WEB PHISHING DETECTION
Project Report

Submitted By:
Team ID: PNT2022TMID35411

SUDHARSHAN THIRUKKUMARAN	2019105061
JAYAANT N	2019105540
SUBRAMANIA RAJA PIRAMUTHU RAJA	2019105060
LOGESH SURESH	2019105026

In Partial Fulfillment for the Award of the Degree
OF
BACHELOR OF ENGINEERING
IN
ELECTRONICS AND COMMUNICATION ENGINEERING

COLLEGE OF ENGINEERING GUINDY
ANNA UNIVERSITY

CONTENTS

1. INTRODUCTION

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. Phishing attack is a simplest way to obtain sensitive information from innocent users. Aim of the phishers is to acquire critical information like username, password and bank account details.

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.

The main objective of this project is to determine and detect the websites and classify them as authentic or illegitimate.

1.2 Purpose

Nowadays, since it is really simple to develop a phoney website that closely resembles a legitimate website, phishing has recently become a major topic of worry for security researchers. Although experts are able to spot bogus websites, not all users are, and those who aren't fall prey to phishing scams. The attacker's primary goal is to obtain bank account passwords.

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. In order to classify the legitimacy of the phishing datasets, we used classification algorithms and approaches to extract the relevant characteristics. The final phishing detection rate can be used to identify the e-banking phishing website based on several key features like URL and domain identity, security and encryption standards, and other factors. Our system will utilize a data mining technique to determine whether an e-banking website is a phishing website or not after a user completes an online transaction and makes a payment through it.

2. LITERATURE SURVEY

2.1 Existing Problem

Phishing is a social engineering attack that aims at exploiting the weakness found in system processes as caused by system users. For instance, a system may be technically safe enough to prevent password theft, but uninformed end users may reveal their passwords if an attacker requested that they change them via a specific Hypertext Transfer Protocol (HTTP) connection, endangering the system's overall security.

Technical flaws can also be exploited by attackers to create much more convincing social engineering communications, such as DNS cache poisoning (i.e., use of legitimate, but spoofed, domain names can be far more persuading than using different domain names).

Since phishing attacks aim at exploiting weaknesses found in humans (i.e., system end-users), it is difficult to mitigate them.
For example, end-users failed to detect 29% of phishing attacks even when trained with the best performing user awareness program. On the other hand, software phishing detection techniques are evaluated against bulk phishing attacks, which makes their performance practically unknown with regards to targeted forms of phishing attacks. These limitations in phishing mitigation techniques have practically resulted in security breaches against several

organizations including leading information security providers.

2.2 References

- Gunter Ollmann, “The Phishing Guide Understanding & Preventing Phishing Attacks”, IBMInternet Security Systems, 2007.
- https://resources.infosecinstitute.com/category/enterprise /phishing/the-phishing-landscape/phishing-data-attackstatistics/#gref
- Mahmoud Khonji, Youssef Iraqi, "Phishing Detection: A Literature Survey IEEE, and Andrew Jones, 2013
- Mohammad R., Thabtah F. McCluskey L., (2015) Phishing websites dataset. Available: https://archive.ics.uci.edu/ml/datasets/Phishing+Websites Accessed January 2016
- http://dataaspirant.com/2017/01/30/how-decision-treealgorithm-works/
- www.phishtank.com
- www.alexa.com

2.3 Problem Statement Definition

The definition of phishing attacks is not consistent in the literature, which is due to the fact that the phishing problem is broad and incorporates varying scenarios. For example, according to PhishTank:
“Phishing is a fraudulent attempt, usually made through email, to steal your personal information”

PhishTank's definition holds true in a number of scenarios which, roughly, cover the majority of phishing attacks (although no accurate studies have been made to reliably quantify this). However, the definition limits phishing attacks to

stealing personal information, which is not always the case.
Another definition is provided by Colin Whittaker et. al.

