SCHOOL OF COMPUTER SCIENCE ENGINEERING AND INFORMATION SYSTEMS

FALL SEMESTER – 2023-24

CSE 3501 - INFORMATION SECURITY ANALYSIS AND AUDIT

J COMPONENT REVIEW

TITLE:

"Creation and Detection of phishing website(spammers) using PyPhisher tool and XGBoost algorithm"

FACULTY:

Dr.C.Navaneethan

Submitted by:

Ramya Sree B (20MIS0065) - F2

Sujana kola (20MIS0171) - F2

Ragavarshini S (20MIS0192) - F2

ABSTRACT:

Phishing websites pose a severe cybersecurity threat, tricking users into divulging sensitive information. As these attacks grow in complexity, it becomes increasingly challenging for users to distinguish between genuine and fraudulent websites. Machine learning techniques offer a solution by analysing attributes like URLs, HTML code, and content to identify phishing websites, with XGBoost standing out for its precision and efficiency.

This project introduces an innovative approach to identifying phishing websites by combining PyPhisher, a Python module for crafting such sites, and the XGBoost algorithm. The method relies on analysing unique website features for identification. PyPhisher generates a dataset of phishing websites for training an XGBoost model, enabling it to classify new websites as phishing or legitimate. Evaluation of a comprehensive dataset shows that this method is highly accurate and efficient, providing a valuable means of protecting users from phishing attempts.

INTRODUCTION:

Phishing attacks have become a prevalent and sophisticated cyber threat, aiming to deceive users into divulging sensitive information by impersonating legitimate websites. To combat this menace, this project proposes the use of the PyPhisher tool in conjunction with the XGBoost algorithm to enhance the creation and detection of phishing websites. PyPhisher is an open-source Python tool that enables the generation of phishing websites easily, while XGBoost is a machine learning algorithm known for its excellent performance in classification tasks.

One of the identified issues with the existing system is the lack of realistic testing scenarios for phishing detection tools. Traditional methods often rely on static datasets that may not accurately represent the dynamic nature of phishing attacks. PyPhisher addresses this by allowing the creation of authentic-looking phishing websites, providing a more realistic testing environment for security solutions.

The contribution of this proposed work lies in the synergy between PyPhisher and XGBoost. By combining a tool that facilitates realistic phishing website creation with a powerful machine learning algorithm for detection, we create a comprehensive framework. This framework not only assists in understanding the anatomy of phishing attacks but also empowers security professionals to develop and evaluate more effective countermeasures.

The results of this will demonstrate the potential of PyPhisher and XGBoost in both creating and detecting phishing websites. By simulating the creation of phishing websites, security professionals can better understand the tactics employed by attackers and develop more effective defense strategies. Additionally, the XGBoost-based detection model exhibits promising accuracy and can be integrated into security systems to strengthen protection against phishing attacks. This project contributes to the ongoing efforts to enhance cybersecurity measures in an ever-evolving threat landscape.

EXISTING SYSTEMS :

There are a variety of existing systems for detecting phishing websites. Some of the mostcommon approaches include:

Blacklists and whitelists: A blacklist is a list of known phishing websites, while a whitelist isa list of known legitimate websites. When a user visits a website, the system checks it against the blacklist and whitelist. If the website is on the blacklist, it is blocked. If the website is on the whitelist, it is allowed. However, blacklists and whitelists can be easily bypassed by attackers, and they cannot detect new phishing websites quickly enough.

Heuristic-based detection: Heuristic-based detection systems use a set of rules to identifyphishing websites. These rules are based on common characteristics of phishing websites, such as suspicious URLs, typos and grammatical errors in website text, and the use of images and logos from legitimate websites. Heuristic-based detection systems are more effective than blacklists and whitelists at detecting new phishing websites, but they can alsogenerate false positives

LITERATURE SURVEY:

S.NO	TITLE	METHODOLOGY	APPLICATION	ADVANTAGES	LIMITATIONS
1	An Investigation on Vulnerability Analysis of Phishing Attacks and Countermeasures.	Investigates and analyzes various phishing tools, including Zphisher, CamPhish, and PyPhisher, to simulate phishing attacks. Explores prevention methods, countermeasures, and real-world threats. Focuses on raising awareness about phishing.	Addresses cybersecurity concerns related to phishing attacks, emphasising online identity theft and the need for realistic countermeasures. Aims to educate both experts and non-experts on identifying and responding to phishing threats.	Focuses on a critical cybersecurity issue - phishing attacks, Aim s to make phishing awareness accessible toa broad audience, not just experts.	Lacks detailed information on specific methodologies used to analyse phishing tools, Does not mentionthe size or diversity of datasets used.

2	Spammer Detection and Fake User Identification on Social Networks	Review of techniques used for detecting spammers on Twitter. Taxonomy of Twitter spam detection approaches based on the ability to detect fake content, URL spam, trending topicspam, and fake users. Comparison of techniques using various features like user, content, graph, structure, and time features.	Enhancing the detection of spammers and fake users on Twitter to mitigate the spread of irrelevant and harmful content, improving the overall user experience and safety on the platform.	Provides a comprehensi ve review and taxonomy of Twitter spam detection techniques,A valuable resource for researchers in the field of online social network security.	Limited detail on the specific algorithms or approaches covered, May notcover the most recent developments as it was published in 2019.
3	A Neural Network-Based Ensemble Approach for Spam Detection in Twitter	Ensemble approach combining deep learning(CNNs) and feature-based models using various word embeddings. Employs a multilayer neural network as a meta- classifier for spam detection at the tweet level.	Aims to enhance spam detection on Twitter by identifying spam at the tweet level in real time, addressing the challenge of spammers creating new accounts.	Combines the strengths of deep learning and traditional feature- based techniques, Successfully outperforms existing methods according to experimental results.	Lack of detailed information on specific deep learning architectures and feature- based techniques, Insufficient information about the datasets used, except for their balance characteristics.
4	Email Spam Detection Using Machine Learning Algorithms	Utilises machine learning techniques to identify fraudulent spam emails. Applies multiple machine learning algorithms toemail	Addresses the growing issue of email spam, especially for illegal and unethical purposes like phishing and fraud. Aims to protect users from harm by detecting	Focuses on real-world problems of email spam with potential for practical application., Employs machine learning algorithms to enhance	Lacks detailed information on specific machine learning algorithms used, Doesn't provide insight into the size or diversity of datasets used for evaluation.

		datasets and selects the best-performing algorithm based on precision and accuracy.	fraudulent emails that may appear genuine.	email spam detection.	
5	Phishing URL Detection Using URL Ranking	Automatic classification of URLs based on lexical and host-based features. Clustering used to derive cluster IDs for each URL. Online URL reputation services employed for categorization.	Protecting users from phishing host URLs across web services. URL ranking based on classification and categorization.	High accuracy (93-98%) in detecting phishing hosts. Utilises online URL reputation services for additional information.	Specifics of the lexical and host-based features are not detailed. No information provided on the size and diversity of the dataset used for testing.
6	Spears Against Shields: Are Defenders Winning the Phishing War?	Assesses the current phishing landscape by generating 1,000 phishing emails with phishing links using natural language generation technology. Tests five popular anti-phishing tools and evaluates anti-phishing training technologies.	Focuses on the ongoing battle between hackers and defenders in the realm of phishing attacks. Aims to determine the effectiveness of current anti-phishing filters and training tools in detecting and mitigating phishing threats.	Provides an assessment of the current state of phishing defence tools and training technologies, Uses a practical approach by generating realistic phishing emails.	Limited details on the specific anti-phishing tools and training technologies tested., No information on the diversity or characteristics of the email dataset used.

7	Fresh-Phish: A Framework for Auto-Detection of Phishing Websites	Develops "Fresh-Phish," a framework for creating up-to-date machine learning data for phishing websites. Utilizes 30 different website features queried using Python to construct a large labeled dataset	Addresses the increasing frequency and sophistication of phishing attacks on the internet by automating the detection of nefarious websites. Aims to adapt to evolving social engineering techniques.	Introduces a framework for creating current and adaptable datasets for phishing website detection, Analyses both accuracy and training time, considering practicality.	Lacks detailed information on the specific machine learning classifiers used, Doesn't provide insight into the size or diversity of the dataset
8	Detection of Phishing Websites using Machine Learning	Blacklisting and semantic analysis	Identify and prevent phishing attacks	The proposed model is efficient and accurate, with a high detection rate and a low false positive rate.	May not be able to detect all phishing attacks, especially those that are new or targeted
9	Real Time Detection of Phishing Websites	URL-based phishing detection, Content-Based Approach (e.g., CANTINA, GoldPhish), Heuristic-Based Approach	Detect phishing attacks in real time	Distinguishing legitimate vs. fake URLs,Low false alarm rate,Real-time prevention and reporting	Inability to address all forms of web spoofing,Ineffectiveness of antivirus, firewall, and SSL in some cases,Difficulty in detecting sophisticated attacks,Requires constant updating

		(e.g., SpoofGuard), Blacklist-Based Approach (e.g., Net Craft Toolbar)			and maintenance of the blacklist
10	CatchPhish:detection of phishing websites by inspecting URLs	URL-based features, Term Frequency-Inverse Document Frequency (TF-IDF), Random Forest classifier	Predicting the legitimacy of a URL without visiting the website	Independent of third-party services, Fast computation, Independent of drive-by downloads, Language independent	Not suitable for handling drive-by downloads, Limited to URL-based features, May not detect zero-day attacks, May misclassify new legitimate sites
11	An Evaluation of Machine Learning-Based Methods for Detection of Phishing Sites	Machine learning techniques (AdaBoost, Bagging, SVM, CART, LR, RF, NN, NB, BART) combined with heuristics	Can be used to develop more accurate and efficient phishing detection systems.	High detection accuracy, Adjustment capability	Limited number of features for phishing site detection, No confirmation of the effectiveness of multiple machine learning techniques
12	Favicon – a Clue to Phishing Sites Detection	Analyzing and detecting phishing sites based on the presence and characteristics of favicons	Favicon detection can be used in web browsers, email clients, security software, and DNS providers to detect and block phishing websites.	Effective in detecting phishing sites that use favicons; relatively easy to implement,Raises awareness in the research community, Encourages user vigilance towards favicons	Some phishing sites do not have favicons (about 9%), Limited coverage for favicon-less phishing sites,Potential risk of agitating phishers, Limited effectiveness for favicon-less phishing sites

