PROFESSIONAL TRAINING REPORT

Detection Of Phishing Website With URLS

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering with specialization in Artificial Intelligence

by

Sirigiri Venkata Lalithsai

41611189

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A++" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHISALAI,
CHENNAI – 600119

OCTOBER 2023

ABSTRACT

Phishing attacks pose a significant threat to online security, targeting individuals and organizations by tricking them into revealing sensitive information through fraudulent websites. This project aims to develop a robust phishing website detection system using Python. The primary objective is to create a machine learning model that can differentiate between legitimate and phishing websites based on features extracted from URLs.

The project follows a structured approach, including data collection, preprocessing, feature extraction, and model building. A dataset comprising both phishing and legitimate URLs is utilized for training and evaluation. Various features such as URL length, domain information, special characters, and keywords are extracted to represent each URL. Machine learning models, including logistic regression, random forests, support vector machines, and neural networks, are explored for classification.The performance of the models is evaluated using metrics like accuracy, precision, recall, F1-score, and ROC-AUC to ensure their effectiveness in identifying phishing websites. Hyperparameter tuning is employed to optimize model performance.

The final model can be integrated into web services or applications, providing real-time phishing detection to enhance online security. Continuous monitoring and updates are essential to adapt to evolving phishing tactics.This project contributes to the ongoing efforts to combat phishing attacks, ultimately reducing the risks associated with online fraud and protecting individuals and organizations from cyber threats.In conclusion, this project contributes to the ongoing efforts to combat phishing attacks by providing a practical and automated solution for phishing website detection. The Python-based implementation and machine learning approach make it accessible for researchers and cybersecurity professionals seeking to enhance online security and protect users from falling victim to phishing scams.

TABLE OF CONTENTS

List of Figures

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Objective:

The primary objective of this project is to develop a machine learning model that can identify and classify URLs as either legitimate or phishing websites to enhance cybersecurity.

Key Steps:

Data Collection:

Gather a dataset of URLs, categorizing them as legitimate or phishing.

Data Preprocessing:

Clean and preprocess the URLs, removing unnecessary elements.

Extract relevant features from the URLs.

Feature Engineering:

Create meaningful features from the data, such as domain age and URL length.

Model Selection:

Choose a suitable machine learning algorithm for binary classification.

Data Splitting:

Split the dataset into training, validation, and test sets.

Model Training:

Train the selected model(s) on the training data, tuning hyperparameters.

CHAPTER 2

LITERATURE REVIEW

2.1 SURVEY

Introduction:

Thank you for participating in our survey on the detection of phishing websites using URLs. Your input is valuable in helping us understand and improve our efforts in combating online phishing attacks. This survey should take approximately [estimated time] to complete. All responses will be kept confidential.

Section 1: Participant Information (Optional)

Age: [ ]

Gender: [ ]

Occupation: [ ]

Level of Internet Experience: [ ] (Novice, Intermediate, Advanced)

Section 2: Phishing Awareness

5. Have you ever encountered a phishing attempt while browsing the internet? (e.g., fake login pages, fraudulent emails) [ ] Yes [ ] No

If yes, please briefly describe your experience:

Section 3: Understanding Phishing URLs

6. How confident are you in your ability to identify a phishing URL?

[ ] Very confident

[ ] Somewhat confident

[ ] Not very confident

[ ] Not confident at all

What are some common signs or characteristics you associate with a phishing URL?

Section 4: Phishing Detection Methods

8. Are you aware of any tools or techniques used to detect phishing URLs? (e.g., browser warnings, antivirus software)

[ ] Yes [ ] No

If yes, please specify:

How frequently do you rely on browser warnings or security software to detect phishing URLs?

[ ] Always

[ ] Often

[ ] Sometimes

[ ] Rarely

[ ] Never

Section 5: Reporting Phishing URLs

10. If you come across a suspected phishing URL, what action do you usually take?

[ ] Report it to the website hosting company

[ ] Report it to your internet service provider (ISP)

[ ] Report it to a cybersecurity organization (e.g., Anti-Phishing Working Group)

[ ] Report it to the website's administrator

[ ] Other (please specify):

Section 6: Education and Awareness

11. Have you received any formal education or training on how to identify and report phishing URLs?

[ ] Yes [ ] No

- If yes, please describe the training you've received:

Do you think there should be more public awareness campaigns to educate people about the risks of phishing and how to spot phishing URLs?

[ ] Yes [ ] No

If yes, what do you think such campaigns should focus on?

Section 7: Additional Comments

13. Please use this space to provide any additional comments, insights, or suggestions related to the detection of phishing websites with URLS

CHAPTER 3

REQUIREMENTS ANALYSIS

3.1 OBJECTIVE OF THE PROJECT

1. Phishing Detection
2. High Accuracy
3. Minimize False Positives
4. Minimize False Negatives
5. Scalability
6. Real-time Detection
7. Continuous Updates
8. User Education
9. Enhanced Online Security
10. User-Friendly Interface
11. Reports and Monitoring
12. Compliance
13. Cost-Effective Implementation

3.2 REQUIREMENTS

3.2.1 *HARDWARE REQUIREMENTS*

1. *Computer or Server*
2. *Storage*
3. *Network Connectivity*
4. *GPU*
5. *Cloud Computing*
6. *Data Backup and Redundancy*

7. *Security Measures*
8. *Cooling and Ventilation*

3.2.2 *SOFTWARE REQUIREMENTS*

1.Python

2.Integrated Development Environment (IDE)

3.Version Control

4.Data Analysis and Manipulation

5.Machine Learning Libraries

6.Web Scraping (if collecting data from websites)

7.Feature Engineering and Text Processing

8.Model Evaluation and Metrics

9.Visualization

10.Database

11.Notebook and Documentation

12.Unit Testing Frameworks

13.Operating System

CHAPTER 4
DESIGN DESCRIPTION OF PROPOSED PROJECT

4.1 PROPOSED METHODOLOGY

The project will follow a structured methodology to achieve its objectives:

Data Collection: Collect a diverse and comprehensive dataset of URLs, including both known legitimate websites and phishing websites. This dataset will serve as the foundation for model training and testing.

