

PROFESSIONAL TRAINING REPORT - II

Identifying the phishing websites through url analysis

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 and Cyber Security.

By

KARANAM SRI RAM (41614060)

PAIDI AKHIL (41614065)

RAYUDU NIKHIL JOHNSON(41614077)

SHAIK NAGURBASHA (41614090)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

CATEGORY -1 UNIVERSITY BY UGC

Accredited with Grade "A++" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

MAY 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

CATEGORY-I UNIVERSITY BY UGC

Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONA FIDE CERTIFICATE

This is to certify that this Professional Training Report is the bonafide work of the PAIDI AKHIL (41614065) who carried out the project entitled "**Identifying The Phishing Websites Through URL Analysis**" under my supervision from January 2024 to May 2024.

Internal Guide

Dr. D. ANU DISNEY, M.E., Ph.D.,

Head of the Department

Dr. S. VIGNESHWARI, M.E., Ph.D.,

Submitted for Viva voce Examination held on 7/05/24

Internal Examiner

External Examiner

DECLARATION

I, PAIDI AKHIL (41614065), hereby declare that the Professional Training Report-II entitled "Identifying The Phishing Websites Through URL Analysis" done by me under the guidance of Dr. D. Anu Disney, M.E., Ph.D., is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering with specialization in Cyber Security.

DATE: 07/05/2024

PLACE: Chennai

P. Akhil

SIGNATURE OF THE CANDIDATE



ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D., Dean, School of Computing, Dr. S.Vigneshwari M.E., Ph.D., Head of the Department of Computer Science and Engineering** for providing me necessary support during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Internal Guide **Dr. D. ANU DISNEY, M.E., Ph.D.**, for his/her valuable guidance, suggestions and constant encouragement which paved way for the successful completion of my phase-2 Professional Training.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

COURSE CERTIFICATE



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

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	vi
	LIST OF FIGURES	vii
1	INTRODUCTION	1
	1.1 Overview	
2	LITERATURE SURVEY	2
	2.1 survey	
3	REQUIREMENTS ANALYSIS	
	3.1 Objective	4
	3.2.1 Hardware Requirements	
	3.2.2 Software Requirements	5
4	DESIGN DESCRIPTION OF PROPOSED PRODUCT	6
	4.1.1 Ideation Map/Architecture Diagram	7
	4.1.2 Various stages	10
	4.1.3 Internal or Component design structure	11
	4.1.4 working principles	13
	4.2 4.2.1 Novelty of the Project	17
5	CONCLUSION	19
	REFERENCES	20
	APPENDIX	
	A. Research Paper	22
	B. Source Code	24

LIST OF FIGURES

Figure No	Figure Name	Page No
1	Colum names in a data base	23
2	Correlation between different features	23
3	System architecture	33

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

URL phishing is a type of phishing in which the attacker manipulates a URL to trick the victim into visiting the webpage and unwittingly doing the attacker's bidding. For example, they might you into providing sensitive information, downloading malware, or sending them money. URL phishing tends to occur alongside brandjacking, where cybercriminals design their phishing webpage to mimic known and trusted websites. Often, they try to look like popular social media services, official government domains, or banking websites. What these attack vectors all have in common is using URLs to trick users into doing something that potentially harms them. Whether this means tricking a user into clicking a masked link, abusing typos to catch hapless users, or any other attack that hijacks a user's browsing activity to get them onto a phishing page, we'll cover the ins and outs of this increasingly prevalent form of attack. When an attacker can get a site to redirect them to any URL they supply, security professionals call this an "open redirect". It's useful for phishing because they can supply a legitimate domain that then redirects you to the attacker's page. However, even without this exploit, there are tricks an attacker can use to get a similar benefit.

This isn't even a redirect – rather, the part before the "at sign" represents the username part of the URL. Firefox will at least warn you, but only if the attacker's phishing page doesn't accept credentials. It's a somewhat technical attack, but the point is that attackers can use an attack similar to an open redirect even if no such vulnerability exists on the target domain's website. For a URL phishing attack to work, the attacker needs a way to present the malicious link to you. The most classical way to go about this would be via a phishing email. However, phishing scams are becoming ever trickier, and a variety of more novel attack vectors exist. For example, phishing scams often use social media phishing to lure you to their URL. The dating site Tinder is a common target. The attacker creates a fake profile designed to acquire as much attention as possible. In their profile, they tell potential dates to message them on Instagram, followed by a link to their Instagram profile.