13	PhishStorm: Detecting Phishing With Streaming Analytics	Analyzes URL-relatedness to identify potential phishing sites in real-time. Utilizes search engine query data and machine learning for classification.	Phishing detection in email, HTTP traffic, and other online applications	High correct classification rate (94.91%). Real-time analytics with Big Data architectures.	Dependency on real-time data sources like search engine query data. Potential false positives and false negatives. Delay in features calculation.
14	PhiKitA: A Novel Dataset for Phishing Website Identification	Crawled phishing kits and phishing websites generated with the kits	PhiKitA is a valuable resource for phishing detection, investigation, and education.	Released a dataset with phishing kits, phishing website samples, and legitimate websites	Need to extend the dataset with more samples and add additional data
15	Phishing URL Detection: A Real-Case Scenario Through Login URLs	Proposed a phishing detection method using machine learning and deep learning techniques on login pages.	Can be used to develop a phishing detection system that is more accurate and efficient than traditional methods.	1. Low false-positive rate when classifying login URLs. 2. No dependence on external services. 3. Trained with updated legitimate login URLs. 4. More representativ e of a real case scenario than using homepage URLs.	1. Overall accuracy is slightly reduced due to the similarity between phishing and legitimate login pages. 2. Machine learning models using handcrafted URL features decreased their performance over time.
16	OFS-NN: An Effective Phishing	Proposed a new phishing detection	Can be used to develop a phishing	Alleviates the over-fitting problem of	May be computationally expensive due to

	Websites Detection Model Based on Optimal Feature Selection and Neural Network	model, OFS-NN, which combines optimal feature selection and neural network.	detection system that is more accurate and efficient than traditional methods.	neural networks by using feature validity value (FVV) to select the optimal features.	the use of neural networks.
17	Phishing Detection System Through Hybrid Machine Learning Based on URL	Proposed a hybrid machine learning model (LSD) that combines logistic regression, support vector machine, and decision tree with soft and hard voting to detect phishing URLs.	Can be used to develop a phishing detection system that is more accurate and efficient than traditional blacklist-based systems.	Achieved the best results in terms of precision, accuracy, recall, F1-score, and specificity compared to other machine learning models.	May be computationally expensive due to the use of multiple machine learning models.
18	AI Meta-Learners and Extra-Trees Algorithm for Phishing Website Detection	Proposed four meta-learner models (AdaBoost-Extra Tree (ABET), Bagging – Extra tree (BET), Rotation Forest - Extra Tree (RoFBET) and LogitBoost-Extra Tree (LBET)) developed using the extra-tree	Can be used to develop a phishing detection system that is more accurate and efficient than traditional blacklist-based systems.	Achieved a detection accuracy not lower than 97% with a drastically low false-positive rate of not more than 0.028. Outperforms existing ML-based models in phishing attack detection.	May not be interpretable as other black-box AI methods.

		base classifier.			
19	A Multi Layered Stacked Ensemble Learning Technique for Phishing Detection	A multi-layered stacked ensemble learning technique that combines various classifiers at different layers to detect phishing websites.	Can be used to develop a phishing detection system that is more accurate and efficient than traditional blacklist-based systems.	Achieves a high accuracy in detecting phishing websites, ranging from 96.79% to 98.90%. Is able to capture multiple characteristic s of data due to data diversity.	Is computationally expensive, as it requires training multiple classifiers at different layers.
20	Intelligent phishing website detection using particle swarm optimization-based feature weighting	PSO-based feature weighting, a machine learning technique that uses particle swarm optimization to assign weights to website features.	Phishing website detection. Can be used to develop a phishing website detection system that is more accurate and efficient than traditional blacklist-based systems.	Accuracy and efficiency. PSO is a powerful evolutionary algorithm that can be used to find the optimal solution to a problem. The proposed approach also removes irrelevant features, which improves the performance of the machine learning models.	Computational cost. PSO is a wrapper method, which means that it is computationally more expensive than filter methods. However, the improved accuracy and performance of the machine learning models justify the additional computational cost.

The provided information outlines a diverse range of research studies and approachesrelated to the detection and prevention of phishing attacks. These studies encompassvarious methodologies, including machine learning, deep learning, ensemble methods, and feature-based techniques. They primarily focus on identifying and mitigating phishing threats in different contexts, such as email, social networks, and websites. While these studies offer valuable insights into phishing detection and countermeasures, they often exhibit limitations, such as insufficient detail on methodologies or dataset characteristics. Additionally, some studies propose innovative frameworks, like the use of favicons or real-time analytics, to enhance

phishing detection. Overall, these research efforts contribute to the ongoing battle against phishing attacks, but further research and data sharing are needed to addressexisting limitations and adapt to evolving threats.

PROPOSED SYSTEM:

The proposed system is a dynamic and comprehensive approach to address the rising threat of phishing attacks. Leveraging the PyPhisher tool in conjunction with the XGBoost machine learning algorithm, our system is designed to both emulate the creation of realistic phishing websites and enhance the detection capabilities of cybersecurity measures. In the creation phase, PyPhisher enables the easy generation of phishing websites that closely mimic trusted entities, allowing security professionals to understand and anticipate the evolving tactics of malicious actors. This phase provides a crucial realistic testing environment, addressing a significant limitation of traditional static datasets.

In the detection phase, the generated phishing websites are utilized to train and test an XGBoost classifier. The strength of XGBoost in handling high-dimensional data and its robustness against overfitting make it an ideal choice for effectively distinguishing between legitimate and malicious websites. Features extracted from the phishing websites, such as HTML content, URL structure, and page layout, serve as input for the classifier, resulting in a powerful and adaptive phishing detection model. The integration of this model into security systems contributes to a proactive defense strategy, bolstering protection against the ever-evolving landscape of phishing attacks.

Overall, the proposed system offers a holistic approach to cybersecurity, combining realistic threat simulation with advanced machine learning techniques. By understanding the creation tactics through PyPhisher and deploying a robust detection mechanism with XGBoost, the system empowers security professionals to stay ahead of cyber threats and reinforce the resilience of their defenses against phishing attacks.

OVERVIEW:

Phishing attacks present a significant cybersecurity challenge, exploiting users' vulnerability to trick them into revealing sensitive information. With the increasing complexity of these attacks, distinguishing between legitimate and fraudulent websites has become a formidable task. Machine learning techniques, particularly the precision and efficiency of XGBoost, offer a compelling solution. This project introduces an innovative strategy by integrating PyPhisher, a Python tool for crafting phishing sites, with the robust XGBoost algorithm. By analyzing unique website features, the method effectively identifies phishing websites, showcasing high accuracy and efficiency in shielding users from malicious attempts.

Addressing the limitations of traditional phishing detection tools, the project emphasizes the necessity for realistic testing scenarios. The dynamic nature of phishing attacks often surpasses the capabilities of static datasets. PyPhisher addresses this gap by enabling the creation of authentic-looking phishing websites, providing a more lifelike testing environment for security solutions. The project's significant contribution lies in the synergy between PyPhisher and XGBoost, forming a comprehensive framework that not only enhances understanding of phishing attacks but also empowers security professionals to develop and evaluate more effective countermeasures. The results underscore the potential of this approach in both creating and detecting phishing websites, offering a proactive stance in the face of evolving cybersecurity threats. The use of PyPhisher on Kali Linux underscores the practical implementation of the project, accompanied by a cautionary note on ethical and lawful usage to prevent potential legal and ethical consequences.

SYSTEM ARCHITECTURE :

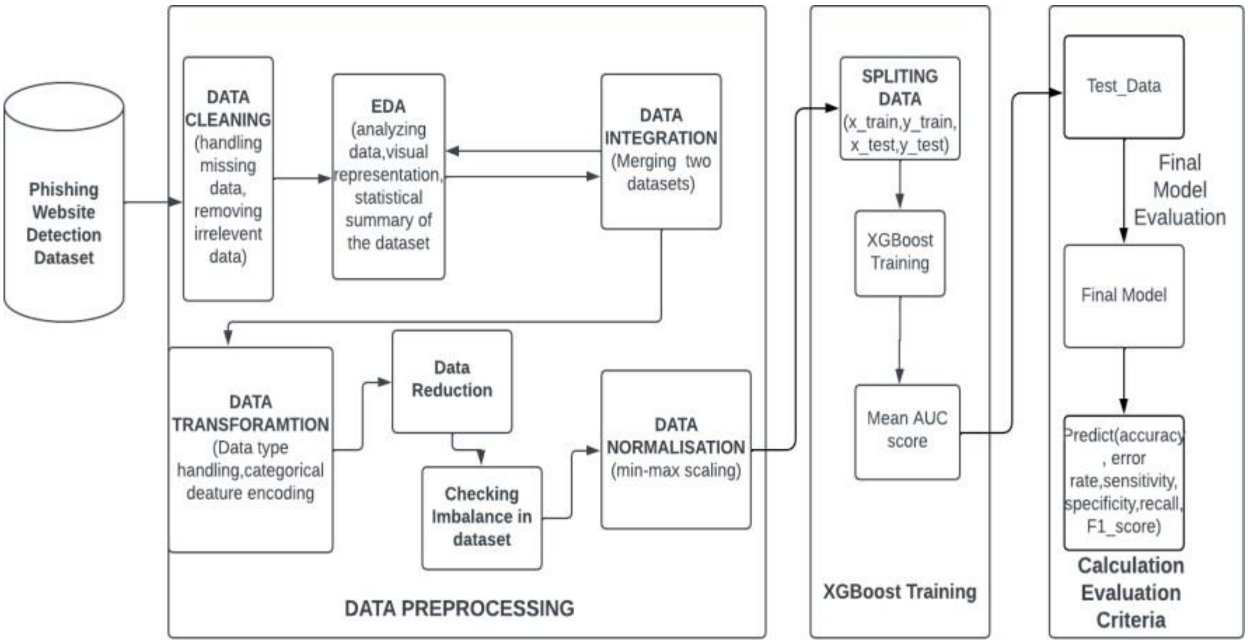


Fig 1:System Architecture

To enhance cybersecurity against phishing attacks, we start by curating a diverse dataset containing both legitimate and malicious websites. Following this, data preprocessing ensures data quality and relevance. Leveraging the XGBoost machine learning algorithm, our model undergoes training to learn patterns and relationships within the dataset. Evaluation criteria such as accuracy, precision, recall, F1 score, and ROC-AUC are then calculated to assess the model's performance. These metrics collectively provide a nuanced understanding of the model's effectiveness, guiding decisions for its deployment in real-world scenarios.

