

WEB PHISHING DETECTION USING MACHINE LEARNING

APPLIED DATA SCIENCE DOMAIN

TEAM ID: PNT2022TMID11486

A PROJECT REPORT

Submitted by

BALAJEE A V(910619106006)

ASWIN T S(910619106005)

BALAJI S(910619106007)

ABISHEIK R(910619106002)

ELECTRONICS AND COMMUNICATION ENGINEERING

K.L.N. COLLEGE OF ENGINEERING

POTTAPALAYAM



TEAM MENTOR

- DR. T.R.MUTHU

EVALUATOR

- DR. S.SUBHA

INDUSTRY MENTOR

- SANDESH P

November 2022

ANNA UNIVERSITY : CHENNAI 600025

BONAFIDE CERTIFICATE

**Certified that this project report “WEB PHISHING DETECTION USING
MACHINE LEARNING” is the bonafide work of**

“BALAJEE A V (910619106006),

ASWIN T S (910619106005),

BALAJI S (910619106007),

ABISHEIK R (910619106002)”,

Who carried out the project work under our supervision.

**SIGNATURE
FACULTY MENTOR**

DR. T.R. MUTHU

ASSISTANT PROFESSOR (Sr.Gr)

**ELECTRONICS AND
COMMUNICATION ENGINEERING
K.L.N COLLEGE OF ENGINEERING**

**SIGNATURE
FACULTY EVALUATOR**

DR. S.SUBHA

ASSISTANT PROFESSOR

**ELECTRONICS AND
COMMUNICATION ENGINEERING
K.L.N COLLEGE OF ENGINEERING**

SIGNATURE

DR.V.KEJALAKSHMI

HEAD OF THE DEPARTMENT

ELECTRONICS AND COMMUNICATION ENGINEERING

K.L.N COLLEGE OF ENGINEERING

ABSTRACT

A web service is one of the most important Internet communications software services. Using fraudulent methods to get personal information is becoming increasingly widespread these days. However, it makes our lives easier, it leads to numerous security vulnerabilities to the Internet's private structure. Web phishing is just one of the many security risks that web services face. Phishing assaults are usually detected by experienced users however, security is a primary concern for system users who are unaware of such situations. Phishing is the act of portraying malicious web runners as genuine web runners to obtain sensitive information from the end-user. Phishing is currently regarded as one of the most dangerous threats to web security. Vicious Web sites significantly encourage Internet criminal activity and inhibit the growth of Web services. As a result, there has been a tremendous push to build a comprehensive solution to prevent users from accessing such websites. We suggest a literacy-based strategy to categorize Web sites into three categories: benign, spam, and malicious. Our technology merely examines the Uniform Resource Locator (URL) itself, not the content of Web pages. As a result, it removes run-time stillness and the risk of drug users being exposed to cyber surfer-based vulnerabilities. When compared to a blacklisting service, our approach performs better on generality and content since it uses learning techniques.

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
2.3.	Problem Statement Definition	6
3.1	Empathy map canvas	7
3.2	Ideation & Brainstorming	7
3.4	Problem solution fit	9
5.1	Data Flow Diagrams	11
6.2	Burndown Chart	15
6.3	JIRA Report	15
7.3	Test URL-1	32

LIST OF TABLE

FIGURE NO	TITLE	PAGE NO
3.3	Proposed Solution	8
4.1	Functional Requirement	10
4.2	Non-Functional Requirements	10
5.2.1	Components & Technologies	12
5.2.2	Application Characteristics	12
5.3	User Stories	13
6.1	Sprint Planning & Estimation	15
6.2	Sprint Delivery Schedule	15
8.1	Test Cases	34
8.2	User Acceptance Testing - UAT	34
9.1	Performance Metrics	36

TABLE OF CONTENTS

CH NO.	TITLE	PAGE NO.
	ABSTRACT	iii
	LIST OF FIGURE	iv
	LIST OF TABLE	v
1	INTRODUCTION	1
1.1	PROJECT OVERVIEW	1
1.2	PURPOSE	2
2	LITERATURE SURVEY	3
2.1	EXISTING PROBLEM	4
2.2	REFERENCES	5
2.3	PROBLEM STATEMENT DEFINITION	6
3	IDEATION & PROPOSED SOLUTION	7
3.1	EMPATHY MAP CANVAS	7
3.2	IDEATION & BRAINSTORMING	7
3.3	PROPOSED SOLUTION	8
3.4	PROBLEM SOLUTION FIT	9
4	REQUIREMENT ANALYSIS	10
4.1	FUNCTIONAL REQUIREMENT	10
4.2	NON-FUNCTIONAL REQUIREMENTS	10
5	PROJECT DESIGN	11

5.1	DATA FLOW DIAGRAMS	11
5.2	SOLUTION & TECHNICAL ARCHITECTURE	12
5.3	USER STORIES	13
6	PROJECT PLANNING & SCHEDULING	14
6.1	SPRINT PLANNING & ESTIMATION	14
6.2	SPRINT DELIVERY SCHEDULE	14
6.3	REPORTS FROM JIRA	15
7	CODING & SOLUTIONING	16
7.1	FEATURE 1	16
7.2	FEATURE 2	21
7.3	CLOUD DEPLOYMENT	32
8	TESTING	34
8.1	TEST CASES	34
8.2	USER ACCEPTANCE TESTING	35
9	RESULTS	36
9.1	PERFORMANCE METRICS	36
10	ADVANTAGES & DISADVANTAGES	40
11	CONCLUSION	41
12	FUTURE SCOPE	42
13	APPENDIX	43
	SOURCE CODE	

CHAPTER 1

INTRODUCTION

Introduction to Web Phishing

Phishing is a type of social engineering where an attacker sends a fraudulent (e.g., spoofed, fake, or otherwise deceptive) message designed to trick a person into revealing sensitive information to the attacker or to deploy malicious software on the victim's infrastructure like ransomware. Phishing attacks have become increasingly sophisticated and often transparently mirror the site being targeted, allowing the attacker to observe everything while the victim is navigating the site, and transverse any additional security boundaries with the victim. As of 2020, phishing is by far the most common attack performed by cybercriminals, the FBI's Internet Crime Complaint Centre recording over twice as many incidents of phishing than any other type of computer crime.

The first recorded use of the term "phishing" was in the cracking toolkit AOHell created by Koceilah Rekouche in 1995; however, it is possible that the term was used before this in a print edition of the hacker magazine *2600*. The word is a variant of *fishing*, influenced by phreaking, and alludes to the use of increasingly sophisticated lures to "fish" for users' sensitive information.

Attempts to prevent or mitigate the impact of phishing incidents include legislation, user training, public awareness, and technical security measures. Phishing awareness has become important at home and at the work place. For instance, from 2017 to 2020, phishing attacks have increased from 72% to 86% among businesses.

1.1 PROJECT OVERVIEW

There are a number of users who purchase products online and make payments through e-banking. There are e-banking websites that ask users to provide sensitive data such as username, password & credit card details, etc., often for malicious reasons. This type of e-banking website is known as a phishing website. Web service is one of the key communications software services for the Internet. Web phishing is one of many security threats to web services on the Internet.

Common threats of web phishing:

- Web phishing aims to steal private information, such as usernames, passwords, and credit card details, by way of impersonating a legitimate entity.
- It will lead to information disclosure and property damage.
- Large organizations may get trapped in different kinds of scams.

In order to detect and predict e-banking phishing websites, we proposed an intelligent, flexible and effective system that is based on using classification algorithms. We implemented classification algorithms and techniques to extract the phishing datasets criteria to classify their legitimacy. The e-banking phishing website can be detected based on some important characteristics like URL and domain identity, and security and encryption criteria in the final phishing detection rate. Once a user makes a transaction online when he makes payment through an e-banking website our system will use a data mining algorithm to detect whether the e-banking website is a phishing website or not.

1.2 PURPOSE

The purpose of this project is to predict phishing website using the data from Kaggle Dataset, and compare different classification models. There have been famous detection problems such as Credit card Fraud Detection, while people have not done great phishing detection because they don't have data with enough attributes. The provided dataset includes 11430 URLs with 87 extracted features. The dataset is designed to be used as benchmarks for machine learning-based phishing detection systems. Features are from three different classes: 56 extracted from the structure and syntax of URLs, 24 extracted from the content of their correspondent pages, and 7 are extracted by querying external services. The dataset is balanced, it contains exactly 50% phishing and 50% legitimate URLs. The gradient boosting algorithm often provides predictive accuracy that cannot be trumped and can optimize on different loss functions and provides several hyper parameter tuning options that make the function fit very flexible. We finally proved that these features are important and good for classification accuracy. Gradient Boosting has repeatedly proven to be one of the most powerful techniques to build predictive models in both classification and regression. Because of the fact that Gradient Boosting algorithms can easily overfit on a training data set, different constraints or regularization methods can be utilized to enhance the algorithm's performance and combat overfitting.

CHAPTER 2

LITERATURE SURVEY

In emerging technology, industry, which deeply influence today's security problems, has given a headache to many employers and home users. Occurrences that exploit human vulnerabilities have been on the upsurge in recent years. In these new times there are many security systems being enabled to ensure security is given the outmost priority and prevention to be taken from being hacked by those who are involved in cyber-offenses and essential prevention is taken as high importance in organization to ensure network security is not being compromised. Cyber security employee are currently searching for trustworthy and steady detection techniques for phishing websites detection. Due to wide usage of internet to perform various activities such as online bill payment, banking transaction, online shopping, etc. Customer face numerous security threats like cybercrime. Many cybercrime is being casually executed for example spam, fraud, identity theft cyber terrorisms and phishing. Among this phishing is known as the most common cybercrime today. Phishing has become one amongst the top three most current methods of law breaking in line with recent reports, and both frequency of events and user weakness has increased in recent years, more combination of all these methods result in greater danger of economic damage. Phishing is a social engineering attack that targets and exploiting the weakness found in the system at the user's end. This paper proposes the Agile Unified Process (AUP) to detect duplicate websites that can potentially collect sensitive information about the user. The system checks the blacklisted sites in dataset and learns the patterns followed by the phishing websites and applies it to further given inputs. The system sends a pop-up and an e-mail notification to the user, if the user clicks on a phishing link and redirects to the site if it is a safe website. This system does not support real time detection of phishing sites; user has to supply the website link to the system developed with Microsoft Visual Studio 2010 Ultimate and MySQL stocks up data and to implement database in this system. Phishing costs Internet user's lots of money. It refers to misusing weakness on the user side, which is vulnerable to such attacks. The basic ideology of the proposed solution is use to all the three-hybrid solution blacklist and whitelist, heuristics and visual similarity. The proposed system carries out a set of procedures before giving out the results. First, it tracks all "http" traffic of client system by creating a browser extension. Then compare domain of each URL with the white list of trusted domains and the blacklist of illegitimate domains.

