techniques using machine learning. Phishing has become a main area of concern for security researchers because it is not difficult to create fake websites which look so close to legitimate websites. Experts can identify fake websites but not all the users can identify the fake website and such users become the victim of phishing attacks.

1. Identify the criteria that can recognize fake URLs
2. Build a decision tree that can iterate through the criteria
3. Train our model to recognize fake vs real URLs
4. Evaluate our model to see how it performs
5. Check for false positives/negatives

CHAPTER 2

LITERATURE SURVEY

1. URL-based Phishing Websites Detection via Machine Learning 2021 - IEEE - International Conference on Data Analytics for Business and Industry (ICDABI)

Phishing is a social engineering cybersecurity attack that involves an attacker who provides a counterfeit piece of information that is hand-crafted skillfully to trick a user (human victim usually) to provide sensitive information to the attacker or to install malicious software on the victim's computing platform. The system developed in this paper for phishing websites is detected using URL addresses and solves binary classification problems where websites are classified into either authentic or phishing websites. The system utilized machine learning techniques such as shallow neural networks and decision trees to learn data patterns in website URLs. The system has been evaluated on the Phishing Websites Dataset which includes small (balanced-class) and large (unbalanced-class) datasets. Aimed at improving the computational efficiency of our system by providing an optimized implementation using fast machine learning frameworks built using low-level programming languages.

2. A Machine Learning Approach for URL Based Web Phishing Using Fuzzy Logic as Classifier 2019 - International Conference on Communication and Electronics Systems (ICCES)

Phishing is the major problem of the internet era. In this era of the internet the security of our data on the web is gaining an increasing importance. Phishing is one of the most harmful ways to unknowingly access the credential information like username, password or account number from the users. Users are not aware of this type of attack and later they will also become a part of the phishing attacks. It may be the losses of financial funds,personal information,reputation of brand name or trust of brand.So the detection of phishing sites is necessary. In this paper we design a framework of phishing detection using URL.

3. Detecting Phishing Websites Using Machine Learning 2019 - 2nd International Conference on Computer Applications & Information Security (ICCAIS)

Phishing websites are one of the internet security problems that target human vulnerabilities rather than software vulnerabilities. It can be described as the process of attracting online users to obtain their sensitive information such as usernames and passwords. In this paper, we offer an intelligent system for detecting phishing websites. The system acts as an additional functionality to an internet browser as an extension that automatically notifies the user when it detects a phishing website. The system is based on a machine learning method, particularly supervised learning. We have selected the Random Forest technique due to its good performance in classification. Our focus is to pursue a higher performance classifier by studying the features of phishing websites and choosing the better combination of them to train the classifier.

4. A Methodical Overview on Detection, Identification and Proactive Prevention of Phishing Websites 2021 - Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV).

This project is a natural way to deal with quality variables. It will present an approach to solving the fuzziness in the phishing website assessment and propose an accurate and smart model for detecting phishing websites. This phishing detection technique is based on fuzzy logic and machine learning algorithms in order to distinguish different factors on the phishing website. According to their study, for phishing identification with greater precision, a total of 30 characteristics and phishing site variables can also be used. A real-time phishing dataset is used which is downloaded from the UCI machine learning repository. This approach is based on smooth logic and machine learning algorithms that define various factors on the phishing website.

5. Feature extraction and classification phishing websites based on URL 2015 - IEEE Conference on Communications and Network Security (CNS)

Phishing is a malicious form of online theft that aims at stealing users' personal information, such as online banking passwords, credit card numbers and other financial data. In the last decade, many users suffered monetary losses as a result of the increasing number of phishing attacks. The motivation of our study is to

propose a safer framework for detecting phishing websites with high accuracy in less time. The detection of phishing can be achieved by either increasing user awareness or using software based detection. Although there are several software detection techniques that address the problem of phishing detection, phishing has become more and more complicated and sophisticated, and can bypass the filter set by anti-phishing techniques. In this study, we extracted more URL features and analyzed subset based feature selection methods which have not been used previously for the purpose of phishing websites detection based on URL.

REFERENCES:

1. URL-based Phishing Websites Detection via Machine Learning |

IEEE Conference Publication | IEEE Xplore

2. A Machine Learning Approach for URL Based Web Phishing Using Fuzzy Logic as Classifier | IEEE Conference Publication | IEEE Xplore

3.IEEE Xplore - Conference Table of Contents

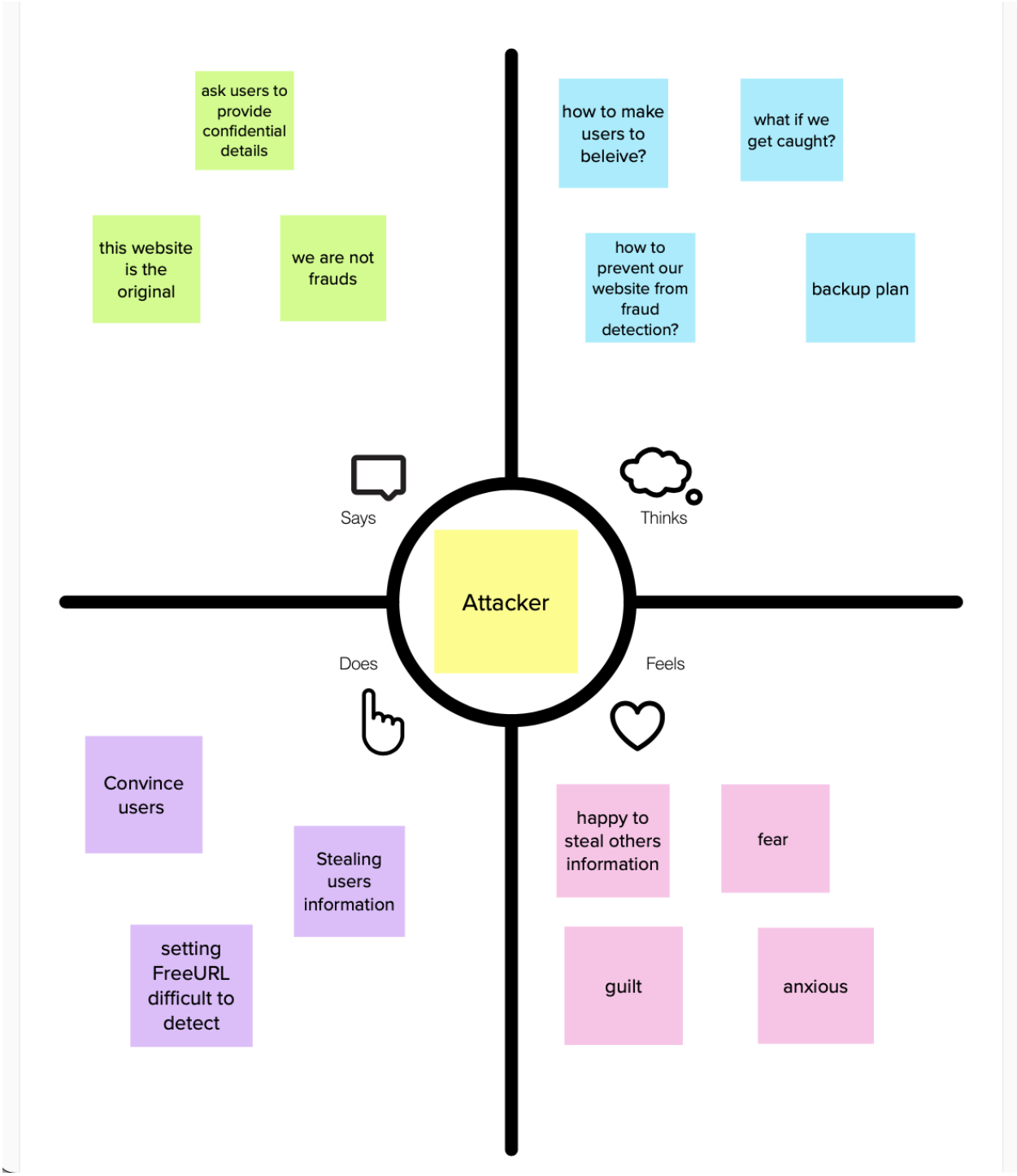
4.A Methodical Overview on Detection, Identification and Proactive Prevention of Phishing Websites | IEEE Conference Publication | IEEE Xplore

5. Feature extraction and classification phishing websites based on URL | IEEE Conference Publication | IEEE Xplore

CHAPTER 3
IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas:

An empathy map is a collaborative tool teams can use to gain a deeper insight into their customers. Much like a user persona, an empathy map can represent a group of users, such as a customer segment. Empathy maps should be used throughout any UX process to establish common ground among team members and to understand and prioritize user needs. In user-centered design, empathy maps are best used from the very beginning of the design process.