FUNCTIONAL ARCHITECTURE :

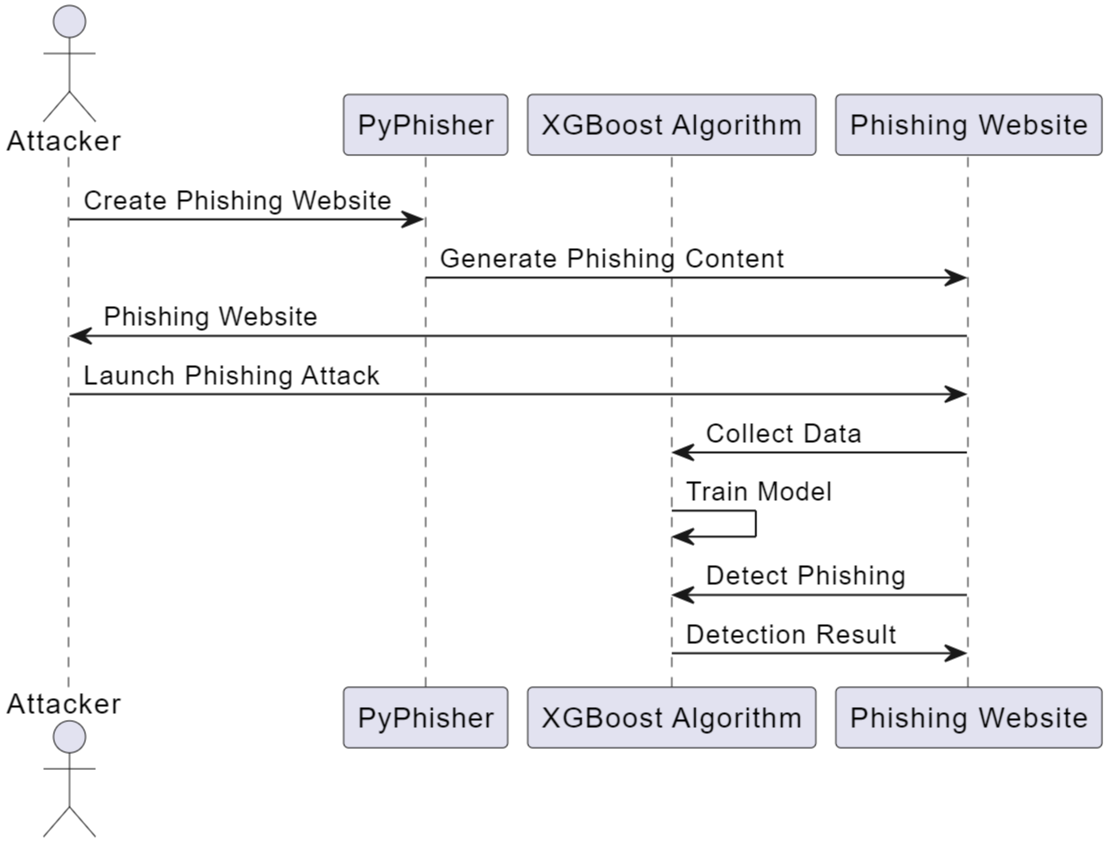


Fig 2 : Functional Architecture

In a phishing attack, the attacker creates a deceptive website resembling legitimate entities, generates convincing messages to lure victims, launches the attack via emails or texts, collects personal information on the fake site, and exploits the gathered data for identity theft or fraud.

MODULAR DESIGN(ACITIVITY DIAGRAM) :

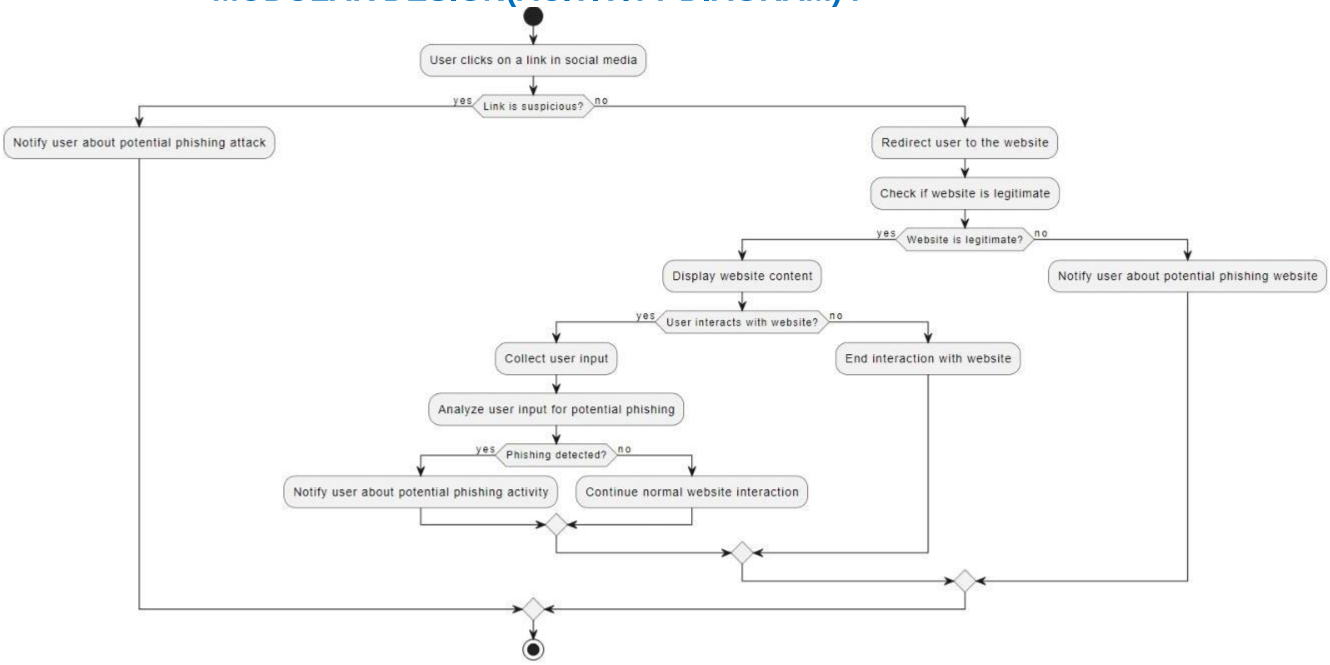


Fig 3 : Activity diagram

1. Start:

- The activity begins with the initiation of the process, denoted by the start symbol.

2. Create Phishing Website (PyPhisher):

- The first major activity involves the creation of a phishing website using PyPhisher.
- Sub-activities include:
- Launching PyPhisher on the Kali Linux platform.
- Setting up parameters and configurations for the phishing website.
- Utilizing PyPhisher features such as spoofing and website cloning to generate a convincing phishing site.
- Saving the generated phishing website for future use.

3. Generate Dataset:

- After creating the phishing website, the next activity involves generating a dataset using the crafted phishing site.
- This includes extracting relevant features like URL structure, HTML content, and page layout from the generated phishing website.

4. Train XGBoost Model:

- The dataset is then used to train the XGBoost machine learning model.- Sub-activities encompass:

- Preprocessing the dataset to prepare it for training.
- Configuring XGBoost parameters for optimal performance.
- Training the XGBoost model on the phishing website dataset.

5. Save Trained Model:

- Once the XGBoost model is trained successfully, it is saved for future use in phishing website detection.

6. Detect Phishing Website (Using XGBoost):

- The detection phase begins by utilizing the trained XGBoost model to identify whether a given website is legitimate or a phishing attempt.
- Features extracted from a new website, such as URL structure and HTML content, are input into the XGBoost model.

7. Evaluate Detection Results:

- The detection results are then evaluated using various metrics to assess the accuracy and effectiveness of the XGBoost model in distinguishing phishing websites from legitimate ones.

8. End:

- The activity concludes with the end symbol, representing the completion of the entire process.

This activity diagram provides a visual representation of the sequential steps involved in the creation of a phishing website using PyPhisher and its subsequent detection using the XGBoost algorithm.

INNOVATIVE IDEA:

The innovative idea in this project lies in the integration of PyPhisher, a tool for creating phishing websites, with the powerful XGBoost algorithm for detection. While existing works often focus on either simulation or detection, our project combines both aspects to provide a comprehensive solution.

1. Realistic Testing Environment:

- Unlike traditional methods that rely on static datasets, our project introduces a more realistic testing environment. PyPhisher allows the creation of authentic-looking phishing websites, mimicking the dynamic tactics employed by attackers. This realistic simulation enhances the effectiveness of the testing phase for security solutions.

2. Synergy of Creation and Detection:

- The synergy between PyPhisher and XGBoost is a unique feature of this project. By utilizing PyPhisher to simulate the creation of phishing websites, security professionals gain insights into the evolving tactics of attackers. The XGBoost algorithm then takes these insights and efficiently detects phishing websites, creating a proactive defense mechanism.

3. Comprehensive Framework:

- The project contributes to cybersecurity by offering a comprehensive framework. PyPhisher's ability to simulate real-world phishing scenarios complements XGBoost's robustness in detecting malicious websites. This holistic approach provides security professionals with a toolset that covers both sides of the phishing threat landscape.

4. Practical Implementation on Kali Linux:

- The use of PyPhisher on Kali Linux adds a practical layer to the project. Kali Linux is a widely recognized distribution for penetration testing and ethical hacking. This integration ensures that the project aligns with industry standards and practices, making it a valuable tool for ethical cybersecurity professionals.

In summary, the innovative aspect of this project lies in the synergy of creating realistic phishing scenarios using PyPhisher and detecting them with the precision of the XGBoost algorithm. This combination offers a unique and comprehensive approach to combating phishing attacks, contributing to the ongoing efforts to enhance cybersecurity measures in an ever-evolving threat landscape.