Further various characters in the URL is considered like number of '@', number of '-' and many more. Next approach is to extract and compare CSS of doubtful URL and compare it with the CSS of each of the legitimate domains in queue. This method will look into visual based features of the phished websites and machinelearning classifiers such as decision tree, logistic regression, random forest are applied to the collected data, and a score is generated. The match score and similarity score is evaluated. If the score is greater than threshold then the URL marked as phishing and blocked. This approach provides a three level security block. Phishing is a dangerous effort to steal private data from users like address, Aadhar number, PAN card details, credit or debit card details, bank account details, personal details etc. The various types of phishing attacks like spoofing, instant spam spoofing, Hosts file poisoning, malware-based phishing, Man-in-the middle, session hijacking, DNS based phishing, deceptive phishing, key loggers/loggers, Web Trojans, Data theft, Content-injection phishing, Search engine phishing, Email /Spam, Web based delivery, Link Manipulation, System reconfiguration, Phone phishing, etc.

2.1 EXISTING PROBLEMS

The importance to safeguard online users from becoming victims of online fraud, divulging confidential information to an attacker among other effective uses of phishing as an attacker's tool, phishing detection tools play a vital role in ensuring a secure online experience for users. Unfortunately, many of the existing phishing-detection tools, especially those that depend on an existing blacklist, suffer limitations such as low detection accuracy and high false alarm that is often caused by either a delay in blacklist update as a result of human verification process involved in classification or perhaps, it can be attributed to human error in classification which may lead to improper classification of the classes.

These critical issues have drawn many researchers to work on various approaches to improve detection accuracy of phishing attacks and to minimize false alarm rate. The inconsistent nature of attacks behaviors and continuously changing URL phish patterns require timely updating of the reference model. Therefore, it requires an effective technique to regulate retraining as to enable machine learning algorithm to actively adapt to the changes in phish patterns. The ML based phishing techniques depend on website functionalities to gather information that can help classify websites for detecting phishing sites. The problem of phishing cannot be eradicated,

nonetheless can be reduced by combating it in two ways, improving targeted anti-phishing procedures and techniques and informing the public on how fraudulent phishing websites can be detected and identified. To combat the ever evolving and complexity of phishing attacks and tactics, ML anti-phishing techniques are essential.

2.2 REFERENCES

- ▶ Detecting Phishing Websites Using Machine Learning by Sagar Patil, Yogesh Shetye, Nilesh Shendage published in the year 2020.
- ▶ Machine Learning-Based Phishing Attack Detection by Sohrab Hossain, Dhiman Sarma, Rana Joythi Chakma published in the year 2020.
- ▶ Phishing website detection based on effective machine learning approach by Gururaj Harinahalli Lokesh published in the year 2020.
- ▶ Research on Website Phishing Detection Based on LSTM RNN by Yang Su published in the year 2020.
- ▶ Detecting Phishing Website Using Machine Learning by Mohammed Hazim Alkawaz, Stephanie Joanne Steven, Asif Iqbal Hajamydeen published in the year 2020.
- ▶ Fette, I., Sadeh, N.M., Tomasic, A. "Learning to detect phishing emails." In Proceedings of the 16th International Conference on World Wide Web (WWW'07), May 2017.
- ▶ Abu-Nimeh, S., Nappa, D., Wang, X., Nair, S. "A comparison of machine learning techniques for phishing detection." In Proceedings of eCrime Researchers Summit (eCryme '07), Oct 2010.
- ▶ Basnet, R., Mukkamala, S., Sung, A.H. "Detection of phishing attacks: A machine learning approach." *Studies in Fuzziness and Soft Computing*, 226:373–383, 2014.
- ▶ Lakshmi, V. Santhana, and M. S. Vijaya. "Efficient prediction of phishing websites using supervised learning algorithms." *Procedia Engineering* 30 (2012): 798-805.
- ▶ Ramzan, Zulfikar (2010). "Phishing attacks and countermeasures". In Stamp, Mark; Stavroulakis, Peter (eds.). *Handbook of Information and Communication Security*. Springer.
- ▶ Quintin, Cooper (August 27, 2015). "New Spear Phishing Campaign Pretends to be EFF". EFF. Archived from the original on August 7, 2019. Retrieved November 29, 2016.

2.3 PROBLEM STATEMENT DEFINITION

A problem statement is a concise description of the problem or issues a project seeks to address. The problem statement identifies the current state, the desired future state and any gaps between the two. A problem statement is an important communication tool that can help ensure everyone working on a project knows what the problem they need to address is and why the project is important.



Figure 2.3 Problem Statement Definition

3.3 PROPOSED SOLUTION

Table 3.3 Proposed Solution

Team ID	PNT2022TMID11486
Project Name	Web Phishing Detection
Date	09.10.2022
Team Members	1. Balajee A V 2. Aswin T S 3. Balaji S 4. Abisheik R

S.NO.	PARAMETER	DESCRIPTION
1.	Problem Statement (problem to be solved)	A particular challenge in this domain is that notorious hackers are constantly making new strategies to break into our defense measures. The drawback of the existing systems is detecting some minor false positive and false negative results. These disadvantages can be abolished by introducing a much-enhanced feature to feed to the machine learning algorithm that would result in much higher accuracy.
2.	Idea/ Solution Description	We focus on the direct implementation of the project to the chrome extension so that as the user clicks on the particular URL and if that URL is a phishing site then the user gets a pop-up warning message.
3.	Novelty/ Uniqueness	1.Using Machine Learning we developed the Web application as Web Phishing Detection 2. It checks the URL and no of users visited in that particular webpage or website and creates an alert to the user if the website is dangerous or not .
4.	Social Impact/ Customer Satisfaction	1. To spread the cyber awareness on multiple attacks mainly on this phishing attack. 2. An individual can unlearn and relearn this model in various types of aspects in Cyber security and Data theft.
5.	Business Model (Financial Benefit)	1.This model helps Banking and Financial sectors from data loss and data attack which leads to zero financial loss externally. 2.In Business Organization they can use this tool to get rid from cyber attack and can implement how to improve the security when this attack occur next time.
6.	Scalability of Solution	1. We deliver the Good feasible UI/UX design on Web phishing detection. 2. The model is tested and trained in multiple types of datasets to get high accuracy than other algorithms.

3.4 PROBLEM SOLUTION FIT

Problem/solution fit is the very beginning stage of a startup when founders are using design thinking to shape their business idea and validate it with their users.

For tech and tech-enabled startups there are several steps that are usually completed at this stage:

- 1. Problem research - it includes customer segments, pain points for each customer segment and value proposition;
- 2. Articulating the solution - that includes the product channels, the marketing channels, and the revenue streams;
- 3. Validating solution hypothesis with potential users - user research
- 4. PoC development - it can vary from only high-level wireframes to some outcome of the “no code required” platforms.
- 5. User testing
- 6. Clickable prototype using available prototyping tools like **InVision** or a more advanced version of the “no code required” platforms mentioned above.

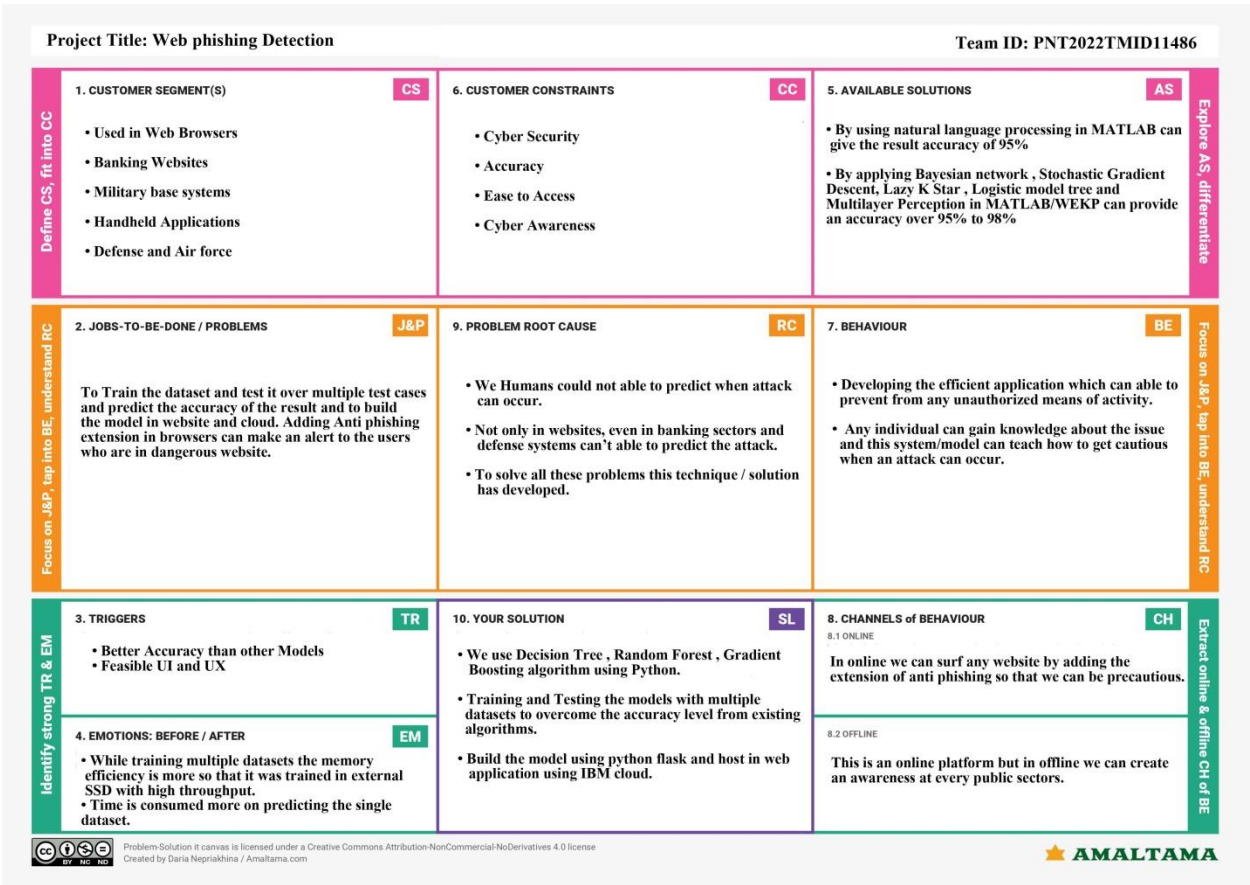


Figure 3.4 Problem Solution Fit

CHAPTER 4

REQUIREMENT ANALYSIS

Solution Requirements (Functional & Non-functional)

Date	13 October 2022
Team ID	PNT2022TMID11486
Project Name	Project - Web Phishing Detection
Maximum Marks	4 Marks

TABLE 4.1 FUNCTIONAL REQUIREMENT

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	- Registration through Form Registration through Gmail - Registration through LinkedIn
FR-2	User Confirmation	- Confirmation via Email - Confirmation via OTP
FR-3	User Authentication	Confirmation of Google Firebase
FR-4	User Security	Strong Passwords , 2FA and FIDO2.0 Webauthn
FR-5	User Performance	Usage of Legitimate websites, Optimize Network Traffic

TABLE 4.2 NON-FUNCTIONAL REQUIREMENTS

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Responsive UI / UX Design and users can easily configure the settings based on their preference.
NFR-2	Security	Implementation of Updated security algorithms and techniques.
NFR-3	Reliability	Reliability Factor determines the possibility of a suspected site to be Valid or Fake.
NFR-4	Performance	The two main characteristics of a phishing site are that it looks extremely similar to a legitimate site and that it has at least one field to enable users to input their credentials.
NFR-5	Availability	It occurs when an attacker, masquerading as a trusted entity, dupes a victim into opening an email, instant message, or text message.
NFR-6	Scalability	Scalable detection and isolation of phishing, the main ideas are to move the protection from end users towards the network provider and to employ the novel bad neighbourhood concept, in order to detect and isolate both phishing e mail senders and phishing web servers.