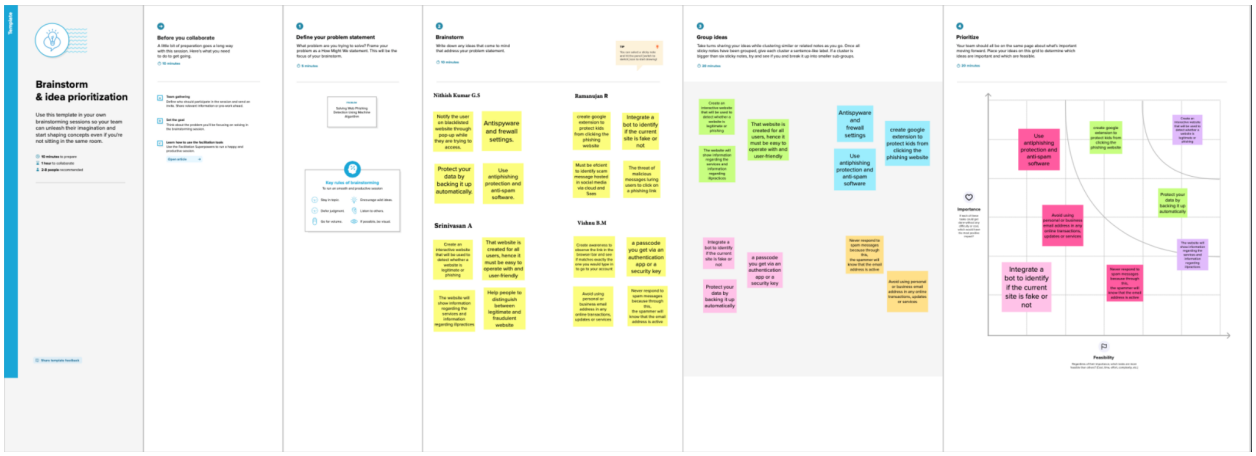


3.2 Ideation & Brainstorming:

Ideation essentially refers to the whole creative process of coming up with and communicating new ideas. Ideation is innovative

thinking, typically aimed at solving a problem or providing a more efficient means of doing or accomplishing something.

Ideation is often closely related to the practice of brainstorming, a specific technique that is utilized to generate new ideas. A principal difference between ideation and brainstorming is that ideation is commonly more thought of as being an individual pursuit, while brainstorming is almost always a group activity.



PROPOSED SOLUTION:-

S.NO	PARAMETER	DESCRIPTION
1	Problem Statement(Problem to be solved)	There are a number of users who purchase products online and make payments through e-banking. There are e-banking websites that ask users to provide sensitive data such as username, password & credit card details etc., often for malicious reasons. This type of e-banking website

		is known as a phishing website. Web service is one of the key communications software services for the internet.
2	Idea/Solution description	Inorder to detect and predict e-banking phishing websites, we proposed an effective system using Machine Learning and data mining techniques like classification algorithms. This system gives user like a warning signal to notify these phishing websites. It helps them to safeguard their identities and their login credentials.
3	Novelty/Uniqueness	Most of the projects only identify the phishing websites. Here we not only identify them but also automatically block these kinds of websites completely in future and also block some mails/ads from these malicious websites.
4	Social Impact/Customer satisfaction	The web phishing detection project attains customer satisfaction by discarding various kinds of malicious websites to protect their privacy. This project does not only support single users but also supports a large social community.It can also help an organization to protect their privacy.
5	Business Model(Revenue Model)	This developed model helps the banking sector as it secures the legitimate website from other malware that are set by hackers. It can also be used by enterprise applications by organizations which handle sensitive information and also can

		be sold to government agencies to prevent the loss of potentially important data.
6	Scalability of the solution	This model provides many capabilities to the user without reducing its efficiency to detect the malicious websites. It is a user-friendly model. The performance rate will also be high.

3.4 Problem Solution fit

The Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realized for it solves the customer's problem. It helps entrepreneurs, marketers and corporate innovators identify behavioral patterns and recognize what would work and why.

<table>
<tr><td rowspan="2">Define CS, fit into CC</td><td>1. CUSTOMER SEGMENT(S)
Ecommerce customers.
Online transaction users.</td><td>6. CUSTOMER CONSTRAINTS
Lack of awareness.
No age limit.</td><td>5. AVAILABLE SOLUTIONS
Antivirus software.
Get free anti-phishing add-ons.
Install firewalls.</td><td rowspan="2">Explore AS, differentiate</td></tr>
<tr><td>2. JOBS-TO-BE-DONE / PROBLEMS
Warn users when malicious URLS are detected.</td><td>9. PROBLEM ROOT CAUSE
Hackers trying to steal sensitive information for making money.</td><td>7. BEHAVIOUR
Rotate passwords regularly.
Don't give out information unnecessarily.</td></tr>
<tr><td rowspan="2">Focus on J&P, tap into BE, understand RC</td><td>3. TRIGGERS
Loss of money.
Data loss.</td><td rowspan="2">10. YOUR SOLUTION
Using Machine learning and data mining techniques to detect web phishing attacks.</td><td rowspan="2">8. CHANNELS of BEHAVIOUR
8.1 ONLINE
Use secure websites from prior knowledge.
8.2 OFFLINE
Take legal actions by filing a police complaint.</td><td rowspan="2">Focus on J&P, tap into BE, understand RC</td></tr>
<tr><td>Identify Strong TR & EM</td><td>4. EMOTIONS: BEFORE / AFTER
Frustrated.
Annoyed.</td><td>Identify Strong TR & EM</td></tr>
</table>

Purpose:

- ❑ Solve complex problems in a way that fits the state of your customers.
- ❑ Succeed faster and increase your solution adoption by

tapping into existing mediums and channels of behavior.

❑ Sharpen your communication and marketing strategy with the right triggers and messaging.

❑ Increase touchpoints with your company by finding the right problem-behavior fit and building trust by solving frequent annoyances, or urgent or costly problems.

❑ Understand the existing situation in order to improve it for your target group.

CHAPTER 4

REQUIREMENT ANALYSIS

4.1 Functional requirements:

FR No.	Functional Requirement (Epic)	Description
FR-1	User Input	User inputs an URL in the form to check whether it is a malicious website.
FR-2	Website comparison	The model compares the given URL with the list of phishing URLs present in the database.
FR-3	Feature Extraction	If it is found none on the comparison it extracts the HTML and domain-based features from the URL.
FR-4	Prediction	The model predicts the URL using machine Learning algorithms such as

		Random Forest technique.
FR-5	Classifier	Model then sends the output to the classifier and produces the result.
FR-6	Announcement	The model finally displays whether the given URL is phishing or not.

4.2 Non-functional requirements:

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	It is an easy to use and access interface which results in greater efficiency.
NFR-2	Security	It is a secure website which protects the sensitive information of the user and prevents malicious attacks.
NFR-3	Reliability	The system can detect phishing websites with greater accuracy using ML algorithms.
NFR-4	Performance	The system produces responses within seconds and execution is faster.
NFR-5	Availability	Users can access the website via any browser from anywhere at any time.
NFR-6	Scalability	This application can be accessed online without paying. It can detect any web site with high accuracy.

CHAPTER 5

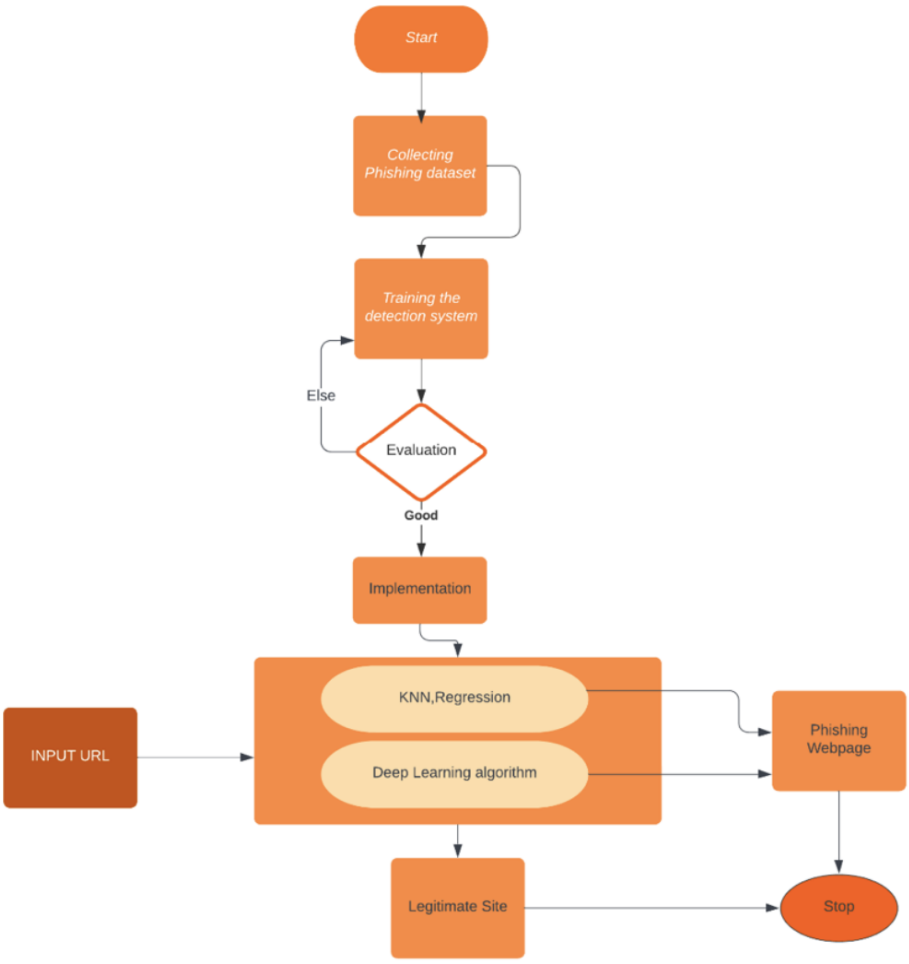
PROJECT DESIGN

5.1 Data Flow diagram:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

DFD level 0

DFD Level 0:



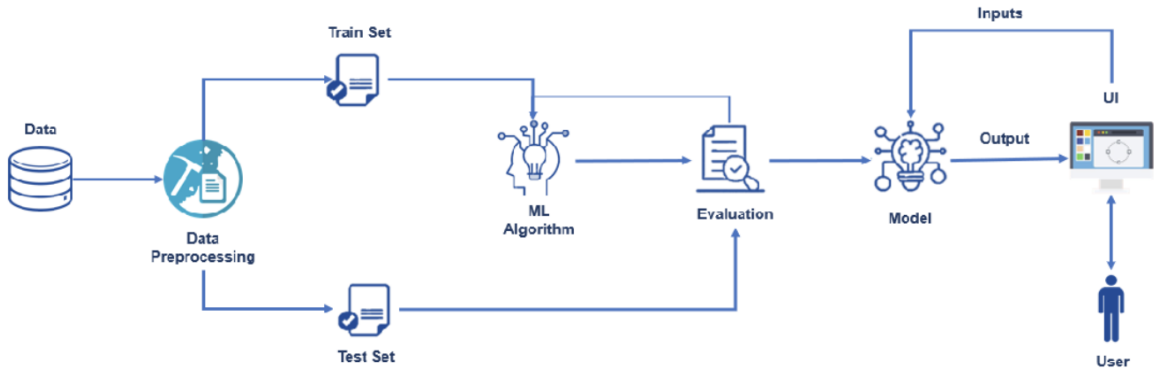
5.2 Solution & Technical Architecture:

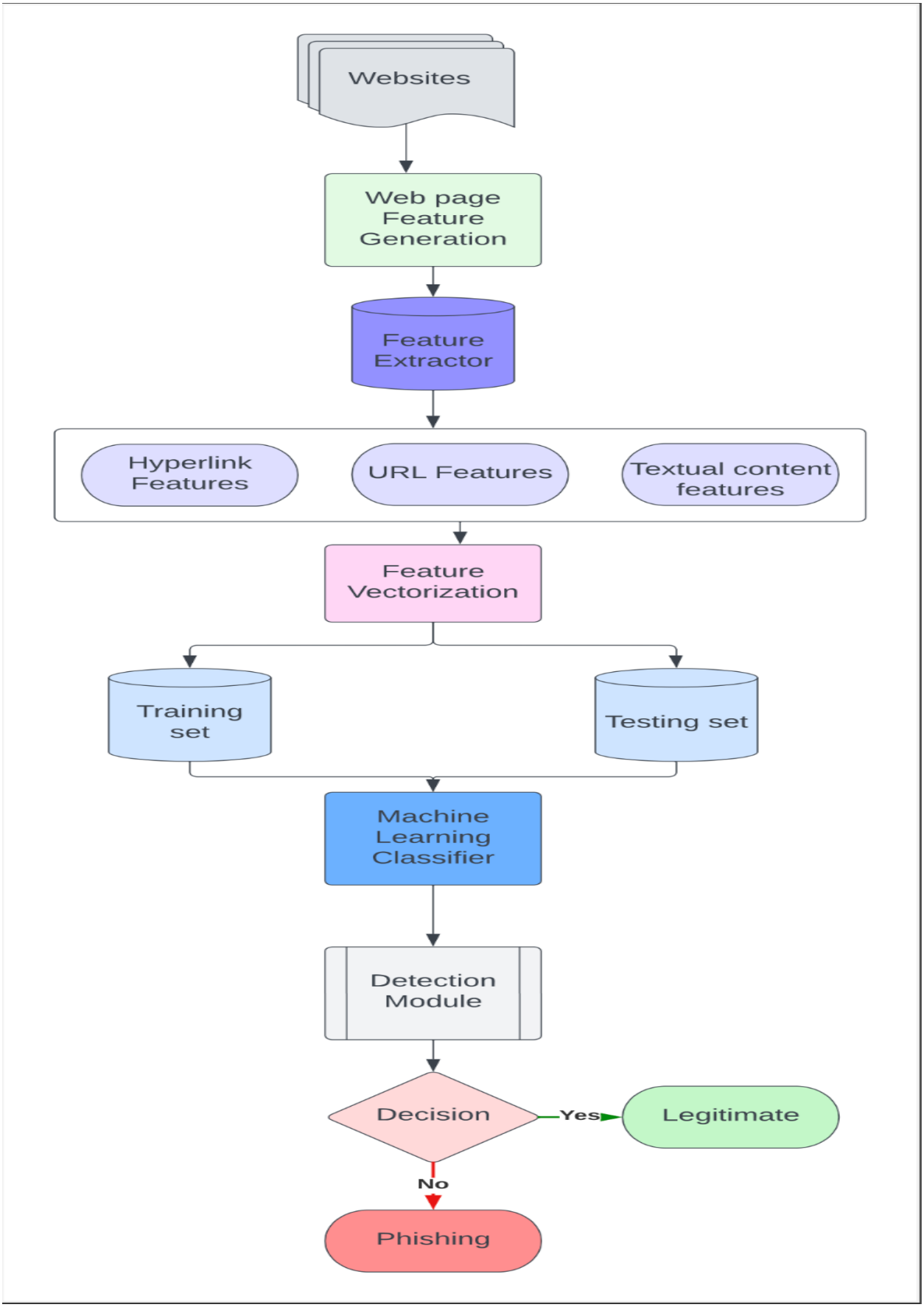
SOLUTION:

Our solution is to build an efficient and intelligent system to detect phishing sites by applying a machine learning algorithm which implements classification algorithms and techniques to extract the phishing datasets criteria to classify their legitimacy by carefully analyzing and identifying various factors that could be used to detect a phishing site. These factors fall under the categories of address bar-based features, domain-based features, HTML & JavaScript based features. Using these features, we can identify a phishing site with high accuracy.

TECHNICAL ARCHITECTURE:

Technical architecture which is also often referred to as application architecture includes the major components of the system, their relationships, and the contracts that define the interactions between the components. The goal of technical architects is to achieve all the business needs with an application that is optimized for both performance and security.




Websites
Web page Feature Generation
Feature Extractor
Hyperlink Features
URL Features
Textual content features
Feature Vectorization
Training set
Testing set
Machine Learning Classifier
Detection Module
Decision
Yes
Legitimate
No
Phishing

CHAPTER 6

PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority
Sprint-1	Login	USN-1	As a user, I can navigate into the website.	1	High
Sprint-1	Dashboard	USN-2	As a user, I will input any site's URL in the form to check its genuineness.	1	High
Sprint-1		USN-3	As a user, I can see the output.	2	High
Sprint-2	Backend	USN-4	As an admin, if a new URL is found, I can add the new state into the database.	3	Medium

Sprint-3	Report	USN-5	As a user, I can ask my queries and report suspicious sites in the report box.	1	Low
Sprint-4		USN-6	As an admin, I can take actions to the queries asked by the user.	2	Low

6.2 Sprint Delivery Schedule:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

CHAPTER 7

CODING & SOLUTIONING

7.1 Feature 1 – Classification of URL:

The primary feature of this project is to classify the given URL as phishing or benign. Various classification algorithms are used to achieve this.

7.1.1 Methodology:

7.1.1.1 Data collection:

URL features of legitimate websites and phishing websites were collected. The data set consists of 11,055 URLs which include 6,157 legitimate URLs and 4,898 phishing URLs. Legitimate URLs are labeled as “1” and phishing URLs are labeled as “-1”. The features that are present in the data set include:

· IP Address in URL
· Length of URL
· Using URL Shortening Services
· "@" Symbol in URL
· Redirection "//" in URL
· Prefix or Suffix "-" in Domain
· Having Sub Domain
· Length of Domain Registration
· Favicon
· Port Number
· HTTPS Token
· Request URL
· URL of Anchor
· Links in Tags
· SFH
· Email Submission
· Abnormal URL
· Status Bar Customization (on mouse over)
· Disabling Right Click

· Presence of Popup Window
· IFrame Redirection
· Age of Domain
· DNS Record
· Web Traffic
· Page Rank
· Google Index
· Links pointing to the page
· Statistical Report
· Result

Using IBM Cloud Storage this data is accessed throughout the project. The code written below is used to import the dataset.

```
import os, types

import pandas as pd

from botocore.client import Config

import ibm_boto3

def iter (self): return 0

# The following code accesses a file in your IBM Cloud Object
Storage. It includes your credentials.

# You might want to remove those credentials before you share
the notebook.

cos_client = ibm_boto3.client(service_name='s3',

    ibm_api_key_id='',

    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",

    config=Config(signature_version='oauth'),

    endpoint_url='https://s3.private.us.cloud-object
storage.appdomain.cloud')


bucket ='webphishingdetection-donotdelete-pr-icmjtvktnzli2s'



object_key = 'dataset_website.csv'
```

```
body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']

    # add missing iter method, so pandas accepts body as
    file-like object

    if not hasattr(body, " iter "): body. iter =
      types.MethodType( iter , body )


    data0 = pd.read_csv(body)

    data0.head()
```

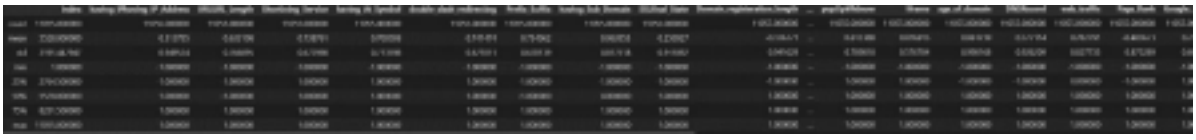
7.1.1.2 Data pre-processing and Exploratory Data Analysis:

Few plots and graphs were drawn to find how the data is distributed and how features are related to each other.

