Final Project Report Template

1. INTRODUCTION:

Phishing is one of the most prevalent forms of cybercrime, where attackers impersonate legitimate entities to deceive users into revealing sensitive information such as login credentials, financial details, or personal data. These malicious attacks often use fake websites that appear similar to authentic ones, luring unsuspecting victims into a trap.

One of the most efficient ways to detect phishing websites is through URL analysis. Phishing websites often have distinctive characteristics in their URLs, such as suspicious domain names, abnormal subdomains, or misleading paths. By analyzing these elements, it is possible to predict whether a URL is associated with a phishing attempt.

This project focuses on developing a system that can automatically detect phishing websites based on the characteristics of URLs. The approach involves extracting key features from URLs and using machine learning techniques to classify them as either legitimate or malicious. The goal is to provide a robust and scalable solution to mitigate phishing attacks and protect users from falling victim to online fraud.

The significance of this project lies in its potential to enhance online security, reduce phishing incidents, and contribute to the broader field of cybersecurity.

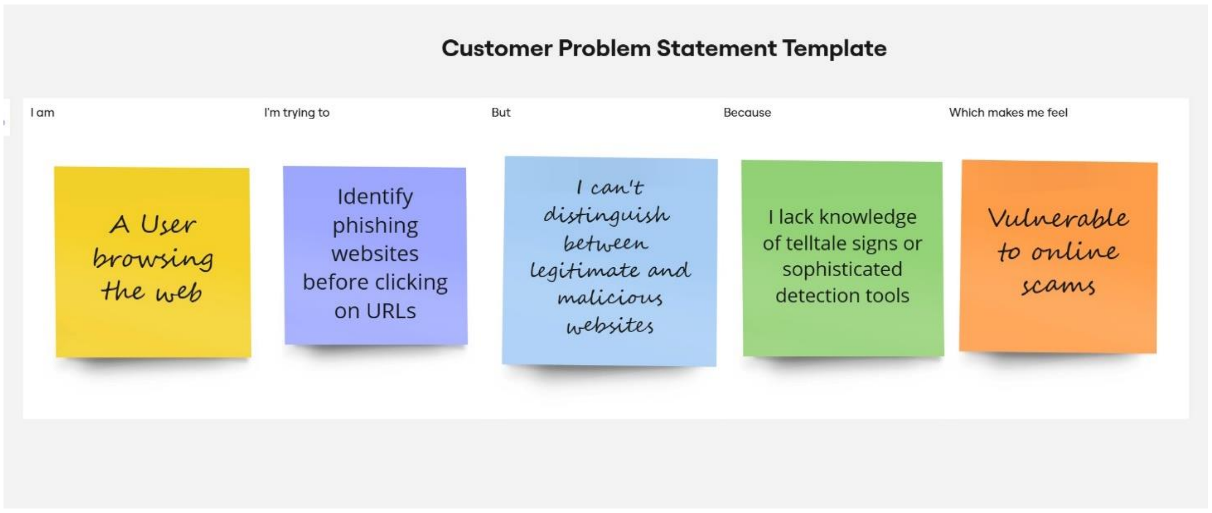
Project Overview	
Objective	To develop a machine learning-based solution that automatically detects and flags phishing websites by analyzing URLs.
Scope	This project will focus on collecting a dataset of URLs, developing and training machine learning models, and creating a user interface for users to input URLs for phishing detection. The project will be completed over a six-month period.

2.PROJECT INITIALIZATION AND PLANNING PHASE:

Problem Statements (Customer Problem Statement Template):

The major trouble is that phishing technique is bad accuracy and low adaptability to new phishing links. We plan to apply machine learning to overcome these limitation through imposing some classification algorithms and evaluating the overall performance of these algorithms on our dataset. We have decided on the Random Forest method because of its excellent performance in classification but random forest and decision tree are not good with nlp data

Example:



Customer Problem Statement Template

I am	I'm trying to	But	Because	Which makes me feel
A User browsing the web	Identify phishing websites before clicking on URLs	I can't distinguish between legitimate and malicious websites	I lack knowledge of telltale signs or sophisticated detection tools	Vulnerable to online scams

Proposed Solution:

 Data Collection: Gather a dataset of known phishing and legitimate URLs from various sources.

 Feature Extraction: Analyze URLs to extract relevant features (e.g., length, use of HTTPS, presence of suspicious characters).

 Model Training: Use supervised learning techniques to train models on the dataset for classification.

 Implementation: Develop a web-based interface for users to input URLs and receive phishing risk assessments.

Key Features:

 Real-time URL Analysis: Immediate classification of input URLs as phishing or legitimate.

 Feature Visualization: Show users how certain features contribute to the phishing risk assessment.

 User-Friendly Interface: Simple design for easy URL input and display of results.

Feature selection:

Feature selection is crucial when developing a machine learning model to detect phishing websites from URLs. The goal is to extract informative and relevant features from URLs that can help distinguish between phishing and legitimate websites.

These features are derived directly from the structure and content of the URL itself.

- **URL Length:** Phishing URLs are often longer than legitimate ones, as attackers add multiple parameters or obfuscate URLs.
- **Number of Dots:** Phishing URLs tend to use more dots (.) in an attempt to confuse users by mimicking subdomains.
- **Presence of Special Characters:** Symbols like @, -, _, &, %, =, ?, and / are often more frequent in phishing URLs.
- **Number of Subdomains:** Phishing sites tend to use long subdomain chains (e.g., http://login.yourbank.com.phishing.com).
- **Use of IP Address:** URLs containing IP addresses instead of domain names (e.g., http://192.168.0.1/login) are often phishing attempts.
- **Suspicious Keywords:** Look for common phishing keywords such as “login,” “secure,” “verify,” “account,” “update,” “banking,” etc.
- **Use of HTTPS:** Check whether the URL uses HTTP or HTTPS. Many phishing websites may still use HTTP, though more sophisticated attacks use HTTPS with fake certificates.
- **URL Encoding:** Phishing URLs may use hexadecimal encoding to hide malicious portions of the URL (e.g., %20).

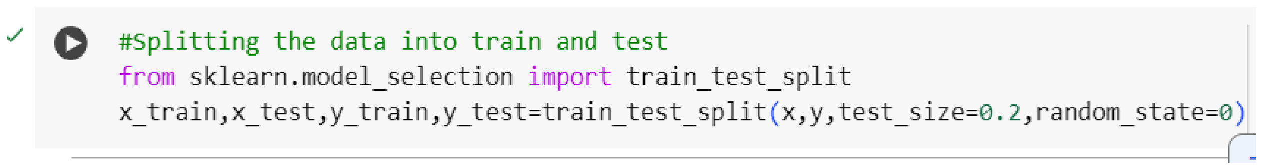
Feature Selection Techniques:

- ❖ Tree-Based Methods (Feature Importance):

Random Forest, XGBoost, Logistic Regression, Decision Trees can provide insights into feature importance.

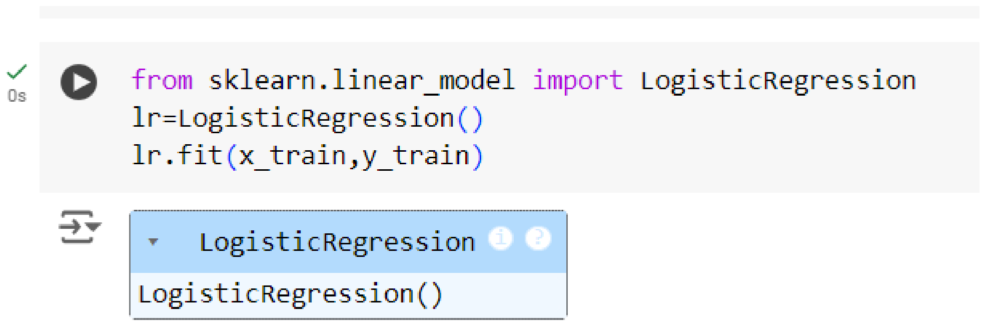
3. Data Collection and Preprocessing phase:

Splitting the data:



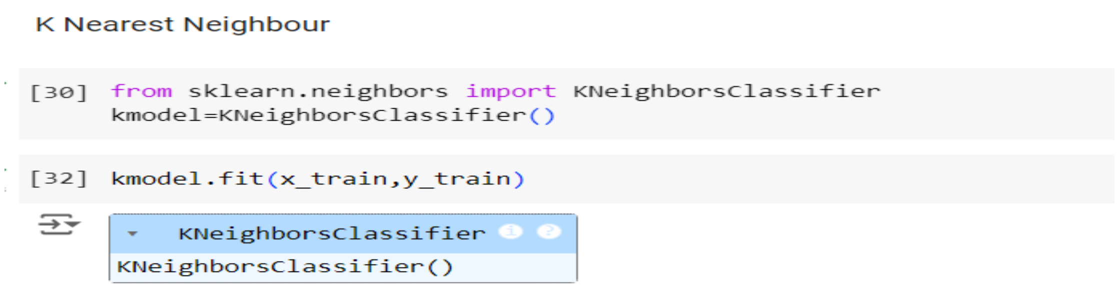
```
#Splitting the data into train and test
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

Training the model:



```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(x_train,y_train)
```

```
LogisticRegression
LogisticRegression()
```

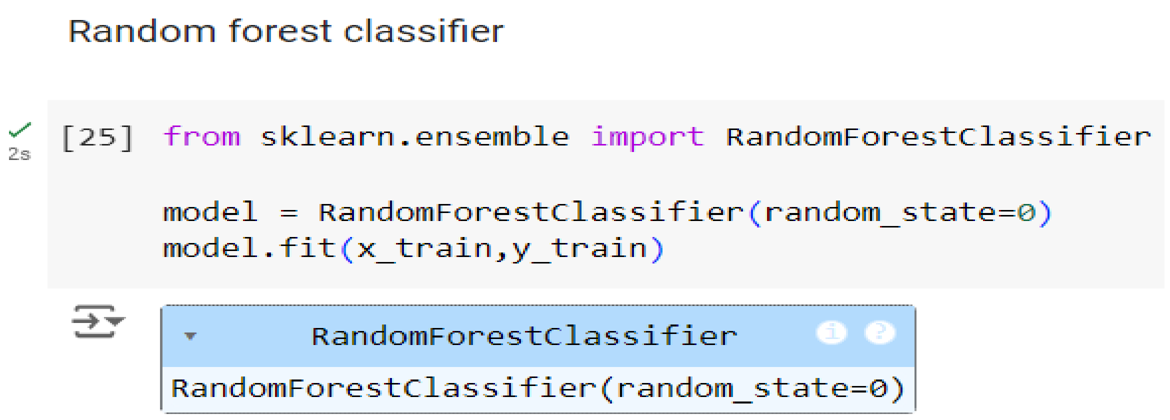


K Nearest Neighbour

```
[30] from sklearn.neighbors import KNeighborsClassifier
     kmodel=KNeighborsClassifier()
```

```
[32] kmodel.fit(x_train,y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier()
```



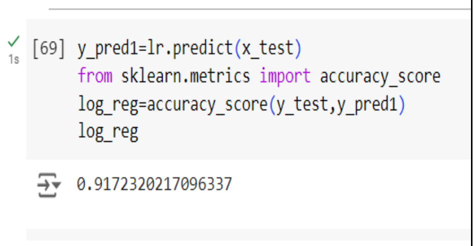
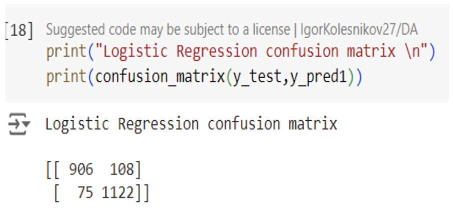
Random forest classifier

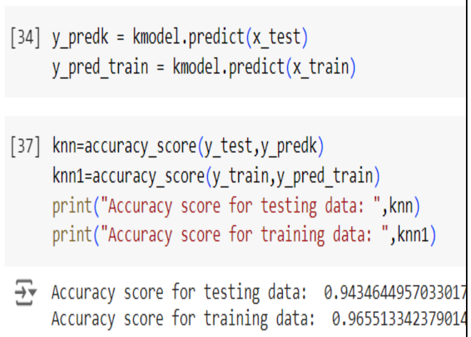
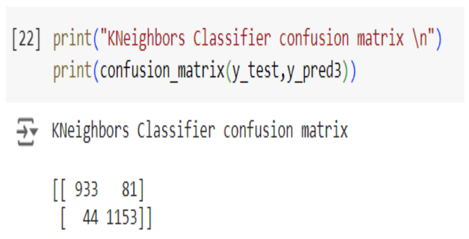
```
[25] from sklearn.ensemble import RandomForestClassifier

     model = RandomForestClassifier(random_state=0)
     model.fit(x_train,y_train)
```

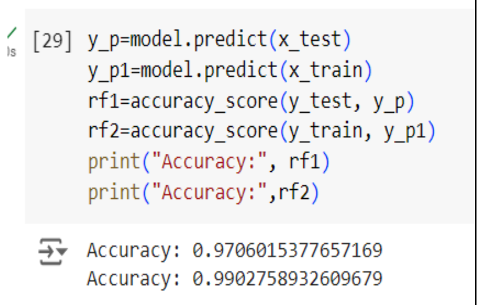
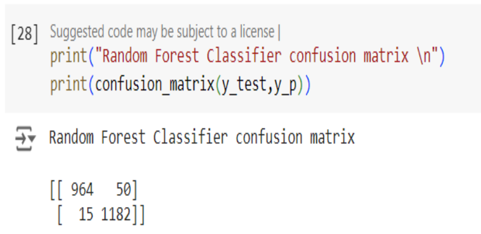
```
RandomForestClassifier
RandomForestClassifier(random_state=0)
```

Model Validation and Evaluation Report:

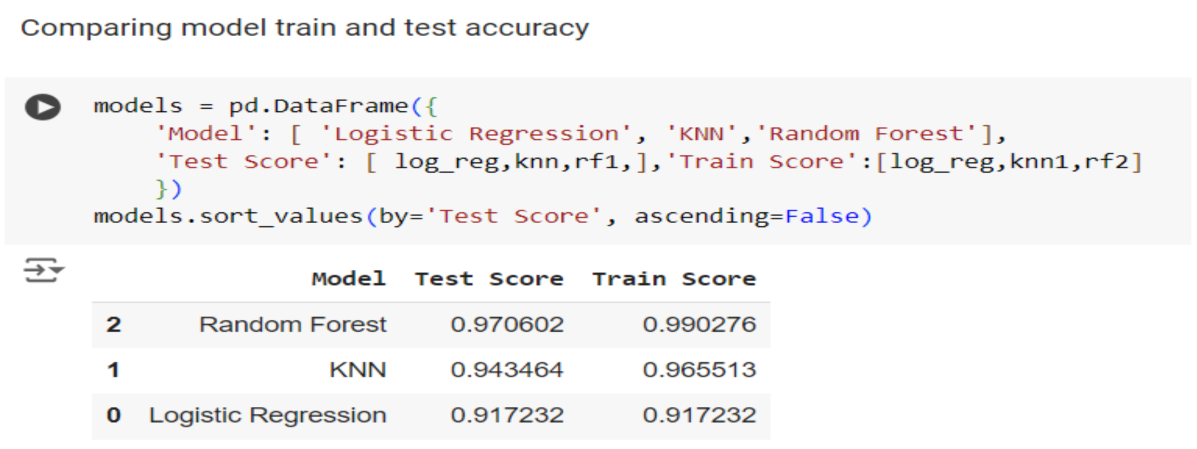
Model	Classification Report	Accuracy	Confusion Matrix

Linear regression	 `[69] y_pred1=lr.predict(x_test)`
	`from sklearn.metrics import accuracy_score`
	`log_reg=accuracy_score(y_test,y_pred1)`
	`log_reg`
	`0.9172320217096337`
		0.91	 `[18]` Suggested code may be subject to a license | IgorKolesnikov27/DA
			`print("Logistic Regression confusion matrix \n")`
			`print(confusion_matrix(y_test,y_pred1))`
			`Logistic Regression confusion matrix`
			`[[ 906  108]`
			` [  75 1122]]`

KNN	 `[34] y_predk = kmodel.predict(x_test)`
	`y_pred_train = kmodel.predict(x_train)`
	`[37] knn=accuracy_score(y_test,y_predk)`
	`knn1=accuracy_score(y_train,y_pred_train)`
	`print("Accuracy score for testing data: ",knn)`
	`print("Accuracy score for training data: ",knn1)`
	`Accuracy score for testing data:  0.9434644957033017`
	`Accuracy score for training data:  0.965513342379014`
		0.94	 `[22] print("KNeighbors Classifier confusion matrix \n")`
			`print(confusion_matrix(y_test,y_pred3))`
			`KNeighbors Classifier confusion matrix`
			`[[ 933   81]`
			` [  44 1153]]`

Random Forest	 ```[29] y_p=model.predict(x_test) y_p1=model.predict(x_train) rf1=accuracy_score(y_test, y_p) rf2=accuracy_score(y_train, y_p1) print("Accuracy:", rf1) print("Accuracy:",rf2)``` ```Accuracy: 0.9706015377657169 Accuracy: 0.9902758932609679```	0.97	 ```[28] Suggested code may be subject to a license | print("Random Forest Classifier confusion matrix \n") print(confusion_matrix(y_test,y_p))``` ```Random Forest Classifier confusion matrix [[964 50] [15 1182]]```

Comparing of model train test accuracy:



Comparing model train and test accuracy