CHAPTER 5

PROJECT DESIGN

Date	13 October 2022
Team ID	PNT2022TMID11486
Project Name	Project – Web Phishing Detection
Maximum Marks	4 Marks

5.1 DATA FLOW DIAGRAMS

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

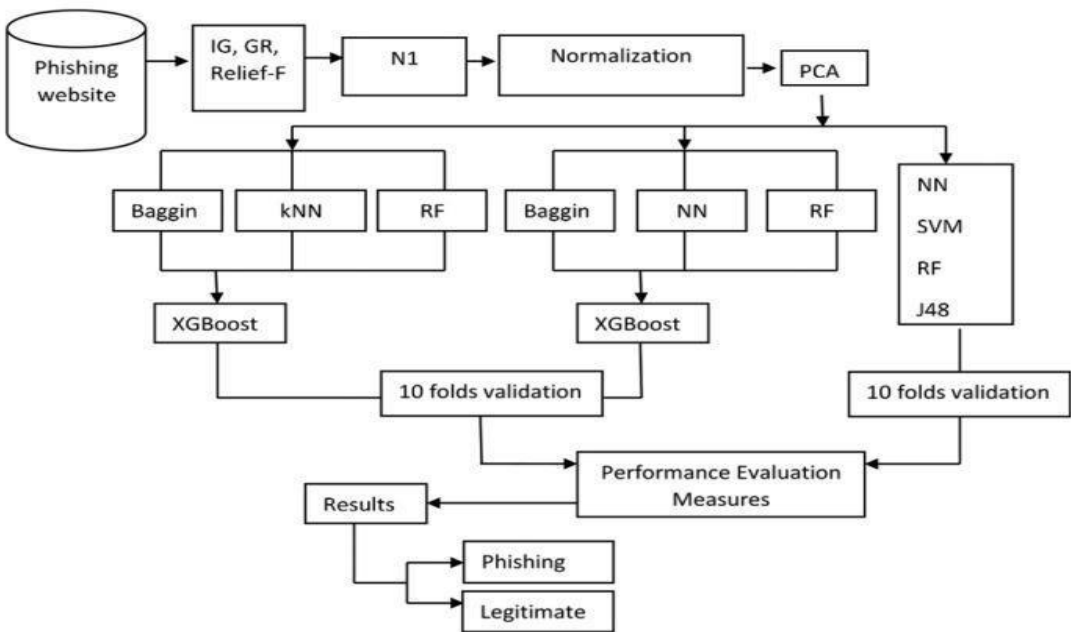


Figure 5.1 Data Flow Diagrams

5.2 SOLUTION & TECHNICAL ARCHITECTURE

Project Design Phase-II Technology Stack (Architecture & Stack)

Date	13 October 2022
Team ID	PNT2022TMID11486
Project Name	Project - Web Phishing Detection
Maximum Marks	4 Marks

Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table 5.2.1 & table 5.2.2

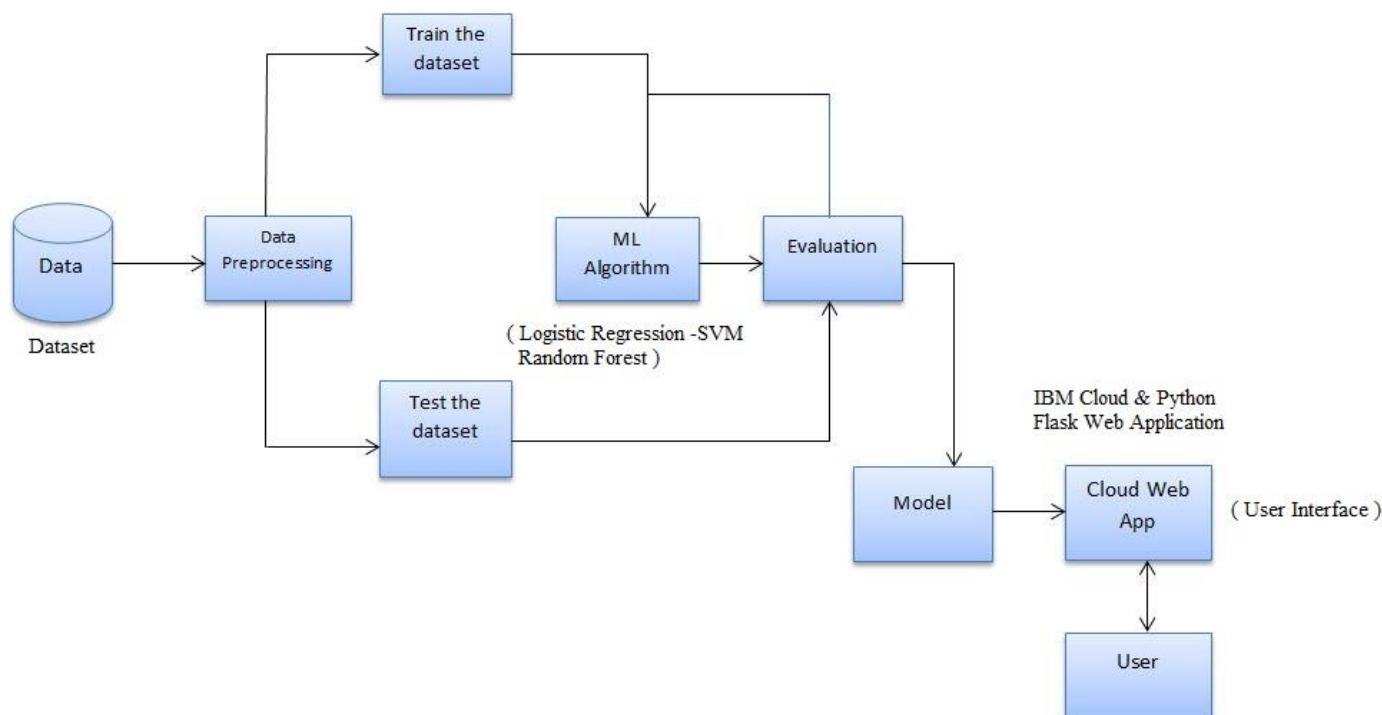


Table-1.; Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	Web Application, Cloud UI	HTML, CSS, JavaScript / Angular Js / React Js etc.
2.	Application Logic-1	Machine Learning Algorithms such as Gradient Boost, Random forest, Decision Tree, Logistic Regression and SVM. Python Flask Application for Web App	Java / Python
3.	Application Logic-2	IBM Watson Speech to Text technology enables fast and accurate speech transcription in multiple languages for a variety of use cases, including but not limited to customer self-service, agent assistance and speech analytics.	IBM Watson STT service
4.	Application Logic-3	The IBM Watson Assistant service combines machine learning, natural language understanding, and an integrated dialog editor to create conversation flows between your apps and your users.	IBM Watson Assistant
5.	Database	Stored Procedure (EXEC)	MySQL, NoSQL, etc.
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Gophish is a powerful, open-source phishing framework that makes it easy to test your organization's exposure to phishing.	Machine Learning
2.	Security Implementations	In our prototype we use encryption techniques and security algorithms on web application	AES 256 , Cofense PDR
3.	Scalable Architecture	Scalability is high due to accuracy provided by the model and Responsive UI/UX	React Framework, jQuery, Bootstrap, Cloudflare
4.	Availability	Available at NLP, Spam Detection ,Blacklisting or Reporting, and machine learning techniques	Acunetix, Intruder, Ghost Phisher
5.	Performance	Deployed and Tested with multiple algorithms and this system gives greater accuracy and better performance than other.	Deep Learning

5.3 USER STORIES

User stories are one of the core components of an agile program. They help provide a user-focused framework for daily work — which drives collaboration, creativity, and a better product overall.

User stories are written by or for users or customers to influence the functionality of the system being developed. In some teams, the product manager (or product owner in Scrum), is primarily responsible for formulating user stories and organizing them into a product backlog. In other teams, anyone can write a user story. User stories can be developed through discussion with stakeholders, based on personas or are simply made up.

A user story is the smallest unit of work in an agile framework. It’s an end goal, not a feature, expressed from the software user’s perspective. In software development and product management, a user story is an **informal, natural language description of features of a software system**.

They are written from the perspective of an end user or user of a system, and may be recorded on index cards, Post-it notes, or digitally in project management software.

Table 5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register my personal details only in official websites.	I can access my account / dashboard	Medium	Sprint-1
		USN-2	As a user, I should create strong passwords.	I can access my account securely	High	Sprint-1
		USN-3	As a user, I can register in websites which doesn't navigate me to any other websites.	I can store the data in legitimate website	Low	Sprint-2
	Login	USN-4	As a user, I can login into required websites.	I can access my account	Low	Sprint-1
Customer (Mobile user)	Registration	USN-5	As a user, I can register with verification code.	Authorized Login	High	Sprint-1
		USN-6	As a user, I should not register at unknown or random calls.	I can be prevented from Cyber Attacks	Medium	Sprint-1
		USN-7	As a user, I should not register in other devices.	I can access in my authorized device.	Low	Sprint-2
Administrator		USN-8	Admin should maintain his/her database securely.	Prevented from Phishing Attacks	High	Sprint-2
Customer Care		USN-9	As a user, If my account is Phished or Attacked.	I can report / Complain	High	Sprint-1
		USN-10	As a user, I should not take others information	I can be punished for it.	Medium	Sprint-1

CHAPTER 6

PROJECT PLANNING & SCHEDULING

Project Planning (Product Backlog, Sprint Planning, Stories, Story points)

Date	18 October 2022
Team ID	PNT2022TMID11486
Project Name	Project – Web Phishing Detection
Maximum Marks	8 Marks

TABLE 6.1 SPRINT PLANNING & ESTIMATION

Product Backlog, Sprint Schedule and Estimation (4 Marks)

Product backlog and sprint schedule:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User input	USN-1	User inputs an URL in the required field to check its validation.	5	Medium	Abisheik R
Sprint-1	Website Comparison	USN-2	Model compares the websites using Blacklist and Whitelist approach.	10	High	Aswin T S
Sprint-1	Storage	USN-3	Storing the Blacklisted websites in Database using IBM Cloud.	15	High	Balajee A V
Sprint-2	Feature Extraction	USN-4	After comparison, if none found on comparison then it extract feature using heuristic and visual similarity.	10	High	Aswin TS
Sprint-2	Prediction	USN-5	Model predicts the URL using Machine learning algorithms such as logistic Regression, MLP.	10	Medium	Balaji S
Sprint-2	Accuracy Test	USN-6	Selecting the best accurate model and to process further steps.	15	High	Balajee A V
Sprint-3	Classifier	USN-7	Model sends all the output to the classifier and produces the final result.	5	Medium	Abisheik R
Sprint-3	Hosting	USN-8	Setting Up the Application and hosting in IBM cloud	10	Medium	Balaji S
Sprint-4	Announcement	USN-9	Model then displays whether the website is legal site or a phishing site.	15	High	Balajee A V
Sprint-4	Events	USN-10	This model needs the capability of retrieving and displaying accurate result for a website.	10	High	Aswin T S

TABLE 6.2 SPRINT DELIVERY SCHEDULE

Project Tracker, Velocity & Burndown Chart (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	12 Nov 2022

Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let’s calculate the team’s average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

We have a 6-day sprint duration, and the velocity of the team is 20 (points per sprint). So our team’s average velocity (AV) per iteration unit (story points per day)

$$AV = (Sprint\ Duration / Velocity) = 20 / 6 = 3.33$$

Burndown Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.

