PROJECT-1 PHISHING WEBSITE DETECTION TOOL

Objective: To design and implement a tool capable of identifying phishing websites based on specific URL patterns and characteristics.

Task Description: The goal is to create a script that accepts a list of URLs, analyzes them for known phishing indicators, and categorizes them as either 'Legitimate' or 'Phishing'. The task is focused on practical detection without using machine learning, instead relying on rule-based analysis.

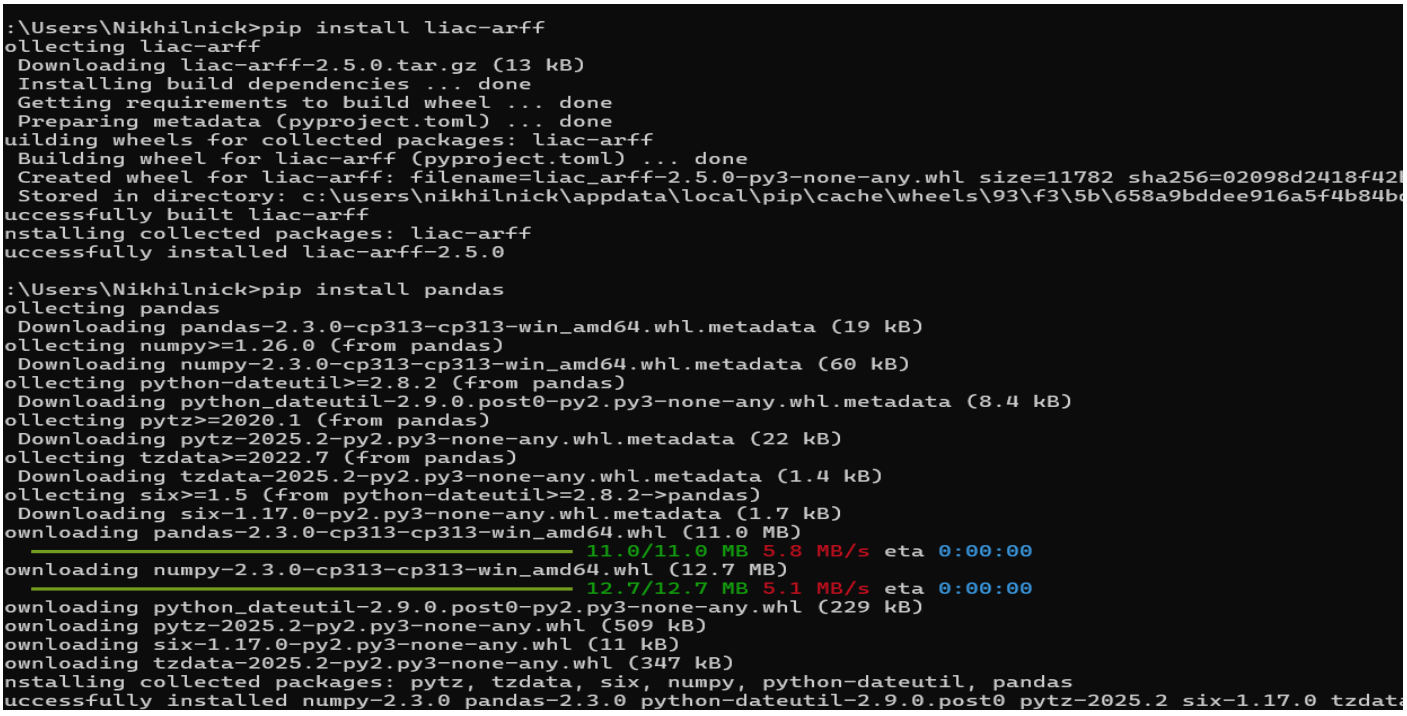
Procedure

First we need to download a dataset from a website

https://archive.ics.uci.edu/dataset/327/phishing+websites.

Now go to the location where the data set is downloaded and copy the path and open the command prompt and go to that location now run the command install liac-arff

The dataset is in **ARFF format** (Attribute-Relation File Format), which is commonly used in Weka — but **not natively supported** by pandas or most Python libraries



Without this library, Python can't understand .arff files easily.