“We define a phishing page as any web page that, without permission, alleges to act on behalf of a third party with the intention of confusing viewers into performing an action with which the viewer would only trust a true agent of the third party”

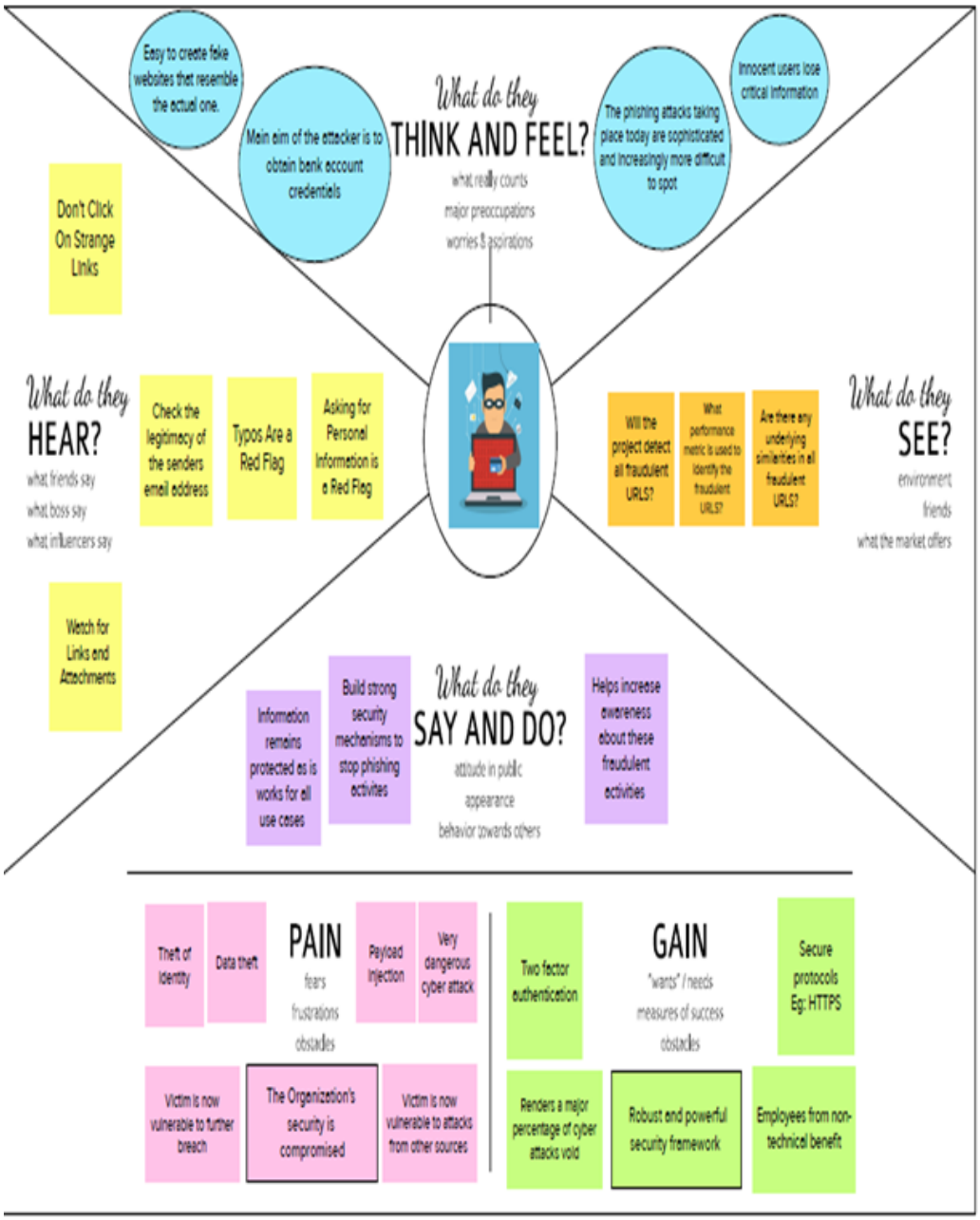
The definition by Colin Whittaker et. al. aims to be broader than PhishTank's definition in a sense that attackers’ goals are no longer restricted to stealing personal information from victims. On the other hand, the definition still restricts phishing attacks to ones that act on behalf of third parties, which is not always true.
For example, phishing attacks may communicate socially engineered messages to lure victims into installing MITB malware by attracting the victims to websites that are supposed to deliver safe content (e.g., video streaming). Once the malware (or crimeware as often named by Anti-Phishing Working Group (APWG)2) is installed, it may log the victim's keystrokes to steal their passwords.
In order to address the limitations of the previous definitions above, we consider phishing attacks as semantic attacks which use electronic communication channels (such as E-Mails, HTTP, SMS, VoIP, etc…) to communicate socially engineered messages to persuade victims to perform certain actions (without restricting the actions) for an attacker's benefit (without restricting the benefits).