IMPLEMENTATION DETAILS AND ANALYSIS :

1. Software details and Screen Shots:

I) “Pyphisher -Kali linux”

PyPhisher is an application included in Kali Linux, an important distribution for penetration testing and ethical hacking. PyPhisher is intended for use in phishing attacks, which include tricking people into disclosing personal information such asusernames, passwords, and credit card numbers.

Name: PyPhisher
Purpose: Phishing
Attack Tool**Platform:**
Kali Linux **Features:**

- Spoofing: PyPhisher allows the production of illegal email messages that lookto be issued from legitimate sources, boosting the possibility of victim involvement.
- Website Cloning: It has the ability to clone real websites, creating

phoney login pages or forms that look identical to the original and duping victims intoentering their credentials.

Note(caution):

PyPhisher and other penetration testing tools must be used ethically and lawfully.Unauthorised or evil use of such tools can have major legal and ethical consequences.

Fig 4 : PyPhisher tool

II) JUPYTER NOTEBOOK

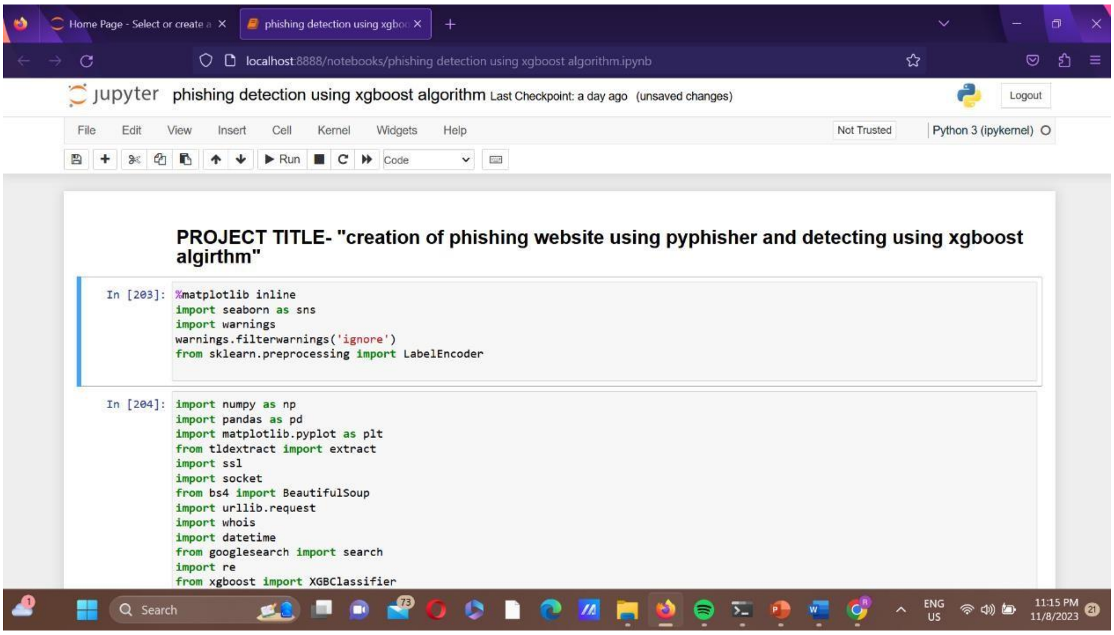


Fig 5 : Jupyter Notebook

2. Sample Code

https://drive.google.com/drive/folders/1m59ZGQei5mb6LTGqLHn7wlV4Z_wXZUCc

3. Outputs :

I. KALI LINUX :-

i) Instagram

Fig 6 : Opening PyPhisher tool

Fig 7 : Giving command for Instagram

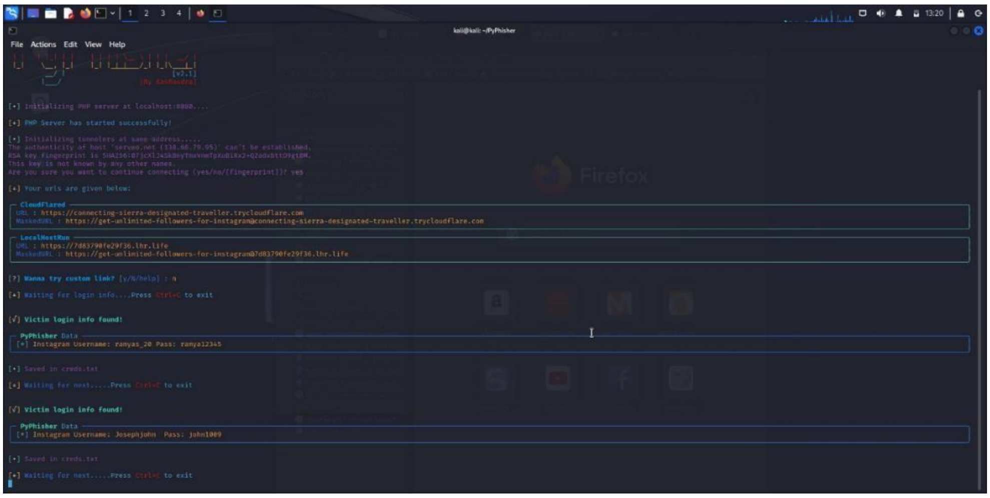


Fig 8 : Login info gathered using phishing URL

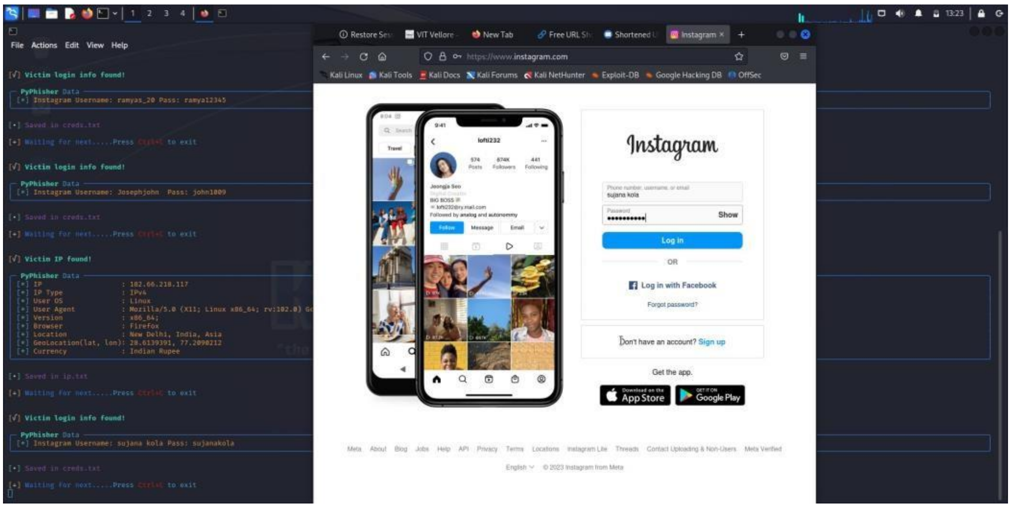


Fig 9 : Login Instagram using gathered credentials

ii) Zomato

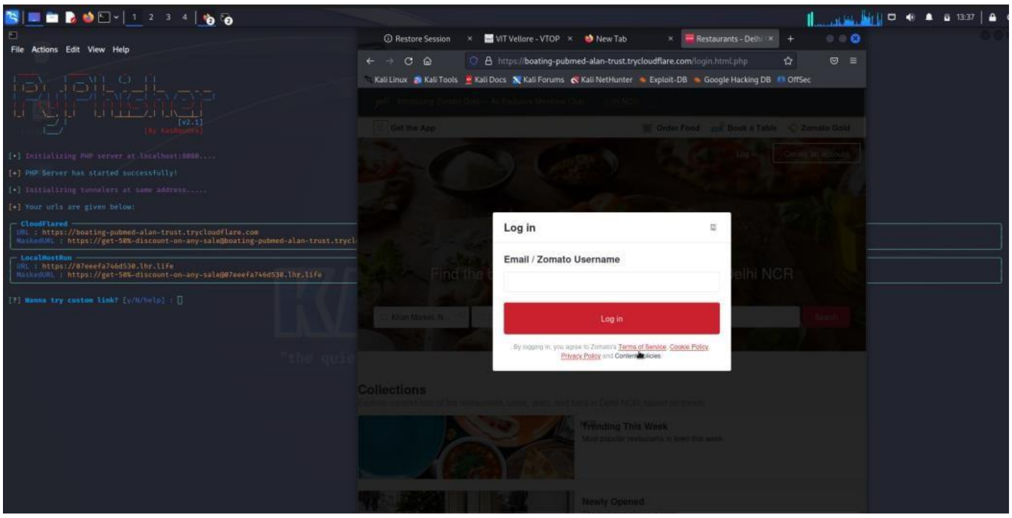


Fig 10 : Zomato phishing URL

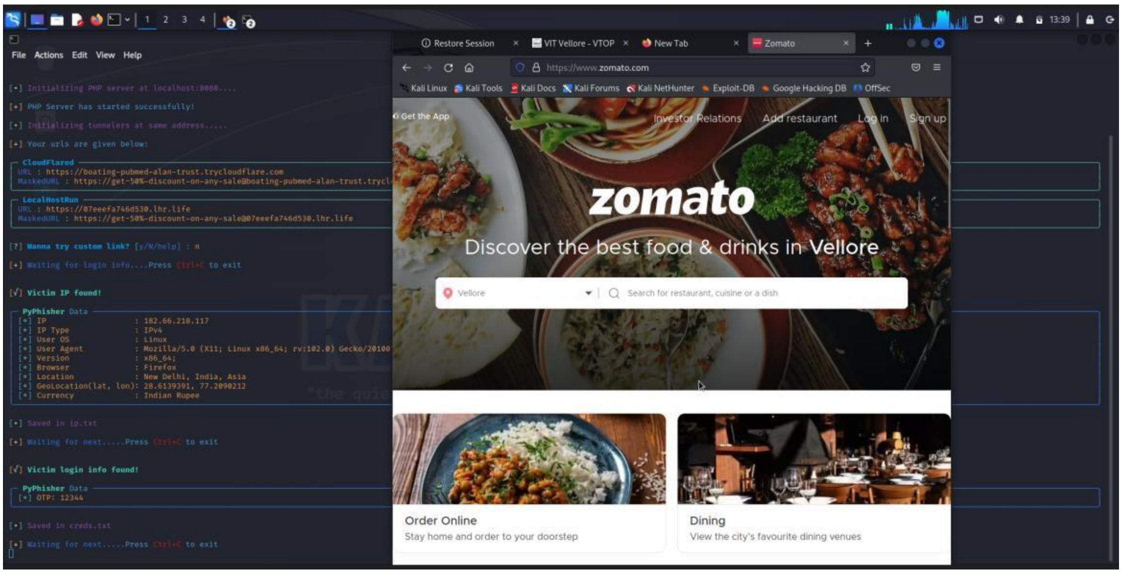


Fig 11 : Logging in Zomato with the gathered credentials

iii) Amazon

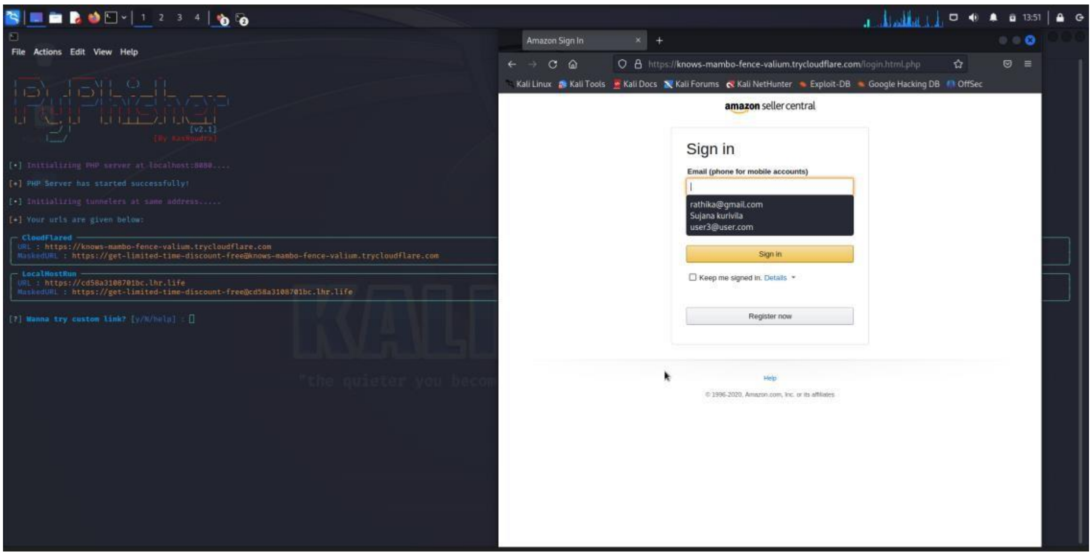


Fig 12 : Amazon phishing url

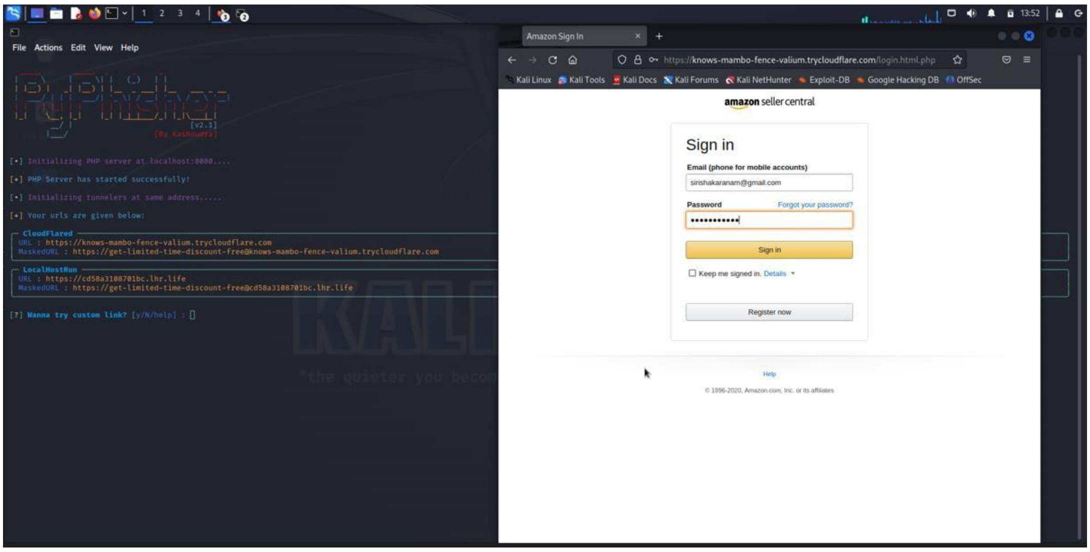


Fig 13 : User trying to login to his account

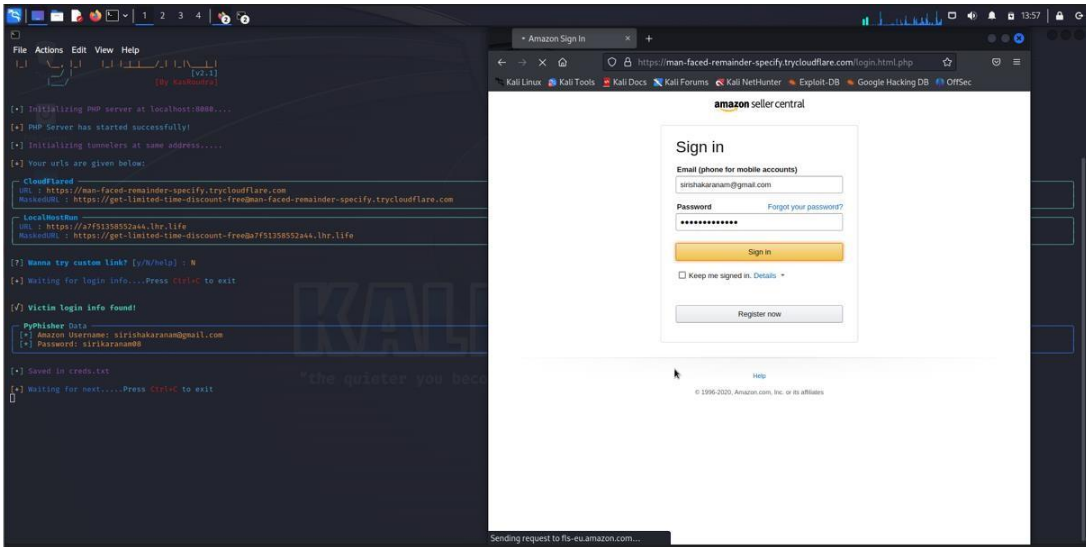


Fig 14 : Gathered credentials using phishing URL

II. JUPYTER NOTEBOOK :-

MACHINE LEARNING MODEL:

DATASET LINK: https://www.kaggle.com/datasets/shashwatwork/web-page-phishing-detection-dataset

```
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from sklearn.preprocessing import LabelEncoder
```

```
[ ] import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from tldextract import extract
import ssl
import socket
from bs4 import BeautifulSoup