Imports the pandas library — used for handling and analyzing data in table (DataFrame) format. It's like Excel for Python but more powerful.

Create a new file named something like phishing_loader.py and write a python code .

Save and run the file by opening a terminal and typing py phishing_loader.py.

```
NameError: name 'df' is not defined
>>> import arff
... import pandas as pd
...
... # Step 1: Load the ARFF file from the extracted folder
... with open("C:/Users/Nikhilnick/Downloads/phishing+websites/Training Dataset.arff", 'r') as f:
...     data = arff.load(f)
...
... # Step 2: Extract attribute names
... columns = [attr[0] for attr in data['attributes']]
...
... # Step 3: Create a DataFrame
... df = pd.DataFrame(data['data'], columns=columns)
...
... # Step 4: Show the first 5 rows
... print(df.head())
...
  having_IP_Address URL_Length Shortining_Service  ... Links_pointing_to_page Statistical_report Result
0                -1          1                  1  ...                      1                -1     -1
1                 1          1                  1  ...                      1                 1     -1
2                 1          0                  1  ...                      0                -1     -1
3                 1          0                  1  ...                     -1                 1     -1
4                 1          0                 -1  ...                      1                 1      1

[5 rows x 31 columns]
```

Now run this commands given below:

```
from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score

# Step 1: Separate features and label

X = df.drop('Result', axis=1)  # all input columns

y = df['Result']              # output column


# Step 2: Split into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# Step 3: Create and train the model
```

model = RandomForestClassifier()

model.fit(X_train, y_train)

Step 4: Make predictions and check accuracy

y_pred = model.predict(X_test)

print("Model Accuracy:", accuracy_score(y_test, y_pred))

```
SyntaxError: invalid syntax
>>> from sklearn.model_selection import train_test_split
... from sklearn.ensemble import RandomForestClassifier
... from sklearn.metrics import accuracy_score
...
>>> from sklearn.model_selection import train_test_split
... from sklearn.ensemble import RandomForestClassifier
... from sklearn.metrics import accuracy_score
...
>>> # Step 1: Separate features and label
... X = df.drop('Result', axis=1)  # all input columns
... y = df['Result']               # output column
...
... # Step 2: Split into training and testing sets
... X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
...
... # Step 3: Create and train the model
... model = RandomForestClassifier()
... model.fit(X_train, y_train)
...
... # Step 4: Make predictions and check accuracy
... y_pred = model.predict(X_test)
... print("Model Accuracy:", accuracy_score(y_test, y_pred))
...
Model Accuracy: 0.9656264133876075
>>>
```

The above commands of sklearn will Training set (to teach the model) ,Testing set (to evaluate its accuracy), Learns patterns in the data,Figures out which features (e.g., IP address, URL length, etc.) are important, helps you measure how good your model is using metrics like:

- Accuracy
- Precision
- Recall (you used accuracy)

Now create a file in the same folder train_model.py.

Run it in command prompt and given input values to it for predicitions.

```
C:\Users\Nikhilnick\OneDrive\Desktop>py train_model.py
✅ Model trained and saved as 'phishing_model.pkl'

C:\Users\Nikhilnick\OneDrive\Desktop>py optional.py
Please enter the following feature values (only -1, 0, or 1):
having_IP_Address: -1
URL_Length: 0
Shortining_Service: 1
having_At_Symbol: @
Invalid input. Please enter an integer (-1, 0, or 1).
having_At_Symbol: /
Invalid input. Please enter an integer (-1, 0, or 1).
having_At_Symbol: 1
double_slash_redirecting: -1
Prefix_Suffix: 0
having_Sub_Domain: 1
SSLfinal_State: 0
Domain_registeration_length: 1
Favicon: 1
port: 21
```

```
Favicon: 1
port: 21
Enter only -1, 0, or 1.
port: 0
HTTPS_token: 1
Request_URL: 0
URL_of_Anchor: -1
Links_in_tags: 1
SFH: 1
Submitting_to_email: 1
Abnormal_URL: -1
Redirect: -1
on_mouseover: -1
RightClick: 1
popUpWidnow: 0
Iframe: 0
age_of_domain: 1
DNSRecord: 0
web_traffic: 1
Page_Rank: -1
Google_Index: 0
Links_pointing_to_page: -1
Statistical_report: 1
C:\Users\Nikhilnick\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\utils\validation.py:2749: UserWarn
ing: X does not have valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(

🔍 Prediction: Legitimate ✅

C:\Users\Nikhilnick\OneDrive\Desktop>
```

It is ready for predicitions and it will generate a phishing_model.pkl file.