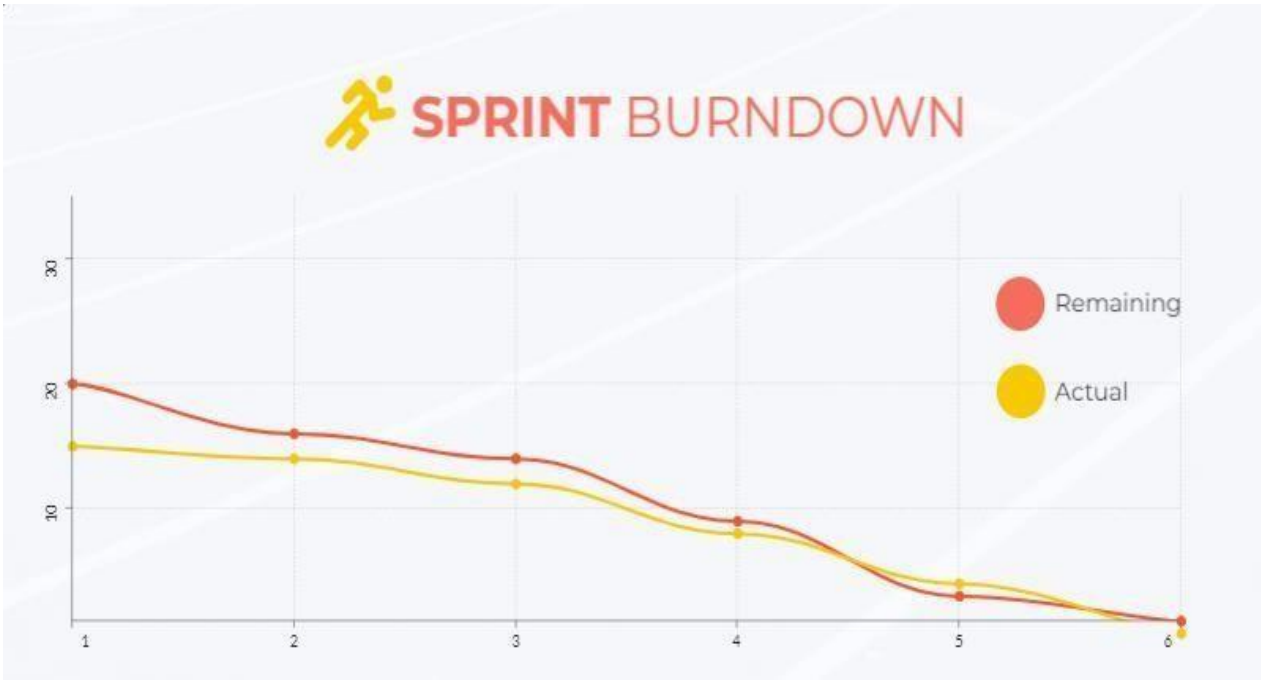


Figure 6.2 Burndown Chart

6.3 REPORTS FROM JIRA

Figure 6.3 JIRA report

	OCT							NOV							NOV							NOV						
	24	25	26	27	28	29	30	31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
Sprints	WPD Sprint 1							WPD Sprint 2							WPD Sprint 3							WPD Sprint 4						
WPD-8 User Input																												
WPD-9 Website Comparison																												
WPD-10 Feature Extraction																												
WPD-11 Prediction																												
WPD-12 Classifier																												
WPD-13 Announcement																												
WPD-14 Events																												

CHAPTER 7

CODING & SOLUTIONING

7.1 FEAUTRE 1

DATA PREPROCESSING & DATA VISUALIZATION

Data preprocessing is an **iterative process for the transformation of the raw data into understandable and useable forms**. Raw datasets are usually characterized by incompleteness, inconsistencies, lacking in behavior, and trends while containing errors . The preprocessing is essential to handle the missing values and address inconsistencies.

Importing Libraries & Dataset

In [12]:

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
```

In [14]:

```
data=pd.read_csv("D:/Collection Of Dataset/dataset_website.csv")
```

In [15]:

```
data
```

Out[15]:

	index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain	SSLfinal_State	Domains
	0	1	-1	1	1	1	-1	-1	-1	-1
	1	2	1	1	1	1	1	-1	0	1
	2	3	1	0	1	1	1	-1	-1	-1
	3	4	1	0	1	1	1	-1	-1	-1
	4	5	1	0	-1	1	1	-1	1	1

	11050	11051	1	-1	1	-1	1	1	1	1
	11051	11052	-1	1	1	-1	-1	-1	1	-1
	11052	11053	1	-1	1	1	1	-1	1	-1
	11053	11054	-1	-1	1	1	1	-1	-1	-1
	11054	11055	-1	-1	1	1	1	-1	-1	-1

11055 rows x 32 columns

In [16]:

```
data.head()
```

Out[16]:

	index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain	SSLfinal_State	Domain_re
	0	1	-1	1	1	1	-1	-1	-1	-1
	1	2	1	1	1	1	1	-1	0	1
	2	3	1	0	1	1	1	-1	-1	-1
	3	4	1	0	1	1	1	-1	-1	-1
	4	5	1	0	-1	1	1	-1	1	1

5 rows x 32 columns

Numerical Analysis

In [17]:

```
data.shape
```

Out[17]:

```
(11055, 32)
```

In [18]:

```
data.size
```

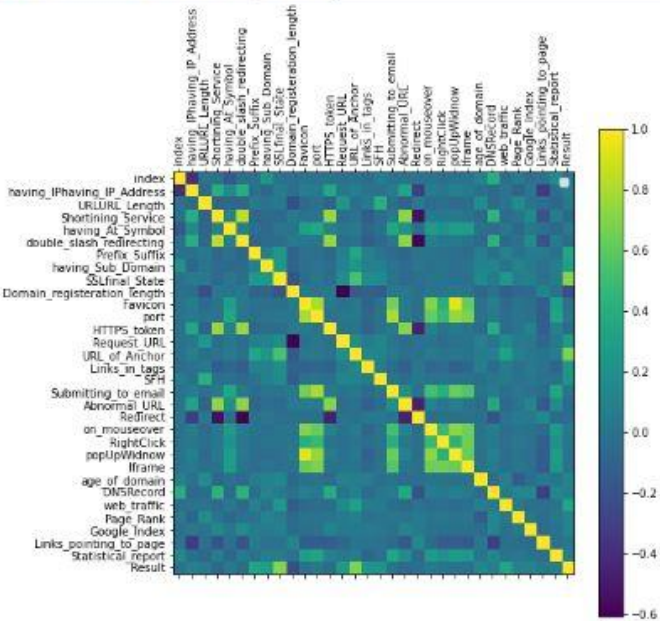
Out[18]:

```
353760
```

Data Visualization

```
In [25]: def plot_corr(df,size=8):
corr=df.corr()
fig,ax=plt.subplots(figsize=(size,size))
ax.legend()
cax=ax.matshow(corr)
fig.colorbar(cax)
plt.xticks(range(len(corr.columns)), corr.columns, rotation='vertical')
plt.yticks(range(len(corr.columns)), corr.columns)
plot_corr(data)
```

No handles with labels found to put in legend.



- 1. There are 11054 instances and 31 features in dataset.
- 2. Out of which 30 are independent features where as 1 is dependent feature.
- 3. Each feature is in int datatype, so there is no need to use LabelEncoder.
- 4. There is no outlier present in dataset.
- 5. There is no missing value in dataset

From below image we can infer that in the dataset contains 5000+ Phishing Websites and 6000+ Legitimate Website Features.



Splitting the data

```
In [27]: x=data.iloc[:,1:31].values
        y=data.iloc[:,0].values

In [28]: x

Out[28]: array([[ -1,  1,  1, ...,  1,  1, -1],
               [  1,  1,  1, ...,  1,  1,  1],
               [  1,  0,  1, ...,  1,  0, -1],
               ...,
               [  1, -1,  1, ...,  1,  0,  1],
               [-1, -1,  1, ...,  1,  1,  1],
               [-1, -1,  1, ..., -1,  1, -1]], dtype=int64)

In [29]: y

Out[29]: array([-1, -1, -1, ..., -1, -1, -1], dtype=int64)
```

Train, Test & Split

```
In [30]: from sklearn.model_selection import train_test_split
        x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)

In [31]: x_train.shape

Out[31]: (8844, 30)

In [32]: y_train.shape

Out[32]: (8844,)

In [33]: x_test.shape

Out[33]: (2211, 30)

In [34]: y_test.shape

Out[34]: (2211,)
```

SELECTING APPROPRIATE MODEL

Model selection is a key step in every data science project and requires perhaps the most conceptual foundational knowledge.

We’d reviewed a number of supervised machine learning models in class like Logistic Regression, K-Nearest Neighbors, Naive Bayes, Random Forest, and Gradient Boost.

Here we used to choose Gradient Boosting Algorithm for predicting best accuracy than other models.

Model Building & Training:

```
In [14]: # Creating holders to store the model performance results
        ML_Model = []
        accuracy = []
        f1_score = []
        recall = []
        precision = []

        #function to call for storing the results
        def storeResults(model, a,b,c,d):
            ML_Model.append(model)
            accuracy.append(round(a, 3))
            f1_score.append(round(b, 3))
            recall.append(round(c, 3))
            precision.append(round(d, 3))
```

Gradient Boosting Classifier

```
In [49]: # Gradient Boosting Classifier Model
from sklearn.ensemble import GradientBoostingClassifier

# Instantiate the model
gbc = GradientBoostingClassifier(max_depth=4, learning_rate=0.7)

# fit the model
gbc.fit(X_train, y_train)

Out[49]: GradientBoostingClassifier(learning_rate=0.7, max_depth=4)
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [50]: #predicting the target value from the model for the samples
y_train_gbc = gbc.predict(X_train)
y_test_gbc = gbc.predict(X_test)

In [51]: #computing the accuracy, f1_score, Recall, precision of the model performance

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)
print("Gradient Boosting Classifier : f1_score on training Data: {:.3f}".format(f1_score_train_gbc))
print("Gradient Boosting Classifier : f1_score on test Data: {:.3f}".format(f1_score_test_gbc))
print()

recall_score_train_gbc = metrics.recall_score(y_train, y_train_gbc)
recall_score_test_gbc = metrics.recall_score(y_test, y_test_gbc)
print("Gradient Boosting Classifier : Recall on training Data: {:.3f}".format(recall_score_train_gbc))
print("Gradient Boosting Classifier : Recall on test Data: {:.3f}".format(recall_score_test_gbc))
print()

precision_score_train_gbc = metrics.precision_score(y_train, y_train_gbc)
precision_score_test_gbc = metrics.precision_score(y_test, y_test_gbc)
print("Gradient Boosting Classifier : precision on training Data: {:.3f}".format(precision_score_train_gbc))
print("Gradient Boosting Classifier : precision on test Data: {:.3f}".format(precision_score_test_gbc))
```