Definition 1
Phishing is a type of computer attack that communicates socially engineered messages to humans via electronic communication channels in order to persuade them to perform certain actions for the attacker's benefit.Due to exponential growth in the number of people using the internet, the internet now controls much of the world. Due to the anonymity offered by the internet and the rapid growth of online transactions, hackers try to trick end users by using techniques like phishing, SQL injection, malware, man-in-the-

middle attacks, domain name system tunneling, ransomware, web Trojans, and so on. Phishing is said to be the most deceptive attack of all these. Our main aim of this paper is classification of a phishing website with the aid of various machine learning techniques to achieve maximum accuracy and concise model.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation and Brainstorming

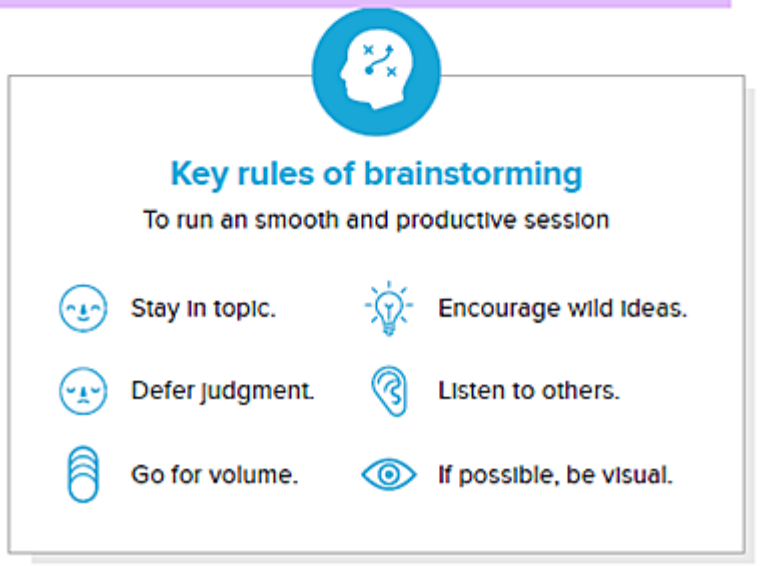
1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

Phishing is a type of social engineering attack often used to steal user data.
Phishing attacks are becoming more and more sophisticated, and our algorithms are suffering to keep up with this level of sophistication. They have low detection rate and high false alarm especially when novel phishing approaches are used. The blacklist-based method is unable to keep up with the current phishing attacks as registering new domains has become easier. Moreover, a comprehensive blacklist can ensure a perfect up-to-date database. Various other techniques such as page content inspection algorithms have been used to combat the false negatives but as each algorithm uses a different approach, their accuracy varies. Therefore, a combination of the two can increase the accuracy while implementing different error detection methods.



2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Sudharshan Thirukkumaran

Long URL's can be used to hide the suspicious part of in the address bar. Hence, keeping a value for the length of the URL to determine the phishing websites

A secure web-page link contains at most 5 dots. If perhaps, there are more than 5 dots in a web page then it may be recognized as a phishing link

Phishing websites stay online for a very limited time. Hence, these websites are hosted with IP-address instead of a fully qualified domain name, which most legal websites don't follow anymore.