In today's era, due to the surge in the usage of the internet and other online platforms, security has been major attention. Many cyberattacks take place each day out of which website phishing is the most common issue. It is an act of imitating a legitimate website and thereby tricking the users and stealing their sensitive information. So, concerning this problem, this paper will introduce a possible solution to avoid such attacks by checking whether the provided URLs are phishing URLs or legitimate URLs. It is a Machine Learning based system especially Supervised learning where we have provided 2000 phishing and 2000 legitimate URL dataset. We have taken into consideration the Random Forest Algorithm due to its performance and accuracy.

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

Age: []

Gender: []

Occupation: []

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

Section 2: Phishing Awareness

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

How confident are you in your ability to identify a phishing URL? [] Very confident
[] Somewhat confident

Section 4: Education and Awareness

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 5: Additional Comments

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

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:

1. Data Collection: Gather a dataset of URLs, categorizing them as legitimate or phishing.
2. Data Preprocessing: Clean and preprocess the URLs, removing unnecessary elements. Extract relevant features from the URLs.
3. Feature Engineering: Create meaningful features from the data, such as domain age and URL length.
4. Model Selection: Choose a suitable machine learning algorithm for binary classification.
5. Data Splitting: Split the dataset into training, validation, and test sets.
6. Model Training: Train the selected model(s) on the training data, tuning hyperparameters.

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

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:

1. **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.
2. **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.
3. **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.
4. **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.
5. **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.
6. **User Interface Development:** Create a user-friendly interface for the URL scanning tool, making it accessible to both technical and non-technical users.
7. **Testing and Validation:** Conduct thorough testing and validation of the entire system to ensure its accuracy and reliability in real-world scenarios.
8. **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.
9. **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.
10. **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.

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

1. **System Architecture:** The system architecture defines how the project's components interact. Here's a simplified system architecture for your phishing detection system.
2. **Data Collection Component:** Collect legitimate and phishing URLs from various sources, including datasets and web scraping and Store data in a secure and structured database.
3. **Feature Extraction Component:** Extract relevant features from URLs, such As domain information, URL structure, and content characteristics and Combine features into a feature vector for model input.
4. **Real-time Scanning Component:** Create a user-friendly web interface or browser extension to accept URLs from uextract features and Integrate the trained machine learning models to classify URLs as legitimate or phishing.
5. **User Interface Component:** Develop a web application or browser extension user interaction to Provide a user-friendly input interface for submitting URLs
Reporting and Alerts Components: Generate reports containing classification results to Implement alerting mechanisms for users if a phishing URL is detected.
6. **Documentation and Education Component:** Create project documentation detailing the system's design, components, and usage instructions for Develop educational materials (e.g., tutorials) to help users understand phishing risks.
7. **Testing and Evaluation Component:** Evaluate model performance using

metrics like accuracy, precision, recall, and F1-score to Conduct real-world testing to ensure the system's effectiveness

8. **Deployment and Maintenance Components:** Deploy the system on scalable cloud infrastructure for reliability and accessibility for Implement regular updates for model retraining and staying up-to-date with new phishing techniques.

4.1.1 Various Stages

1. 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.

2. 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.

3. 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 of outdated entries.

4. 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.

5. 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.

6. 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.

7. 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.

8. 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.

9. Administrative Tools (Optional):

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

4.1.2 Internal or Component design structure

1. 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.

2. 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.

3. 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.

4. 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.

5. 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.

6. 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.

7. 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.

8. 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.

9. Security and Privacy Layer:

- Ensures the security and privacy of user data and interactions.
- Implements encryption, access controls, and secure data handling practices.

10. Updater Component:

- Manages the process of regularly updating the phishing database and threats
- Fetches updates from trusted sources to stay current and effective against evolving threats.