```
models = pd.DataFrame({
    'Model': [ 'Logistic Regression', 'KNN','Random Forest'],
    'Test Score': [ log_reg,knn,rf1,],'Train Score':[log_reg,knn1,rf2]
    })
models.sort_values(by='Test Score', ascending=False)
```

```
                 Model  Test Score  Train Score
2        Random Forest    0.970602     0.990276
1                  KNN    0.943464     0.965513
0  Logistic Regression    0.917232     0.917232
```

4. Model Development Phase:

Model Selection-

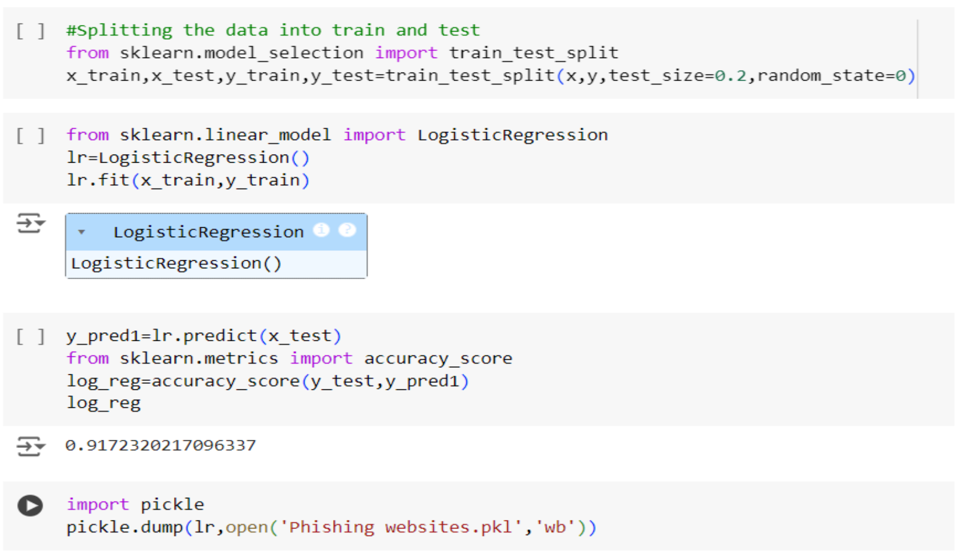
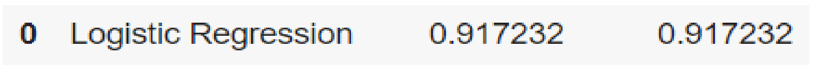
In the forthcoming Model Selection Report, various models will be outlined, detailing their descriptions and performance metrics, including Accuracy or F1 Score. This comprehensive report will provide insights into the chosen models and their effectiveness.

Model Selection Report:

Model	**Description**	**Performance Metric (e.g., Accuracy, F1 Score)**
Logistic regression	Logistic regression is a statistical method used for binary classification problems, where the outcome variable is categorical with two possible outcomes (e.g., success/failure, yes/no). It models the relationship between one or more independent variables and the probability outcome occurring.	91%
KNN	K-Nearest Neighbors (KNN) is a simple yet powerful algorithm used in machine learning for classification and regression tasks. It operates on the principle of proximity, where the output for a given input is determined by the majority class (in classification) or the average (in regression) of its k nearest neighbors in the feature space.	94%
Random Forest	Random Forest is an ensemble learning method primarily used for classification and regression tasks. It builds multiple decision trees during training and merges their outputs to improve accuracy and control overfitting.	97%

5. Model Optimization and Tuning Phase:

The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Model	**Finalized Model Accuracy**
Model 1: Logistic regression	 

```
[ ] #Splitting the data into train and test
    from sklearn.model_selection import train_test_split
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)

[ ] from sklearn.linear_model import LogisticRegression
    lr=LogisticRegression()
    lr.fit(x_train,y_train)

    LogisticRegression
    LogisticRegression()

[ ] y_pred1=lr.predict(x_test)
    from sklearn.metrics import accuracy_score
    log_reg=accuracy_score(y_test,y_pred1)
    log_reg