The phishing URL may include the "@" symbol somewhere within the address because the web browser, when reading an internet address; ignore everything to the left of the @ symbol

Jayaant N

Detect suspicious Pop-Ups If you go to a website or web-page and a pop-up window shows up right away asking you to enter personal information, like a username and password, this is a red flag.

The spelling of the web address is another important aspect to be noted. Fraudsters take advantage of the fact that we tend to skim read information. As such, they will create web addresses that are similar to well-known and trusted ones to launch their phishing attacks.

PageRank could be used as a feature to identify phishing websites. The PageRank score could give relevant information regarding the authenticity of the website.

Enter fake credentials to check if a website is authentic. If logged in,it is likely to be a phishing website.

Logesh Suresh

A website can be considered suspicious depending on its journey to the website. If the link is present in an EMAIL it should be noted if its from a reputable source or not.

Some reviews about the website could give relevant insight regarding the authenticity of the website. The chances that the site has defrauded people in the past or people have already discovered that the site is fake, is pretty good. Victims of scans regularly go online to report and/ or share their experience, warning users to avoid the site..

Some red flags to look for are broken English/ grammar, spelling mistakes and low resolution images. Another indication that it may be a phishing website is the lack of a 'contact us' page. Authentic businesses usually provide contact details, including their postal address, phone number, email address and social media links.

Legitimate websites usually allow credit card/ PayPal payments. Phishing websites on the other hand mostly only allow bank transfers. This could be considered as a red flag and viewed as an alarm.