Univariate analysis:

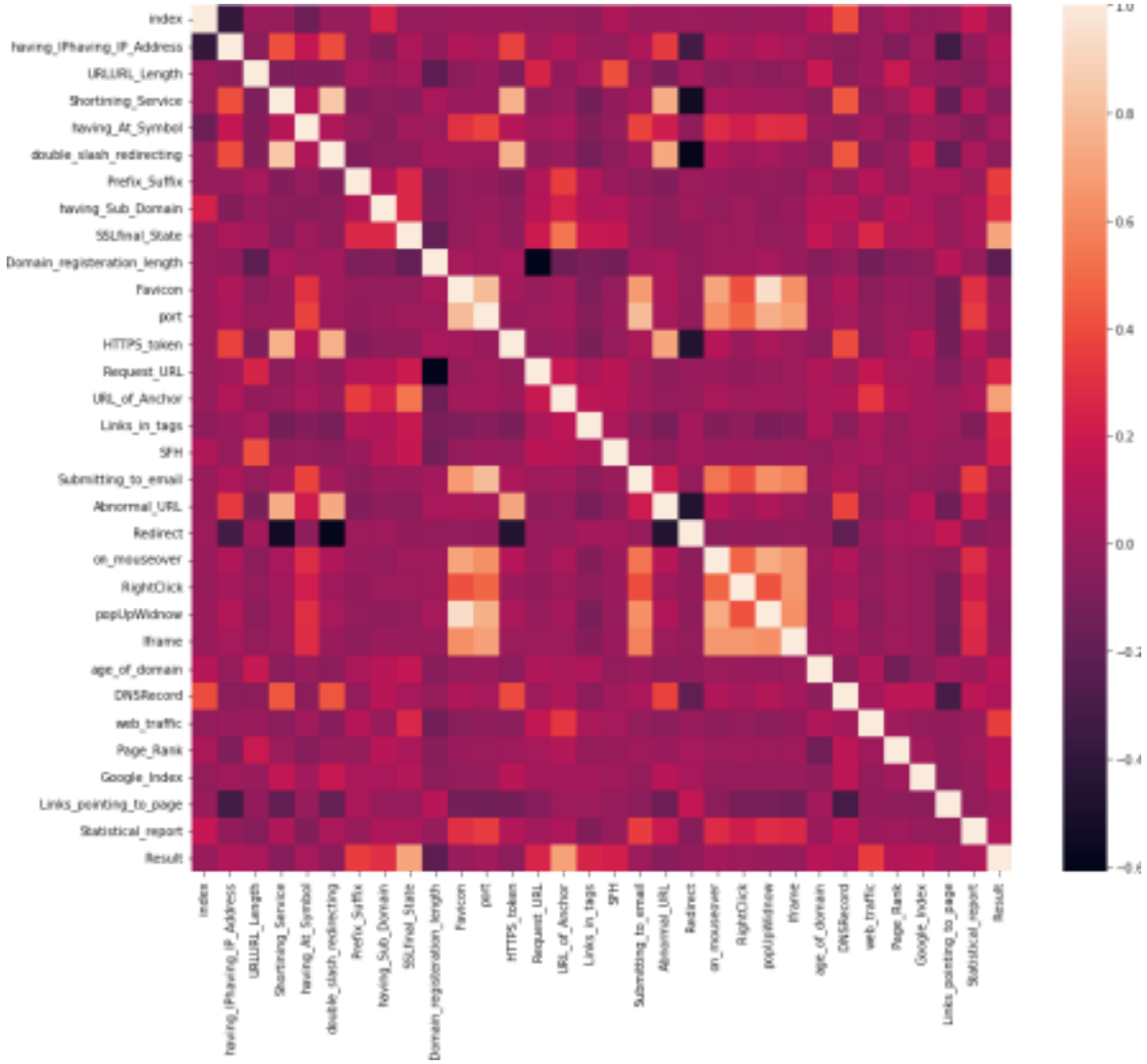
Univariate analysis provides an understanding of the characteristics of each feature in the data set. Different characteristics are computed for numerical and categorical data. For the numerical features characteristics are standard deviation, skewness, kurtosis, percentile, interquartile range (IQR) and range. For the categorical features characteristics are count, cardinality, list of unique values, top and freq.

```
data0.describe()
```



Bivariate analysis:

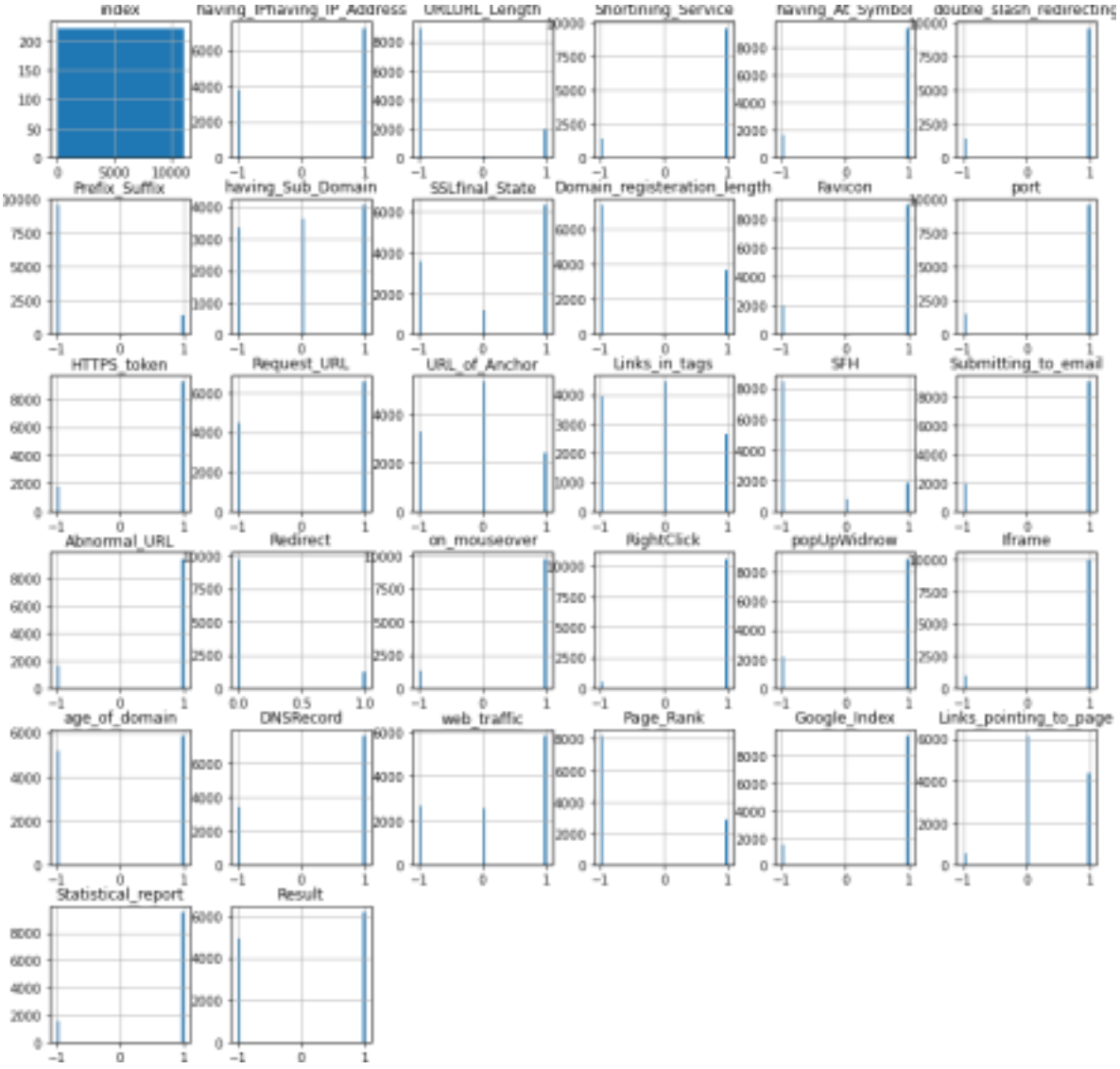
```
plt.figure(figsize=(15,13))

sns.heatmap(data0.corr())

plt.show()
```



From this correlation matrix, it is evident that there is no correlation with many features. So, it is crucial to eliminate these features.

Multivariate analysis:

```
data0.hist(bins = 50,figsize = (15,15))
plt.show()
```



From data distribution graph and correlation matrix, we can conclude that the following features do not have much impact on the result:

· having_Sub_Domain

· Domain_registeration_length

· Favicon

· Request_URL

· URL_of_Anchor

· Links_in_tags

· Submitting_to_email

· Redirect

· web_traffic

· Page_Rank

· Google_Index

· Links_pointing_to_page

All the above features will not be included in further processing.

```
#Removing the features which do not have much impact on
Result
data=data0.iloc[:,[1,2,3,4,5,6,12,20,21,22,23,24,25,30,31]]
data.head()
```

Checking for null values:

This dataset doesn't contain any null values.

```
#checking the data for null or missing values
data.isnull().sum()
```

```
having_IPhaving_IP_Address    0
URLURL_Length                 0
Shortining_Service            0
having_At_Symbol              0
double_slash_redirecting      0
Prefix_Suffix                 0
HTTPS_token                   0
on_mouseover                  0
RightClick                    0
popUpWidnow                   0
Iframe                        0
age_of_domain                 0
DNSRecord                     0
Statistical_report            0
Result                        0
dtype: int64
```

7.1.1.3 Model building:

From the dataset above, it is clear that this is a supervised machine learning task. There are two major types of supervised machine learning problems, called classification and regression.
This data set comes under classification problems, as the input URL is classified as phishing (-1) or legitimate (1). The supervised machine learning models (classification) considered to train the dataset in this notebook are:

· XGBoost

· Decision Tree

· Random Forest

· Support Vector Machines

XGBoost:

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.

```
#XGBoost Classification model
from xgboost import XGBClassifier

import warnings
warnings.filterwarnings("ignore", category=UserWarning)

# instantiate the model
xgb = XGBClassifier(learning_rate=0.4,max_depth=7,verbosity =
0) #fit the model
xgb.fit(X_train, y_train)
#predicting the target value from the model for the samples
y_test_xgb = xgb.predict(X_test)
y_train_xgb = xgb.predict(X_train)
#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))
```

Decision Tree Classifier:

Decision trees are widely used models for classification and regression tasks. Essentially, they learn a hierarchy of if/else questions, leading to a decision. Learning a decision tree means learning the sequence of if/else questions that gets us to the true answer most quickly.