    0.9172320217096337

    import pickle
    pickle.dump(lr,open('Phishing websites.pkl','wb'))
```

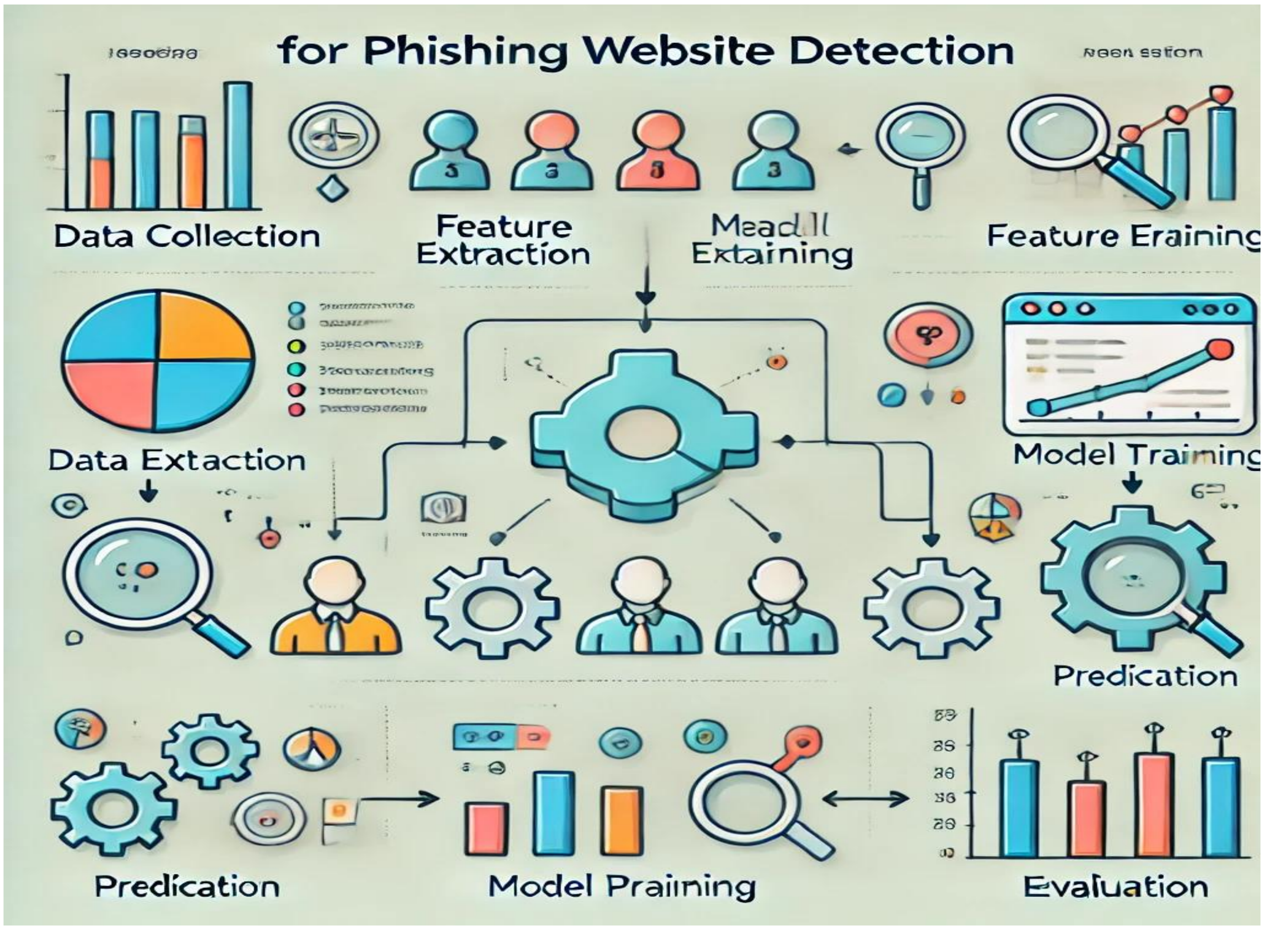
0	Logistic Regression	0.917232	0.917232

Note:

Due to the specific requirements of this project and the dataset's structure, the other models were not implemented in this phase, as Logistic Regression demonstrated superior performance for text classification.

Final Model Selection Justification (2 Marks):

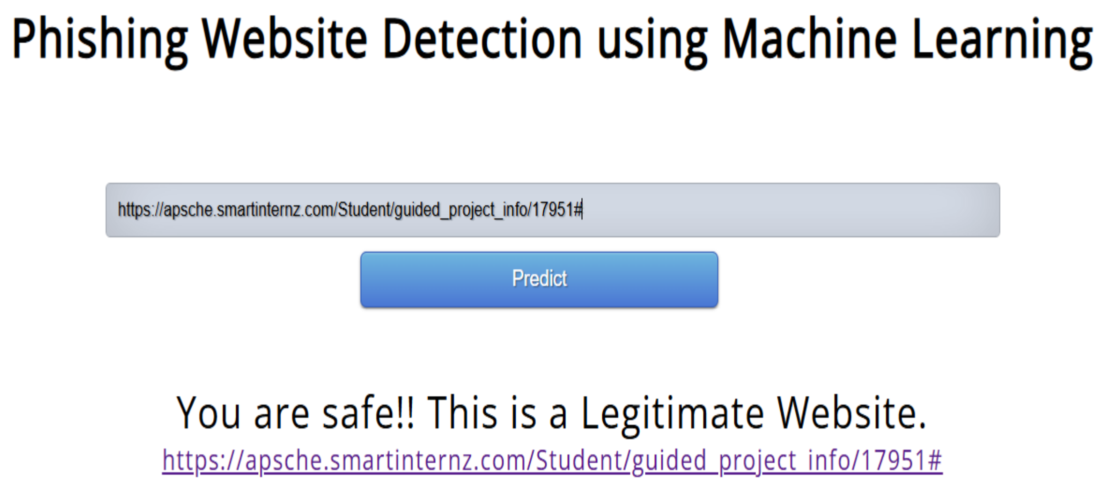
Final Model	**Reasoning**
Logistic Regression	Logistic regression is justified for phishing website detection from URLs because it provides an interpretable, scalable, and efficient solution to a binary classification problem. Its probabilistic output, ease of handling various types of features, and ability to manage imbalanced data make it a strong candidate for this cybersecurity application.



6. Results:

Output Screenshots-

Detection of Correct URL-



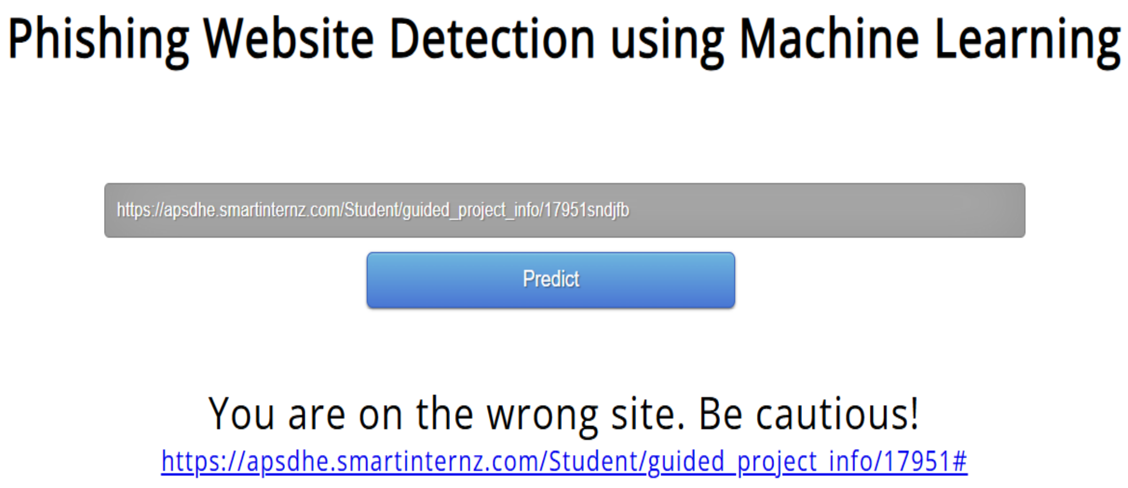
Phishing Website Detection using Machine Learning

https://apsche.smartinternz.com/Student/guided_project_info/17951#

You are safe!! This is a Legitimate Website.

https://apsche.smartinternz.com/Student/guided_project_info/17951#

Detection of Wrong URL-



Phishing Website Detection using Machine Learning

https://apsdhe.smartinternz.com/Student/guided_project_info/17951sndjfb

You are on the wrong site. Be cautious!

https://apsdhe.smartinternz.com/Student/guided_project_info/17951#

7. Advantages:

Detecting phishing websites from URLs has several advantages:

1. **Speed and Efficiency**:
   - URL-based detection is quick, as it does not require downloading the entire webpage or performing complex analysis on the page content. This makes it faster and computationally efficient, ideal for real-time detection.
2. **Early Detection**:
   - By analyzing URLs, phishing websites can be identified even before they fully load or execute malicious scripts. This helps to prevent phishing attacks at an early stage.
3. **Minimal Resource Requirements**:
   - URL-based detection primarily relies on examining specific URL features like domain names, length, use of special characters, and HTTP vs. HTTPS. This involves less resource consumption compared to other techniques that may involve content analysis.
4. **Scalability**:
   - Since URLs are just strings, analyzing them is lightweight and scalable. It allows security tools to efficiently monitor and filter large volumes of web traffic for phishing attempts.
5. **Works in Encrypted Traffic**:
   - When websites are encrypted using HTTPS, content-based detection methods may not have full visibility. However, URL analysis can still be effective, as it focuses on the visible part of the URL structure.
6. **Simplicity**:
   - URL-based detection methods are relatively simpler to implement compared to machine learning models that analyze full page content or behavior. This makes it easier to integrate with existing security systems.

7. **Behavioral Pattern Identification**:
    - URL analysis can help identify patterns in malicious URLs used by attackers. For instance, phishing URLs often include certain keywords (like 'login' or 'verify') or unusual domain structures. These patterns can be leveraged for automated detection.
8. **No Dependency on Visual Similarities**:
    - Unlike some methods that rely on visual similarities between legitimate and phishing sites, URL-based detection focuses purely on the URL's structure, avoiding issues with sophisticated visual disguises.

URL-based phishing detection serves as an effective first line of defense against many attacks while complementing other detection techniques.

Disadvantages:

While detecting phishing websites from URLs has several advantages, there are also some disadvantages and limitations to consider:

1. **Limited Scope**:
    - URL-based detection focuses only on the URL's structure and properties, ignoring the actual content of the website. This means sophisticated phishing attacks that use legitimate-looking URLs can bypass detection.
2. **Obfuscation Techniques**:
    - Attackers often employ obfuscation techniques, such as using URL shorteners, adding misleading subdomains, or using typosquatting (misspelled domain names) to create URLs that appear legitimate. This can make it challenging to distinguish phishing URLs from genuine ones.
3. **Difficulty with Zero-Day Phishing URLs**:

- URL-based models often rely on identifying patterns and known features. Zero-day phishing attacks or newly created URLs may not have these patterns, making them harder to detect accurately.