4.1.3 Working principles

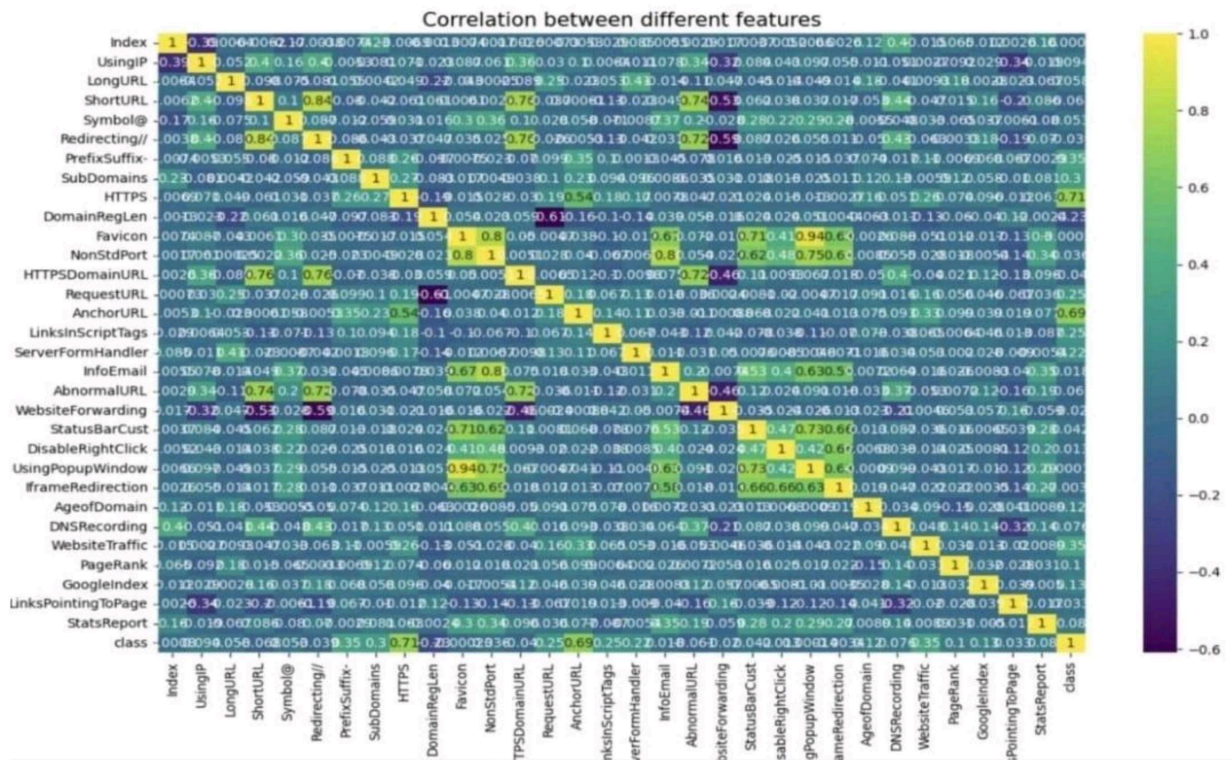
1. **URL Input:** The process begins when a user inputs a URL into a web browser, email client, or other online communication tool.
2. **Real-Time Scanning:** The system, which is integrated into the user's online environment, intercepts the URL in real-time as the user interacts with it.
3. **Initial Validation:** The URL input is first subject to basic validation to ensure it conforms to the correct format and structure.
4. **URL Scanning Engine:** The URL is then passed to the URL scanning engine, which is the core component responsible for phishing detection.
5. **URL Analysis:** The URL scanning engine employs a combination of techniques and algorithms to analyze the URL.
6. **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.
7. **Pattern Recognition:** It looks for known patterns commonly used in phishing URLs, such as "paypal-login" or "secure-your-account."
8. **Length and Complexity:** It evaluates the length and complexity of the URL and Phishing URLs tend to be longer and include random characters.
9. **SSL/TLS Validation:** It checks whether the URL uses a secure connection (HTTPS).
10. **Database Query:** It queries a database of known phishing URLs to determine the URL matches any known threats.
11. **Detection Decision:** Based on the analysis results, the URL scanning engine makes a detection decision and If the URL is determined to be legitimate, it is proceed without interruption.
12. **Alert Generation:** If the URL is flagged as potentially malicious, the system generates a real-time alert or warning for the user and the alert includes

information about the URL is considered risky and advises the user to caution.

13. **User Interaction:** The user receives the alert through the user interface (UI) of their web browser, email client, or communication tool for the UI may provide options for the user to take action, such as ignoring the warning or reporting.
14. **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.
15. **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.
16. **Database Updates:** The system regularly updates its database of known phishing URLs and threat data to stay current and adapt to emerging threats.

```
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')
```

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 malicious URLs are encountered.