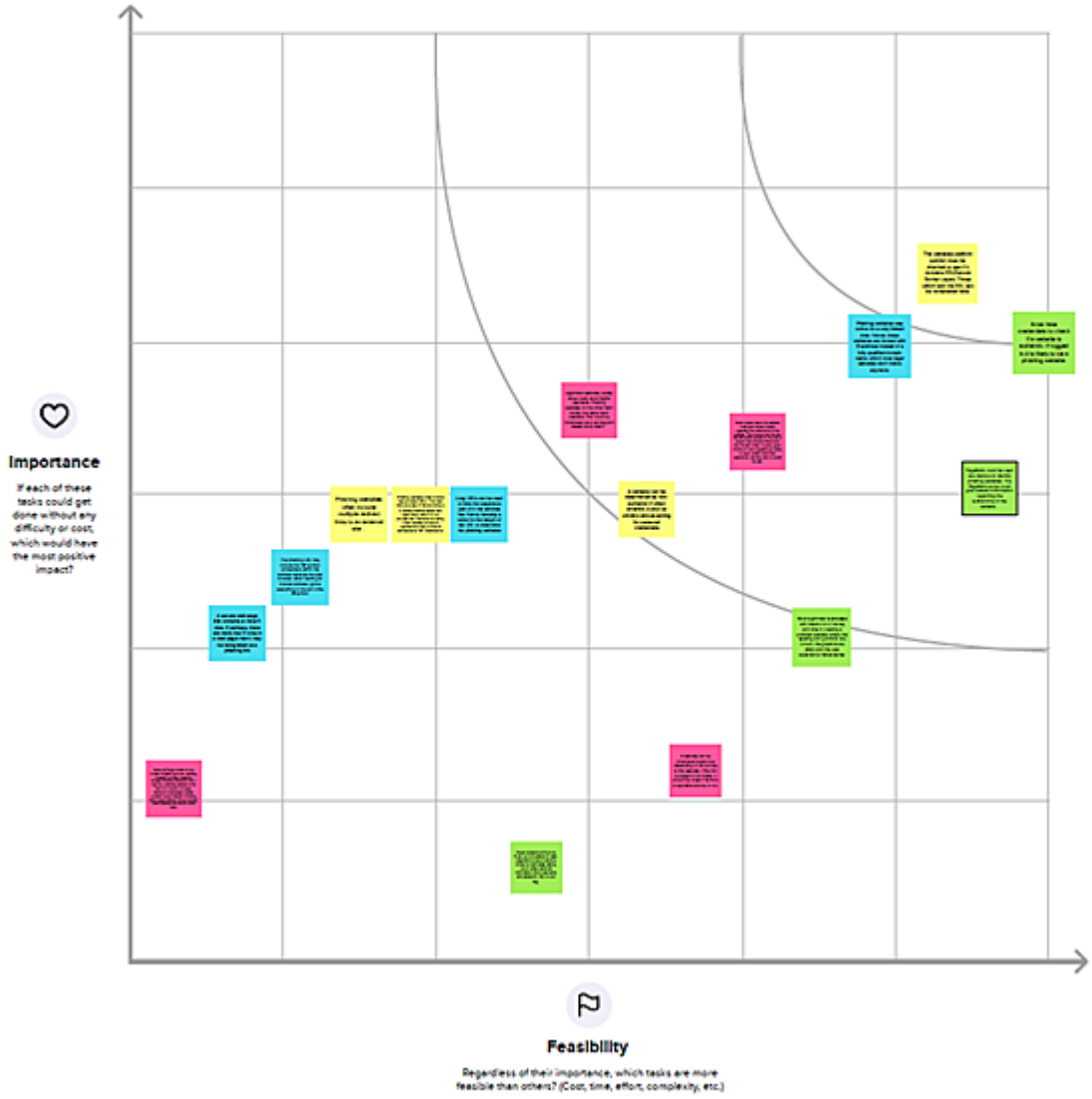
Subramania Raja Piramuthu Raja

Phishing websites often include multiple redirect links to an external site .

Phishing websites often include hex-encoded URLs. The main illicit purpose of this encoding is to evade blacklist based anti-spam filters which do not process hex character encoding. It also evades protection mechanisms that prohibit IP addresses as URL destinations.

A website can be determined as non-authentic if when entered, a pop-up window shows asking for personal credentials

The websites padlock symbol must be checked to see if it contains SSL(Secure Socket Layer). Those which lack the SSL can be considered fake.


Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

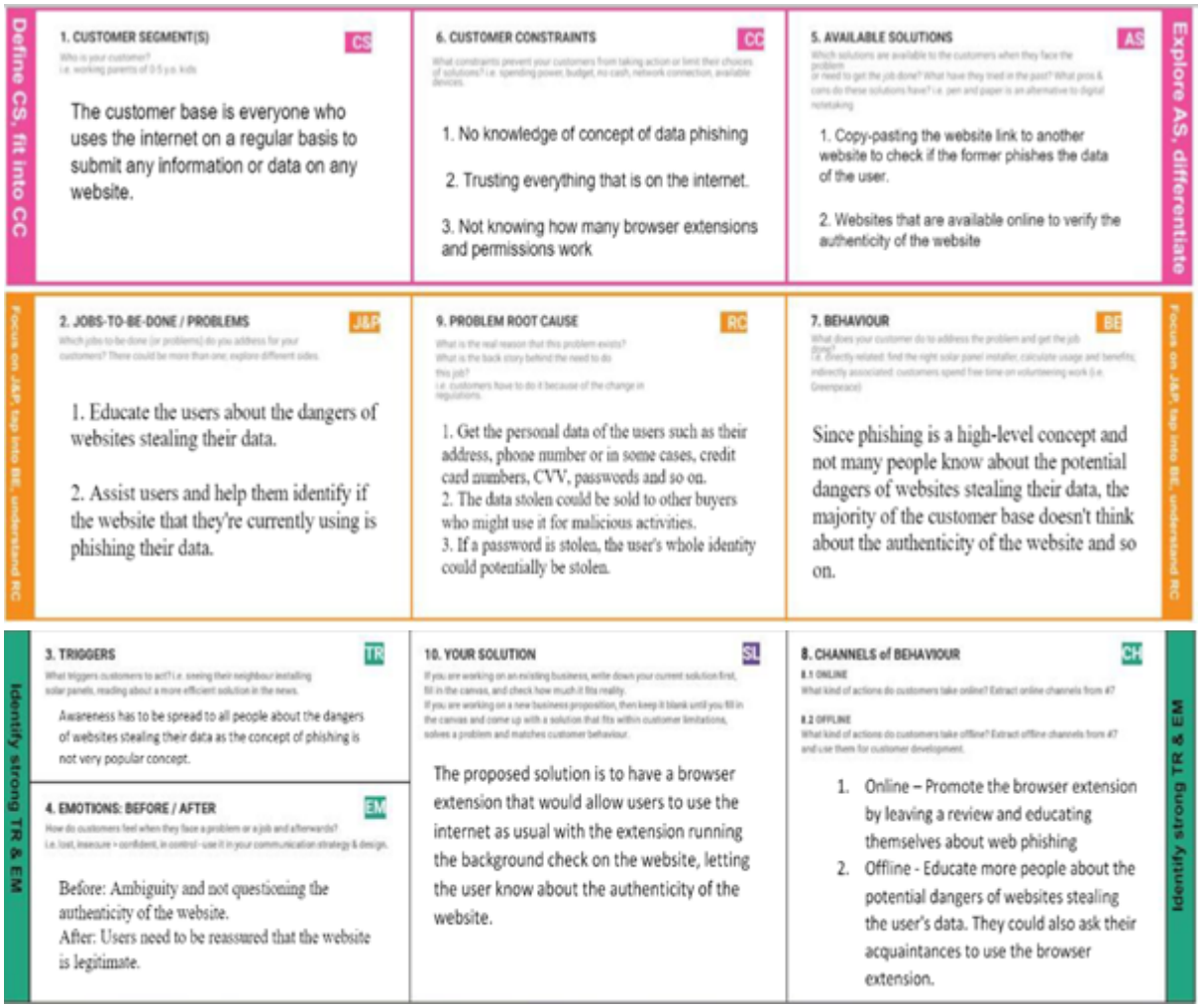
3.3 Proposed solution

S.no	Parameter	Description
1	Problem Statement (Problem to be solved)	Low detection accuracy and high false alarm rates are problems with phishing detection methods, especially when new phishing strategies are introduced. A thorough blacklist cannot guarantee a flawless up-to-date database, and as a result, the most popular methodology, one that relies on blacklists, is ineffective in reacting to phishing assaults that are on the rise as new domain registration has grown simpler. Additionally, some solutions have made use of page content inspection to address the issue of false negatives and to strengthen stale list weaknesses. Additionally, several page content inspection algorithms each take a different approach to accurately identifying phishing websites. Hence, Problem to be solved is: Detection of malicious websites and ransomware; Identify, block, and mitigate targeted threats.

2	Idea / Solution description	Using the brainstorming and idea prioritization template in mural these ideas are considered as feasible and important: • As the problem statement was mapped to a binary classification problem. The idea is to come up with generalized linear models and evaluate the performance of the linear models. • To enhance the robustness of the solution, models like decision trees, random forest could also be tried out.
3	Novelty / Uniqueness	• To come up with effective feature engineering techniques to evaluate the given URL's authenticity. • To come up with models and find the ideal hyperparameter to for the binary classification task (detection of malicious websites).
4	Social Impact / Customer Satisfaction	• Today, e-payment is the most widely used method of payment, and a lot of online purchases rely on it. • The major objective is to identify phishing e-payment websites and safeguard user information from phishing to protect users' privacy.
5	Business Model (Revenue Model)	• This system receives funding from users and online marketplaces for the sale of goods.