4. False Positives:

    Heuristics-based detection methods, which identify phishing URLs based on suspicious patterns, might flag legitimate websites as phishing. This results in false positives, leading to user inconvenience and decreased trust in the detection system.

5. Limited Information:
    - URLs alone may not provide enough information to make an accurate classification in many cases. Without examining the webpage's content or behavior, some sophisticated attacks may slip through undetected.
6. Spoofing and Use of Legitimate Services:
    - Phishers sometimes host their malicious content on legitimate services (such as compromised accounts on trusted domains or cloud storage services). In such cases, the URLs themselves may not appear suspicious.
7. Dynamic and Evasive Techniques:
    - Some phishing attacks use dynamically generated URLs or URLs that change frequently. Attackers can rapidly rotate URLs or redirect users to different phishing pages after initial validation, evading detection.
8. Dependency on Rule Updates:
    - Traditional URL detection methods may depend on predefined rules and feature sets. Phishing techniques are constantly evolving, and if the rules are not updated regularly, the system may fail to detect new threats.

Overall, while URL-based phishing detection is efficient and valuable as a first-line defense, it should ideally be combined with other approaches like content analysis, behavioral analysis, or machine learning models for more robust protection.

8.Conclusion:

The project on detecting phishing websites from URLs demonstrates the effectiveness and feasibility of using machine learning algorithms like Logistic Regression for this purpose. By focusing on the analysis of URL features such as domain names, special characters, protocol usage, and keyword patterns, the detection system can efficiently classify websites as legitimate or phishing. This URL-based approach offers speed, scalability, and minimal resource requirements, making it ideal for real-time protection.

However, the project also highlights that URL-based detection has its limitations, such as susceptibility to obfuscation techniques, zero-day phishing attacks, and false positives. Therefore, while this method serves as a strong first line of defense, it is recommended to complement it with other detection methods like webpage content analysis or behavioral analysis to enhance overall security.

In conclusion, detecting phishing websites from URLs offers a practical, lightweight, and efficient solution to combat a significant portion of phishing attacks. With ongoing updates and improvements, such systems can play a crucial role in safeguarding users from evolving phishing threats.

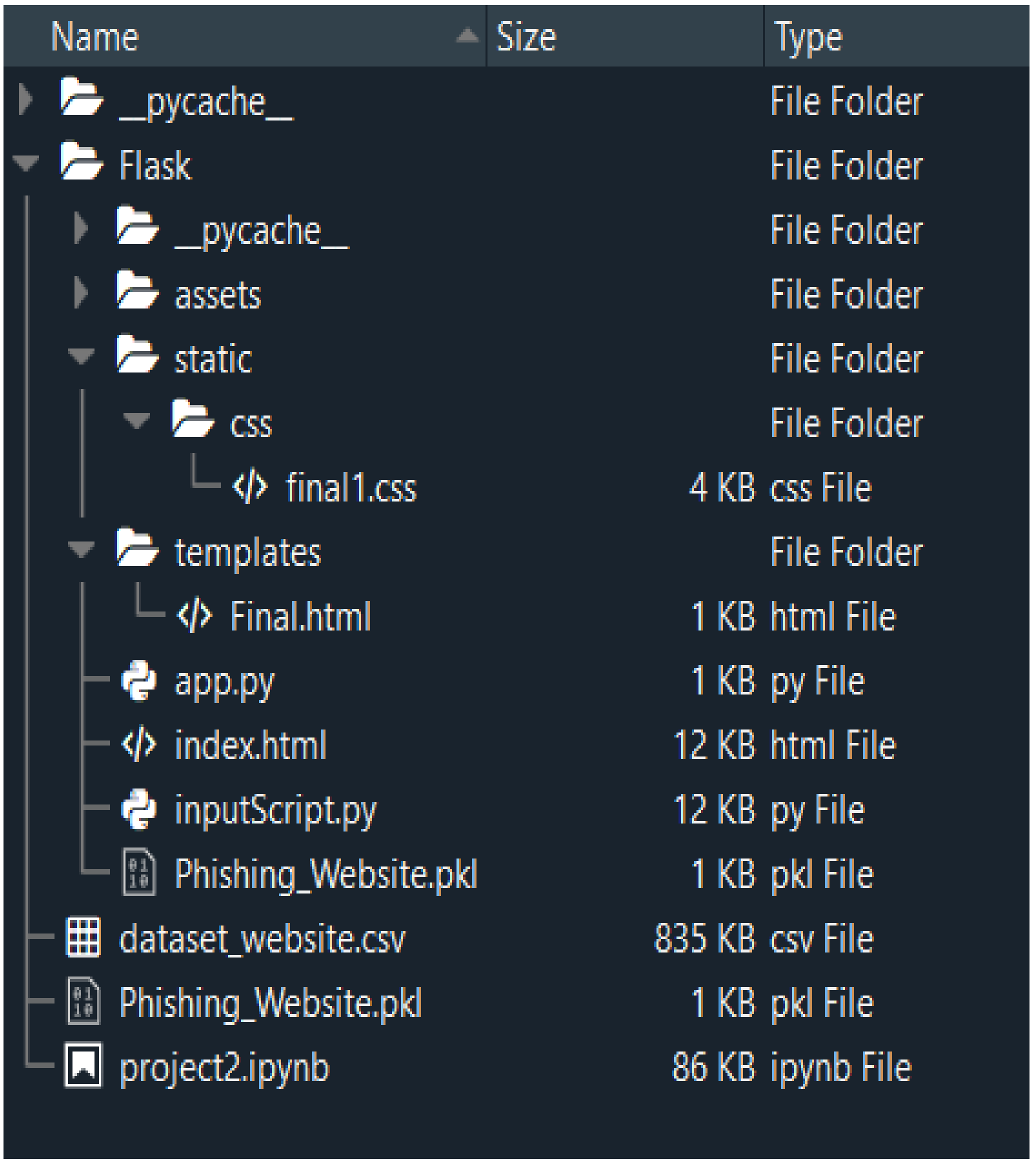
Future scope:

The future scope of detecting phishing websites from URLs focuses on improving the accuracy, adaptability, and scalability of current detection systems. Advancements can include the integration of more sophisticated machine learning techniques such as Deep Learning models, which can identify complex patterns and tackle evasive techniques used by attackers. Real-time threat intelligence and self-learning models could enhance the system's ability to counter zero-day phishing attacks and new types of URL obfuscation methods.

Another key area is the development of hybrid detection approaches that combine URL-based analysis with content and behavioral analysis, reducing false positives and improving robustness. Additionally, integrating detection systems with popular web browsers, email clients, and mobile apps can provide users with real-time protection and alerts against potential phishing threats.

Furthermore, with the rise in multilingual and mobile-based phishing, expanding the system to detect threats across languages and platforms is essential. As the landscape of phishing attacks evolves, the focus should also include developing adaptive models that continuously learn and enhance their detection capabilities, ensuring proactive and comprehensive security against phishing attacks.

Project Folder Structure:



Name	Size	Type
__pycache__		File Folder
Flask		File Folder
__pycache__		File Folder
assets		File Folder
static		File Folder
css		File Folder
final1.css	4 KB	css File
templates		File Folder
Final.html	1 KB	html File
app.py	1 KB	py File
index.html	12 KB	html File
inputScript.py	12 KB	py File
Phishing_Website.pkl	1 KB	pkl File
dataset_website.csv	835 KB	csv File
Phishing_Website.pkl	1 KB	pkl File
project2.ipynb	86 KB	ipynb File

SOURCE CODE:

App.py