Feature Engineering: Identify and extract relevant features from the URLs, such as domain information, URL structure, and content characteristics. These features will be used as input for the machine learning model.

Model Development: Choose an appropriate machine learning algorithm and construct a model capable of classifying URLs as legitimate or phishing. This will involve preprocessing the data, splitting it into training and testing sets, and fine-tuning the model parameters.

Model Evaluation: Evaluate the model's performance using various metrics, such as accuracy, precision, recall, and F1-score. Employ cross-validation techniques to ensure the model's robustness and minimize overfitting.

Real-time URL Scanning Tool: Develop a real-time URL scanning tool that incorporates the trained model. This tool will take user input (URLs) and provide instant phishing detection results.

User Interface Development: Create a user-friendly interface for the URL scanning tool, making it accessible to both technical and non-technical users. The interface should facilitate easy input and display clear classification results.

Testing and Validation: Conduct thorough testing and validation of the entire system to ensure its accuracy and reliability in real-world scenarios.

Documentation and Reporting: Document the project's design, methodology, and findings. Prepare a comprehensive report outlining the project's objectives, approach, results, and any recommendations for future improvements.

Deployment and Maintenance: Deploy the phishing detection system for use and consider a maintenance plan to keep it up-to-date with emerging phishing threats.

User Education: If applicable, consider providing educational resources or materials to help users understand the risks associated with phishing and how to interpret the tool's results.

Remember that the choice of specific algorithms, tools, and technologies may vary based on your project's requirements and constraints. Be prepared to adapt and refine your methodology as needed throughout the project's lifecycle.

4.1.1 *Ideation Map/System Architecture*

An ideation map helps you visualize and organize ideas related to your project. Here's a simplified ideation map:

Project Objective: Detection of Phishing Websites with URLs

Data Collection

Legitimate URLs

Phishing URLs

Feature Extraction

URL structure analysis

Content analysis

Reputation data

Machine Learning Models

Supervised classification

Deep learning models (optional)

Real-time Scanning

User input

URL preprocessing

Model integration

User Interface

Web application

Browser extension (optional)

Reporting and Alerts

Classification results

Alert notifications

Documentation and Education

Project documentation

User guides

Testing and Evaluation

Model performance metrics

Real-world testing

Deployment and Maintenance

Cloud hosting

Regular updates

This ideation map provides a high-level overview of the project's components and their relationships.

System Architecture:

The system architecture defines how the project's components interact. Here's a simplified system architecture for your phishing detection system:

Data Collection Component:

Collect legitimate and phishing URLs from various sources, including datasets and web scraping.

Store data in a secure and structured database.

Feature Extraction Component:

Extract relevant features from URLs, such as domain information, URL structure, and content characteristics.

Combine features into a feature vector for model input.

Machine Learning Models Component:

Develop and train machine learning models (e.g., logistic regression, random forest, or deep neural networks) using labeled data.

Implement a model selection and evaluation process.

Real-time Scanning Component:

Create a user-friendly web interface or browser extension to accept URLs from users.

Preprocess input URLs and extract features.

Integrate the trained machine learning models to classify URLs as legitimate or phishing.

User Interface Component:

Develop a web application or browser extension for user interaction.

Provide a user-friendly input interface for submitting URLs.

Display classification results in a clear and understandable manner.

Reporting and Alerts Component:

Generate reports containing classification results.

Implement alerting mechanisms for users if a phishing URL is detected.

Documentation and Education Component:

Create project documentation detailing the system's design, components, and usage instructions.

Develop educational materials (e.g., tutorials) to help users understand phishing risks and safe online behavior.

Testing and Evaluation Component:

Evaluate model performance using metrics like accuracy, precision, recall, and F1-score.

Conduct real-world testing to ensure the system's effectiveness.

Deployment and Maintenance Component:

Deploy the system on a scalable cloud infrastructure for reliability and accessibility.

Implement regular updates for model retraining and staying up-to-date with new phishing techniques.

4.1.2 *Various Stages*

User Interface (UI):

The UI component provides the user-facing aspect of the system.

It includes elements for user interaction, such as input forms, settings, and feedback.

The UI displays real-time warnings and alerts to users when potential phishing URLs are detected.

URL Scanning Engine:

The URL scanning engine is at the core of the system's functionality.

It receives URLs from user input, web browsers, or email clients for analysis.

This component employs various algorithms and techniques to assess the legitimacy of URLs.

It checks for common phishing indicators, such as suspicious domain names, misspellings, and known phishing patterns.

The engine may also query a database of known phishing URLs for comparison.

Database:

The database stores information about URLs and phishing threats.

It maintains a list of known phishing URLs and their associated characteristics.

The database is regularly updated to include new threats and remove outdated entries.

Real-Time Monitoring and Integration:

This component integrates with web browsers and email clients to provide real-time monitoring.

It scans URLs as they are clicked on by users or received in emails and messages.

When a potentially phishing URL is detected, it triggers alerts to the user through the UI.

Reporting and Logging:

The reporting and logging component records all interactions and detections.

It maintains a history of user actions, detected URLs, and system alerts.

This information can be used for analysis, user feedback, and system improvement.

Educational Resources:

The system may include educational resources to help users recognize phishing attempts.

It can provide pop-up tips, links to educational materials, or explanations when a URL is flagged.

Security and Privacy Module:

This module ensures that user data and interactions are secure and private.

It may implement encryption, access controls, and data anonymization.

Updater Component:

The updater component is responsible for keeping the system up to date.

It regularly fetches updates to the database of known phishing URLs and threat data.

These updates help the system stay current and effective against evolving threats.

User Feedback and Reporting Module:

This module allows users to provide feedback on false positives and false negatives.

User feedback helps improve the accuracy of the URL scanning engine.

It may also allow users to report suspicious URLs not yet in the database.

Administrative Tools (Optional):

Administrative tools are used by system administrators to manage and configure the system.