In the machine learning setting, these questions are called tests (not to be confused with the test set, which is the data we use to test to see how generalizable our model is). To build a tree, the algorithm searches over all possible tests and finds the one that is most informative about the target variable.

```
# Decision Tree model
from sklearn.tree import DecisionTreeClassifier


# instantiate the model
tree = DecisionTreeClassifier(max_depth = 5)
# fit the model
tree.fit(X_train, y_train)
#predicting the target value from the model for the samples
y_test_tree = tree.predict(X_test)
y_train_tree = tree.predict(X_train)
#computing the accuracy of the model performance
acc_train_tree = accuracy_score(y_train,y_train_tree)
acc_test_tree = accuracy_score(y_test,y_test_tree)


print("Decision Tree: Accuracy on training Data:
{:.3f}".format(acc_train_tree))
print("Decision Tree: Accuracy on test Data:
{:.3f}".format(acc_test_tree))
```

Random Forest Classifier:

Random forests for regression and classification are currently among the most widely used machine learning methods. A random forest is essentially a collection of decision trees, where each tree is slightly different from the others. The idea behind random forests is that each tree might do a relatively good job of predicting, but will likely overfit on part of the data.

If we build many trees, all of which work well and overfit in different ways, we can reduce the amount of overfitting by averaging their results. To build a random forest model, you need to decide on the number of trees to build (the n_estimators parameter of RandomForestRegressor or RandomForestClassifier). They are very powerful, often work well without heavy tuning of the parameters, and

don’t require scaling of the data.

```
# Random Forest model
from sklearn.ensemble import RandomForestClassifier
# instantiate the model
forest = RandomForestClassifier(max_depth=5)
# fit the model
forest.fit(X_train, y_train)
#predicting the target value from the model for the samples
y_test_forest = forest.predict(X_test)
y_train_forest = forest.predict(X_train)
#computing the accuracy of the model performance
acc_train_forest = accuracy_score(y_train,y_train_forest)
acc_test_forest = accuracy_score(y_test,y_test_forest)


print("Random forest: Accuracy on training Data:
{:.3f}".format(acc_train_forest))
print("Random forest: Accuracy on test Data:
{:.3f}".format(acc_test_forest))
```

Support Vector Machines:

In machine learning, support-vector machines (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.

```
#Support vector machine model
from sklearn.svm import SVC
```

```
# instantiate the model
svm = SVC(kernel='linear', C=1.0, random_state=12)
#fit the model
svm.fit(X_train, y_train)
#predicting the target value from the model for the samples
y_test_svm = svm.predict(X_test)
y_train_svm = svm.predict(X_train)
#computing the accuracy of the model performance
acc_train_svm = accuracy_score(y_train,y_train_svm)
acc_test_svm = accuracy_score(y_test,y_test_svm)

print("SVM: Accuracy on training Data:
{:.3f}".format(acc_train_svm)) print("SVM : Accuracy on test
Data: {:.3f}".format(acc_test_svm))
```

7.1.2 User interface:

The user opens the site and inputs a URL to check its legitimacy. Necessary features are extracted from this URL and predictions are made.

7.1.2.1 Feature extraction:

We will extract the 13 features that we used to train our model.

IP Address in URL:

Checks for the presence of an 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 the URL has an IP address, the value assigned to this feature is -1 (phishing) or else 1 (legitimate).

```
def having_IPhaving_IP_Address(self):
        try:
            ipaddress.ip_address(self.url)
            return -1
        except:
            return 1
```

Length of URL:

Computes the length of the URL. Phishers can use long URL to hide the doubtful part in the address bar. In this project, if the length of the URL is greater than or equal 54 characters then the URL is classified as phishing otherwise legitimate.

If the length of URL >= 54, the value assigned to this feature is -1 (phishing) or else 1 (legitimate).

```
def URLURL_Length(self):
   if len(self.url) < 54:
      return 1
   else:
      return -1
```

Using URL Shortening Services:

URL shortening is a method on the “World Wide Web” in which a URL may be made considerably smaller in length and still lead to the required webpage. This is accomplished by means of an “HTTP Redirect” on a domain name that is short, which links to the webpage that has a long URL.

If the URL is using Shortening Services, the value assigned to this feature is -1 (phishing) or else 1 (legitimate).

```
def Shortining_Service(self):
    shortening_services =
r"bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|t
r\.im|is\.gd|cli\.gs|" \
r"yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twu
rl\.nl|snipurl\.com|" \
r"short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr
\.com|fic\.kr|loopt\. us|" \
r"doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit
```

```
\.do|t\.co|lnkd\.in| db\.tt|" \
r"qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit
\.ly|ity\.im|q\.gs|is\.gd| " \
r"po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.b
b|yourls\.org|x\.co|" \
r"prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1
url\.com|tweez\.me|v \.gd|" \
          r"tr\.im|link\.zip\.net"
    match=re.search(shortening_services,self.url)
    if match:
      return -1
    else:
      return 1
```

"@" 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 1 (legitimate).

```
def having_At_Symbol(self):
  if "@" in self.url:
     return -1
  else:
     return 1
```

Redirection "//" in URL:

Check 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 the 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 1 (legitimate).

```
def double_slash_redirecting(self):
    pos = self.url.rfind('//')
    if pos > 6:
      if pos > 7:
        return -1
```

```
        else:
          return 1
      else:
        return 1
```

Prefix or Suffix "-" in Domain:

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

```
def Prefix_Suffix(self):
    if '-' in urlparse(self.url).netloc:
      return -1
    else:
      return 1
```

HTTPS Token:

Checks for the presence of "http/https" in the domain part of the URL. The phishers may add the “HTTPS” token to the domain part of a URL in order to trick users.

If the URL has "http/https" in the domain part, the value assigned to this feature is -1 (phishing) or else 1 (legitimate).

```
def HTTPS_token(self):
    domain = urlparse(self.url).netloc
    if 'https' in domain:
      return -1
    else:
      return 1
```

Status Bar Customization (on mouse over):

Phishers may use JavaScript to show a fake URL in the status bar to users. To extract this feature, we must dig-out the webpage source code, particularly the “onMouseOver” event, and check if it makes any changes on the status bar. If the response is empty or onmouseover is found then, the value assigned to this feature is -1

(phishing) or else 1 (legitimate).

```
def on_mouseover(self):
    try:
      if re.findall("", self.response.text):
        return -1
      else:
        return 1
    except:
       return -1
```

Disabling Right Click:

Phishers use JavaScript to disable the right-click function, so that users cannot view and save the webpage source code. This feature is treated exactly as “Using onMouseOver to hide the Link”. Nonetheless, for this feature, we will search for the event “event.button==2” in the webpage source code and check if the right click is disabled. If the response is empty or onmouseover is not found then, the value assigned to this feature is -1 (phishing) or else 1 (legitimate).

```
def RightClick(self):
    if self.response == "":
      return -1
    else:
      if re.findall(r"event.button ?== ?2", self.response.text):
        return 1
      else:
        return -1
```

Presence of Popup Window:

Pop up windows are another option used by phishers to redirect users to other pages. They display attractive ads to lure the user to click the link. Nonetheless, for this feature, we will search for an event “alert” in the webpage source code and check if it is present. If the response is empty or alert is not found then, the value assigned to this feature is -1 (phishing) or else 1 (legitimate).

```
def popUpWidnow(self):
    try:
      if re.findall(r"alert\(", self.response.text):
        return 1
      else:
        return -1
    except:
      return -1
```

IFrame Redirection:

IFrame is an HTML tag used to display an additional webpage into one that is currently shown. Phishers can make use of the “iframe” tag and make it invisible i.e. without frame borders. In this regard, phishers make use of the “frameBorder” attribute which causes the browser to render a visual delineation. If the iframe is empty or response is not found then, the value assigned to this feature is -1 (phishing) or else 1 (legitimate).

```
def Iframe(self):
    try:
      if re.findall(r"[<iframe>|<frameBorder>]",
      self.response.text):
        return 1
      else:
        return -1
    except:
      return -1
```

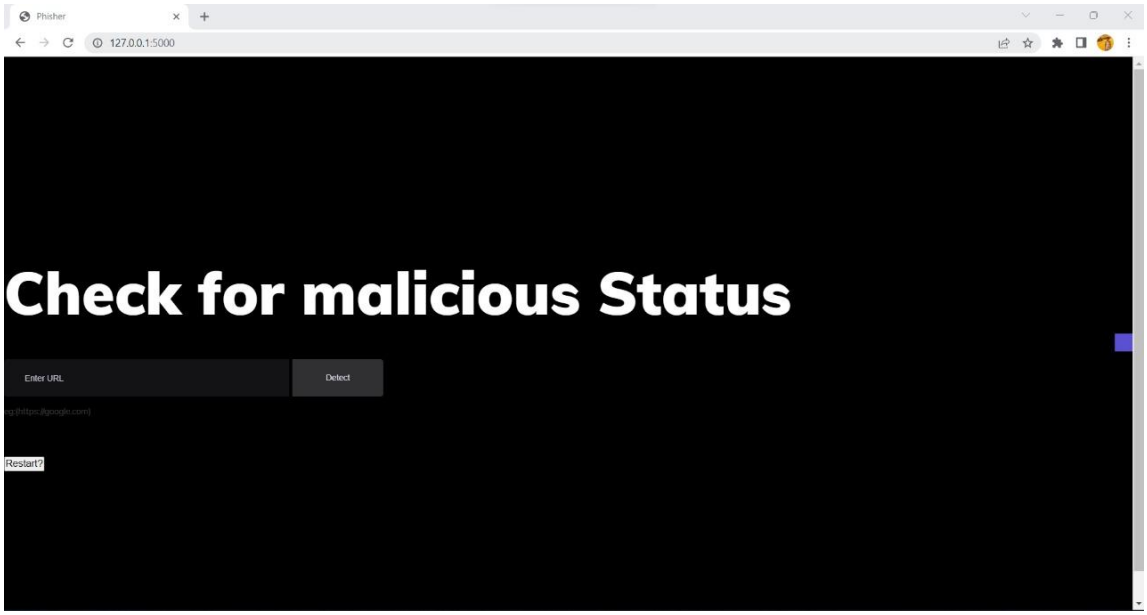
Age of Domain:

This feature can be extracted from the WHOIS database. Most phishing websites live for a short period of time. The minimum age of the legitimate domain is considered to be 12 months for this project. Age here is nothing but different between creation and expiration time. If age of domain > 12 months, the value of this feature is -1 (phishing) else 1 (legitimate).