import urllib.request
import whois
import datetime
from googlesearch import search
import re
from xgboost import XGBClassifier
```

```
[ ] # reading data from csv file
df1 = pd.read_excel('C:\\Users\\Ragavarshini\\Dropbox\\My PC (LAPTOP-7LQ4HL1A)\\Downloads\\Dataset1.xlsx')
df2 = pd.read_csv('C:\\Users\\Ragavarshini\\Dropbox\\My PC (LAPTOP-7LQ4HL1A)\\Downloads\\Dataset2.csv')
```

About Dataset

```
display(df1,df2)
```

	url	length_url	length_hostname	ip	nb_dots	nb_hyphens	nb_at	nb_qm	nb_and	nb_or	...	domain_in_title	domain_with_copyright	whois_registered_domain	domai
0	http://www.progarchives.com/album.asp?id=61737	46	20	zero	3	zero	0	1	0	0	...	1	one	0	
1	http://signin.eday.co.uk.ws.edayisapi.dllsign....	128	120	0	10	0	0	0	0	0	...	1	zero	0	
2	http://www.avevaconstruction.com/blesstool/ima...	52	25	0	3	0	0	0	0	0	...	1	zero	0	
3	http://www.jp519.com/	21	13	0	2	0	0	0	0	0	...	1	one	0	
4	https://www.velocidrone.com/	28	19	0	2	0	0	0	0	0	...	0	zero	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1493	http://www.highlux.co.nz/	25	17	0	3	0	0	0	0	0	...	0	Zero	0	
1494	http://lvk.hop.ru	17	10	0	2	0	0	0	0	0	...	1	zero	0	
1495	https://oeuf38bui-6pyt2jamu.vercel.app/	39	30	0	2	1	0	0	0	0	...	1	Zero	0	
1496	https://www.byggmax.no/	23	14	0	2	0	0	0	0	0	...	0	one	0	
1497	http://psychologydictionary.org/standard-error/	47	24	0	1	1	0	0	0	0	...	1	Zero	0	

1498 rows × 89 columns

	url	length_url	length_hostname	ip	nb_dots	nb_hyphens	nb_at	nb_qm	nb_and	nb_or	...	domain_in_title	domain_with_copyright	whois_registered_domain	domain_
0	https://www.proteca.jp/	23	14	0	2	0	0	0	0	0	...	0	one	1	
1	https://www.samysprints2go.com/	31	22	0	2	0	0	0	0	0	...	1	one	0	
2	http://spaday-men.ru/wp-content/backups/logon.php	49	13	0	2	2	0	0	0	0	...	1	zero	0	
3	http://www.discovery.com/search/guide/earth.html	48	17	0	3	0	0	0	0	0	...	0	zero	0	
4	http://king-pes.blogspot.com	28	21	0	2	1	0	0	0	0	...	1	zero	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9978	https://623112j4j3.codesandbox.io/index.html	44	25	0	3	0	0	0	0	0	...	0	zero	0	
9979	http://en.academic.ru/dic.nsf/enwiki/279719	43	14	0	3	0	0	0	0	0	...	1	Zero	0	
9980	http://www.neutralsources.com/-/re.html	39	22	0	3	1	0	0	0	0	...	1	zero	0	
9981	http://www.pwc.com/gx/en/financial-services/fi...	114	11	0	3	6	0	0	0	0	...	1	one	0	
9982	http://y9o5m.codesandbox.io/onedrive.html	41	20	0	3	0	0	0	0	0	...	0	zero	0	