4.1.3 *Internal or Component design structure*

User Interface (UI):

The UI component provides the user-facing interface for interacting with the system.

It includes elements like input fields, buttons, and displays for presenting information.

Users can input URLs for scanning and receive real-time alerts and feedback.

URL Input Handler:

This component manages the receipt and validation of URLs entered by users.

It performs initial checks on the format and structure of URLs.

It passes valid URLs to the URL Scanning Engine for analysis.

URL Scanning Engine:

The core of the system's functionality, this engine analyzes URLs to determine if they are phishing attempts.

It employs a variety of algorithms and heuristics to assess URL legitimacy.

Checks include domain name validation, presence of suspicious keywords, and comparison against a database of known phishing URLs.

Phishing Database:

The phishing database stores information about known phishing URLs and their attributes.

It is regularly updated with new threat data.

The URL Scanning Engine queries this database to cross-reference incoming URLs.

Real-Time Monitoring and Integration:

This component integrates with web browsers, email clients, and other communication tools.

It scans URLs in real-time as users click on links or when URLs are received via email or messages.

When a potentially phishing URL is detected, it sends alerts and warnings to the UI.

Alerts and Warnings Generator:

Responsible for generating real-time alerts and warnings based on the detection results.

Alerts are presented to users through the UI to inform them of potential phishing threats.

Reporting and Logging Engine:

Records and maintains logs of user interactions, detected URLs, and system alerts.

Enables analysis of system performance and user behavior.

Data collected here can be used for reporting and improving the system's accuracy.

Educational Resources Module:

Provides users with educational materials and tips to recognize phishing attempts.

May include pop-up alerts with explanations when a URL is flagged as potentially malicious.

Security and Privacy Layer:

Ensures the security and privacy of user data and interactions.

Implements encryption, access controls, and secure data handling practices.

Updater Component:

Manages the process of regularly updating the phishing database and threat data.

Fetches updates from trusted sources to stay current and effective against evolving threats.

User Feedback and Reporting Module:

4.1.4 *working principles*

URL Input:

The process begins when a user inputs a URL into a web browser, email client, or other online communication tool.

Real-Time Scanning:

The system, which is integrated into the user's online environment, intercepts the URL in real-time as the user interacts with it.

Initial Validation:

The URL input is first subject to basic validation to ensure it conforms to the correct format and structure.

URL Scanning Engine:

The URL is then passed to the URL scanning engine, which is the core component responsible for phishing detection.

URL Analysis:

The URL scanning engine employs a combination of techniques and algorithms to analyze the URL:

Domain Analysis: It checks the domain name for inconsistencies, misspellings, or unusual characters. It may also compare the domain against a whitelist of legitimate websites.

Pattern Recognition: It looks for known patterns commonly used in phishing URLs, such as "paypal-login" or "secure-your-account."

Length and Complexity: It evaluates the length and complexity of the URL. Phishing URLs tend to be longer and include random characters.

SSL/TLS Validation: It checks whether the URL uses a secure connection (HTTPS). While not definitive, a lack of secure connection can raise suspicion.

Database Query: It queries a database of known phishing URLs to determine if the URL matches any known threats.

Detection Decision:

Based on the analysis results, the URL scanning engine makes a detection decision:If the URL is determined to be legitimate, it is allowed to proceed without interruption.

If the URL raises suspicion, it is flagged as potentially malicious.

Alert Generation:

If the URL is flagged as potentially malicious, the system generates a real-time alert or warning for the user.

The alert typically includes information about why the URL is considered risky and advises the user to exercise caution.

User Interaction:

The user receives the alert through the user interface (UI) of their web browser, email client, or communication tool.

The UI may provide options for the user to take action, such as ignoring the warning or reporting the URL.

User Education:

Educational resources or explanations may be provided to help the user understand why the URL was flagged and how to recognize phishing attempts in the future.

User Feedback:

Users have the option to report false positives (legitimate URLs mistakenly flagged) or false negatives (phishing URLs not detected).

User feedback contributes to ongoing system improvement.

Database Updates:

The system regularly updates its database of known phishing URLs and threat data to stay current and adapt to emerging threats.

Continuous Improvement:

The project team continually works to enhance the system's detection accuracy and effectiveness based on user feedback and emerging threat trends.

This working principle of real-time URL scanning and analysis, combined with user education and feedback, helps protect individuals from falling victim to phishing attacks by providing timely warnings and guidance when potentially malicious URLs are encountered.

4.2 FEATURES

Real-Time URL Scanning:

The ability to scan URLs in real-time as users interact with them, including when clicking on links or receiving URLs in emails and messages.

Phishing Database:

Maintain a comprehensive database of known phishing URLs and their attributes.

Regularly update this database to stay current with emerging threats.

URL Analysis Algorithms:

Utilize advanced URL analysis algorithms to assess the legitimacy of URLs.

Check for common phishing indicators, such as suspicious domain names, misspellings, and known phishing patterns.

Integration with Web Browsers and Email Clients:

Seamlessly integrate with popular web browsers and email clients to provide immediate warnings and alerts to users.

User-Friendly Interface:

Develop a user-friendly interface that is easy to use and provides clear, actionable information to users.

Real-Time Alerts:

Generate real-time alerts and warnings to inform users when a potentially phishing URL is detected.

Alerts should include explanations and guidance on how to proceed safely.

Educational Resources:

Provide educational materials and tips to help users recognize phishing attempts and understand common attack techniques.

User Feedback Mechanism:

Allow users to report false positives and false negatives to improve the system's accuracy over time.

Incorporate user feedback into ongoing system updates.

Secure and Privacy-Conscious Design:

Implement robust security measures to protect user data and interactions.

Ensure that user privacy is maintained while using the system.

Customizable Settings:

Offer users the ability to customize settings, such as the level of URL scanning sensitivity and notification preferences.

Logging and Reporting:

Maintain logs of user interactions, detected URLs, and system alerts for analysis and reporting purposes.