```
import numpy as np

from flask import Flask, request, jsonify, render_template

import pickle

import inputScript
import joblib


app = Flask(__name__)
model = joblib.load(open('Phishing websites.pkl', 'rb'))


@app.route('/')
def index():
    return render_template('index.html')


@app.route('/predict', methods=['GET', 'POST'])
def predict():
    if request.method == 'POST':
        url = request.form.get('URL')
        if not url:
             return render_template('predict.html', prediction_text='Error:
URL is required', url='predict')


        try:
            checkprediction = inputScript.Phishing_Website_Detection(url)
            # Reshape the input data to a 2D array
            checkprediction = np.array(checkprediction).reshape(1, -1)
            prediction = model.predict(checkprediction)
            output = prediction[0]
            if output == 1:
                pred = "You are safe!! This is a Legitimate Website."
```

```
          else:
              pred = "You are on the wrong site. Be cautious!"
            return render_template('predict.html', prediction_text=pred, url=url)
        except Exception as e:
            return render_template('predict.html', prediction_text='An error occurred: {}'.format(e), url=url)
    else:
        return render_template('predict.html', prediction_text='Error: Invalid request method', url='predict')


@app.route('/predict_api', methods=['POST'])
def predict_api():
    try:
        data = request.get_json(force=True)
        if not data:
            raise ValueError("Invalid JSON data")
        input_array = np.array(list(data.values()))
        # Reshape the input data to a 2D array
        input_array = input_array.reshape(1, -1)
        prediction = model.predict(input_array)
        output = prediction[0]
        return jsonify({'output': output})
    except ValueError as e:
        # Handle invalid JSON data or data format errors
        return jsonify({'error': str(e)}), 400
    except Exception as e:# Handle any other exceptions raised during
prediction
        return jsonify({'error': str(e)}), 500

if __name__ == '__main__':
    app.run(debug=True)
```

InputScript.py

```
import regex
from tldextract import extract
import ssl
import socket
from bs4 import BeautifulSoup
import urllib.request
import whois
import datetime
import requests
import favicon
import re
from googlesearch import search

#1. Checking if URL contains any IP address. Returns -1 if contains else returns 1
def having_IPhaving_IP_Address(url):
    match = regex.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])\\\/)|'  # 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-fA-F0-9]{1,4}:){7}[a-fA-F0-9]{1,4}', url)  # IPv6
    return -1 if match else 1


# 2.Checking for the URL length. Returns 1 (Legitimate) if the URL length is less than 54 characters
def URLURL_Length(url):
    length = len(url)
    if length <= 75:
        return 1 if length < 54 else 0
    return -1


#3. Checking with the shortening URLs.
def Shortining_Service(url):
    match = regex.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\.co
m|cutt\.us|u\.bb|yourls\.org|'
                        'x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|q
r\.net|1url\.com|tweez\.me|v\.gd|tr\.im|link\.zip\.net', url)
    return -1 if match else 1

#4.Checking for @ symbol. Returns 1 if no @ symbol found. Else returns -1.
def having_At_Symbol(url):
    return 1 if '@' not in url else -1

# 5.Checking for Double Slash redirections. Returns -1 if // found. Else returns 1
def double_slash_redirecting(url):
    return -1 if '//' in url else 1

#6. Checking for - in Domain. Returns -1 if '-' is found else returns 1.
def Prefix_Suffix(url):
    domain_info = extract(url)
    return -1 if '-' in domain_info.domain else 1

# 7.Checking the Subdomain. Returns 1 if the subDomain contains less than 1 '.'
# Returns 0 if the subDomain contains less than 2 '.'
# Returns -1 if the subDomain contains more than 2 '.'
def having_Sub_Domain(url):
    domain_info = extract(url)
    subdomain = domain_info.subdomain
    return 1 if subdomain.count('.') <= 1 else 0 if subdomain.count('.') <= 2 else -1
```

```
# 8.Checking the SSL. Returns 1 if it returns the response code and -1 if exceptions are thrown.
def SSLfinal_State(url):
    try:
        response = requests.get(url)
        return 1
    except Exception as e:
        return -1

#9.domains expires on ≤ 1 year returns -1, otherwise returns 1
def Domain_registeration_length(url):
    try:
        domain = whois.whois(url)
        exp = domain.expiration_date[0]
        up = domain.updated_date[0]
        domainlen = (exp - up).days
        return -1 if domainlen <= 365 else 1
    except:
        return -1

# 10.Checking the Favicon. Returns 1 if the domain of the favicon image and the URL domain match else returns -1.
def Favicon(url):
    domain_info = extract(url)
    try:
        icons = favicon.get(url)
        icon = icons[0]
        favicon_domain_info = extract(icon.url)

        return 1 if favicon_domain_info.domain == domain_info.domain else -1

    except:

        return -1


#11.Checking the Port of the URL. Returns 1 if the port is available else returns -
1.

def port(url):

    try:

        a_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

        location = (url[7:], 80)

        result_of_check = a_socket.connect_ex(location)

        a_socket.close()

        return 1 if result_of_check == 0 else -1

    except:

        return -1


#12. HTTPS token in part of domain of URL returns -1, otherwise returns 1

def HTTPS_token(url):

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

    if match.start(0) == 0:

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

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

    return -1 if match else 1


# 13.% of request URL<22% returns 1, otherwise returns -1

def Request_URL(url):
```

```
try:
    domain_info = extract(url)
    websiteDomain = domain_info.domain
    opener = urllib.request.urlopen(url).read()
    soup = BeautifulSoup(opener, 'lxml')
    imgs = soup.findAll('img', src=True)
    total = len(imgs)
    linked_to_same = 0
    avg = 0
    for image in imgs:
        image_domain_info = extract(image['src'])
        imageDomain = image_domain_info.domain
        if websiteDomain == imageDomain or imageDomain == '':
            linked_to_same += 1
    vids = soup.findAll('video', src=True)
    total += len(vids)
    for video in vids:
        video_domain_info = extract(video['src'])
        vidDomain = video_domain_info.domain
        if websiteDomain == vidDomain or vidDomain == '':
            linked_to_same += 1
    linked_outside = total - linked_to_same
    if total != 0:
        avg = linked_outside / total
    return 1 if avg < 0.22 else -1
except:
    return -1
```

```
# 14.:% of URL of anchor<31% returns 1, % of URL of anchor ≥ 31% and ≤ 67%
returns 0, otherwise returns -1
def URL_of_Anchor(url):
    try:
        domain_info = extract(url)
        websiteDomain = domain_info.domain
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        anchors = soup.findAll('a', href=True)
        total = len(anchors)
        linked_to_same = 0
        avg = 0
        for anchor in anchors:
            anchor_domain_info = extract(anchor['href'])
            anchorDomain = anchor_domain_info.domain
            if websiteDomain == anchorDomain or anchorDomain == '':
                linked_to_same += 1
        linked_outside = total - linked_to_same
        if total != 0:
            avg = linked_outside / total
        return 1 if avg < 0.31 else 0 if 0.31 <= avg <= 0.67 else -1
    except:
        return 0


#15. :% of links in <meta>, <script>and<link>tags < 25% returns 1, % of links in
<meta>,
# <script> and <link> tags ≥ 25% and ≤ 81% returns 0, otherwise returns -1
def Links_in_tags(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        no_of_meta = 0
        no_of_link = 0
        no_of_script = 0
        anchors = 0
        avg = 0
        for meta in soup.find_all('meta'):
            no_of_meta += 1
        for link in soup.find_all('link'):
            no_of_link += 1
        for script in soup.find_all('script'):
                    no_of_script += 1
        for anchor in soup.find_all('a'):
            anchors += 1
        total = no_of_meta + no_of_link + no_of_script + anchors
        tags = no_of_meta + no_of_link + no_of_script
        if total != 0:
            avg = tags / total
        return -1 if avg < 0.25 else 0 if 0.25 <= avg <= 0.81 else 1
    except:
        return 0

# 16.Server Form Handling
# SFH is "about: blank" or empty → phishing, SFH refers to a different domain →
suspicious, otherwise → legitimate
def SFH(url):
    # ongoing
    return -1


# 17.:using "mail()" or "mailto:" returning -1, otherwise returns 1
def Submitting_to_email(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        if soup.find('mailto:', 'mail():'):
            return -1
        else:
            return 1
    except:
        return -1


#18.Host name is not in URL returns -1, otherwise returns 1
def Abnormal_URL(url):
    domain_info = extract(url)
    try:
        domain = whois.whois(url)
        hostname = domain.domain_name[0].lower()
        match = re.search(hostname, url)
        return 1 if match else -1
    except:
        return -1

# 19.number of redirect page ≤ 1 returns 1, otherwise returns 0
def Redirect(url):
    try:
        request = requests.get(url)
        a = request.history
        return 1 if len(a) <= 1 else 0
    except:
        return 0

#20. onMouseOver changes status bar returns -1, otherwise returns 1
def on_mouseover(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup
        if soup.find(onmouseover=True):
            return -1
        else:
            return 1
    except:
        return -1

# 21.RightClick disables returns -1, otherwise returns 1
def RightClick(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        if soup.find(oncontextmenu="function disableSelection"):
            return -1
        else:
            return 1
    except:
        return -1
#22.popupwindow
def popUpWidnow(url):
    try:
        response = requests.get(url)
        soup = BeautifulSoup(response.text, 'html.parser')
        popup_windows = soup.find_all('input', {'type': 'text'})
        if popup_windows:
            return 1  # phishing
        else:
            return 0  # legitimate
    except Exception as e:
        print(f"Error in popUpWidnow: {e}")
        return 0  # legitimate by default