Now create a file called optional.py and add the python code given below.

import joblib

Load the saved model

```
model = joblib.load('phishing_model.pkl')

# Column order (must match the training set exactly)
features = [
    'having_IP_Address', 'URL_Length', 'Shortining_Service', 'having_At_Symbol',
    'double_slash_redirecting', 'Prefix_Suffix', 'having_Sub_Domain', 'SSLfinal_State',
    'Domain_registeration_length', 'Favicon', 'port', 'HTTPS_token', 'Request_URL',
    'URL_of_Anchor', 'Links_in_tags', 'SFH', 'Submitting_to_email', 'Abnormal_URL',
    'Redirect', 'on_mouseover', 'RightClick', 'popUpWidnow', 'Iframe', 'age_of_domain',
    'DNSRecord', 'web_traffic', 'Page_Rank', 'Google_Index', 'Links_pointing_to_page',
    'Statistical_report'
]

# Collect user input
print("Please enter the following feature values (only -1, 0, or 1):")
user_input = []
for feature in features:
    while True:
        try:
            value = int(input(f"{feature}: "))
            if value in [-1, 0, 1]:
                user_input.append(value)
                break
            else:
                print("Enter only -1, 0, or 1.")
        except ValueError:
```

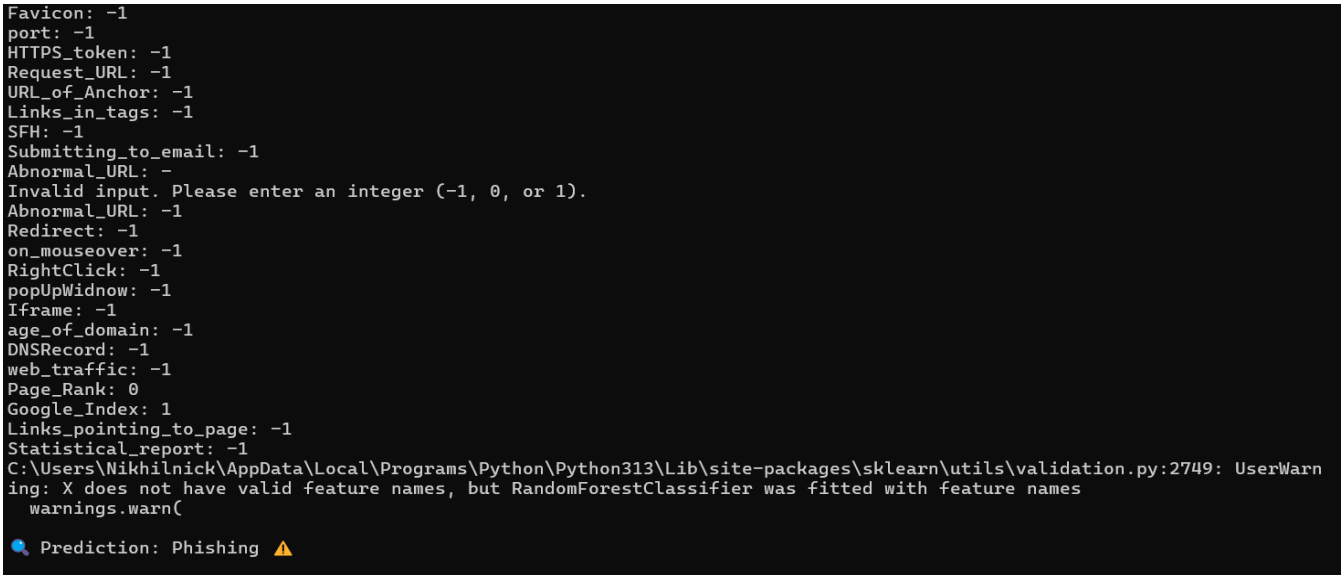
```
        print("Invalid input. Please enter an integer (-1, 0, or 1).")


# Predict using the loaded model
prediction = model.predict([user_input])


# Display result
print("\n🔍 Prediction:", "Legitimate ✅" if prediction[0] == 1 else "Phishing ⚠️")
```