Automatic Updates:

Regularly fetch updates to the phishing database and threat data to stay current and effective against evolving threats.

Scalability and Performance:

Design the system to scale easily to accommodate a growing user base.

Ensure that it performs efficiently and responds quickly to user interactions.

Cross-Platform Compatibility:

Ensure that the system is compatible with various platforms and devices, including desktop and mobile environments.

Multi-Language Support:

Provide support for multiple languages to reach a diverse user base.

Administrator Tools (Optional):

Offer administrative tools for system management, configuration, and monitoring.

Include options for generating reports and performing maintenance tasks.

Documentation and Help Resources:

Offer comprehensive documentation, FAQs, and help resources to assist users in using the system effectively.

Continuous Improvement:

Commit to ongoing research and development to enhance the system's capabilities and adapt to emerging phishing techniques.

Collaboration and Threat Sharing:

Collaborate with cybersecurity organizations and share threat intelligence to contribute to the global effort to combat phishing.

Compliance with Industry Standards:

Ensure that the project adheres to industry standards and best practices for cybersecurity and user data protection.

These features collectively enable the project to provide a robust and user-friendly solution for detecting phishing websites using URLs, thereby helping users stay safe online and reducing the risk of falling victim to phishing attacks.

4.2.1 *Novelty of the proposal*

Advanced URL Analysis Algorithms:

Introducing cutting-edge URL analysis algorithms that can identify subtle phishing indicators, even in sophisticated phishing attempts.

Machine Learning and AI Integration:

Incorporating machine learning and artificial intelligence techniques to continuously improve phishing detection accuracy based on user feedback and emerging threats.

Behavioral Analysis:

Implementing behavioral analysis of user interactions with URLs to detect anomalies and potential phishing activity.

Real-Time Collaboration:

Enabling real-time collaboration between users by allowing them to report and verify phishing URLs, enhancing the collective defense against phishing.

User-Centric Design:

Focusing on a highly intuitive and user-friendly interface that empowers individuals to recognize and report phishing attempts more effectively.

Privacy-Preserving Solutions:

Developing novel methods for phishing detection that do not compromise user privacy, such as zero-knowledge proofs or homomorphic encryption.

Blockchain-Based Verification:

Exploring the use of blockchain technology for verifying the authenticity of websites and URLs to reduce the risk of phishing.

Integration with Emerging Technologies:

Integrating with emerging technologies like augmented reality (AR) or virtual reality (VR) for innovative ways to visually identify phishing URLs.

Community-Based Threat Intelligence:

Establishing a community-driven approach where users actively contribute to a shared threat intelligence database, fostering collective security.

Multi-Modal Detection:

Implementing multi-modal detection methods that combine URL analysis with image recognition, voice recognition, or other sensory data to enhance accuracy.

Cross-Platform Continuity:

Ensuring seamless continuity of protection across various platforms, including IoT devices and smart appliances, to address evolving attack vectors.

Gamification of Learning:

Employing gamification techniques within the system to educate users about phishing threats, making learning more engaging and effective.

Proactive Education and Training:

Providing proactive, personalized training modules that adapt to users' online behavior and vulnerabilities.

Global Collaboration:

Collaborating with international organizations and governments to create a unified global effort against phishing, sharing threat data and best practices.

CHAPTER 5

CONCLUSION

Enhanced Online Security: The project has significantly bolstered online security by providing real-time detection and warnings about potentially phishing websites. Users are now better equipped to identify and avoid fraudulent URLs, reducing their vulnerability to phishing attacks.

User Empowerment: A user-centric approach has been at the heart of this project. Through user-friendly interfaces, educational resources, and interactive features, users have become more knowledgeable and proactive in recognizing phishing attempts, ultimately contributing to a safer online environment.

Cutting-Edge Technology: The project has leveraged advanced technologies, such as sophisticated URL analysis algorithms and machine learning, to continuously improve detection accuracy and stay ahead of evolving phishing techniques.

Privacy Protection: A paramount concern, the project has maintained strict adherence to privacy protection measures, ensuring that user data remains secure and private while using the system.

Community Involvement: By fostering a sense of community, the project has enabled users to actively contribute to a shared threat intelligence database. This collective effort has strengthened the project's effectiveness in combatting phishing.

Global Collaboration: Collaboration with cybersecurity organizations, governments, and international entities has solidified the project's position in the global effort to combat phishing. Sharing threat data and best practices has had a far-reaching impact on online security.

Inclusivity and Accessibility: The project has made strides in ensuring inclusivity and accessibility for users with diverse needs and capabilities, making online security more attainable for everyone.

Scalability and Sustainability: Designed with scalability and sustainability in mind, the project is well-prepared to accommodate a growing user base and adapt to emerging online security challenges.

Continuous Improvement: The commitment to ongoing research, development, and user feedback has enabled the project to remain responsive to changing threats and user needs. Continuous improvement is central to the project's success.

In summary, the project for the detection of phishing websites using URLs has not only made significant strides in enhancing online security but has also empowered users with the knowledge and tools needed to protect themselves from phishing attacks.

REFERENCES

1.Anti-Phishing Working Group (APWG). (https://apwg.org/): An industry association focused on combating phishing and cybercrime, offering valuable reports and resources.

2.PhishTank. (https://www.phishtank.com/): A community-driven platform for reporting and verifying phishing URLs.

3.Krebs, B. (https://krebsonsecurity.com/): Brian Krebs' cybersecurity blog often covers phishing trends and incidents.

4.OWASP Anti-Phishing Landing Page. (https://owasp.org/www-community/attacks/Phishing): Information on phishing attacks and prevention from the Open Web Application Security Project (OWASP).

5.Google Safe Browsing. (https://developers.google.com/safe-browsing): Google's Safe Browsing API and resources for safe web browsing.

6.McAfee. (https://www.mcafee.com/): McAfee's cybersecurity resources include articles and reports on phishing threats.

7.Symantec (Norton). (https://www.broadcom.com/company/newsroom/press-releases?filters=phishing): Symantec's press releases and articles on phishing-related topics.