# 23.IFrame Redirection returns -1, otherwise returns 1
def iFrame(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        if soup.find(iframe=True):
            return -1
        else:
            return 1
    except:
        return -1
```

#24. Age of Domain returns -1 if age is less than 6 months, otherwise returns 1

```
def Domain_Age(url):
    try:
        domain = whois.whois(url)
        creation_date = domain.creation_date[0]
        end_date = datetime.datetime.now()
        age = (end_date - creation_date).days
        return -1 if age < 180 else 1
    except:
        return -1
```

#25. DNS Record availability returns -1 if DNS record is not available, otherwise returns 1

```
def DNS_Record(url):
    try:
        domain = whois.whois(url)
        return 1 if domain else -1
    except:
        return -1
```

#26. Web Traffic returns -1 if web traffic is less than 1000, otherwise returns 1

```
def Web_Traffic(url):
    try:
        search_results = list(search(url, num_results=10))
        return -1 if len(search_results) < 1000 else 1
    except:
        return -1

#27. Page Rank returns -1 if page rank is less than 0.2, otherwise returns 1
def Page_Rank(url):
    try:
        rank = 0  # implement page rank algorithm
        return -1 if rank < 0.2 else 1
    except:
        return -1

#28. Google Index returns -1 if google index is less than 100, otherwise returns 1
def Google_Index(url):
    try:
        search_results = list(search(url, num_results=10))
        return -1 if len(search_results) < 100 else 1
    except:
        return -1

#29. Links Pointing to Page returns -1 if links pointing to page is less than 2,
otherwise returns 1
def Links_Pointing_to_Page(url):
    try:
        search_results = list(search(url, num_results=10))
        return -1 if len(search_results) < 2 else 1
    except:
        return -1

# 30.Statistical Report returns -1 if statistical report is less than 60, otherwise returns 1
def Statistical_Report(url):
    try:
        rank = 0  # implement statistical report algorithm
        return -1 if rank < 60 else 1
    except:
        return -1