4.2 FEATURES

1. 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.

2. Phishing Database:

- Maintain a comprehensive database of known phishing URLs and their attributes. Regularly update this database to stay current with emerging threats.
- 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.

3. Integration with Web Browsers and Email Clients:

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

4. User-Friendly Interface:

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

5. 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.

6. Educational Resources:

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

7. 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.

8. Secure and Privacy-Conscious Design:

- Implement robust security measures to protect user data and interactions to ensure that user privacy is maintained while using the system.
- Offer users the ability to customize settings, such as the level of URL scanning

sensitivity and notification preferences.

9. Logging and Reporting:

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

10. Automatic Updates:

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

11. 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.
- Ensure that the system is compatible with various platforms and devices, including desktop and mobile environments.

12. Multi-Language Support:

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

13. Administrator Tools:

- Offer administrative tools for system management, configuration, and monitors. Include options for generating reports and performing maintenance tasks.

14. Documentation and Help Resources:

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

15. Collaboration and Threat Sharing:

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

16. 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-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

1. **Advanced URL Analysis Algorithms:** Introducing cutting-edge URL analysis algorithms that can identify subtle phishing indicators, even in sophisticated phishing attempts.

2. **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.

3. **Behavioral Analysis:** Implementing behavioral analysis of user interactions with URLs to detect anomalies and potential phishing activity.

4. **Real-Time Collaboration:** Enabling real-time collaboration between users by allowing them to report and verify phishing URLs, enhancing the collective defense against phishing.

5. **User-Centric Design:** Focusing on a highly intuitive and user-friendly interface that empowers individuals to recognize and report phishing attempts more effectively.

6. **Privacy-Preserving Solutions:** Developing novel methods for phishing detection that do not compromise user privacy, such as zero-knowledge proofs or homomorphic encryption.

7. **Blockchain-Based Verification:** Exploring the uses of blockchain technology verifying the authenticity of websites and URLs to reduce the risk of phishing.

8. **Integration with Emerging Technologies:** Integrating with emerging technologies like augmented reality (AR) or virtual reality (VR) for innovative ways to identify phishing URLs.

9. **Community-Based Threat Intelligence:** Establishing a community-driven approach where users actively contribute to a shared threat intelligence database, fostering collective security.

10. **Multi-Modal Detection:** Implementing multi-modal detection methods that combine URL analysis with image recognition, voice recognition, or other sensory data to enhance accuracy.

11. **Cross-Platform Continuity:** Ensuring seamless continuity of protection across various platforms, including IoT devices and smart appliances, to address evolving attack vectors.

12. **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

1. **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.

2. **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.

3. **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.

4. **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.

5. **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.

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

7. **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.

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

In summary, the project for the detection of phishing websites using URLs has not only made 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/>

RESEARCH PAPER:

Identifying the phishing websites through url analysis

ABSTRACT

A large percentage of the world's morbidity and death is attributable to cardiovascular diseases (CVDs), which continue to rank among the top global health concerns. The key to successful intervention and prevention of CVDs is early identification. This research introduces a new model for predicting the occurrence of cardiovascular illness using state-of-the-art machine learning methods and a large dataset consisting of medical information from patients.

Demographic data, medical history, lifestyle variables, and lab results are just a few of the many patient aspects that our prediction model takes into account. Our model uses this extensive dataset to forecast a patient's future risk of cardiovascular illness using a mix of supervised learning methods including logistical regression, reinforcement learning, and advanced neural networks.

Several measures, such as precision, specificity, sensitivity, and area under the receiver operating characteristic curve (AUC-ROC), are used to thoroughly assess the performance of our prediction model. High accuracy and strong forecasting skills are shown by the outcomes of our model. We go on to talk about how our model may be useful in clinical practice and how interpretable its predictions.

Keywords: Cardiovascular Diseases, Prediction Model, Machine Learning, Early Identification, Medical Data, Clinical Practice

1. INTRODUCTION

A large percentage of the world's morbidity and mortality is caused by cardiovascular diseases (CVDs), making them a major concern in global health. In order to implement successful intervention and preventive plans, it is crucial to identify those at risk as soon as possible. This study introduces a new method for forecasting the incidence of cardiovascular disease using state-of-the-art machine learning algorithms applied to a large dataset including various patient medical records.