```
Gradient Boosting Classifier : Accuracy on training Data: 0.989
Gradient Boosting Classifier : Accuracy on test Data: 0.974

Gradient Boosting Classifier : f1_score on training Data: 0.990
Gradient Boosting Classifier : f1_score on test Data: 0.977

Gradient Boosting Classifier : Recall on training Data: 0.994
Gradient Boosting Classifier : Recall on test Data: 0.989

Gradient Boosting Classifier : precision on training Data: 0.986
Gradient Boosting Classifier : precision on test Data: 0.966
```

CLASSIFICATION REPORT OF THE MODEL:

It is one of the performance evaluation metrics of a classification-based machine learning model. It displays your model’s precision, recall, F1 score and support. It provides a better understanding of the overall performance of our trained model.

Precision - Precision is defined as the ratio of true positives to the sum of true and false positives.

Recall - Recall is defined as the ratio of true positives to the sum of true positives and false negatives.

F1 Score - The F1 is the weighted harmonic mean of precision and recall. The closer the value of the F1 score is to 1.0, the better the expected performance of the model is.

Support - Support is the number of actual occurrences of the class in the dataset. It doesn't vary between models, it just diagnoses the performance evaluation process.

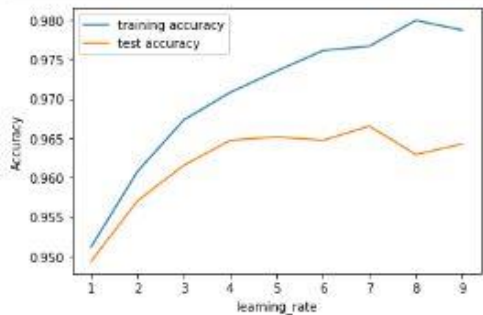
```
In [52]: #computing the classification report of the model
print(metrics.classification_report(y_test, y_test_gbc))
```

	precision	recall	f1-score	support
-1	0.99	0.96	0.97	976
1	0.97	0.99	0.98	1235
accuracy			0.97	2211
macro avg	0.98	0.97	0.97	2211
weighted avg	0.97	0.97	0.97	2211

```
In [53]: training_accuracy = []
test_accuracy = []
# try learning_rate from 0.1 to 0.9
depth = range(1,10)
for n in depth:
    forest_test = GradientBoostingClassifier(learning_rate = n*0.1)

    forest_test.fit(X_train, y_train)
    # record training set accuracy
    training_accuracy.append(forest_test.score(X_train, y_train))
    # record generalization accuracy
    test_accuracy.append(forest_test.score(X_test, y_test))

#plotting the training & testing accuracy for n_estimators from 1 to 50
plt.figure(figsize=None)
plt.plot(depth, training_accuracy, label="training accuracy")
plt.plot(depth, test_accuracy, label="test accuracy")
plt.ylabel("Accuracy")
plt.xlabel("learning_rate")
plt.legend();
```



```
In [82]: #Sorting the dataframe on accuracy
sorted_result=result.sort_values(by=['Accuracy', 'f1_score'],ascending=False).reset_index(drop=True)

In [83]: # dispalying total result
sorted_result
```

Out[83]:

	ML Model	Accuracy	f1 score	Recall	Precision
0	Gradient Boosting Classifier	0.974	0.977	0.994	0.986
1	CatBoost Classifier	0.972	0.975	0.994	0.989
2	Random Forest	0.969	0.972	0.992	0.991
3	Support Vector Machine	0.964	0.968	0.980	0.965
4	Decision Tree	0.958	0.962	0.991	0.993
5	K-Nearest Neighbors	0.956	0.961	0.991	0.989
6	Logistic Regression	0.934	0.941	0.943	0.927
7	Naive Bayes Classifier	0.605	0.454	0.292	0.997
8	XGBoost Classifier	0.548	0.548	0.993	0.984
9	Multi-layer Perceptron	0.543	0.543	0.989	0.983

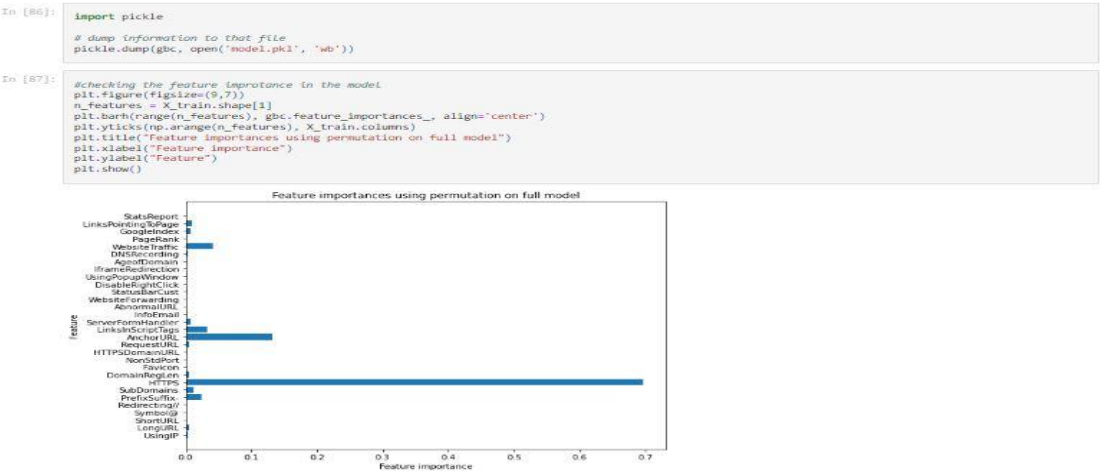
Storing Best Model

```
In [84]: # XGBoost Classifier Model
from xgboost import XGBClassifier

# instantiate the model
gbc = GradientBoostingClassifier(max_depth=4,learning_rate=0.7)

# fit the model
gbc.fit(X_train,y_train)
```

Out[84]: GradientBoostingClassifier(learning_rate=0.7, max_depth=4)
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.



7. Conclusion

Gradient Boosting Classifier correctly classify URL upto 97.4% respective classes and hence reduces the chance of malicious attachments. So we choose this as appropriate model

7.2 FEAUTRE 2

BUILDING FLASK APP

```
import ipaddress
import re
import urllib.request
from bs4 import BeautifulSoup
import socket
import requests
from googlesearch import search
```

```

import whois
from datetime import date, datetime
import time
from dateutil.parser import parse as date_parse
from urllib.parse import urlparse
class FeatureExtraction:
    features = []
    def __init__(self,url):
        self.features = []
        self.url = url
        self.domain = ""
        self.whois_response = ""
        self.urlparse = ""
        self.response = ""
        self.soup = ""
        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.longUrl())
        self.features.append(self.shortUrl())
        self.features.append(self.symbol())
        self.features.append(self.redirecting())
        self.features.append(self.prefixSuffix())
        self.features.append(self.SubDomains())
        self.features.append(self.Hppts())
        self.features.append(self.DomainRegLen())
        self.features.append(self.Favicon())

        self.features.append(self.NonStdPort())
        self.features.append(self.HTTPSDomainURL())
        self.features.append(self.RequestURL())
        self.features.append(self.AnchorURL())
        self.features.append(self.LinksInScriptTags())
        self.features.append(self.ServerFormHandler())
        self.features.append(self.InfoEmail())
        self.features.append(self.AbnormalURL())
        self.features.append(self.WebsiteForwarding())
        self.features.append(self.StatusBarCust())

```

```

self.features.append(self.DisableRightClick())
self.features.append(self.UsingPopupWindow())
self.features.append(self.IframeRedirection())
self.features.append(self.AgeofDomain())
self.features.append(self.DNSRecording())
self.features.append(self.WebsiteTraffic())
self.features.append(self.PageRank())
self.features.append(self.GoogleIndex())
self.features.append(self.LinksPointingToPage())
self.features.append(self.StatsReport())
# 1.UsingIp
def UsingIp(self):
    try:
        ipaddress.ip_address(self.url)
        return -1
    except:
        return 1
# 2.longUrl
def longUrl(self):
    if len(self.url) < 54:
        return 1
    if len(self.url) >= 54 and len(self.url) <= 75:
        return 0
    return -1
# 3.shortUrl
def shortUrl(self):
    match =
re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|'
'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|'
'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|'
'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|'
'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|'
'q\.gs|is\.gd|po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|'
'x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.me|v\.gd|tr\.im|link\.zip\.net', self.url)
    if match:
        return -1
    return 1
# 4.Symbol@
def symbol(self):
    if re.findall("@",self.url):
        return -1

```



```

    return 1

# 5.Redirecting//
def redirecting(self):
    if self.url.rfind('/')>6:
        return -1
    return 1

# 6.prefixSuffix
def prefixSuffix(self):
    try:
        match = re.findall('-', self.domain)
        if match:
            return -1
        return 1
    except:
        return -1

# 7.SubDomains
def SubDomains(self):
    dot_count = len(re.findall(".", self.url))
    if dot_count == 1:
        return 1
    elif dot_count == 2:
        return 0
    return -1
# 8.HTTPS
def Hppts(self):
    try:
        https = self.urlparse.scheme
        if 'https' in https:
            return 1
        return -1
    except:
        return 1
# 9.DomainRegLen
def DomainRegLen(self):
    try:
        expiration_date = self.whois_response.expiration_date
        creation_date = self.whois_response.creation_date
        try:
            if(len(expiration_date)):
                expiration_date = expiration_date[0]
        except:
            Pass
        try:
            if(len(creation_date)):
                creation_date = creation_date[0]
        except:
            Pass
        age = (expiration_date.year-creation_date.year)*12+ (expiration_date.month-

```

```

creation_date.month)
    if age >=12:
        return 1
    return -1
except:
    return -1
# 10. Favicon
def Favicon(self):
    try:
        for head in self.soup.find_all('head'):
            for head.link in self.soup.find_all('link', href=True):
                dots = [x.start(0) for x in re.finditer('\.', head.link['href'])]
                if self.url in head.link['href'] or len(dots) == 1 or domain in
head.link['href']:
                    return 1
            return -1
        except:
            return -1
# 11. NonStdPort
def NonStdPort(self):
    try:
        port = self.domain.split(":")
        if len(port)>1:
            return -1
        return 1
    except:
        return -1
# 12. HTTPSDomainURL
def HTTPSDomainURL(self):
    try:
        if 'https' in self.domain:
            return -1
        return 1
    except:
        return -1

# 13. RequestURL
def RequestURL(self):
    try:
        for img in self.soup.find_all('img', src=True):
            dots = [x.start(0) for x in re.finditer('\.', img['src'])]
            if self.url in img['src'] or self.domain in img['src'] or len(dots) == 1:
                success = success + 1
            i = i+1
        for audio in self.soup.find_all('audio', src=True):
            dots = [x.start(0) for x in re.finditer('\.', audio['src'])]
            if self.url in audio['src'] or self.domain in audio['src'] or len(dots) == 1:
                success = success + 1
            i = i+1
        for embed in self.soup.find_all('embed', src=True):
            dots = [x.start(0) for x in re.finditer('\.', embed['src'])]

```