# Phishing Website Detection
def Phishing_Website_Detection(url):
    try:
        features = [having_IPhaving_IP_Address(url),
URLURL_Length(url),
Shortining_Service(url),
having_At_Symbol(url),
double_slash_redirecting(url),
Prefix_Suffix(url),
having_Sub_Domain(url),
SSLfinal_State(url),
Domain_registeration_length(url),
Favicon(url),
port(url),
HTTPS_token(url),
Request_URL(url),
URL_of_Anchor(url),
Links_in_tags(url),
SFH(url),
Submitting_to_email(url),
Abnormal_URL(url),
Redirect(url),
on_mouseover(url),
RightClick(url),
popUpWidnow(url),
iFrame(url),
Domain_Age(url),
DNS_Record(url),
Web_Traffic(url),
Page_Rank(url),
Google_Index(url),
Links_Pointing_to_Page(url),
Statistical_Report(url)]
        return features
    except:
        return [-1]
# Example usage:
#url = "http://example.com"
#features = Phishing_Website_Detection(url)
#print(features)
```

WEB PAGE DESIGNING:

INDEX Page-

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1.0" name="viewport">

  <title>Phishing Website Detection</title>
  <meta content="" name="description">
  <meta content="" name="keywords">


  <!-- Google Fonts -->
                                                                  <link
href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,6
00,600i,700,700i|Jost:300,300i,400,400i,500,500i,600,600i,700,700i|Poppins:3
00,300i,400,400i,500,500i,600,600i,700,700i" rel="stylesheet">

  <!-- Vendor CSS Files -->
        <!--link        href="assets/vendor/bootstrap/css/bootstrap.min.css"
rel="stylesheet">
  <link href="assets/vendor/icofont/icofont.min.css" rel="stylesheet">
  <link href="assets/vendor/boxicons/css/boxicons.min.css" rel="stylesheet">
  <link href="assets/vendor/remixicon/remixicon.css" rel="stylesheet">
  <link href="assets/vendor/venobox/venobox.css" rel="stylesheet">
  <link href="assets/vendor/owl.carousel/assets/owl.carousel.min.css" rel="stylesheet">
  <link href="assets/vendor/aos/aos.css" rel="stylesheet"-->



  <link href="{{ url_for('static', filename='/vendor/bootstrap/css/bootstrap.min.css') }}" rel="stylesheet">
  <link href="{{ url_for('static', filename='/vendor/icofont/icofont.min.css') }}" rel="stylesheet">
  <link href="{{ url_for('static', filename='/vendor/boxicons/css/boxicons.min.css') }}" rel="stylesheet">
  <link href="{{ url_for('static', filename='/vendor/remixicon/remixicon.css') }}" rel="stylesheet">
  <link href="{{ url_for('static', filename='/vendor/venobox/venobox.css') }}" rel="stylesheet">
  <link href="{{ url_for('static', filename='/vendor/owl.carousel/assets/owl.carousel.min.css') }}" rel="stylesheet">
  <link href="{{ url_for('static', filename='/vendor/aos/aos.css') }}" rel="stylesheet">

  <!-- Template Main CSS File -->
<!-- href="assets/css/style.css" rel="stylesheet"-->
  <link href="{{ url_for('static', filename='css/style.css') }}" rel="stylesheet">



</head>

<body>
```

```
 <!-- ======= Header ======= -->
 <header id="header" class="fixed-top ">
   <div class="container d-flex align-items-center">

     <h1 class="logo mr-auto"><a href="index.html">AntiPhish</a></h1>
     <!-- Uncomment below if you prefer to use an image logo -->
    <!--a href="index.html" class="logo mr-auto"><img src="assets/img/logo.png"
alt="" class="img-fluid"></a-->

     <nav class="nav-menu d-none d-lg-block">
       <ul>
         <li class="active"><a href="http://localhost:5000">Home</a></li>
         <li><a href="#about">About</a></li>
         <li><a href="#contact">Contact</a></li>

       </ul>
     </nav><!-- .nav-menu -->

     <a href="http://localhost:5000/predict" class="get-started-btn scrollto">Get
Started</a>

   </div>
 </header><!-- End Header -->

 <!-- ======= Hero Section ======= -->
 <section id="hero" class="d-flex align-items-center">
```

```
  <div class="container">

    <div class="row">

        <div class="col-lg-6 d-flex flex-column justify-content-center pt-4 pt-lg-0
order-2 order-lg-1" data-aos="fade-up" data-aos-delay="200">

        <h1>Solution to Detect Phishing Websites  </h1>

        <h2>Be aware of what's happening with your confidential data</h2>

        <div class="d-lg-flex">

                    <a href="http://localhost:5000/predict" class="btn-get-started
scrollto">Get Started</a>

                        <a href="https://www.youtube.com/watch?v=jDDaplaOz7Q"
class="venobox btn-watch-video" data-vbtype="video" data-autoplay="true">
Watch Video <i class="icofont-play-alt-2"></i></a>

        </div>

      </div>

       <div class="col-lg-6 order-1 order-lg-2 hero-img" data-aos="zoom-in" data-
aos-delay="200">

                              <img src="https://www.technologyvisionaries.com/wp-
content/uploads/2020/01/bigstock-Data-Phishing-Hacker-Attack-t-319270852-
scaled.jpg" class="img-fluid animated" alt="">

      </div>

    </div>

  </div>


 </section><!-- End Hero -->


 <main id="main">




  <!-- ======= About Us Section ======= -->
  <section id="about" class="about">
    <div class="container" data-aos="fade-up">


      <div class="section-title">
        <h2>About</h2>
      </div>


      <div class="row content">
        <div class="col-lg-6">
          <p>
            Web service is one of the key communications software services for the
Internet. Web phishing is one of many security threats to web services on the
Internet.  Web phishing aims to steal private information, such as usernames,
passwords, and credit card details, by way of impersonating a legitimate entity.
          </p>


        </div>
        <div class="col-lg-6 pt-4 pt-lg-0">
          <p>
           The recipient is then tricked into clicking a malicious link, which can lead
to the installation of malware, the freezing of the system as part of a ransomware
attack or the revealing of sensitive information. It will lead to information
disclosure and property damage.
          </p>


        </div>
      </div>


    </div>
  </section><!-- End About Us Section -->

 <!-- ======= Cta Section ======= -->
  <section id="cta" class="cta">
   <div class="container" data-aos="zoom-in">

    <div class="row">
     <div class="col-lg-9 text-center text-lg-left">
      <h3>Check your Website</h3>
       <p>Understanding if the website is a valid one or not is important and
plays a vital role in securing the data. To know if the URL is a valid one or you are
information is at risk check your website. </p>
     </div>
     <div class="col-lg-3 cta-btn-container text-center">
                                     <a        class="cta-btn        align-middle"
href="http://localhost:5000/predict">Check your website</a>
     </div>
    </div>

   </div>
  </section><!-- End Cta Section -->

  <!-- ======= Services Section ======= -->
  <section id="services" class="services section-bg">
   <div class="container" data-aos="fade-up">

    <div class="section-title">
     <h2>Protect yourself from Phishing Attacks</h2>
```

```
          <p>As a report from the Anti-Phishing Working Group (APWG) revealed earlier this year, there has been a notable rise in the number phishing attacks. It’s a widespread problem, posing a huge risk to individuals and organizations.</p>

         <p>Follow the tips below and stay better protected against phishing attacks.</p>

       </div>


       <div class="row">

     <div class="col-xl-3 col-md-6 d-flex align-items-stretch mt-4 mt-xl-0" data-aos="zoom-in" data-aos-delay="400">

          <div class="icon-box">

            <div class="icon"><i class="bx bx-layer"></i></div>

            <h4>Browse securely with HTTPs</h4>

             <p>You should always, where possible, use a secure website (indicated by https:// and a security “lock” icon in the browser’s address bar) to browse, and especially when submitting sensitive information online, such as credit card details.</p>

          </div>

         </div>

        <div class="col-xl-3 col-md-6 d-flex align-items-stretch" data-aos="zoom-in" data-aos-delay="100">

          <div class="icon-box">


            <div class="icon"><i class="bx bxl-dribbble"></i></div>

            <h4>Watch out for shortened links</h4>

              <p>Cybercriminals often use these – from Bitly and other shortening services – to trick you into thinking you are clicking a legitimate link, when in fact you’re being inadvertently directed to a fake site.</p>

          </div>

         </div>
```

```
          <div class="col-xl-3 col-md-6 d-flex align-items-stretch mt-4 mt-md-0" data-aos="zoom-in" data-aos-delay="200">
            <div class="icon-box">
              <div class="icon"><i class="bx bx-file"></i></div>
              <h4>Does that email look suspicious? Read it again</h4>
              <p>Plenty of phishing emails are fairly obvious. They will be punctuated with plenty of typos, words in capitals and exclamation marks.</p>
            </div>
          </div>

          <div class="col-xl-3 col-md-6 d-flex align-items-stretch mt-4 mt-xl-0" data-aos="zoom-in" data-aos-delay="300">
            <div class="icon-box">
              <div class="icon"><i class="bx bx-tachometer"></i></div>
              <h4>Be wary of threats and urgent deadlines</h4>
              <p>Some of these threats may include notices about a fine, or advising you to do something to stop your account from being closed. Ignore the scare tactics and contact the company separately via a known and trusted channel.</p>
            </div>
          </div>




        </div>

      </div>
```

```
</section><!-- End Services Section -->
<!-- ======= Contact Section ======= -->
<section id="contact" class="contact">
  <div class="container" data-aos="fade-up">

    <div class="section-title">
      <h2>Contact</h2>
      <p>Get in contact with us to build many more amazing projects.</p>
    </div>

    <div class="row">

      <div class="col-lg-9 mt-5 mt-lg-0 d-flex align-items-stretch">
        <div class="info">
          <div class="address">
            <i class="icofont-google-map"></i>
            <h4>Location:</h4>
            <p>Guntur</p>
          </div>

          <div class="email">
            <i class="icofont-envelope"></i>
            <h4>Email:</h4>
            <p>gharika2004@gmail.com</p>
          </div>

          <div class="phone">
```

```
            <i class="icofont-phone"></i>

            <h4>Call:</h4>

            <p>9160509160</p>

           </div>


        <iframe width="100%" height="370" frameborder="0" allowfullscreen=""
style="border:0"
src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d15742.2
144952721!2d76.54167835902591!3d9.460481059436212!2m3!1f0!2f0!3f0!3
m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x3b0625fed0fbdc3d%3A0x92d3856e640
4922e!2sKristu+Jyoti+College+of+Management+and+Technology!5e0!3m2!1se
n!2sin!4v1460176408676" frameborder="0" style="border:0; width: 100%;
height: 290px;" allowfullscreen></iframe>


          </div>


        </div>


        <div class="col-lg-7 mt-5 mt-lg-0 d-flex align-items-stretch">



        </div>


       </div>


     </div>

    </section><!-- End Contact Section -->


  </main><!-- End #main -->
```

```
<!-- ======= Footer ======= -->
<footer id="footer">



  <div class="footer-top">
    <div class="container">
      <div class="row">


      </div>
    </div>
  </div>

  <div class="container footer-bottom clearfix">

</footer><!-- End Footer -->

<a href="#" class="back-to-top"><i class="ri-arrow-up-line"></i></a>
<div id="preloader"></div>

<!-- Vendor JS Files >
<script src="assets/vendor/jquery/jquery.min.js"></script>
<script src="assets/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>
<script src="assets/vendor/jquery.easing/jquery.easing.min.js"></script>
<script src="assets/vendor/php-email-form/validate.js"></script>
```

```
<script src="assets/vendor/waypoints/jquery.waypoints.min.js"></script>
<script src="assets/vendor/isotope-layout/isotope.pkgd.min.js"></script>
<script src="assets/vendor/venobox/venobox.min.js"></script>
<script src="assets/vendor/owl.carousel/owl.carousel.min.js"></script>
<script src="assets/vendor/aos/aos.js"></script-->

<!-- Template Main JS File >
<script src="assets/js/main.js"></script-->
 <script src="{{ url_for('static',filename='assets/js/main.js') }}"></script>

<script src="{{ url_for('static',filename='vendor/jquery/jquery.min.js') }}"></script>
<script src="{{ url_for('static',filename='vendor/bootstrap/js/bootstrap.bundle.min.js') }}"></script>
<script src="{{ url_for('static',filename='vendor/jquery.easing/jquery.easing.min.js') }}"></script>
<script src="{{ url_for('static',filename='vendor/php-email-form/validate.js') }}"></script>
<script src="{{ url_for('static',filename='vendor/waypoints/jquery.waypoints.min.js') }}"></script>
<script src="{{ url_for('static',filename='vendor/isotope-layout/isotope.pkgd.min.js') }}"></script>
<script src="{{ url_for('static',filename='vendor/owl.carousel/owl.carousel.min.js') }}"></script>
<script src="{{ url_for('static',filename='vendor/aos/aos.js') }}"></script>
<script src="{{ url_for('static',filename='vendor/venobox/venobox.min.js') }}"></script>
```

```
  <script src="{{ url_for('static',filename='js/main.js') }}"></script>

</body>

</html>
```

PREDICTION Page-

```
<!DOCTYPE html>
<html >
<!--From https://codepen.io/frytyler/pen/EGdtg-->
<head>
  <meta charset="UTF-8">
  <title>Prediction</title>
    <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
      <link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
   <link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
   <link rel="stylesheet" href="{{ url_for('static', filename='css/final1.css') }}">
     <!--link rel="stylesheet" href="G:\Gayatri Files\Smartbridge\Nidhi\Phishing Website\static\css\style1.css"-->
```

```
<style>
.login{
top: 20%;
}
</style>
</head>

<body>
<div class="header">
<div>AntiPhish</div>
   <ul>



      <li><a href="http://127.0.0.1:5000/#contact">Contact</a></li>
      <li><a href="http://localhost:5000/#about">About</a></li>
      <li><a href="http://localhost:5000">Home</a></li>
   </ul>
</div>

<div class="main">
<h1>Phishing Website Detection using Machine Learning<h1>
</div>
<form action="{{ url_for('predict')}}"method="post">
      <input type="text" name="URL" placeholder="Enter the URL to be verified"
required="required" />
```

```
            <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
        </form>
        <br>
        <br>

            <div id='result',class='result' style='color:black;font-size:30px;'>{{ prediction_text }}</div>
        <a href=" {{ url }} "> {{ url }} </a>
</body>

</html>
```

GitHub & Project Demo Link:

Project Demo Link-

https://drive.google.com/file/d/15f05QewTDGzOII_Rx863N57sHW42q5Lp/view?usp=sharing

Github link-

Garikapati-Harika/Detection-of-phishing-websites-from-urls