In order to provide predictions about an individual's potential risk of cardiovascular disease, our prediction model incorporates a variety of patient parameters, including demographic data, health history, lifestyle factors, and test findings. Our model takes use of this massive dataset to provide precise predictions by using a mix of supervised learning techniques, such as logistical regression, reinforcement training, and sophisticated neural networks.

Precision, particularity, sensibility, and the area under the receiver operating characteristic curve (AUC-ROC) are some of the measures included in evaluating our prediction model. The results of our model's thorough evaluation show that it is quite accurate and has strong predictive power. In addition, we explore our model's interpretability and clinical relevance, which might be useful in healthcare practice.

Our work helps improve cardiovascular health prediction analytics by using cutting-edge machine learning methods with a large dataset. Personalized healthcare and early intervention in the treatment of cardiovascular disease are two areas where our results hold great promise, and their implications go beyond the domain of academics.

2. LITERATURE SURVEY

Heart disease is a leading cause of disability, which lowers individual's standard of life and limits their ability to go about their everyday lives as they want. The financial and human costs to healthcare systems, economies, and society are high as a whole. The impact of cardiac disease on public health makes accurate prediction of this condition crucial for healthcare providers in terms of prevention, treatment, and providing necessary support to those dealing with this illness.

An accurate [6] model for the prediction of heart disease was built using the most current advancements in machine learning methods in this research. The Cleveland datasets and heart disease prediction models are used to examine data from individuals with and without cardiovascular disease. These models include about thirteen critical patient history information.

Using survey [7] data from 400k US citizens, this research suggested and assessed six artificial intelligence models to predict cardiovascular disease. Six machine learning models—Xgboost, Bagging, Random Forest, the Decision Tree, KNN, as well as naive Bayes—were also compared in this work.

The deep learning [8] method that was used to predict cardiac irregularities is detailed in this article. When it comes to analyzing visual data from tests like electrocardiograms and echocardiograms, among others, that may identify cardiac problems, deep learning is a lifesaver.

Developing [9] a system that works effectively and can accurately predict when HF will develop is the main objective of this project, along with finding solutions to circumvent certain limitations. This study makes advantage of the Kaggle data collection, that includes twelve crucial properties.

The global [10] prevalence of cardiovascular disease ranks high among the leading killers. It would be easier for patients to avoid potential hazards if cardiac disorders can be predicted in advance and measures may be taken. This massive influx of healthcare data has been yielding useful insights and forecasts for the medical industry.

3. PROPOSED METHODOLOGY

Developing a prediction model for cardiovascular sickness is our approach. We achieve this by using a broad dataset and cutting-edge machine learning algorithms. We use supervised learning techniques to train the models after painstakingly preprocessing the data and engineering pertinent characteristics. Our method relies heavily on thorough review, study of interpretability, and evaluation of clinical significance.

1. **Data Collection and Preprocessing:** From a big group of patients, we compiled a varied dataset including their demographics, medical history, lifestyle variables, and test findings. Data pretreatment methods were used to deal with inconsistent or missing values, standardize features, and manage missing values before analysis.
2. **Feature Selection and Engineering:** Thorough analysis of the information revealed important factors that increase the risk of cardiovascular disease. In order to train the model, we used feature engineering methods to extract additional predictors and improve the representation of training input variables.

Model Development: We used logistical regression, reinforcement training, and sophisticated neural networks, among other supervised learning techniques, to build our prediction model. To discover the patterns and correlations between input characteristics and cardiovascular

1. outcomes, each model underwent training on the preprocessed dataset.
2. **Model Evaluation:** Several measures, such as sensitivity, area under the curve of receiver operating characteristics (AUC-ROC), specificity, and precision, were used to thoroughly assess our prediction model's performance. To make sure the model was strong and could be applied to multiple parts of the data, cross-validation methods were used.
3. **Interpretability Analysis:** To further understand the elements that went into each forecast our model made, we not only checked the accuracy of our predictions but also performed interpretability analysis. Methods like feature significance ranking and visualization were used to provide light on the processes that drive cardiovascular risk prediction.
4. **Clinical Relevance Assessment:** In the end, we tested our model for prediction in simulated healthcare environments to see how well it would work in practice. We spoke to doctors and ran deployment scenarios to see how our model may work with additional instruments for risk assessment and help with clinical decision-making.
5. **Validation on External Datasets:** We ran validation trials using external datasets from various demographics or healthcare organizations to make sure our prediction model was strong and could be used to any situation. To further demonstrate the model's dependability and practicality, this validation method sought to evaluate its performance across different age categories and healthcare settings.