```

        if self.url in embed['src'] or self.domain in embed['src'] or len(dots) == 1:
            success = success + 1
        i = i+1
    for iframe in self.soup.find_all('iframe', src=True):
        dots = [x.start(0) for x in re.finditer('\.', iframe['src'])]
        if self.url in iframe['src'] or self.domain in iframe['src'] or len(dots) == 1:
            success = success + 1
        i = i+1
    try:
        percentage = success/float(i) * 100
        if percentage < 22.0:
            return 1
        elif((percentage >= 22.0) and (percentage < 61.0)):
            return 0
        else:
            return -1
    except:
        return 0
except:
    return -1

```

14. AnchorURL

```
def AnchorURL(self):
```

```

    try:
        i,unsafe = 0,0
        for a in self.soup.find_all('a', href=True):
            if "#" in a['href'] or "javascript" in a['href'].lower() or "mailto" in
a['href'].lower() or not (url in a['href'] or self.domain in a['href']):
                unsafe = unsafe + 1
            i = i + 1
        try:
            percentage = unsafe / float(i) * 100
            if percentage < 31.0:
                return 1
            elif ((percentage >= 31.0) and (percentage < 67.0)):
                return 0
            else:
                return -1
        except:
            return -1
    except:
        return -1

```

15. LinksInScriptTags

```
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
# 16. ServerFormHandler
def ServerFormHandler(self):
    try:
        if len(self.soup.find_all('form', action=True))==0:
            return 1
        else :
            for form in self.soup.find_all('form', action=True):
                if form['action'] == "" or form['action'] == "about:blank":
                    return -1
                elif self.url not in form['action'] and self.domain not in form['action']:
                    return 0
                else:
                    return 1
    except:
        return -1
# 17. InfoEmail
def InfoEmail(self):
    try:
        if re.findall(r"[mail\(\)|mailto:?}", self.soap):
            return -1
        else:
            return 1
    except:
        return -1
# 18. AbnormalURL
def AbnormalURL(self):
    try:
        if self.response.text == self.whois_response:
            return 1
        else:
            return -1
    except:
        return -1

```

```

# 19. WebsiteForwarding
def WebsiteForwarding(self):
    try:
        if len(self.response.history) <= 1:
            return 1
        elif len(self.response.history) <= 4:
            return 0
        else:
            return -1
    except:
        return -1
# 20. StatusBarCust
def StatusBarCust(self):
    try:
        if re.findall("<script>.+onmouseover.+</script>", self.response.text):
            return 1
        else:
            return -1
    except:
        return -1
# 21. DisableRightClick
def DisableRightClick(self):
    try:
        if re.findall(r"event.button ?== ?2", self.response.text):
            return 1
        else:
            return -1
    except:
        return -1
# 22. UsingPopupWindow
def UsingPopupWindow(self):
    try:
        if re.findall(r"alert\(", self.response.text):
            return 1
        else:
            return -1
    except:
        return -1
# 23. IframeRedirection
def IframeRedirection(self):
    try:
        if re.findall(r"<iframe>|<frameBorder>]", self.response.text):
            return 1
        else:
            return -1
    except:
        return -1
# 24. AgeofDomain
def AgeofDomain(self):
    try:
        creation_date = self.whois_response.creation_date

```

```

try:
    if(len(creation_date)):
        creation_date = creation_date[0]
except:
    pass
today = date.today()
age = (today.year-creation_date.year)*12+(today.month-
creation_date.month)
if age >=6:
    return 1
return -1
except:
    return -1
# 25. DNSRecording
def DNSRecording(self):
    try:
        creation_date = self.whois_response.creation_date
    try:
        if(len(creation_date)):
            creation_date = creation_date[0]
    except:
        Pass
    today = date.today()
    age = (today.year-creation_date.year)*12+(today.month-
creation_date.month)
    if age >=6:
        return 1
    return -1
    except:
        return -1
# 26. WebsiteTraffic
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
# 27. PageRank
def PageRank(self):
    try:
        prank_checker_response =
requests.post("https://www.checkpagerank.net/index.php", {"name": self.domain})
        global_rank = int(re.findall(r"Global Rank: ([0-9]+)",
rank_checker_response.text)[0])
        if global_rank > 0 and global_rank < 100000:
            return 1
        return -1

```

```
except:
    return -1
```

```
# 28. GoogleIndex
```

```
def GoogleIndex(self):
```

```
    try:
        site = search(self.url, 5)
        if site:
            return 1
        else:
            return -1
```

```
    except:
        return 1
```

```
# 29. LinksPointingToPage
```

```
def LinksPointingToPage(self):
```

```
    try:
        number_of_links = len(re.findall(r"<a href=", self.response.text))
        if number_of_links == 0:
            return 1
        elif number_of_links <= 2:
            return 0
        else:
            return -1
```

```
    except:
        return -1
```

```
# 30. StatsReport
```

```
def StatsReport(self):
```

```
    try:
        url_match = re.search(
```

```
'at\.ua|usa\.cc|baltazarpresentes\.com\.br|pe\.hu|esy\.es|hol\.es|sweddy\.com|myjino\.ru|96\.lt|ow\.ly', url)
```

```
        ip_address = socket.gethostbyname(self.domain)
```

```
        ip_match =
```

```
re.search('146\.112\.61\.108|213\.174\.157\.151|121\.50\.168\.88|192\.185\.217\.116|78\.46\.211\.158|181\.174\.165\.13|46\.242\.145\.103|121\.50\.168\.40|83\.125\.22\.219|46\.242\.145\.98|'
```

```
'107\.151\.148\.44|107\.151\.148\.107|64\.70\.19\.203|199\.184\.144\.27|107\.151\.148\.108|107\.151\.148\.109|119\.28\.52\.61|54\.83\.43\.69|52\.69\.166\.231|216\.58\.192\.225|'
```

```
'118\.184\.25\.86|67\.208\.74\.71|23\.253\.126\.58|104\.239\.157\.210|175\.126\.123\.219|141\.8\.224\.221|10\.10\.10\.10|43\.229\.108\.32|103\.232\.215\.140|69\.172\.201\.153|'
```

```
'216\.218\.185\.162|54\.225\.104\.146|103\.243\.24\.98|199\.59\.243\.120|31\.170\.160\.61|213\.19\.128\.77|62\.113\.226\.131|208\.100\.26\.234|195\.16\.127\.102|195\.16\.127\.157|'
```

```
'34\.196\.13\.28|103\.224\.212\.222|172\.217\.4\.225|54\.72\.9\.51|192\.64\.147\.141|'
```

```
198\200\56\183|23\253\164\103|52\48\191\26|52\214\197\72|87\98\255\18|
209\99\17\27|'
```

```
'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
```

TESTING THE FLASK APP WITH LOCAL ENVIRONMENT

```
from flask import Flask, request, render_template
import numpy as np
import pandas as pd
from sklearn import metrics
import warnings
import pickle
warnings.filterwarnings('ignore')
from feature import FeatureExtraction
file = open("model.pkl", "rb")
gbc = pickle.load(file)
file.close()
app = Flask(__name__)
@app.route("/", methods=["GET", "POST"])
def index():
    if request.method == "POST":
        url = request.form["url"]
        obj = FeatureExtraction(url)
        x = np.array(obj.getFeaturesList()).reshape(1,30)
        y_pred = gbc.predict(x)[0]
        #1 is safe
        #-1 is unsafe
        y_pro_phishing = gbc.predict_proba(x)[0,0]
        y_pro_non_phishing = gbc.predict_proba(x)[0,1]
        # if(y_pred ==1 ):
        pred = "It is {0:.2f} % safe to go ".format(y_pro_phishing*100)
        return render_template('index.html',xx
        and(y_pro_non_phishing,2),url=url )
        return render_template("index.html", xx =-1)
if __name__ == "__main__":
    app.run(debug=True,port=2002)
```

7.3 STORING THE APP IN IBM CLOUD STORAGE SERVICE

IBM Watson Studio

Search in your workspaces

Buy

Balajee A V's Account

Dallas

B4

Deployments / models / phishing /

newdeployment

Deployed

Online

API reference

Test

https://us-south.ml.cloud.ibm.com/ml/v4/deployments/084b5c52-f617-40ef-a0e8-3e6cf79ae447/predictions?version=2022-11-06

IAM

Code snippets

cURL

Java

JavaScript

Python

Scala

NOTE: you must set \$API_KEY below using information retrieved from your IBM Cloud account.

curl --insecure -X POST --header "Content-Type: application/x-www-form-urlencoded" --header "Accept: application/json" --data-urlencode "grant_type=urn:ibm:params:oauth:grant-type:apikey" --data-urlencode "apikey=\$API_KEY" "https://iam.cloud.ibm.com/identity/token"

the above CURL request will return an auth token that you will use as \$IAM_TOKEN in the scoring request below
TODO: manually define and pass values to be scored below
curl -X POST --header "Content-Type: application/json" --header "Accept: application/json" --header "Authorization: Bearer \$IAM_TOKEN" -d '{"input_data": [{"fields": [\$ARRAY_OF_INPUT_FIELDS], "values": [\$ARRAY_OF_VALUES_TO_BE_SCORED, \$ANOTHER_ARRAY_OF_VALUES_TO_BE_SCORED]}]}' "https://us-south.ml.cloud.ibm.com/ml/v4/deployments/084b5c52-f617-40ef-a0e8-3e6cf79ae447/predictions?version=2022-11-06"

SCORING ENDPOINT IN IBM CLOUD

In [17]:

```
import requests

# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
API_KEY = "cWGD5yTjEpEGtqPpvHPDBEIN5eXFS7eh2JRDyUwhYSMW"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":
    API_KEY, "grant_type": "urn:ibm:params:oauth:grant-type:apikey"})
mltoken = token_response.json()[["access_token"]]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}

# NOTE: manually define and pass the array(s) of values to be scored in the next line
payload_scoring = {"input_data": [{"field": [{"UsingIP", "LongURL", "ShortURL", "Symbol@", "Redirecting//", "PrefixSuffix-", "SubDomains", "HTTPS", "DomainReg
    }, "values": [[1,1,1,1,1,-1,-1,-1,-1,1,1,1,1,-1,-1,1,1,0,1,1,1,-1,-1,-1,-1,-1,1,0,1]]}]]}

response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/084b5c52-f617-40ef-a0e8-3e6cf79ae447/predictions?version=2022-11-06', headers=header, json=payload_scoring)
print("Scoring response")
predictions=response_scoring.json()
#print(predictions)
pred=print(predictions['predictions'][0]['values'][0][0])
if(pred != 1):
    print("The Website is secure.. Continue")
else:
    print("The Website is not Legitimate... BEWARE!!")