9983 rows × 89 columns

```
[ ] df=pd.merge(df1,df2, how='outer')
```

```
df
```

	url	length_url	length_hostname	ip	nb_dots	nb_hyphens	nb_at	nb_qm	nb_and	nb_or	...	domain_in_title	domain_with_copyright	whois_registered_domain	doma
0	http://www.progarchives.com/album.asp?id=61737	46	20	zero	3	zero	0	1	0	0	...	1	one	0	
1	http://signin.eday.co.uk.ws.edayisapi.dllsign....	128	120	0	10	0	0	0	0	0	...	1	zero	0	
2	http://www.avevaconstruction.com/blesstool/ima...	52	25	0	3	0	0	0	0	0	...	1	zero	0	
3	http://www.jp519.com/	21	13	0	2	0	0	0	0	0	...	1	one	0	
4	https://www.velocidrone.com/	28	19	0	2	0	0	0	0	0	...	0	zero	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
11460	http://newlifeschooloftheology.com/wp-includes...	62	27	0	1	1	0	0	0	0	...	1	zero	0	
11461	https://www.marsbahis234.com/tr/	32	20	0	2	0	0	0	0	0	...	1	zero	0	
11462	http://www.cdcenterpa.com/	26	18	0	2	0	0	0	0	0	...	1	zero	0	
11463	https://623112j4j3.codesandbox.io/index.html	44	25	0	3	0	0	0	0	0	...	0	zero	0	
11464	http://www.neutralsources.com/-/re.html	39	22	0	3	1	0	0	0	0	...	1	zero	0	

11465 rows × 89 columns

```
df.head()
```

	url	length_url	length_hostname	ip	nb_dots	nb_hyphens	nb_at	nb_qm	nb_and	nb_or	...	domain_in_title	domain_with_copyright	whois_registered_domain	domain_r
0	http://www.progarchives.com/album.asp?id=61737	46	20	zero	3	zero	0	1	0	0	...	1	one	0	
1	http://signin.eday.co.uk.ws.edayisapi.dllsign....	128	120	0	10	0	0	0	0	0	...	1	zero	0	
2	http://www.avevaconstruction.com/blesstool/ima...	52	25	0	3	0	0	0	0	0	...	1	zero	0	
3	http://www.jp519.com/	21	13	0	2	0	0	0	0	0	...	1	one	0	
4	https://www.velocidrone.com/	28	19	0	2	0	0	0	0	0	...	0	zero	0	

5 rows × 89 columns

```
[ ] # dropping the url column and storing in a new data dataframe 'data'
    data = df.drop(['url','nb_or'], axis = 1)
```

```
[ ] #data = df.drop(['domain_with_copyright'], axis = 1)
```

```
data.head()
```

	length_url	length_hostname	ip	nb_dots	nb_hyphens	nb_at	nb_qm	nb_and	nb_eq	nb_underscore	...	domain_in_title	domain_with_copyright	whois_registered_domain	domain_registration_length	domain_
0	46	20	zero	3	zero	0	1	0	1	0	...	1	one	0	627	66
1	128	120	0	10	0	0	0	0	0	0	...	1	zero	0	300	
2	52	25	0	3	0	0	0	0	0	0	...	1	zero	0	119	17
3	21	13	0	2	0	0	0	0	0	0	...	1	one	0	130	13
4	28	19	0	2	0	0	0	0	0	0	...	0	zero	0	164	16

5 rows × 87 columns

```
[ ] # printing the shape of data
    data.shape
```

```
(11465, 87)
```

```
[ ] # checking whether there is any missing value in the dataframe or not
    data.isnull().values.any()
```

```
False
```

```
data.isnull().sum()
```

```
length_url         0
length_hostname    0
ip                 0
nb_dots            0
nb_hyphens         0
                  ..
web_traffic        0
dns_record         0
google_index       0
page_rank          0
status             0
Length: 87, dtype: int64
```

```
[ ] data.isnull().sum().sum()
```

```
0
```

```
[ ] for i in data.columns:
        data[i].fillna(data[i].mode()[0],inplace=True)
```

```
[ ] data.isnull().values.any()
```

```
False
```

```
#Displaying a summary of the dataset
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11465 entries, 0 to 11464
Data columns (total 87 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   length_url                  11465 non-null  int64
 1   length_hostname             11465 non-null  int64
 2   ip                          11465 non-null  object
 3   nb_dots                     11465 non-null  int64
 4   nb_hyphens                  11465 non-null  object
 5   nb_at                       11465 non-null  int64
 6   nb_qm                       11465 non-null  int64
 7   nb_and                      11465 non-null  int64
 8   nb_eq                       11465 non-null  int64
 9   nb_underscore               11465 non-null  int64
 10  nb_tilde                    11465 non-null  int64
 11  nb_percent                  11465 non-null  int64
 12  nb_slash                    11465 non-null  int64
 13  nb_star                     11465 non-null  int64
 14  nb_colon                    11465 non-null  int64
 15  nb_comma                    11465 non-null  int64
 16  nb_semicolumn               11465 non-null  int64
 17  nb_dollar                   11465 non-null  int64
 18  nb_space                    11465 non-null  int64
 19  nb_www                      11465 non-null  int64
 20  nb_com                      11465 non-null  int64
 21  nb_dslash                   11465 non-null  int64
 22  http_in_path                11465 non-null  int64
 23  https_token                 11465 non-null  int64
 24  ratio_digits_url            11465 non-null  float64
 25  ratio_digits_host           11465 non-null  float64
 26  punycode                    11465 non-null  int64
 27  port                        11465 non-null  int64
 28  tld_in_path                 11465 non-null  int64
```

DATA TRANSFORMATION

```
# filtering the rows which have data type 'object'
list_o_dtype = []
for i in data.columns:
    if data[i].dtype == 'O':
        list_o_dtype.append(i)
```

```
[ ] # taking the last column name out as it is dependent variable
list_o_dtype = list_o_dtype[:-1]
```

```
[ ] # printing the 'list_o_dtype'
list_o_dtype
```

```
['ip', 'nb_hyphens', 'domain_with_copyright']
```

```
[ ] data.dtypes
```

```
length_url          int64
length_hostname     int64
ip                 object
nb_dots             int64
nb_hyphens         object
                    ...
web_traffic         int64
dns_record          int64
google_index        int64
page_rank           int64
status             object
Length: 87, dtype: object
```

```
[ ] data['nb_hyphens']=pd.to_numeric(data['nb_hyphens'],errors='coerce').astype(float)
    data['nb_hyphens'].dtypes
```

```
dtype('float64')
```

```
data['ip']=pd.to_numeric(data['ip'],errors='coerce').astype(float)
data['ip'].dtypes
```

```
dtype('float64')
```

```
data['domain_with_copyright']=pd.to_numeric(data['domain_with_copyright'],errors='coerce').astype(float)
data['domain_with_copyright'].dtypes
```

```
dtype('float64')
```

```
[ ] from sklearn import preprocessing
    le=preprocessing.LabelEncoder()
    data['ip']=le.fit_transform(data['ip'])
    data['ip'].dtypes
    data['nb_hyphens']=le.fit_transform(data['nb_hyphens'])
    data['nb_hyphens'].dtypes
    data['domain_with_copyright']=le.fit_transform(data['domain_with_copyright'])
    data['domain_with_copyright'].dtypes
```

```
dtype('int64')
```

```
[ ] data.isnull().values.any()
```

```
False
```

```
data.dtypes
```

```
length_url          int64
length_hostname     int64
ip                  int64
nb_dots             int64
nb_hyphens          int64
                    ...
web_traffic         int64
dns_record          int64
google_index        int64
page_rank           int64
status             object
Length: 87, dtype: object
```

```
[ ] #Displaying a stastical summary of the dataset
    data.describe()
```

	length_url	length_hostname	ip	nb_dots	nb_hyphens	nb_at	nb_qm	nb_and	nb_eq	nb_underscore	...	empty_title	domain_in_title	domain_with_copyright	who:
count	11465.00000	11465.000000	11465.000000	11465.000000	11465.000000	11465.000000	11465.000000	11465.000000	11465.000000	11465.000000	...	11465.000000	11465.000000	11465.0	
mean	61.21413	21.132316	0.223899	2.496206	1.802791	0.022067	0.140689	0.169036	0.299084	0.334060	...	0.125512	0.773659	0.0	
std	56.82026	10.634692	0.506780	1.422212	4.721049	0.155560	0.365570	0.871668	1.040586	1.166172	...	0.331313	0.418481	0.0	
min	13.00000	4.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.0	
25%	33.00000	15.000000	0.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	1.000000	0.0	
50%	47.00000	19.000000	0.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	1.000000	0.0	
75%	71.00000	24.000000	0.000000	3.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	1.000000	0.0	
max	1641.00000	214.000000	2.000000	24.000000	25.000000	4.000000	3.000000	19.000000	19.000000	18.000000	...	1.000000	1.000000	0.0	

8 rows × 86 columns

Checking imbalances in data

```
# checking whether the dataset is balanced or not
data['status'].value_counts()
```

```
status
legitimate    5733
phishing      5732
Name: count, dtype: int64
```

Unsupported cell type. Double-click to inspect/edit the content.