now run the file in command prompt.

```
C:\Users\Nikhilnick\OneDrive\Desktop>py train_model.py
✅ Model trained and saved as 'phishing_model.pkl'

C:\Users\Nikhilnick\OneDrive\Desktop>py optional.py
Please enter the following feature values (only -1, 0, or 1):
having_IP_Address: -1
URL_Length: 0
Shortining_Service: 1
having_At_Symbol: @
Invalid input. Please enter an integer (-1, 0, or 1).
having_At_Symbol: /
Invalid input. Please enter an integer (-1, 0, or 1).
having_At_Symbol: 1
double_slash_redirecting: -1
Prefix_Suffix: 0
having_Sub_Domain: 1
SSLfinal_State: 0
Domain_registeration_length: 1
Favicon: 1
port: 21
```

Give the input values to know whether it is legitimate or phishing.

The input should be -1 or 0 or 1 because machine can learn through digits only it cannot understand the languages.



It is predicting correctly .

Now create a new file(url_predictor.py) for advanced features

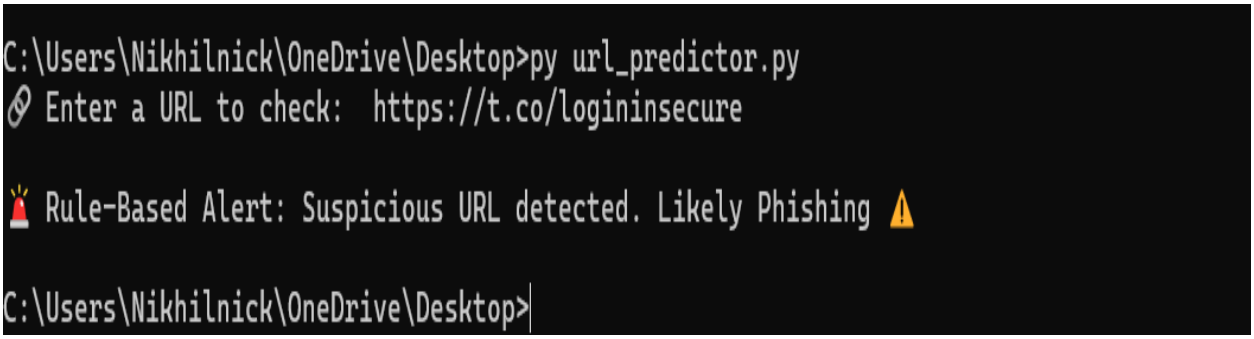
This script demonstrates a practical and efficient approach to phishing website detection by:

1. **Leveraging rule-based logic** to quickly identify suspicious URLs based on obvious red flags like:
    - Use of shortening services (e.g., bit.ly)
    - Presence of @ symbols
    - IP addresses instead of domain names

2. **Applying a trained machine learning model** (phishing_model.pkl) for deeper analysis of more complex features extracted from the URL, improving accuracy where rule-based logic alone may not suffice.

3. **Ensuring auditability and record-keeping** by logging every scan result with a timestamp, URL, and prediction outcome in a CSV file.

Overall, this tool offers a reliable and scalable solution for phishing detection, suitable for integration into broader security systems, browser extensions, or standalone utilities.



When we run this it show the result .

the above code will generate the scan_results.csv file (if it doesn't already exist) and append one row containing the:

- **Current timestamp**
- **Scanned URL**
- **Result** (e.g., "Phishing" or "Legitimate").