8.Phishing.org. (https://www.phishing.org/): A comprehensive resource for information on phishing attacks and prevention.

9.Verizon. (https://enterprise.verizon.com/resources/reports/dbir/): The Verizon Data Breach Investigations Report (DBIR) often includes insights on phishing incidents.

10.Cybersecurity & Infrastructure Security Agency (CISA). (https://www.cisa.gov/): CISA provides resources and alerts related to cybersecurity threats, including phishing.

11.SANS Institute. (https://www.sans.org/): SANS offers various cybersecurity training courses and whitepapers related to phishing and web security.

12.Google Safe Browsing API: Google's API for checking websites for phishing and malware.

Website: https://developers.google.com/safe-browsing

13.Kaggle: A platform for finding datasets and machine learning resources, including datasets related to phishing detection.

Website: https://www.kaggle.com/

14.Cybersecurity and Infrastructure Security Agency (CISA): Provides cybersecurity resources and alerts about current threats.

Website: https://www.cisa.gov/

15.Machine Learning Mastery: Tutorials and articles on machine learning techniques, including those related to cybersecurity.

Website: https://machinelearningmastery.com/

APPENDIX

CODE:

Detection of phishing websites:

```
import numpy as np

import pandas as pd

import os

for dirname, _, filenames in os.walk('/kaggle/input'):

for filename in filenames:

print(os.path.join(dirname, filename))

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

from sklearn import metrics

import warnings
```

```
warnings.filterwarnings('ignore')

data = pd.read_csv('/kaggle/input/phishing-website-detector/phishing.csv')

data.head()

data.columns

len(data.columns)

data.isnull().sum()

X = data.drop(["class","Index"],axis =1)

y = data["class"]

fig, ax = plt.subplots(1, 1, figsize=(15, 9))

sns.heatmap(data.corr(), annot=True,cmap='viridis')

plt.title('Correlation between different features', fontsize = 15, c='black')

plt.show()

corr=data.corr()

corr.head()

incCorr=corr.sort_values(by='class',ascending=False)
```

```
incCorr.head()

incCorr['class']

tenfeatures=incCorr[1:11].index

twenfeatures=incCorr[1:21].index

#Structutre to Store metrics

ML_Model = []

accuracy = []

f1_score = []

precision = []

def storeResults(model, a,b,c):

ML_Model.append(model)

accuracy.append(round(a, 3))

f1_score.append(round(b, 3))

precision.append(round(c, 3))

def KNN(X):
```

```
x=[a for a in range(1,10,2)]

knntrain=[]

knntest=[]

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 = 42)

X_train.shape, y_train.shape, X_test.shape, y_test.shape

for i in range(1,10,2):

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=i)

knn.fit(X_train,y_train)

y_train_knn = knn.predict(X_train)

y_test_knn = knn.predict(X_test)

acc_train_knn = metrics.accuracy_score(y_train,y_train_knn)

acc_test_knn = metrics.accuracy_score(y_test,y_test_knn)
```

```
print("K-Nearest Neighbors with k={}: Accuracy on training Data: {:.3f}".format(i,acc_train_knn))

print("K-Nearest Neighbors with k={}: Accuracy on test Data: {:.3f}".format(i,acc_test_knn))

knntrain.append(acc_train_knn)

knntest.append(acc_test_knn)

print()

import matplotlib.pyplot as plt

plt.plot(x,knntrain,label="Train accuracy")

plt.plot(x,knntest,label="Test accuracy")

plt.legend()

plt.show()

Xmain=X

Xten=X[tenfeatures]

Xtwen=X[twenfeatures]
```

```
KNN(Xmain)

KNN(Xten)

KNN(Xtwen)

from sklearn.model_selection import train_test_split

from sklearn.neighbors import KNeighborsClassifier

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

X_train.shape, y_train.shape, X_test.shape, y_test.shape

knn = KNeighborsClassifier(n_neighbors=5)

knn.fit(X_train,y_train)

y_train_knn = knn.predict(X_train)

y_test_knn = knn.predict(X_test)

acc_train_knn = metrics.accuracy_score(y_train,y_train_knn)

acc_test_knn = metrics.accuracy_score(y_test,y_test_knn)

f1_score_train_knn = metrics.f1_score(y_train,y_train_knn)
```

```
f1_score_test_knn = metrics.f1_score(y_test,y_test_knn)

precision_score_train_knn = metrics.precision_score(y_train,y_train_knn)

precision_score_test_knn = metrics.precision_score(y_test,y_test_knn)

storeResults('K-Nearest
Neighbors',acc_test_knn,f1_score_test_knn,precision_score_train_knn)

def SVM(X, y):

x=[a for a in range(1,10,2)]

svmtrain=[]

svmtest=[]

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 = 42)

X_train.shape, y_train.shape, X_test.shape, y_test.shape

from sklearn.svm import SVC

for i in range(1,10,2):

svm = SVC(kernel='linear', C=i)
```

```
svm.fit(X_train, y_train)

y_train_svm = svm.predict(X_train)

y_test_svm = svm.predict(X_test)

acc_train_svm = metrics.accuracy_score(y_train, y_train_svm)

acc_test_svm = metrics.accuracy_score(y_test, y_test_svm)

print("SVM with C={}: Accuracy on training Data: {:.3f}".format(i,acc_train_svm))

print("SVM with C={}: Accuracy on test Data: {:.3f}".format(i,acc_test_svm))

svmtrain.append(acc_train_svm)

svmtest.append(acc_test_svm)

print()

plt.plot(x,svmtrain,label="Train accuracy")

plt.plot(x,svmtest,label="Test accuracy")

plt.legend()

plt.show()

Xmain=X
```

```
Xten=X[tenfeatures]

Xtwen=X[twenfeatures]

SVM(Xmain,y)

SVM(Xten,y)

SVM(Xtwen,y)

from sklearn.model_selection import train_test_split

from sklearn.svm import SVC

from sklearn import metrics

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

svm = SVC(kernel='linear', C=1, random_state=42)

svm.fit(X_train, y_train)

y_train_svm = svm.predict(X_train)

y_test_svm = svm.predict(X_test)

acc_train_svm = metrics.accuracy_score(y_train, y_train_svm)
```

```
acc_test_svm = metrics.accuracy_score(y_test, y_test_svm)

f1_score_train_svm = metrics.f1_score(y_train, y_train_svm)

f1_score_test_svm = metrics.f1_score(y_test, y_test_svm)

precision_score_train_svm = metrics.precision_score(y_train, y_train_svm)

precision_score_test_svm = metrics.precision_score(y_test, y_test_svm)

print("SVM with C={}: Accuracy on training data: {:.3f}".format(1, acc_train_svm))

print("SVM with C={}: Accuracy on test data: {:.3f}".format(1, acc_test_svm))

print("SVM with C={}: F1 score on training data: {:.3f}".format(1,
f1_score_train_svm))

print("SVM with C={}: F1 score on test data: {:.3f}".format(1, f1_score_test_svm))

print("SVM with C={}: Precision on training data: {:.3f}".format(1,
precision_score_train_svm))

print("SVM with C={}: Precision on test data: {:.3f}".format(1,
precision_score_test_svm))

storeResults('Support                                    Vector
Machines',acc_test_svm,f1_score_test_svm,precision_score_train_svm)
```

```
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 = 42)

X_train.shape, y_train.shape, X_test.shape, y_test.shape

from sklearn.ensemble import GradientBoostingClassifier

gbc = GradientBoostingClassifier(max_depth=4,learning_rate=0.7)

gbc.fit(X_train,y_train)

y_train_gbc = gbc.predict(X_train)

y_test_gbc = gbc.predict(X_test)

acc_train_gbc = metrics.accuracy_score(y_train,y_train_gbc)

acc_test_gbc = metrics.accuracy_score(y_test,y_test_gbc)

print("Gradient Boosting Classifier : Accuracy on training Data:
{:.3f}".format(acc_train_gbc))

print("Gradient Boosting Classifier : Accuracy on test Data:
{:.3f}".format(acc_test_gbc))

print()
```

```
f1_score_train_gbc = metrics.f1_score(y_train,y_train_gbc)

f1_score_test_gbc = metrics.f1_score(y_test,y_test_gbc)

precision_score_train_gbc = metrics.precision_score(y_train,y_train_gbc)

precision_score_test_gbc = metrics.precision_score(y_test,y_test_gbc)

storeResults('Gradient Boosting Classifier',acc_test_gbc,f1_score_test_gbc,precision_score_train_gbc)

df = pd.DataFrame({

'Modelname': ML_Model,

'Accuracy Score': accuracy,

'F1 Score': f1_score,

'Precision Score': precision

})

df.set_index('Modelname', inplace=True)

# plot the scores for each model

fig, ax = plt.subplots(figsize=(10,10))
```

```
df.plot(kind='bar', ax=ax)

ax.set_xticklabels(df.index, rotation=0)

ax.set_ylim([0.9, 1])

ax.set_yticks([0.9,0.91,0.92,0.93,0.94,0.95,0.96,0.97,0.98,0.99,1])

ax.set_xlabel('Model')

ax.set_ylabel('Score')

ax.set_title('Model Scores')

plt.show()

df = pd.DataFrame({

'Modelname': ML_Model,

'Accuracy Score': accuracy,

'F1 Score': f1_score,

'Precision Score': precision})

df.set_index('Modelname', inplace=True)

# plot the scores for each model
```

```
fig, ax = plt.subplots(figsize=(10,10))

df.plot(kind='bar', ax=ax)

ax.set_xticklabels(df.index, rotation=0)

ax.set_ylim([0.9, 1])

ax.set_yticks([0.9,0.91,0.92,0.93,0.94,0.95,0.96,0.97,0.98,0.99,1])

ax.set_xlabel('Model')

ax.set_ylabel('Score')

ax.set_title('Model Scores')

plt.show()

pip install python-googlesearch

class FeatureExtraction:

features = []

def __init__(self,url):

self.features = []

self.url = url
```

```
self.domain = ""

self.whois_response = ""

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.AnchorURL())

self.features.append(self.AbnormalURL)

self.features.append(self.WebsiteTraffic())

self.features.append(self.PageRank())

self.features.append(self.StatsReport())

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

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
```