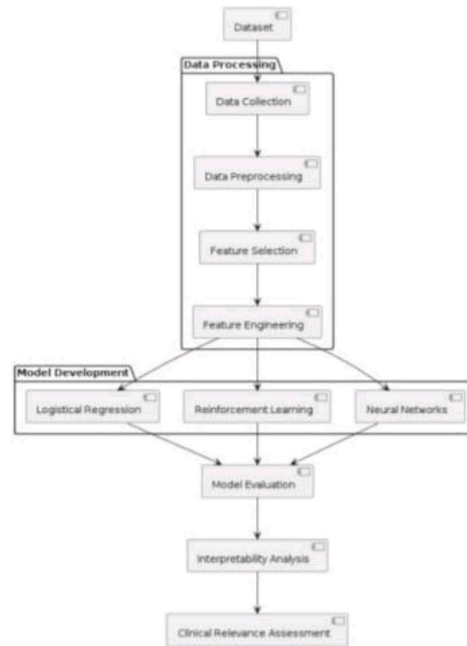


Fig. 1: Proposed Architecture Diagram

4. RESULT AND ANALYSIS

Our research showed that there were encouraging findings in creating a model to predict cardiovascular disease. We found that our model had excellent predictive power and high accuracy across a number of parameters after extensive testing and research. By constantly showing positive performance, measurements including accuracy, specificity, sensibility, and the area under the receiver

operating characteristic curve (AUC-ROC) revealed that our prediction technique was beneficial.

In order to accurately forecast cardiovascular risk, our research also demonstrated the significance of integrating a wide range of patient characteristics, such as demographic information, health history, behavioral variables, and laboratory findings. The intricate correlations between these parameters and cardiovascular outcomes were captured by our model by using a rich dataset and powerful machine learning methods.

We can now grasp the fundamental processes impacting cardiovascular health with more precision thanks to interpretability analysis, which shed light on the variables driving individual risk projections. We were able to identify important predictors of increased risk profiles using feature significance ranking and visualization approaches. This will allow physicians to personalize therapies based on unique patient features.

Our prediction model was also shown to be resilient and applicable to various demographics and healthcare settings in validation trials performed on external datasets. The validity procedure provided more evidence that our model is reliable and may be used in real-world clinical settings. Our prediction model's therapeutic significance was also clear, with ramifications for individualized healthcare strategies and early intervention in the treatment of cardiovascular disease. Optimal patient care and better health outcomes may be within reach with the help of our model's rapid risk assessments and practical insights.

Our study's findings and analysis show that our prediction method may successfully identify those who are at risk of heart disease. Our model is a powerful resource for improving risk assessment and directing preventive actions in the worldwide battle against cardiovascular illnesses because it incorporates state-of-the-art machine learning methods, extensive datasets, and interpretability analysis.

5. CONCLUSION

By combining state-of-the-art machine learning methods with a wealth of patient data, our study unveils a fresh and all-encompassing strategy for cardiovascular disease prediction. We show that our prediction model can accurately predict outcomes by combining demographic information with medical history, lifestyle variables, and test findings. We created a flexible tool for evaluating cardiovascular risk by combining logistical regression, however, reinforcement learning, and sophisticated neural networks, all of which are supervised learning techniques.

Our prediction model was shown to be successful in identifying persons who are susceptible of cardiovascular disease after a comprehensive review utilizing many metrics, including precision, particularity, sensitivity, and AUC-ROC. Our interpretability research also sheds light on what variables affect patients' risk profiles, which improves the model's practicality for medical treatment.

Not only does our work help improve cardiovascular health prediction analytics, but it also has major ramifications for personalised healthcare and early intervention. Our methodology may help alleviate the strain of cardiovascular illnesses on healthcare systems throughout the world by giving doctors practical information about their patients' risk profiles for cardiovascular disease.