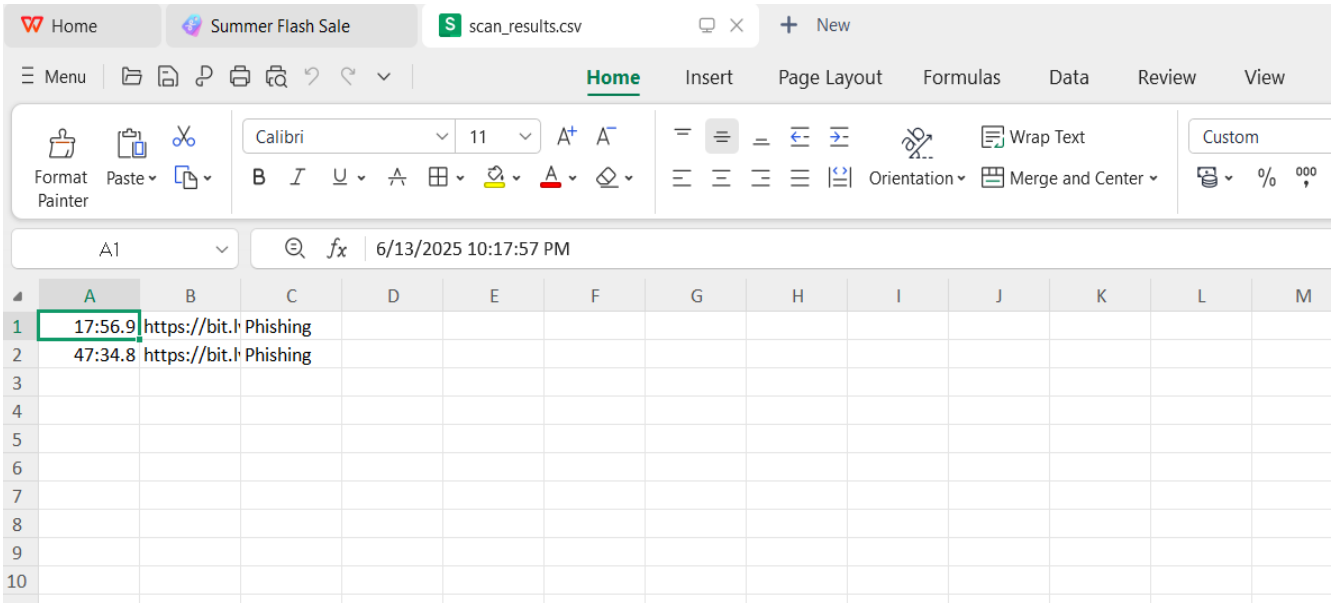
Now last step is to create a file and add the below python code.

```
import csv
from datetime import datetime

url = "https://bit.ly/pay-now"
result = "Phishing"

with open("scan_results.csv", "a", newline="") as file:
    writer = csv.writer(file)
    writer.writerow([datetime.now(), url, result])

print("✅ Scan result saved to scan_results.csv").
```

now run this code in command prompt .

```
C:\Users\Nikhilnick\OneDrive\Desktop>py logger.py
✅ Scan result saved to scan_results.csv

C:\Users\Nikhilnick\OneDrive\Desktop>
```

we can see the results in scan_results.csv.



In this way we can detect the phishig links by using the scripts.

Conclusion:

The Phishing Website Detection Tool is a successful demonstration of combining **rule-based heuristics** with **machine learning** to accurately detect potentially malicious websites. By extracting key features from URLs and applying logical checks along with a trained model (phishing_model.pkl), the tool provides a fast and effective way to classify websites as *Legitimate* or *Phishing*.

The inclusion of:

- **Manual feature-based input prediction**
- **Automated URL feature extraction**
- **Rule-based overrides for high-risk indicators**

- **Result logging into a CSV file**shows a well-rounded approach to building an intelligent, auditable, and extensible detection system.

This project has strengthened core cybersecurity skills like:

- URL analysis
- Threat detection
- Python scripting
- Machine learning application in real-world security use cases

It also lays the foundation for further enhancements such as real-time scanning, integration with web browsers, deployment as a browser extension, or hosting it as an API-based tool.