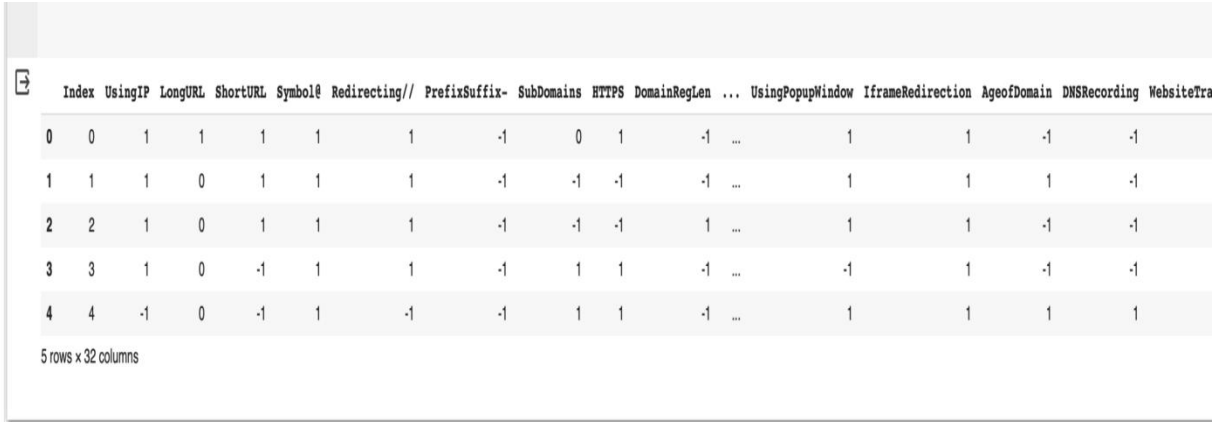
```
else:

return -1

except:

return 0

except:

return -1

def AbnormalURL(self):

try:

if self.response.text == self.whois_response:

return 1

else:

return -1

except:

return -1

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

def StatsReport(self):

try:

url_match = re.search(

'at\.ua|usa\.cc|baltazarpresentes\.com\.br|pe\.hu|esy\.es|hol\.es|sweddy\.com|m
yjino\.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|46\.242\.145\.103|121\.50\.168\.40|83\.125\.22\.219|46\.242\.145\.9\'216\.38\.62\.18|104\.130\.124\.96|47\.89\.58\.141|78\.46\.211\.158|54\.86\.225\.156|54\.82\.156\.19|37\.157\.192\.102|204\.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
```

```
gbc = GradientBoostingClassifier(max_depth=4,learning_rate=0.7)

gbc.fit(X_train,y_train)

url="http://8csdg3iejj.lilagoraj.pl/"

#can provide any URL. this URL was taken from PhishTank

obj = FeatureExtraction(url)

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

y_pred =gbc.predict(x)[0]

if y_pred==1:

print("We guess it is a safe website")

else:

print("Caution! Suspicious website detected")
```