Scaling data

+ Code + Text

```
[ ] # diving all the columns with the maximum value
    # this will scale everything in between 0 and 1
    for i in data.columns[:-1]:
        data[i] = data[i] / data[i].max()
```

```
[ ] data.head()
```

	length_url	length_hostname	ip	nb_dots	nb_hyphens	nb_at	nb_qm	nb_and	nb_eq	nb_underscore	...	domain_in_title	domain_with_copyright	whois_registered_domain	domain_registration_length	don
0	0.028032	0.093458	1.0	0.125000	1.0	0.0	0.333333	0.0	0.052632	0.0	...	1.0	NaN	0.0	0.021020	
1	0.078001	0.560748	0.0	0.416667	0.0	0.0	0.000000	0.0	0.000000	0.0	...	1.0	NaN	0.0	0.010057	
2	0.031688	0.116822	0.0	0.125000	0.0	0.0	0.000000	0.0	0.000000	0.0	...	1.0	NaN	0.0	0.003989	
3	0.012797	0.060748	0.0	0.083333	0.0	0.0	0.000000	0.0	0.000000	0.0	...	1.0	NaN	0.0	0.004358	
4	0.017063	0.088785	0.0	0.083333	0.0	0.0	0.000000	0.0	0.000000	0.0	...	0.0	NaN	0.0	0.005498	

5 rows × 87 columns

splitting data

```
# label encoding the categorical data
# in this case only the dependent variable (the last column of the datset)
y = pd.get_dummies(df['status'], prefix='type')
# dropping only 1 of the columns as only 1 column can serve our purpose
y = y.drop(['type_phishing'], axis = 1)
```

```
[ ] # splitting the dataset into train and test split

    # taking all the columns except the last column
    x = data.iloc[:, :-1]

    # importing required modules from sklearn for train and test split
    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.shape
```

```
(11465, 86)
```

```
y.shape
```

```
(11465, 1)
```

XGBoost algorithm

```
import sys
print(sys.executable)
```

```
C:\Users\Ragavarshini\anaconda3\python.exe
```

```
[ ] # importing XGBoost
from xgboost import XGBClassifier
#importing accuracy_score from sklearn module for testing accuracy
from sklearn.metrics import accuracy_score
from sklearn import metrics
from sklearn.metrics import classification_report
```

```
[ ] from xgboost import XGBClassifier

# Create an instance of the XGBoost model
xgb_model = XGBClassifier()

# Fit the model to your training data
xgb_model.fit(x_train, y_train)
```

```
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)
```

```
x_test.head()
```

	length_url	length_hostname	ip	nb_dots	nb_hyphens	nb_at	nb_qm	nb_and	nb_eq	nb_underscore	...	empty_title	domain_in_title	domain_with_copyright	whois_registered_domain	domain_registration_
7562	0.035344	0.130841	0.0	0.083333	0.04	0.0	0.0	0.0	0.0	0.000000	...	0.0	1.0	NaN	0.0	0.
8156	0.021328	0.126168	0.0	0.083333	0.04	0.0	0.0	0.0	0.0	0.000000	...	1.0	1.0	NaN	0.0	0.
9055	0.034126	0.065421	0.5	0.125000	0.00	0.0	0.0	0.0	0.0	0.000000	...	0.0	1.0	NaN	1.0	0.
9742	0.024375	0.074766	0.0	0.083333	0.00	0.0	0.0	0.0	0.0	0.000000	...	0.0	0.0	NaN	0.0	0.
9254	0.070079	0.084112	0.5	0.125000	0.00	0.0	0.0	0.0	0.0	0.111111	...	0.0	1.0	NaN	0.0	0.

5 rows × 86 columns

```
[ ] x_test[:5]
```

	length_url	length_hostname	ip	nb_dots	nb_hyphens	nb_at	nb_qm	nb_and	nb_eq	nb_underscore	...	empty_title	domain_in_title	domain_with_copyright	whois_registered_domain	domain_registration_
7562	0.035344	0.130841	0.0	0.083333	0.04	0.0	0.0	0.0	0.0	0.000000	...	0.0	1.0	NaN	0.0	0.
8156	0.021328	0.126168	0.0	0.083333	0.04	0.0	0.0	0.0	0.0	0.000000	...	1.0	1.0	NaN	0.0	0.
9055	0.034126	0.065421	0.5	0.125000	0.00	0.0	0.0	0.0	0.0	0.000000	...	0.0	1.0	NaN	1.0	0.
9742	0.024375	0.074766	0.0	0.083333	0.00	0.0	0.0	0.0	0.0	0.000000	...	0.0	0.0	NaN	0.0	0.
9254	0.070079	0.084112	0.5	0.125000	0.00	0.0	0.0	0.0	0.0	0.111111	...	0.0	1.0	NaN	0.0	0.

5 rows × 86 columns

```
np.array(y_test)
```

```
array([[ True],
       [False],
       [False],
       ...,
       [ True],
       [ True],
       [False]])
```

```
[ ] y_pred = xgb_model.predict(x_test)
    predictions =[round(value) for value in y_pred]
```

```
[ ] accuracy = accuracy_score(y_test, y_pred)
    print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
Accuracy: 98.39%
```

```
[ ] error=(1-accuracy)
    error
```

```
0.016136066288704787
```

```
[ ] print("Recall:  %.2f%%"%(metrics.recall_score(y_test,y_pred,zero_division=1)*100.0))
```

```
Recall:  98.06%
```

```
[ ] print("precision: ",metrics.precision_score(y_test,y_pred,zero_division=1))
```

```
precision:  0.9867021276595744
```

```
print("CL report:",metrics.classification_report(y_test,y_pred,zero_division=1))
```

```
CL report:              precision    recall  f1-score   support

       False       0.98      0.99      0.98      1158
        True       0.99      0.98      0.98      1135

    accuracy                           0.98      2293
   macro avg       0.98      0.98      0.98      2293
weighted avg       0.98      0.98      0.98      2293
```

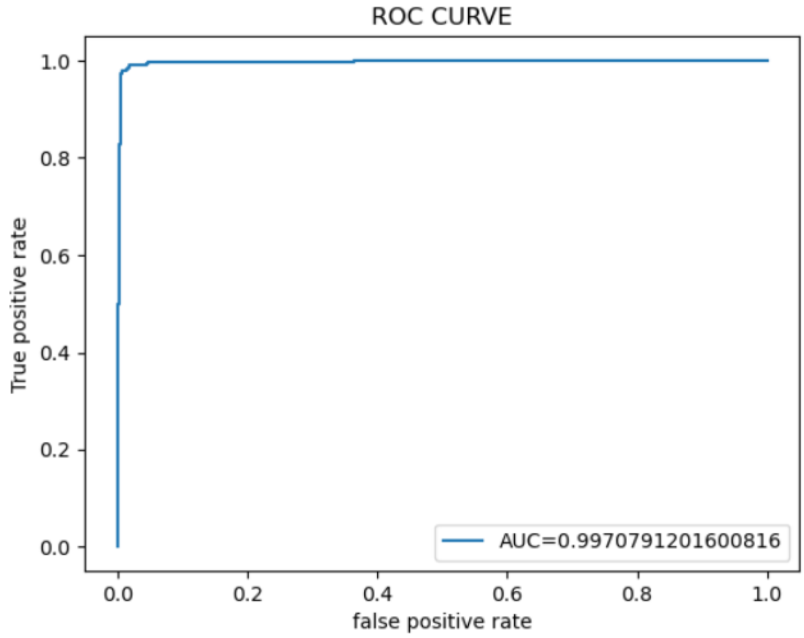
```
[ ] y_pred_proba=xgb_model.predict_proba(x_test) [::,1]
```

```
[ ] false_positive_rate,true_positive_rate,_=metrics.roc_curve(y_test,y_pred_proba)
```

```
[ ] auc=metrics.roc_auc_score(y_test,y_pred_proba)
```

```
plt.plot(false_positive_rate,true_positive_rate,label="AUC="+str(auc))
plt.title('ROC CURVE')
plt.ylabel('True positive rate')
plt.xlabel('false positive rate')
plt.legend(loc=4)
```

```
<matplotlib.legend.Legend at 0x1ff034561d0>
```

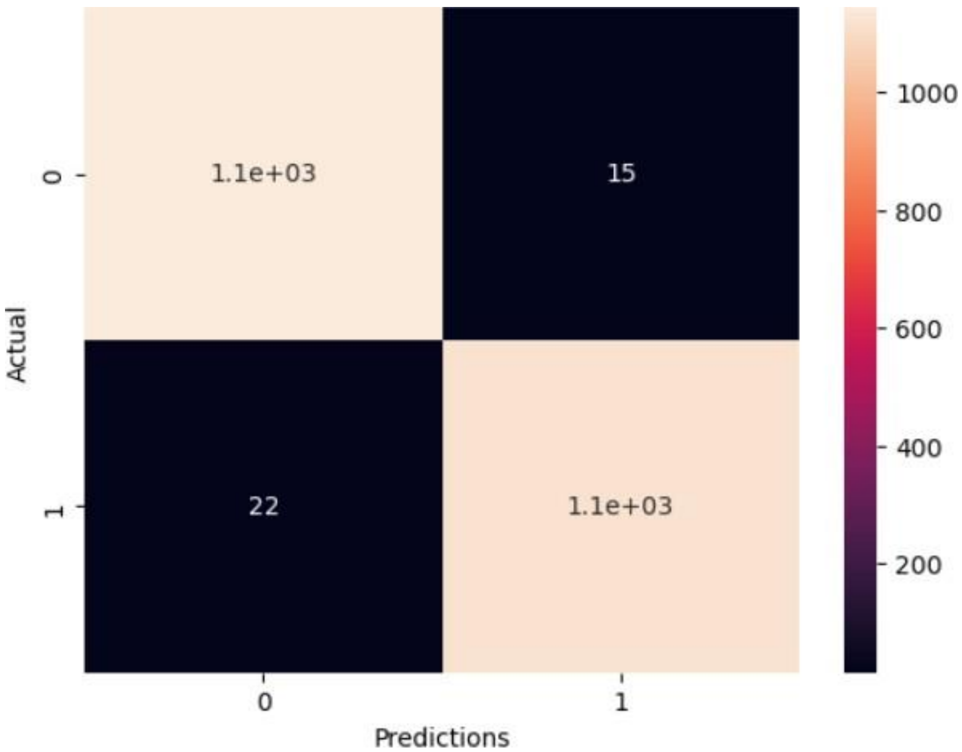


Plotting confusion matrix

```
# importing the necessary libraries for plottinf confusion matrix
from sklearn.metrics import confusion_matrix
import seaborn as sn
```

```
[ ] cnm = confusion_matrix(y_test, predictions)
```

```
[ ] # using heatmap for plotting confusion matrix
    sn.heatmap(cnm, annot=True)
    plt.xlabel('Predictions')
    plt.ylabel('Actual')
    plt.show()
```




```
[ ] y_pred = xgb_model.predict(x_test)
    predictions =[round(value) for value in y_pred]
```

```
predictions
```

```
[1,
 0,
 0,
 1,
 0,
 0,
 0,
 1,
 0,
 0,
 1,
 1,
 1,
 0,
 0,
 0,
 1,
 1,
 1,
 0,
 1,
 0,
 1,
 1,
 1,
 0,
 0,
 1,
 1,
 1,
```

```
[ ] xgb_model.predict(x_test[7:8])
```

```
array([1])
```

```
[ ] df=np.array(xgb_model.predict(x_test[7:8]))
```

```
[ ] x_test[7:8]
```

	length_url	length_hostname	ip	nb_dots	nb_hyphens	nb_at	nb_qm	nb_and	nb_eq	nb_underscore	...	empty_title	domain_in_title	domain_with_copyright	whois_registered_domain	domain_registration_
6308	0.01036	0.042056	0.0	0.041667	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	NaN	0.0	0.