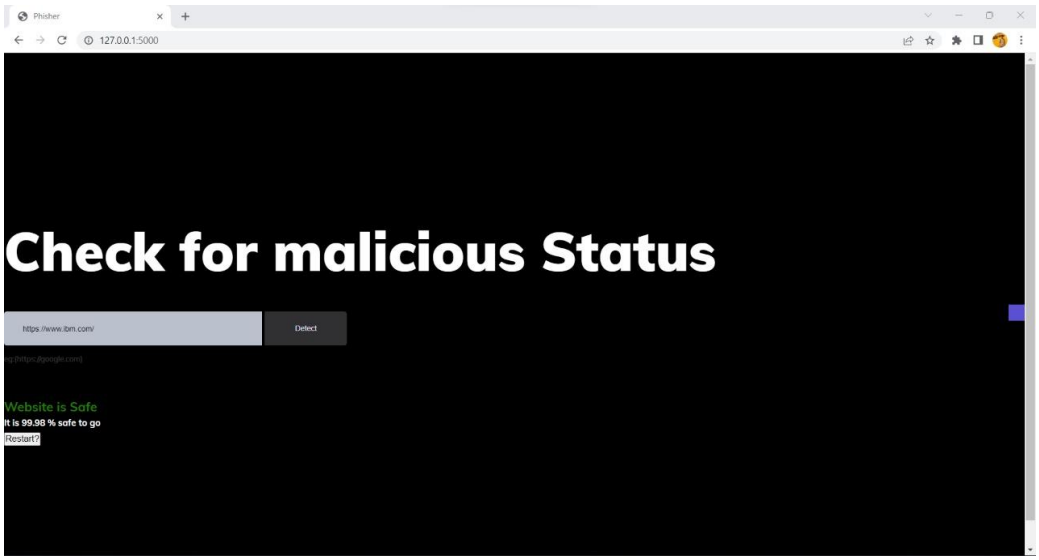
```
def age_of_domain(self):
    creation_date = self.domain_name.creation_date
    expiration_date = self.domain_name.expiration_date
    if (isinstance(creation_date,str) or
    isinstance(expiration_date,str)):
      try:
```

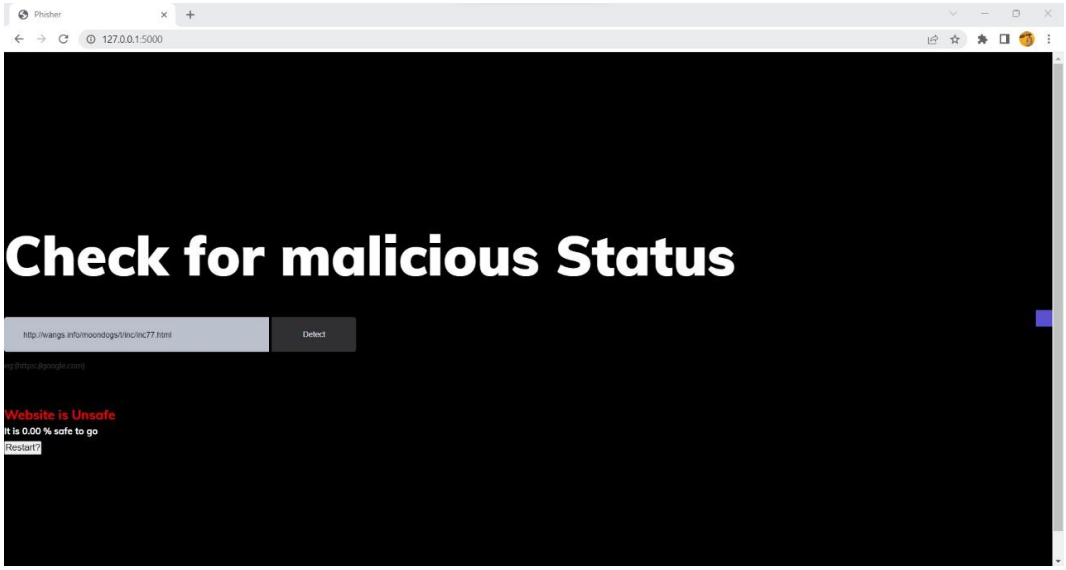
```
        creation_date = datetime.strptime(creation_date,'%Y-%m-%d')
        expiration_date = datetime.strptime(expiration_date,"%Y-%m-%d")
      except:
        return -1
    if ((expiration_date is None) or (creation_date is None)):
      return -1
    elif ((type(expiration_date) is list) or (type(creation_date) is
list)):
```

7.1.2.2 Dashboard



(a) Home Page





(b) Phishing URL result displayed to the user

CHAPTER 8

TESTING

8.1 Test Cases:

Test case ID	Feature Type	Component	Test Scenario	Steps to Execute	Test Data	Expected Result	Actual Result	Status
DashBoard _TC_OO1	Functional	Home Page	Verify user is able to enter the URL in the form	1.Open Hookwebsite 2.Enter a URL and click submit	https://google.com/	Result of classification will be displayed	Working as expected	Pass

DashBoard _TC_OO2	UI	Home Page	Verify the UI elements in the form	1.Enter URL and click go 2.The services and teams' sections are visible 3.Enter a URL and click submit	https://google.com/	Application should show below UI elements: a. input form b. submit button c. services d. team	Working as expected	Pass
DashBoard _TC_OO3	Functional	Home Page	Verify user is able to see an alert when nothing is entered in the textbox	1.Enter URL and click go 2.Enter nothing and click submit 3.An alert is displayed to provide proper input		Alert of incomplete input	Working as expected	Pass
DashBoard _TC_OO4	Functional	Home Page	Verify user is able to see the result when URL is entered in the textbox	1.Enter URL and click go 2.Enter any URL and click submit 3.The result of the classification is displayed.	https://google.com/	Result of classification will be displayed	Working as expected	Pass

8.2 User Acceptance Testing:

Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not	0	0	1	0	1

Reproduced					
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	14	13	26	77

Test Case Analysis:

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	5	0	0	5-
Client Application	51	0	0	51
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

CHAPTER 9

RESULTS

9.1 Performance metrics:

The median efficiency is used to assess each categorization model's effectiveness. The final item will appear in the way it was envisioned. Graphical representations are used to depict information during

classification. The percentage of predictions made using the testing dataset is used to gauge accuracy. By dividing the entire number of forecasts even by properly predicted estimates, it is simple to calculate. The difference between actual and anticipated output is used to calculate accuracy.

$$Accuracy = (TP + TN)/(TP + TN + FP + FN)$$

Where TP = True Positives, TN = True Negatives, FN = False Negatives and FP= False Positives.

Thus, accuracy for all the four used models were calculated and ranked. Gradient Boost Classifier performed better than other models.

Out[83]:

	ML Model	Accuracy	f1_score	Recall	Precision
0	Gradient Boosting Classifier	0.974	0.977	0.994	0.986
1	CatBoost Classifier	0.972	0.975	0.994	0.989
2	Random Forest	0.969	0.972	0.992	0.991
3	Support Vector Machine	0.964	0.968	0.980	0.965
4	Decision Tree	0.958	0.962	0.991	0.993
5	K-Nearest Neighbors	0.956	0.961	0.991	0.989
6	Logistic Regression	0.934	0.941	0.943	0.927
7	Naive Bayes Classifier	0.605	0.454	0.292	0.997
8	XGBoost Classifier	0.548	0.548	0.993	0.984
9	Multi-layer Perceptron	0.543	0.543	0.989	0.983

CHAPTER 10

ADVANTAGES & DISADVANTAGES

ADVANTAGES:

· **Increases user alertness to phishing risks** Whenever the user navigates into the website and provide the URL of the website that needs to be verified for legitimacy, the system detects phishing sites by applying a machine learning algorithm which implements classification algorithms and techniques to extract the phishing datasets criteria to classify their legitimacy which in turn helps the customers to eliminate the risks of cyber threat and protect their valuable corporate or personal data.

· **Users will also be able to pose any query to the admin through the report page designed** Our system is also provided with an option for the clients to report to the administrator which helps them to ask their questions significantly improving their experience on our site.

DISADVANTAGES:

· Not a generalized model
· Huge number of rules
· Needs feed continuously

CHAPTER 11

CONCLUSION

Phishing detection is now an area of great interest among the researchers due to its significance in protecting privacy and providing security. There are many methods to perform phishing detection. Our system aims to enhance the detection method to detect phishing websites using machine learning technology. We achieved a high detection accuracy, and the results show that the classifiers give better performance when we use more data as training data. In future, hybrid technology will be implemented to detect phishing websites more accurately.

CHAPTER-12

FUTURE SCOPE

In the future we intend to build add-ons for our system and if we get a structured dataset of phishing, we can perform phishing detection much faster than any other technique. We can also use a combination of any two or more classifiers to get maximum accuracy. We plan to explore various phishing techniques which use Network based features, Content based features, Webpage based features and HTML and JavaScript features of web pages which will improve the performance of the system. In particular, we extract features from URLs and pass it through the various classifiers.