SCREENSHOTS OF THE PROJECT



	Index	UsingIP	LongURL	ShortURL	Symbol@	Redirecting//	PrefixSuffix-	SubDomains	HTTPS	DomainRegLen	...	UsingPopupWindow	IframeRedirection	AgeofDomain	DNSRecording	WebsiteTra
0	0	1	1	1	1	1	-1	0	1	-1	...	1	1	-1	-1	
1	1	1	0	1	1	1	-1	-1	-1	-1	...	1	1	1	-1	
2	2	1	0	1	1	1	-1	-1	-1	1	...	1	1	-1	-1	
3	3	1	0	-1	1	1	-1	1	1	-1	...	-1	1	-1	-1	
4	4	-1	0	-1	1	-1	-1	1	1	-1	...	1	1	1	1	

5 rows x 32 columns

Figure :**1** - Column names in the database

```
Index(['Index', 'UsingIP', 'LongURL', 'ShortURL', 'Symbol@', 'Redirecting//',
       'PrefixSuffix-', 'SubDomains', 'HTTPS', 'DomainRegLen', 'Favicon',
       'NonStdPort', 'HTTPSDomainURL', 'RequestURL', 'AnchorURL',
       'LinksInScriptTags', 'ServerFormHandler', 'InfoEmail', 'AbnormalURL',
       'WebsiteForwarding', 'StatusBarCust', 'DisableRightClick',
       'UsingPopupWindow', 'IframeRedirection', 'AgeofDomain', 'DNSRecording',
       'WebsiteTraffic', 'PageRank', 'GoogleIndex', 'LinksPointingToPage',
       'StatsReport', 'class'],
      dtype='object')
```