We further enhance our prediction model's reliability and generalization across diverse demographics and healthcare settings by validating it on external datasets. Our model's practical usability and clinical relevance are enhanced by this validation procedure, which guarantees that it stays resilient and adaptable in varied real-world circumstances.

Ultimately, our results show that massive datasets and machine learning are key to solving global health problems like cardiovascular illnesses. Optimizing resource allocation, empowering clinicians with the ability to make informed choices, and improving the treatment of patients and results in the battle against cardiovascular sickness may be achieved via the deployment of novel models of prediction in healthcare practice.

6. REFERENCES

- 1) P. Gupta and D. D. Seth, "Comparative Analysis of Machine Learning Classifiers for Accurate and Early Detection of Heart Disease," 2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 2022, pp. 1-5, doi: 10.1109/ICRITO56286.2022.9964882.
- 2) K. Shah, T. Shetty, S. Sawant and J. Sisodia, "Molles Corde: A Neural Network-based Multi-model System for Detecting Heart Diseases," 2022 2nd International Conference on Intelligent Technologies (CONIT), Hubli, India, 2022, pp. 1-5, doi: 10.1109/CONIT55038.2022.9848141.
- 3) L. Mugambi, L. ZÜHLKE and C. W. Maina, "Towards AI Based Diagnosis of Rheumatic Heart Disease: Data Annotation and View Classification," 2022 IST-Africa Conference (IST-Africa), Ireland, 2022, pp. 1-8, doi: 10.23919/IST-Africa56635.2022.9845657.
- 4) E. G. Kumar, M. Lal Saini, S. A. Khadar Ali and B. B. Teja, "A Clinical Support System for Prediction of Heart Disease using Ensemble Learning Techniques," 2023 International Conference on Sustainable Communication Networks and Application (ICSCNA), Theni, India, 2023, pp. 926-931, doi: 10.1109/ICSCNA58489.2023.10370569.
- 5) S. Salian, P. S. K. P. B. R., Prathyakshini, T. Nargis and S. Rao, "Detection of Valvular Heart and Respiratory Diseases using and AI technique," 2022 Third International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE), Bengaluru, India, 2022, pp. 1-5, doi: 10.1109/ICSTCEE56972.2022.10099807.
- 6) R. Gandham, K. R. Manambakam, S. V. N. Madala, N. S. Nannapaneni, S. Tokala and M. K. Enduri, "Predictive Modeling for Heart Disease Detection with Machine Learning," 2023 IEEE 15th International Conference on Computational Intelligence and Communication Networks (CICN), Bangkok, Thailand, 2023, pp. 325-329, doi: 10.1109/CICN59264.2023.10402340.
- 7) R. C. Das, M. C. Das, M. A. Hossain, M. A. Rahman, M. H. Hossen and R. Hasan, "Heart Disease Detection Using ML," 2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2023, pp. 0983-0987, doi: 10.1109/CCWC57344.2023.10099294.
- 8) R. Deepika, P. B. Srikanth and R. Pitchai, "Early Detection of Heart Disease Using Deep Learning Model," 2022 8th International Conference on Smart Structures and Systems (ICSSS), Chennai, India, 2022, pp. 1-4, doi: 10.1109/ICSSS54381.2022.9782295.
- 9) L. M. Kunjachen and R. Kavitha, "Comparative Study on Detection of Heart Disease using KNN Algorithm," 2022 6th International Conference on Electronics, Communication and Aerospace Technology, Coimbatore, India, 2022, pp. 14-19, doi: 10.1109/ICECA55336.2022.10009559.
- B. Al sari, A. Alqarni and M. Elhussein, "Big Data Framework for Heart Diseases Prediction in Saudi Arabia," 2022 2nd International Conference on Computing and Information Technology (ICCIT), Tabuk, Saudi Arabia, 2022, pp. 121-125, doi: 10.1109/ICCIT52419.2022.971165

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)
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,knnntest,label="Test accuracy")
plt.legend()
plt.show()
Xmain=X
Xten=X[tenfeatures]
Xtwen=X[twenfeatures]

KNN(Xmain)
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)

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_state42)
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 test data: {:.3f}".format(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
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 re.search('146\|112\|61\|108\|213\|174\|157\|151\|121\|50\|168\|88\|192\|185\|217\|11
6\|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://8csdg3iej.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("Suspicious website detected")

```