Scoring response
-1
The Website is not Legitimate... BEWARE!!
```

PUBLISHING AND TESTING WEBPAGE ON HEROKUAPP


AVBalajee/Phish-shield

URL detection

+

https://phishing-shield.herokuapp.com

IBM Project Based Learning



Detect Phishing URLs using Python

PHISHING URL DETECTION

Website Safe

Enter URL

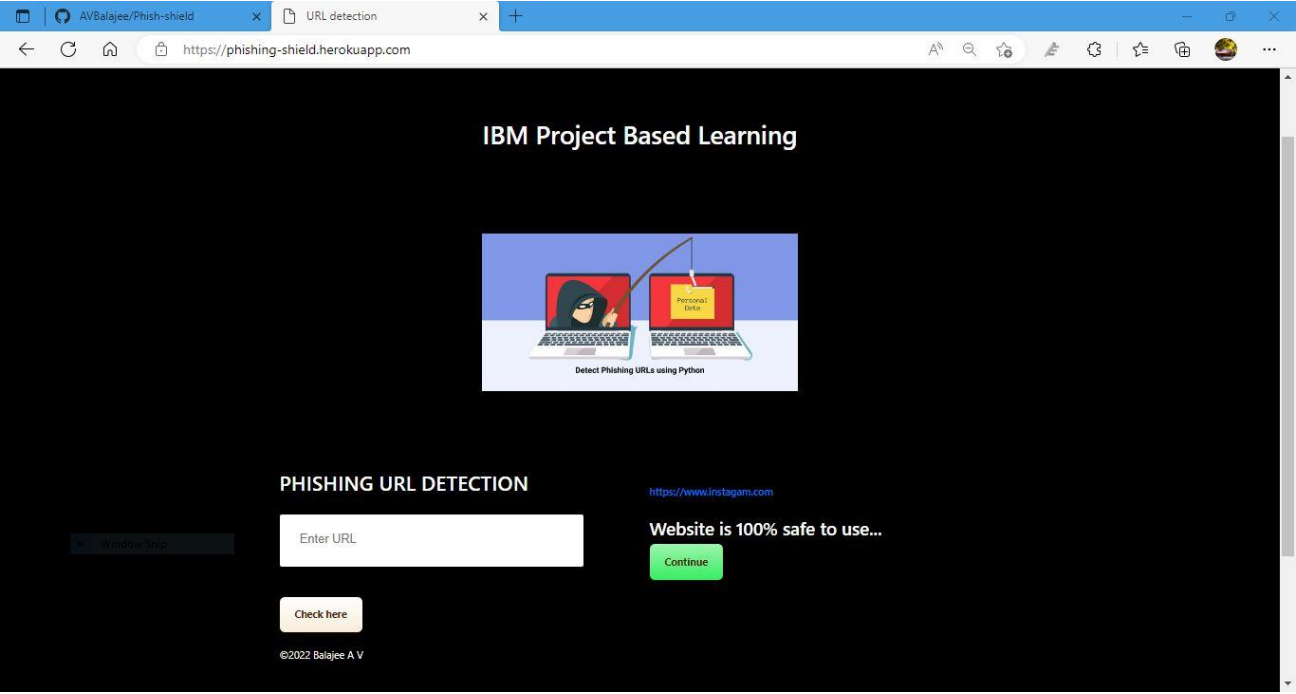
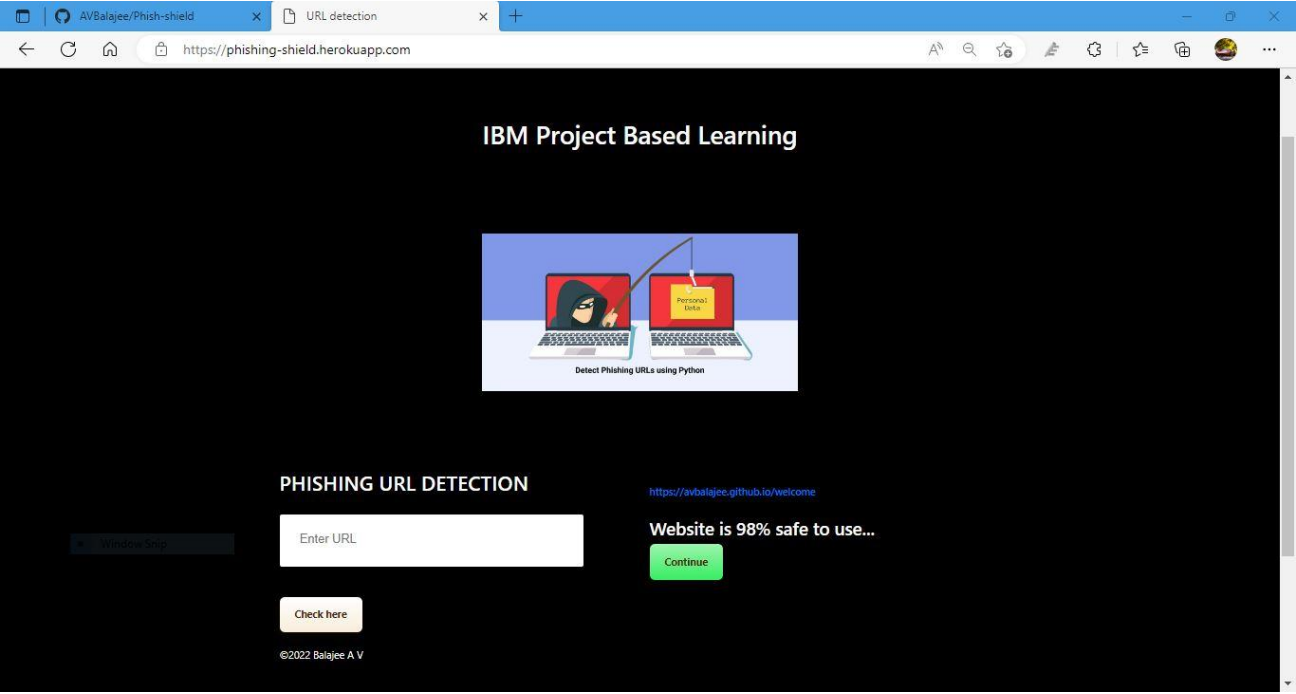
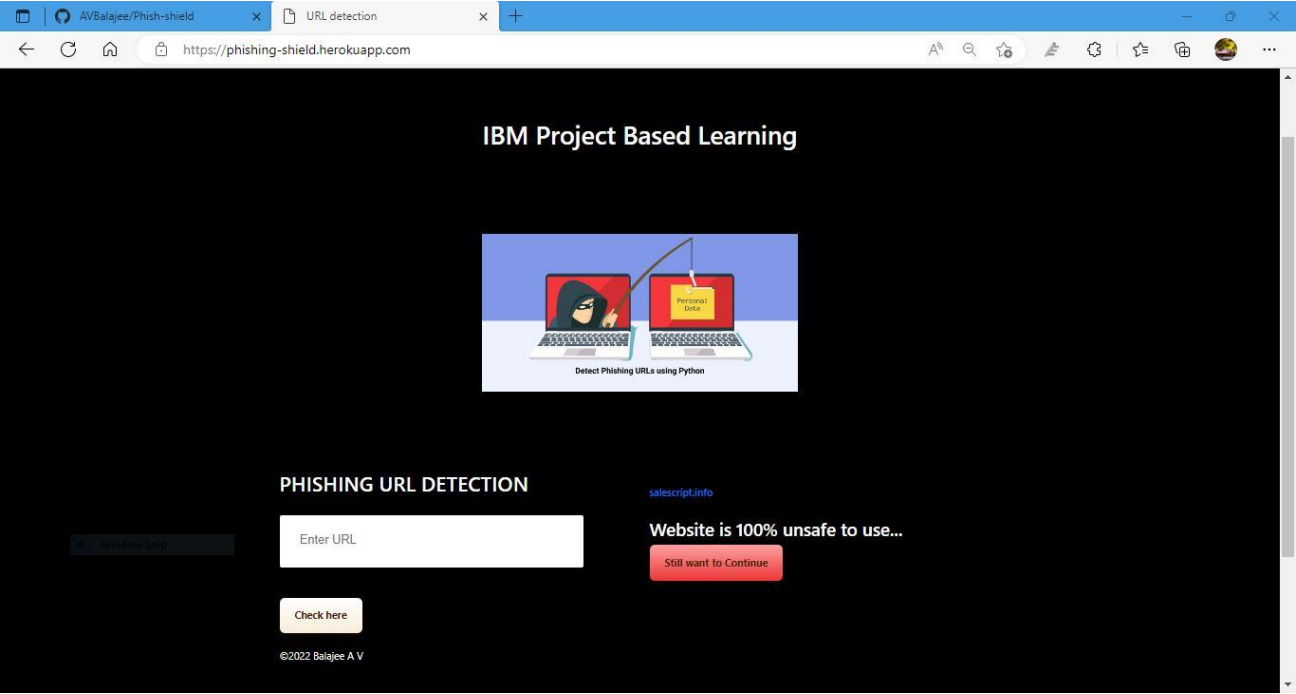
Check here

Website is 100% safe to use...

Continue

©2022 Balajee A V

Figure 7.3 Test URL -1



CHAPTER 8

TESTING

TABLE 8.1 TEST CASES

				Date	15-Nov-22								
				Team ID	PNT2022TMD11486								
				Project Name	Project - Web Phishing Detection								
				Maximum Marks	4 marks								
Test case ID	Feature Type	Component	Test Scenario	Pre-Requsite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
LoginPage_TC_OO 1	Functional	Home Page	Verify user is able to see the Landing Page when user can type the URL in the box		1.Enter URL and click go 2.Type the URL 3.Verify whether it is processing or not.	https://phishing-shield.herokuapp.com/	Should Display the Webpage	Working as expected	Pass		N		S Balaji
LoginPage_TC_OO 2	UI	Home Page	Verify the UI elements is Responsive		1.Enter URL and click go 2. Type or copy paste the URL 3. Check whether the button is responsive or not 4. Reload and Test Simultaneously	https://phishing-shield.herokuapp.com/	Should Wait for Response and then gets Acknowledge	Working as expected	Pass		N		R Abisheik
LoginPage_TC_OO 3	Functional	Home page	Verify whether the link is legitimate or not		1.Enter URL and click go 2. Type or copy paste the URL 3. Check the website is legitimate or not 4. Observe the results	https://phishing-shield.herokuapp.com/	User should observe whether the website is legitimate or not.	Working as expected	Pass		N		T S Aswin
LoginPage_TC_OO 4	Functional	Home Page	Verify user is able to access the legitimate website or not		1.Enter URL and click go 2. Type or copy paste the URL 3. Check the website is legitimate or not 4. Continue if the website is legitimate or be cautious if it is not legitimate.	https://phishing-shield.herokuapp.com/	Application should show that Safe Webpage or Unsafe.	Working as expected	Pass		N		Balajee A V
LoginPage_TC_OO 5	Functional	Home Page	Testing the website with multiple URLs		1.Enter URL (https://phishing-shield.herokuapp.com/) and click go 2. Type or copy paste the URL to test 3. Check the website is legitimate or not 4. Continue if the website is secure or be cautious if it is not secure	1. https://avbalajee.github.io/welcome 2. totalpad.com 3. https://www.kinca.edu 4. salescript.info 5. https://www.google.com/delights.com	User can able to identify the websites whether it is secure or not	Working as expected	Pass		N		Balajee A V

TABLE 8.2 USER ACCEPTANCE TESTING

Acceptance Testing

UAT Execution & Report Submission

Date	15 November 2022
Team ID	PNT2022TMID11486
Project Name	Web Phishing Detection
Maximum Marks	4 Marks

Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the [Web Phishing Detection] project at the time of the release to User Acceptance Testing (UAT).

Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	10	2	4	20	36
Not Reproduced	0	0	1	0	1
Skipped	0	0	0	0	0
Won't Fix	0	0	2	1	3
Totals	23	9	12	25	60

Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	10	0	0	10
Client Application	50	0	0	50
Security	5	0	0	4
Outsource Shipping	3	0	0	3
Exception Reporting	10	0	0	9
Final Report Output	10	0	0	10
Version Control	4	0	0	4

CHAPTER 9

RESULTS

TABLE 9.1 PERFORMANCE METRICS

Project Development Phase

Model Performance Test

Date	13 November 2022
Team ID	PNT2022TMID11486
Project Name	Project – Web Phishing Detection
Maximum Marks	10 Marks

Model Performance Testing:

Project team shall fill the following information in model performance testing template.

S.N o.	Parameter	Values	Screenshot																														
1.	Metrics	<div>Classification Model:</div> <div>Gradient Boosting Classification</div> <div>Accuray Score- 97.4%</div>	<div><pre>In [52]: #computing the classification report of the model print(metrics.classification_report(y_test, y_test_gbc))</pre><table><thead><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr></thead><tbody><tr><td>-1</td><td>0.99</td><td>0.96</td><td>0.97</td><td>976</td></tr><tr><td>1</td><td>0.97</td><td>0.99</td><td>0.98</td><td>1235</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.97</td><td>2211</td></tr><tr><td>macro avg</td><td>0.98</td><td>0.97</td><td>0.97</td><td>2211</td></tr><tr><td>weighted avg</td><td>0.97</td><td>0.97</td><td>0.97</td><td>2211</td></tr></tbody></table></div>		precision	recall	f1-score	support	-1	0.99	0.96	0.97	976	1	0.97	0.99	0.98	1235	accuracy			0.97	2211	macro avg	0.98	0.97	0.97	2211	weighted avg	0.97	0.97	0.97	2211
	precision	recall	f1-score	support																													
-1	0.99	0.96	0.97	976																													
1	0.97	0.99	0.98	1235																													
accuracy			0.97	2211																													
macro avg	0.98	0.97	0.97	2211																													
weighted avg	0.97	0.97	0.97	2211																													
2.	Tune the Model	<div>Hyperparameter Tuning - 97%</div> <div>Validation Method – KFOLD & Cross Validation Method</div>	<div><div>Wilcoxon signed-rank test</div><pre>In [78]: #KFOLD and Cross Validation Model from sklearn.metrics import wilcoxon from sklearn.datasets import load_iris from sklearn.ensemble import GradientBoostingClassifier from sklearn import metrics from sklearn.model_selection import cross_val_score, kfold # Load the dataset X = load_iris().data y = load_iris().target # Prepare model and select your CV method model1 = GradientBoostingClassifier(n_estimators=100) model2 = GradientBoostingClassifier(n_estimators=100) kf = KFold(n_splits=10, random_state=None) # Extract results for each model on the same folds results_model1 = cross_val_score(model1, X, y, cv=kf) results_model2 = cross_val_score(model2, X, y, cv=kf) stat, p = wilcoxon(results_model1, results_model2, zero_method='loglik') stat Out[78]: 95.8</pre></div>																														

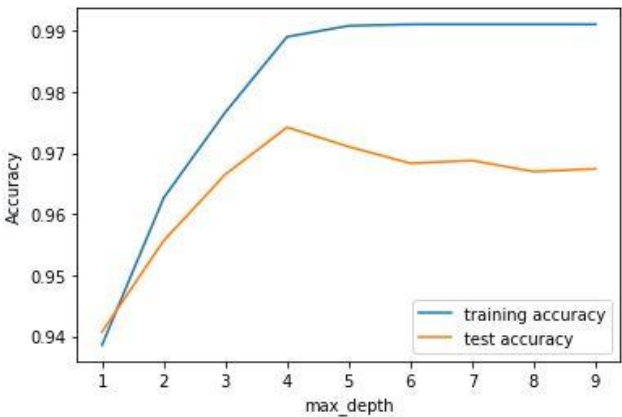
1. METRICS:

CLASSIFICATION REPORT:

```
In [52]: #computing the classification report of the model
print(metrics.classification_report(y_test, y_test_gbc))
```

	precision	recall	f1-score	support
-1	0.99	0.96	0.97	976
1	0.97	0.99	0.98	1235
accuracy			0.97	2211
macro avg	0.98	0.97	0.97	2211
weighted avg	0.97	0.97	0.97	2211

PERFORMANCE :



Out[83]:

	ML Model	Accuracy	f1_score	Recall	Precision
0	Gradient Boosting Classifier	0.974	0.977	0.994	0.986
1	CatBoost Classifier	0.972	0.975	0.994	0.989
2	Random Forest	0.969	0.972	0.992	0.991
3	Support Vector Machine	0.964	0.968	0.980	0.965
4	Decision Tree	0.958	0.962	0.991	0.993
5	K-Nearest Neighbors	0.956	0.961	0.991	0.989
6	Logistic Regression	0.934	0.941	0.943	0.927
7	Naive Bayes Classifier	0.605	0.454	0.292	0.997
8	XGBoost Classifier	0.548	0.548	0.993	0.984
9	Multi-layer Perceptron	0.543	0.543	0.989	0.983

2. TUNE THE MODEL – HYPERPARAMETER TUNING

Hyperparameter tuning is a hit and trial method where every combination of hyperparameters is tested and evaluated, and it selects the best model as the final model. Some scikit-learn APIs like GridSearchCV and RandomizedSearchCV are used to perform hyper parameter tuning.

```
In [58]: #HYPERPARAMETER TUNING
grid.fit(X_train, y_train)
```

Out[58]:

GridSearchCV

GridSearchCV(cv=5,
 estimator=GradientBoostingClassifier(learning_rate=0.7,
 max_depth=4),
 param_grid={'max_features': array([1, 2, 3, 4, 5]),
 'n_estimators': array([10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130,
 140, 150, 160, 170, 180, 190, 200])})

estimator: GradientBoostingClassifier

GradientBoostingClassifier(learning_rate=0.7, max_depth=4)

GradientBoostingClassifier

GradientBoostingClassifier(learning_rate=0.7, max_depth=4)

```
In [59]: print("The best parameters are %s with a score of %.2f"
              % (grid.best_params_, grid.best_score_))

The best parameters are {'max_features': 5, 'n_estimators': 200} with a score of 0.97
```

VALIDATION METHODS: KFOLD & Cross Folding

The validation set is used to evaluate a given model, but this is for frequent evaluation. We, as machine learning engineers, use this data to fine-tune the model hyperparameters. Hence the model occasionally *sees* this data, but never does it “*Learn*” from this. We use the validation set results, and update higher level hyperparameters. So the validation set affects a model, but only indirectly.

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.

The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation.

38

Wilcoxon signed-rank test

```
In [78]: #KFOLD and Cross Validation Model

from scipy.stats import wilcoxon
from sklearn.datasets import load_iris
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.model_selection import cross_val_score, KFold

# Load the dataset
X = load_iris().data
y = load_iris().target

# Prepare models and select your CV method
model1 = GradientBoostingClassifier(n_estimators=100)
model2 = XGBClassifier(n_estimators=100)
kf = KFold(n_splits=20, random_state=None)
# Extract results for each model on the same folds
results_model1 = cross_val_score(model1, X, y, cv=kf)
results_model2 = cross_val_score(model2, X, y, cv=kf)
stat, p = wilcoxon(results_model1, results_model2, zero_method='zsplit');
stat

Out[78]: 95.0
```

The **Wilcoxon Signed-Rank Test** is a statistical test used to determine if 2 measurements from a single group are significantly different from each other on your variable of interest. Your variable of interest should be continuous and your group randomly sampled to meet the assumptions of this test.

The 5x2cv combined **F test** is a procedure for comparing the performance of two models (classifiers or regressors) that was proposed by Alpaydin as a more robust alternative to Dietterich's 5x2cv paired t-test procedure

.

5x2CV combined F test

```
In [89]: from mlxtend.evaluate import combined_ftest_5x2cv
from sklearn.tree import DecisionTreeClassifier, ExtraTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from mlxtend.data import iris_data

# Prepare data and clfs
X, y = iris_data()
clf1 = GradientBoostingClassifier()
clf2 = DecisionTreeClassifier()

# Calculate p-value
f, p = combined_ftest_5x2cv(estimator1=clf1,
                             estimator2=clf2,
                             X=X, y=y,
                             random_seed=1)

print('f-value:', f)
print('p-value:', p)

f-value: 1.727272727272733
p-value: 0.2840135734291782
```


CHAPTER 10

ADVANTAGES & DISADVANTAGES

ADVANTAGES

- High Level of accuracy while comparing to other algorithms and methodology
- Fast in Classification Process
- When it is built in banking sectors , our proposed system will identify the phishing websites and deny the request further more if it is a phishing website.
- By learning this each one can gain an awareness on Phishing attacks
- Preventing financial fraud and embezzlement
- Prevention of cyber espionage
- Prevention of fraud through financial transactions like wire transfers etc.
- Protects your business. One of the most significant advantages of having Cybersecurity is it provides extensive protections for digital anomalies.

DISADVANTAGES

- It is needed to be monitored continuously so the bandwidth is consumed more
- It has huge number of features, so the classifying process is challenging part
- The Web server has some delay due to Python-Flask Environment was implemented.
- Day by day technology improves as well as negative impacts is also increased so as we must be updated to upcoming technologies.

CHAPTER 11

CONCLUSION

We discuss our large-scale system for automatically categorizing phishing runs in this design, which has a false positive rate with less than 0.1. In a fraction of the time, it takes a customized review procedure, our bracket system reviews millions of implicit phishing runner's responses. We reduce the amount of time that phishing runners can be active before we protect our druggies by automatically simplifying our blacklist with our classifier. Indeed, our blacklist strategy keeps us a step ahead of the phishers, thanks to a superb classifier and a robust system. Using the machine literacy method, we can only distinguish between phishing and legitimate URLs. In terms of the delicacy meter, this is what we obtained.

CHAPTER 12

FUTURE SCOPE

Although the use of URL lexical features alone has been shown to result in high accuracy (97%), phishers have learned how to make predicting a URL destination difficult by carefully manipulating the URL to evade detection. Therefore, combining these features with others, such as host, is the most effective approach .

For future enhancements, we intend to build the phishing detection system as a scalable extension and an anti-phish search engine which will incorporate online learning so that new phishing attack patterns can easily be learned and improve the accuracy of our models with better feature extraction.

CHAPTER 13

APPENDIX

GITHUB LINK:

[IBM-EPBL/IBM-Project-25620-1659968640: Web Phishing Detection \(github.com\)](https://github.com/IBM-EPBL/IBM-Project-25620-1659968640)

WEBPAGE LINK:

<https://phishing-shield.herokuapp.com/>