CHAPTER 13
APPENDIX

13.1 Source code:

app.py

```
#importing required libraries


from flask import Flask, request, render_template
import numpy as np
import pandas as pd
from sklearn import metrics
import warnings
import pickle
warnings.filterwarnings('ignore')
from feature import FeatureExtraction


file = open("model.pkl","rb")
gbc = pickle.load(file)
file.close()



app = Flask(__name__)


@app.route("/", methods=["GET", "POST"])
def index():
    if request.method == "POST":


        url = request.form["url"]
        obj = FeatureExtraction(url)
```

```
        x = np.array(obj.getFeaturesList()).reshape(1,30)


        y_pred =gbc.predict(x)[0]

        #1 is safe

        #-1 is unsafe

        y_pro_phishing = gbc.predict_proba(x)[0,0]

        y_pro_non_phishing = gbc.predict_proba(x)[0,1]

        if(y_pred ==1 ):

            result="Website is Safe"

        else:

            result="Website is Unsafe"

        pred = "It is {0:.2f} % safe to go
".format(y_pro_non_phishing*100)

        return
render_template('index.html',pred=pred,result=result,y_pred=
y_pred)



        return render_template('index.html',xx
=round(y_pro_non_phishing,2),url=url,a=y_pro_phishing,b=y_pr
o_non_phishing,c=y_pred )

    return render_template("index.html", xx =-1)




if __name__ == "__main__":

    app.run(debug=True)
```

feature.py

```
import ipaddress

import re

import urllib.request

from bs4 import BeautifulSoup
```

```
import socket
import requests
from googlesearch import search
import whois
from datetime import date, datetime
import time
from dateutil.parser import parse as date_parse
from urllib.parse import urlparse

class FeatureExtraction:
    features = []
    def __init__(self,url):
        self.features = []
        self.url = url
        self.domain = ""
        self.whois_response = ""
        self.urlparse = ""
        self.response = ""
        self.soup = ""

        try:
            self.response = requests.get(url)
            self.soup = BeautifulSoup(response.text,
'html.parser')
        except:
            pass

        try:
            self.urlparse = urlparse(url)
            self.domain = self.urlparse.netloc
```

```
except:
    pass

try:
    self.whois_response = whois.whois(self.domain)
except:
    pass




self.features.append(self.UsingIp())
self.features.append(self.longUrl())
self.features.append(self.shortUrl())
self.features.append(self.symbol())
self.features.append(self.redirecting())
self.features.append(self.prefixSuffix())
self.features.append(self.SubDomains())
self.features.append(self.Hppts())
self.features.append(self.DomainRegLen())
self.features.append(self.Favicon())


self.features.append(self.NonStdPort())
self.features.append(self.HTTPSDomainURL())
self.features.append(self.RequestURL())
self.features.append(self.AnchorURL())
self.features.append(self.LinksInScriptTags())
self.features.append(self.ServerFormHandler())
self.features.append(self.InfoEmail())
```

```
        self.features.append(self.AbnormalURL())
        self.features.append(self.WebsiteForwarding())
        self.features.append(self.StatusBarCust())

        self.features.append(self.DisableRightClick())
        self.features.append(self.UsingPopupWindow())
        self.features.append(self.IframeRedirection())
        self.features.append(self.AgeofDomain())
        self.features.append(self.DNSRecording())
        self.features.append(self.WebsiteTraffic())
        self.features.append(self.PageRank())
        self.features.append(self.GoogleIndex())
        self.features.append(self.LinksPointingToPage())
        self.features.append(self.StatsReport())


    # 1.UsingIp
    def UsingIp(self):
        try:
            ipaddress.ip_address(self.url)
            return -1
        except:
            return 1

    # 2.longUrl
    def longUrl(self):
        if len(self.url) < 54:
            return 1
        if len(self.url) >= 54 and len(self.url) <= 75:
            return 0
```

```
        return -1


    # 3.shortUrl
    def shortUrl(self):
        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\.c
om|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\.c
om|qr\.net|1url\.com|tweez\.me|v\.gd|tr\.im|link\.zip\.net',
self.url)
        if match:
            return -1
        return 1


    # 4.Symbol@
    def symbol(self):
        if re.findall("@",self.url):
            return -1
        return 1


    # 5.Redirecting//
```

```
def redirecting(self):
    if self.url.rfind('//')>6:
        return -1
    return 1

# 6.prefixSuffix
def prefixSuffix(self):
    try:
        match = re.findall('\-', self.domain)
        if match:
            return -1
        return 1
    except:
        return -1

# 7.SubDomains
def SubDomains(self):
    dot_count = len(re.findall("\.", self.url))
    if dot_count == 1:
        return 1
    elif dot_count == 2:
        return 0
    return -1

# 8.HTTPS
def Hppts(self):
    try:
        https = self.urlparse.scheme
        if 'https' in https:
            return 1
```

```
            return -1
        except:
            return 1


    # 9.DomainRegLen
    def DomainRegLen(self):
        try:
            expiration_date =
self.whois_response.expiration_date
            creation_date =
self.whois_response.creation_date
            try:
                if(len(expiration_date)):
                    expiration_date = expiration_date[0]
            except:
                pass
            try:
                if(len(creation_date)):
                    creation_date = creation_date[0]
            except:
                pass


            age =
(expiration_date.year-creation_date.year)*12+
(expiration_date.month-creation_date.month)
            if age >=12:
                return 1
            return -1
        except:
            return -1


    # 10. Favicon
```

```
    def Favicon(self):
        try:
            for head in self.soup.find_all('head'):
                for head.link in self.soup.find_all('link',
href=True):
                    dots = [x.start(0) for x in
re.finditer('\.', head.link['href'])]
                    if self.url in head.link['href'] or
len(dots) == 1 or domain in head.link['href']:
                        return 1
            return -1
        except:
            return -1

    # 11. NonStdPort
    def NonStdPort(self):
        try:
            port = self.domain.split(":")
            if len(port)>1:
                return -1
            return 1
        except:
            return -1

    # 12. HTTPSDomainURL
    def HTTPSDomainURL(self):
        try:
            if 'https' in self.domain:
                return -1
            return 1
        except:
```

```
            return -1


    # 13. RequestURL

    def RequestURL(self):

        try:

            for img in self.soup.find_all('img', src=True):

                dots = [x.start(0) for x in
re.finditer('\.', img['src'])]

                if self.url in img['src'] or self.domain in
img['src'] or len(dots) == 1:

                    success = success + 1

                i = i+1


            for audio in self.soup.find_all('audio',
src=True):

                dots = [x.start(0) for x in
re.finditer('\.', audio['src'])]

                if self.url in audio['src'] or self.domain
in audio['src'] or len(dots) == 1:

                    success = success + 1

                i = i+1


            for embed in self.soup.find_all('embed',
src=True):

                dots = [x.start(0) for x in
re.finditer('\.', embed['src'])]

                if self.url in embed['src'] or self.domain
in embed['src'] or len(dots) == 1:

                    success = success + 1

                i = i+1


            for iframe in self.soup.find_all('iframe',
src=True):

                dots = [x.start(0) for x in
```

```
re.finditer('\.', iframe['src'])]
                if self.url in iframe['src'] or self.domain
in iframe['src'] or len(dots) == 1:
                    success = success + 1
                i = i+1

            try:
                percentage = success/float(i) * 100
                if percentage < 22.0:
                    return 1
                elif((percentage >= 22.0) and (percentage <
61.0)):
                    return 0
                else:
                    return -1
            except:
                return 0
        except:
            return -1

    # 14. AnchorURL
    def AnchorURL(self):
        try:
            i,unsafe = 0,0
            for a in self.soup.find_all('a', href=True):
                if "#" in a['href'] or "javascript" in
a['href'].lower() or "mailto" in a['href'].lower() or not
(url in a['href'] or self.domain in a['href']):
                    unsafe = unsafe + 1
                i = i + 1

            try:
```

```
                percentage = unsafe / float(i) * 100
                if percentage < 31.0:
                    return 1
                elif ((percentage >= 31.0) and (percentage <
67.0)):
                    return 0
                else:
                    return -1
            except:
                return -1

        except:
            return -1

    # 15. LinksInScriptTags
    def LinksInScriptTags(self):
        try:
            i,success = 0,0

            for link in self.soup.find_all('link',
href=True):
                dots = [x.start(0) for x in
re.finditer('\.', link['href'])]
                if self.url in link['href'] or self.domain
in link['href'] or len(dots) == 1:
                    success = success + 1
                i = i+1

            for script in self.soup.find_all('script',
src=True):
                dots = [x.start(0) for x in
re.finditer('\.', script['src'])]
                if self.url in script['src'] or self.domain
```

```
in script['src'] or len(dots) == 1:
                    success = success + 1
                i = i+1


            try:
                percentage = success / float(i) * 100
                if percentage < 17.0:
                    return 1
                elif((percentage >= 17.0) and (percentage <
81.0)):
                    return 0
                else:
                    return -1
            except:
                return 0
        except:
            return -1


    # 16. ServerFormHandler
    def ServerFormHandler(self):
        try:
            if len(self.soup.find_all('form',
action=True))==0:
                return 1
            else :
                for form in self.soup.find_all('form',
action=True):
                    if form['action'] == "" or
form['action'] == "about:blank":
                        return -1
                    elif self.url not in form['action'] and
self.domain not in form['action']:
```

```
                        return 0
                    else:
                        return 1
            except:
                return -1

    # 17. InfoEmail
    def InfoEmail(self):
        try:
            if re.findall(r"[mail\(\)|mailto:?]",
self.soap):
                return -1
            else:
                return 1
        except:
            return -1

    # 18. AbnormalURL
    def AbnormalURL(self):
        try:
            if self.response.text == self.whois_response:
                return 1
            else:
                return -1
        except:
            return -1

    # 19. WebsiteForwarding
    def WebsiteForwarding(self):
        try:
```

```
            if len(self.response.history) <= 1:
                return 1
            elif len(self.response.history) <= 4:
                return 0
            else:
                return -1
        except:
             return -1


    # 20. StatusBarCust
    def StatusBarCust(self):
        try:
            if
re.findall("<script>.+onmouseover.+</script>",
self.response.text):
                return 1
            else:
                return -1
        except:
             return -1


    # 21. DisableRightClick
    def DisableRightClick(self):
        try:
            if re.findall(r"event.button ?== ?2",
self.response.text):
                return 1
            else:
                return -1
        except:
             return -1
```

```
    # 22. UsingPopupWindow
    def UsingPopupWindow(self):
        try:
            if re.findall(r"alert\(", self.response.text):
                return 1
            else:
                return -1
        except:
             return -1


    # 23. IframeRedirection
    def IframeRedirection(self):
        try:
            if re.findall(r"[<iframe>|<frameBorder>]",
self.response.text):
                return 1
            else:
                return -1
        except:
             return -1


    # 24. AgeofDomain
    def AgeofDomain(self):
        try:
            creation_date =
self.whois_response.creation_date
            try:
                if(len(creation_date)):
                    creation_date = creation_date[0]
            except:
```

```
                pass


            today  = date.today()
            age =
(today.year-creation_date.year)*12+(today.month-creation_dat
e.month)
            if age >=6:
                return 1
            return -1
        except:
            return -1


    # 25. DNSRecording
    def DNSRecording(self):
        try:
            creation_date =
self.whois_response.creation_date
            try:
                if(len(creation_date)):
                    creation_date = creation_date[0]
            except:
                pass


            today  = date.today()
            age =
(today.year-creation_date.year)*12+(today.month-creation_dat
e.month)
            if age >=6:
                return 1
            return -1
        except:
            return -1
```

```
    # 26. WebsiteTraffic

    def WebsiteTraffic(self):

        try:

            rank =
BeautifulSoup(urllib.request.urlopen("http://data.alexa.com/
data?cli=10&dat=s&url=" + url).read(),
"xml").find("REACH")['RANK']

            if (int(rank) < 100000):

                return 1

            return 0

        except :

            return -1


    # 27. PageRank

    def PageRank(self):

        try:

            prank_checker_response =
requests.post("https://www.checkpagerank.net/index.php",
{"name": self.domain})


            global_rank = int(re.findall(r"Global Rank:
([0-9]+)", rank_checker_response.text)[0])

            if global_rank > 0 and global_rank < 100000:

                return 1

            return -1

        except:

            return -1




    # 28. GoogleIndex

    def GoogleIndex(self):

        try:
```

```
            site = search(self.url, 5)
            if site:
                return 1
            else:
                return -1
        except:
            return 1

    # 29. LinksPointingToPage
    def LinksPointingToPage(self):
        try:
            number_of_links = len(re.findall(r"<a href=",
self.response.text))
            if number_of_links == 0:
                return 1
            elif number_of_links <= 2:
                return 0
            else:
                return -1
        except:
            return -1

    # 30. StatsReport
    def StatsReport(self):
        try:
            url_match = re.search(

'at\.ua|usa\.cc|baltazarpresentes\.com\.br|pe\.hu|esy\.es|ho
l\.es|sweddy\.com|myjino\.ru|96\.lt|ow\.ly', url)
            ip_address = socket.gethostbyname(self.domain)
            ip_match =
re.search('146\.112\.61\.108|213\.174\.157\.151|121\.50\.168
```

```
\.88|192\.185\.217\.116|78\.46\.211\.158|181\.174\.165\.13|4
6\.242\.145\.103|121\.50\.168\.40|83\.125\.22\.219|46\.242\.
145\.98|'


'107\.151\.148\.44|107\.151\.148\.107|64\.70\.19\.203|199\.1
84\.144\.27|107\.151\.148\.108|107\.151\.148\.109|119\.28\.5
2\.61|54\.83\.43\.69|52\.69\.166\.231|216\.58\.192\.225|'


'118\.184\.25\.86|67\.208\.74\.71|23\.253\.126\.58|104\.239\
.157\.210|175\.126\.123\.219|141\.8\.224\.221|10\.10\.10\.10
|43\.229\.108\.32|103\.232\.215\.140|69\.172\.201\.153|'


'216\.218\.185\.162|54\.225\.104\.146|103\.243\.24\.98|199\.
59\.243\.120|31\.170\.160\.61|213\.19\.128\.77|62\.113\.226\
.131|208\.100\.26\.234|195\.16\.127\.102|195\.16\.127\.157|'


'34\.196\.13\.28|103\.224\.212\.222|172\.217\.4\.225|54\.72\
.9\.51|192\.64\.147\.141|198\.200\.56\.183|23\.253\.164\.103
|52\.48\.191\.26|52\.214\.197\.72|87\.98\.255\.18|209\.99\.1
7\.27|'


'216\.38\.62\.18|104\.130\.124\.96|47\.89\.58\.141|78\.46\.2
11\.158|54\.86\.225\.156|54\.82\.156\.19|37\.157\.192\.102|2
04\.11\.56\.48|110\.34\.231\.42', ip_address)

            if url_match:

                return -1

            elif ip_match:

                return -1

            return 1

        except:

            return 1


    def getFeaturesList(self):

        return self.features
```