1 rows × 86 columns

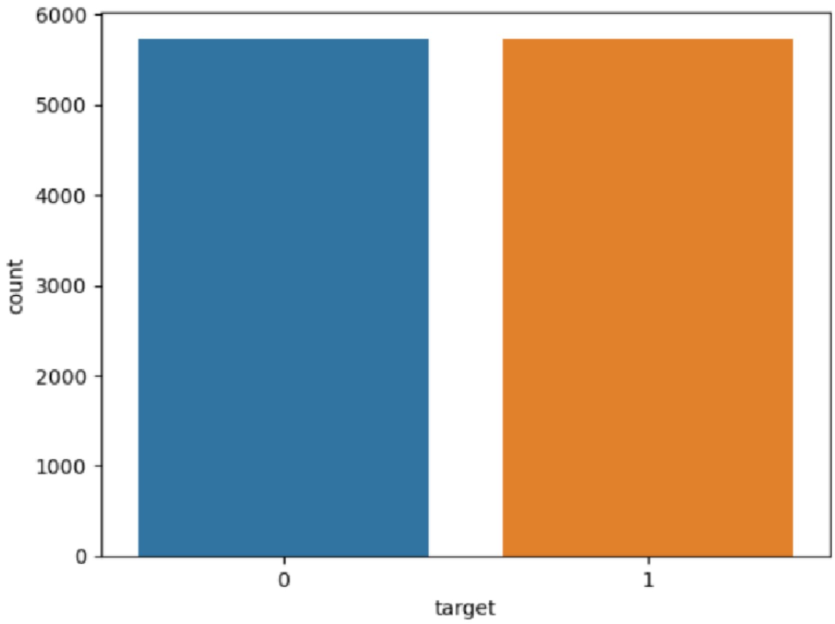
```
[ ] if (np.array(xgb_model.predict(x_test[7:8]).any())==0):
          print("this is legitimate website")
    else:
       print("this is phishing website")
```

```
this is phishing website
```

```
# encoding the target column
label_encode = LabelEncoder()

labels = label_encode.fit_transform(data['status'])

data['target'] = labels

data.drop(columns='status', axis=1, inplace=True)
```

```
sns.countplot(x='target', data=data)
```

```
<Axes: xlabel='target', ylabel='count'>
```



CONCLUSION AND FUTURE WORK :

In conclusion, our proposed system presents a dynamic and comprehensive solution to combat the escalating threat of phishing attacks. By integrating the PyPhisher tool for realistic threat simulation and leveraging the XGBoost machine learning algorithm for robust detection capabilities, our system addresses key limitations of traditional cybersecurity approaches.

The creation phase, facilitated by PyPhisher, enables the generation of phishing websites that closely emulate trusted entities. This not only provides security professionals with a realistic testing environment but also allows them to anticipate and understand the evolving tactics of malicious actors. This innovative approach overcomes the constraints associated with static datasets, offering a more adaptive and proactive defense strategy.

In the detection phase, the XGBoost classifier proves to be a powerful tool, effectively distinguishing between legitimate and malicious websites. Its ability to handle high-dimensional data and resist overfitting enhances the accuracy and reliability of our phishing detection model. By extracting features from phishing websites, such as HTML content, URL structure, and page layout, our system contributes to a more advanced and adaptive defense mechanism against the ever-evolving landscape of phishing attacks.

As part of our future work, we plan to take our system to the next level by creating a user-friendly website using HTML and CSS. This platform will integrate the saved model, stored as a pickle file, into a Flask framework. By allowing users to upload URLs and receive predictions on whether they are phishing websites or not, we aim to democratize the power of our advanced detection model. This step will not only enhance user engagement but also contribute to a wider implementation of our solution in real-world cybersecurity scenarios.

In essence, our proposed system offers a holistic and forward-thinking approach to cybersecurity, bridging the gap between realistic threat simulation and cutting-edge machine learning techniques. As we move towards implementing our model in a user-friendly web application, we anticipate further advancements in the proactive defense against the ever-evolving landscape of phishing attacks.

REFERENCES:

1. Kothamasu, Ganga Abhirup, et al. "An Investigation on Vulnerability Analysis of Phishing Attacks and Countermeasures." International Journal of Safety & Security Engineering 13.2 (2023). https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&as_ylo=2022&q=An+Investigation+on+Vulnerability+Analysis+of+Phishing+Attacks+and+Countermeasur+es.&btnG=
2. Spammer Detection and Fake User Identification on Social Networks. (2019). IEEE Journals& Magazine | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/8719906
3. A Neural Network-Based Ensemble Approach for Spam Detection in Twitter. (2018,December 1). IEEE Journals & Magazine | IEEE Xplorer https://ieeexplore.ieee.org/abstract/document/8540077

4. Email Spam Detection Using Machine Learning Algorithms. (2020, July 1). IEEE Conference Publication | IEEE Xplore.
https://ieeexplore.ieee.org/abstract/document/9183098
5. Phishing URL Detection Using URL Ranking. (2015, June 1). IEEE Conference Publication |IEEE Xplore.
https://ieeexplore.ieee.org/abstract/document/7207281/authors#aut hors
6. Spears Against Shields | Proceedings of the ACM International Workshop on Security and Privacy Analytics. (n.d.). ACM Conferences.
https://dl.acm.org/doi/abs/10.1145/3309182.3309191
7. Fresh-Phish: A Framework for Auto-Detection of Phishing Websites. (2017, August 1). IEEE Conference Publication | IEEE Xplore.
https://ieeexplore.ieee.org/abstract/document/8102930
8. Razaque, A., Frej, M. B. H., Sabyrov, D., Shaikhyn, A., Amsaad, F., & Oun, A. (2020, October). Detection of phishing websites using machine learning. In 2020 IEEE Cloud Summit(pp. 103-107). IEEE.
https://ieeexplore.ieee.org/abstract/document/9283682
9. Ahmed, A. A., & Abdullah, N. A. (2016, October). Real time detection of phishing websites. In2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON) (pp. 1-6). IEEE.
https://ieeexplore.ieee.org/abstract/document/7746247
10. Rao, Routhu Srinivasa, Tatti Vaishnavi, and Alwyn Roshan Pais. "CatchPhish: detection of phishing websites by inspecting URLs." Journal of Ambient Intelligence and Humanized Computing 11 (2020): 813-825.
https://link.springer.com/article/10.1007/s12652-019-01311-4
11. Miyamoto, D., Hazeyama, H., & Kadobayashi, Y. (2009). An evaluation of machine learning-based methods for detection of phishing sites. In Advances in Neuro-Information Processing:15th International Conference, ICONIP 2008, Auckland, New Zealand, November 25-28, 2008, Revised Selected Papers, Part I 15 (pp. 539-546). Springer Berlin Heidelberg.
https://link.springer.com/chapter/10.1007/978-3-642- 02490-0_66
12. Geng, G. G., Lee, X. D., Wang, W., & Tseng, S. S. (2013,September). Favicon-a clue tophishing sites detection. In 2013 APWG eCrime Researchers IEEE
.https://ieeexplore.ieee.org/abstract/document/6805775
13. Marchal, S., François, J., State, R., & Engel, T. (2014). PhishStorm: Detecting phishing with streaming analytics. IEEE Transactions on Network and Service Management, 11(4), 458-471.
https://ieeexplore.ieee.org/abstract/document/6975177
14. F. Castaño, E. F. Fernañdez, R. Alaiz-Rodríguez and E. Alegre, "PhiKitA: Phishing Kit AttacksDataset for Phishing Websites Identification," in IEEE Access, vol. 11, pp. 40779-40789, 2023, doi:10.1109/ACCESS.2023.3268027.
https://ieeexplore.ieee.org/document/10103863
15. E. Zhu, Y. Chen, C. Ye, X. Li and F. Liu, "OFS-NN: An Effective Phishing Websites Detection Model Based on Optimal Feature Selection and Neural Network," in IEEE Access, vol. 7, pp. 73271-73284, 2019
https://ieeexplore.ieee.org/document/8730309
16. A. Karim, M. Shahroz, K. Mustofa, S. B. Belhaouari and S. R. K. Joga, "Phishing DetectionSystem Through Hybrid Machine Learning Based on URL," in IEEE Access, vol. 11, pp. 36805-36822, 2023, doi: 10.1109/ACCESS.2023.3252366 .
https://ieeexplore.ieee.org/document/10058201

17. Y. A. Alsariera, V. E. Adeyemo, A. O. Balogun and A. K. Alazzawi, "AI Meta-Learners andExtra-Trees Algorithm for the Detection of Phishing Websites," in IEEE Access, vol. 8, pp.142532-142542, 2020, doi: 10.1109/ACCESS.2020.3013699.
https://ieeexplore.ieee.org/document/9154378
18. L. R. Kalabarige, R. S. Rao, A. Abraham and L. A. Gabralla, "Multilayer Stacked EnsembleLearning Model to Detect Phishing Websites," in IEEE Access, vol. 10, pp. 79543-79552, 2022, doi: 10.1109/ACCESS.2022.3194672.
https://ieeexplore.ieee.org/document/9843994\
19. W. Ali and S. Malebary, "Particle Swarm Optimization-Based Feature Weighting for ImprovingIntelligent Phishing Website Detection," in IEEE Access, vol. 8, pp. 116766-116780, 2020, doi: 10.1109/ACCESS.2020.3003569
https://ieeexplore.ieee.org/document/912122 7
20. L. Tang and Q. H. Mahmoud, "A Deep Learning-Based Framework for Phishing Website Detection," in IEEE Access, vol. 10, pp. 1509-1521, 2022, doi: 10.1109/ACCESS.2021.3137636.
https://ieeexplore.ieee.org/document/9 661323