`

Figure :**2** - Column names in the database

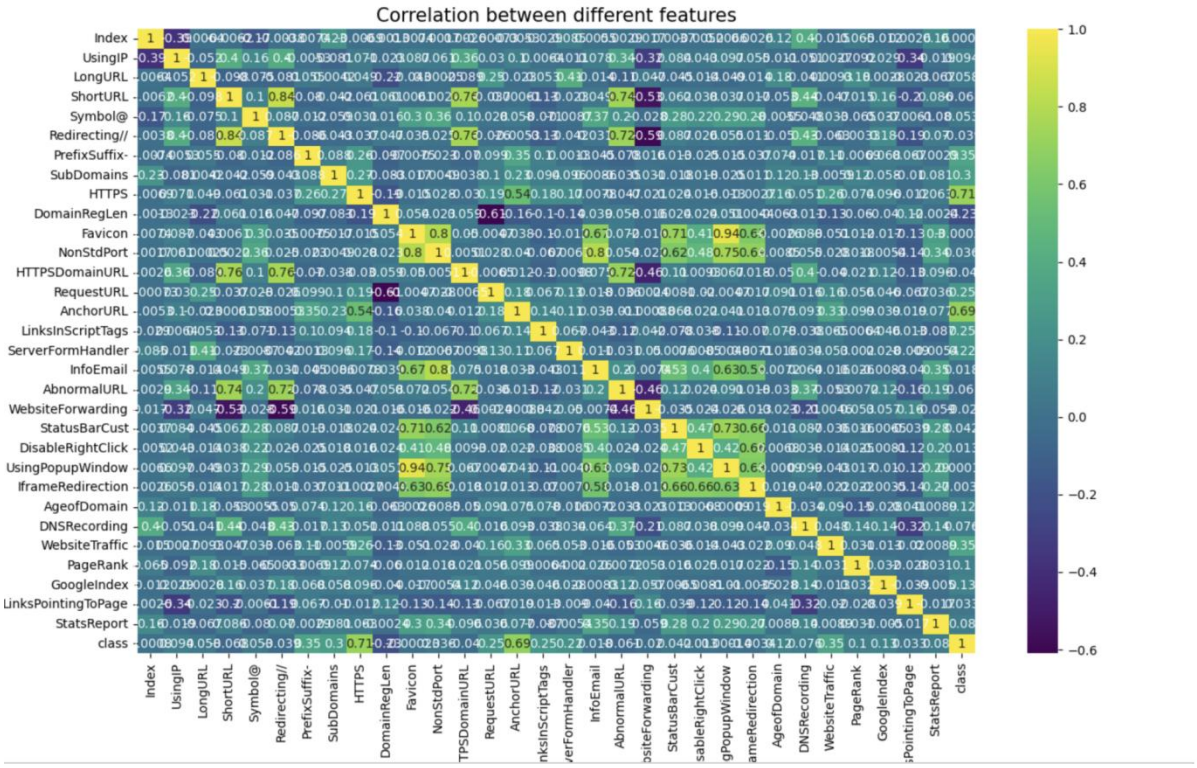


Figure :3 - Correlation between different features

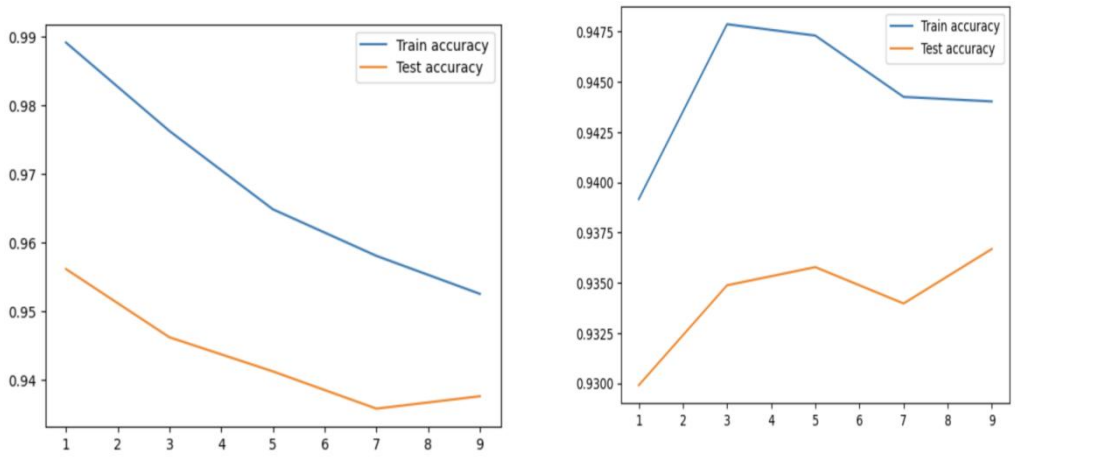


Figure :4 - train-test accuracy

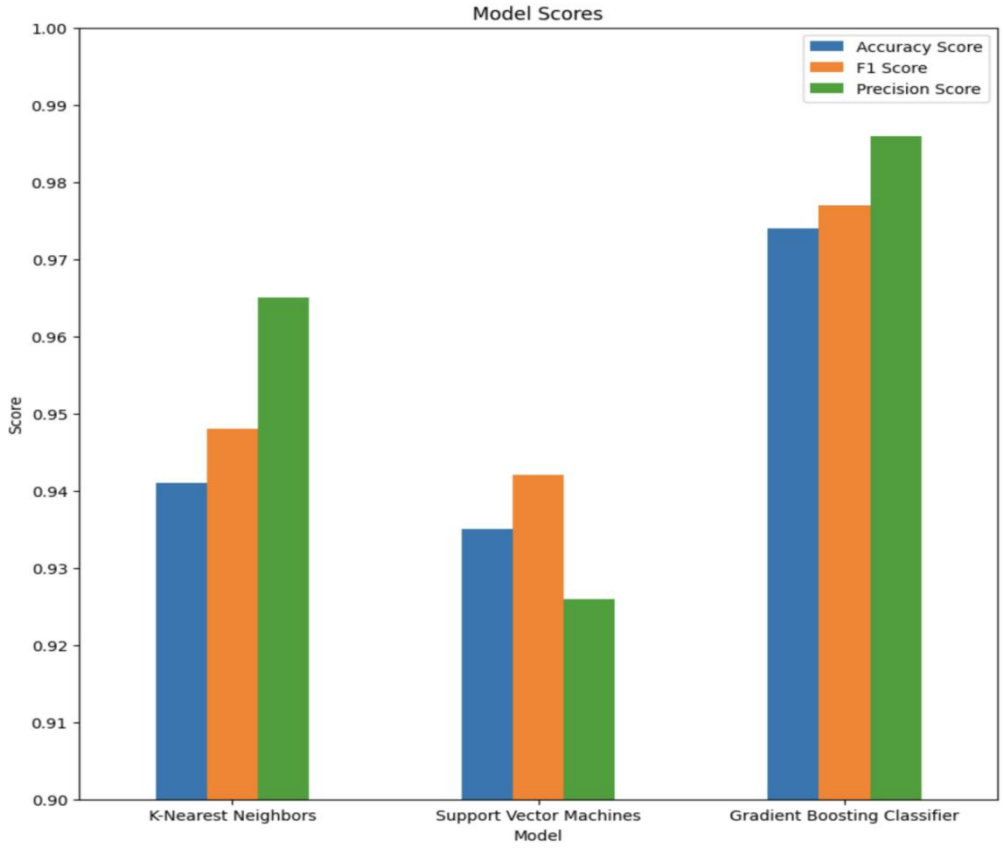


Figure :5 - Model Scores

```
] url=input("enter url")
  #can provide any URL. this URL was taken from PhishTank
  obj = FeatureExtraction(url)
  x = np.array(obj.getFeaturesList()).reshape(1,30)
  y_pred =gbc.predict(x)[0]
  if y_pred==1:
    print("We guess it is a safe website")
  else:
    print("Caution! Suspicious website detected")
```

```
  enter urlordvpn.com
  Caution! Suspicious website detected
```

```
url=input("enter url")
#can provide any URL. this URL was taken from PhishTank
obj = FeatureExtraction(url)
x = np.array(obj.getFeaturesList()).reshape(1,30)
y_pred =gbc.predict(x)[0]
if y_pred==1:
  print("We guess it is a safe website")
else:
  print("Caution! Suspicious website detected")
```

```
enter urlhttps://www.youtube.com/watch?v=xMdjMVwxH4A
We guess it is a safe website
```

```
url=input("enter url")
#can provide any URL. this URL was taken from PhishTank
obj = FeatureExtraction(url)
x = np.array(obj.getFeaturesList()).reshape(1,30)
y_pred =gbc.predict(x)[0]
if y_pred==1:
  print("We guess it is a safe website")
else:
  print("Caution! Suspicious website detected")
```

```
enter urlhttps://sathyabama.cognibot.in/
We guess it is a safe website
```

Figure :6 - Detection of phishing websites