index.html

```
<!DOCTYPE html>

<html lang="en">
```

```
<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width,
initial-scale=1.0">

    <title>Phisher</title>

    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/shorthandcss@1.1.1/dist/s
horthand.min.css" />

    <link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Muli:200,300,4
00,500,600,700,800,900&display=swap" />

    <link rel="stylesheet" type="text/css"


href="https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/
1.9.0/slick.min.css" />

    <link rel="stylesheet" type="text/css"
href="//cdn.jsdelivr.net/npm/slick-carousel@1.8.1/slick/slic
k-theme.css" />

</head>



<body class="bg-black muli">

    <nav class="w-100pc flex flex-column md-flex-row
md-px-10 py-5 bg-black">

        <div class="flex justify-between">



            <a data-toggle="toggle-nav"
data-target="#nav-items" href="#"

                class="flex items-center ml-auto md-hidden
indigo-lighter opacity-50 hover-opacity-100 ease-300 p-1
m-3">

                <i data-feather="menu"></i>

            </a>

        </div>



    </nav>
```

```
    <!-- hero section -->

    <section id="home" class="min-h-100vh flex justify-start
items-center">

            <div>

                <h1 class="white fs-l3 lh-2 md-fs-xl1
md-lh-1 fw-900 ">Check for malicious Status</h1>



                <div class="br-8 mt-10 inline-flex">

                    <form action="/" method ="post">

                    <input type="text"

                        class="input-lg half bw-0 fw-200
bg-indigo-lightest-10 white ph-indigo-lightest focus-white
opacity-80 fs-s3 py-5 min-w-25vw br-r-0"

                        class="form__input" name ='url'
id="url"

                        placeholder="Enter URL" required=""
/>

                    <button

                        class="button-lg
bg-indigo-lightest-20 indigo-lightest focus-white fw-300
fs-s3 mr-0 br-l-0" role="button">Detect</button>

                    </form>

                </div>

                <div class="white opacity-20 fs-s3
mt-3">eg:(https://google.com)</span>

            </div>

            <div class="col-md" id="form2">



                <br>

                <h6 class = "right "><a href= {{ url }}
target="_blank">{{ url }}</a></h6>



                <br>
```

```
{% if y_pred == 1 %}

    <h3 style="color:green;">{{result}}</h3>

    <h5 style="color:white;">{{pred}}</h5>

{% else %}

    <h3 style="color:red;">{{result}}</h3>

    <h5 style="color:white;">{{pred}}</h5>

{% endif %}

                <button class="button2" id="button2"
role="button" onclick="window.open('{{ url_for('index')
}}')" target="_blank" >Restart?</button>

            </div>

        </div>

    </section>


    <script>


        let x = '{{xx}}';

        let num = x*100;

        if (0<=x && x<0.50){

            num = 100-num;

        }

        let txtx = num.toString();

        if(x<=1 && x>=0.50){

            var label = "Website is "+txtx +"% safe to
use...";

            document.getElementById("prediction").innerHTML
= label;


document.getElementById("button1").style.display="block";

        }

        else if (0<=x && x<0.50){

            var label = "Website is "+txtx +"% unsafe to
use..."
```

```
            document.getElementById("prediction").innerHTML
= label ;


document.getElementById("button2").style.display="block";

        }


</script>
```

```
    <a class="fixed top-50pc right-0 p-3 bg-indigo white
hover-scale-up-1 ease-300 no-underline"
href="https://gum.co/tifJM" target="_blank" >

        <i class="w-4" data-feather="download"></i>

    </div>

    <script
src="https://code.jquery.com/jquery-3.4.1.min.js"></script>

    <script src="https://unpkg.com/feather-icons"></script>

    <script
src="https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1
.9.0/slick.min.js"></script>

    <script
src="https://cdn.jsdelivr.net/gh/cferdinandi/smooth-scroll@1
5.0.0/dist/smooth-scroll.polyfills.min.js"></script>

    <script src="assets/js/script.js"></script>

</body>


</html>
```

GitHub & project demo link:

GitHub link: https://github.com/IBM-EPBL/IBM-Project-5220-1658751901

Demolink:https://github.com/IBM-EPBL/IBM-Project-5220-1658751901/blob/mai

n/Final%20Deliverables/demo.mkv