6	Scalability of the Solution	• Machine Learning models and effective feature engineering techniques helps identify phishing websites and come up with key features that are common in most phishing websites

3.4 Proposed solution fit



4. REQUIREMENT ANALYSIS

4.1 Functional requirement

FR No.	Functional Requirement(Epic)	Sub Requirement(Story/sub-task)
FR-1	User Registration	Registration through Gmail Registration using Forms
FR-2	User Confirmation	Confirmation via Email
FR-3	Website identification	Model Detects the malicious website and notifies user
FR-4	Prediction	Model predicts the URL using algorithms such as Regression, KNN.
FR-5	Classifier	Model predicts the output to the classifier and prints the result on the screen.
FR-6	Results	Model predict the website and generates a pop-up message to the user before they enter any confidential details.

4.2 Non-functional requirements

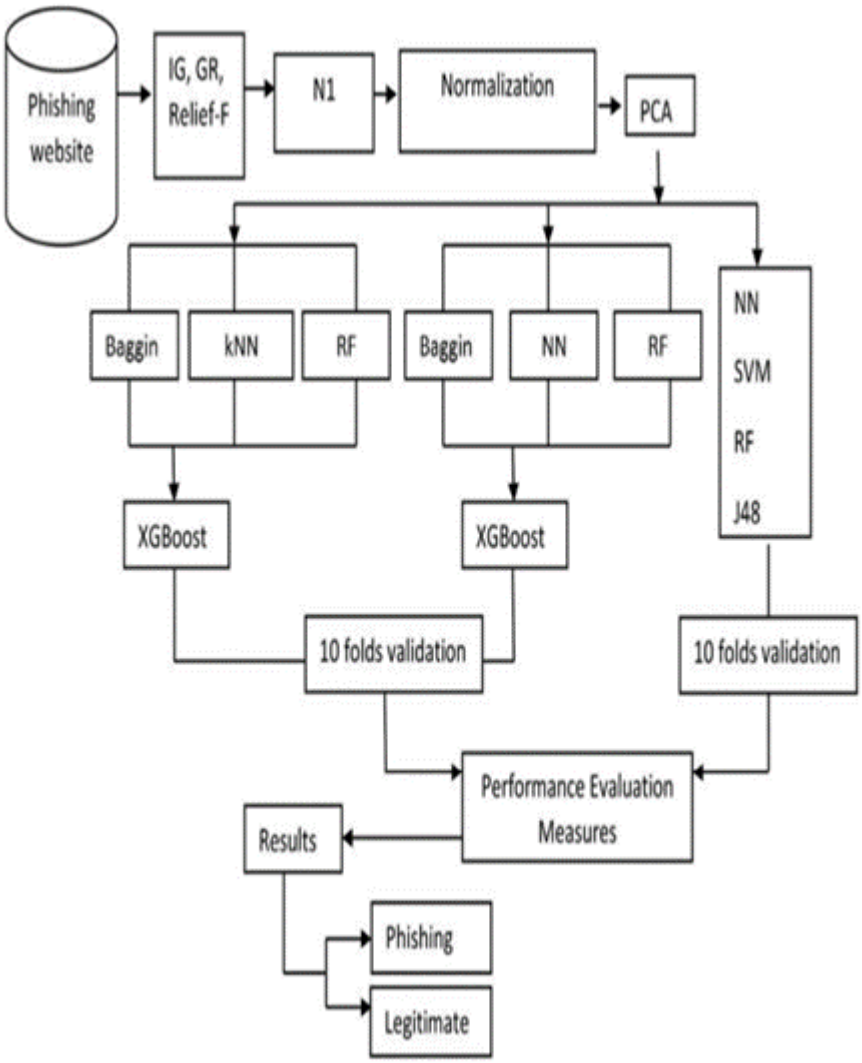
FR No.	Non-Functional Requirement	Description
NFR-1	Usability	User can browse websites easily using web phishing detection without fear of losing data.

NFR-2	Security	User can authenticate websites by getting pop-up messages
NFR-3	Reliability	The product should be reliable for all websites
NFR-4	Performance	The analysis time should be fast as user should noy stay connected to malicious websites for long
NFR-5	Availability	All users should get access to the resources at all possible locations
NFR-6	Scalability	The performance of the website should be efficient to handle increasing user and loads without much disturbance

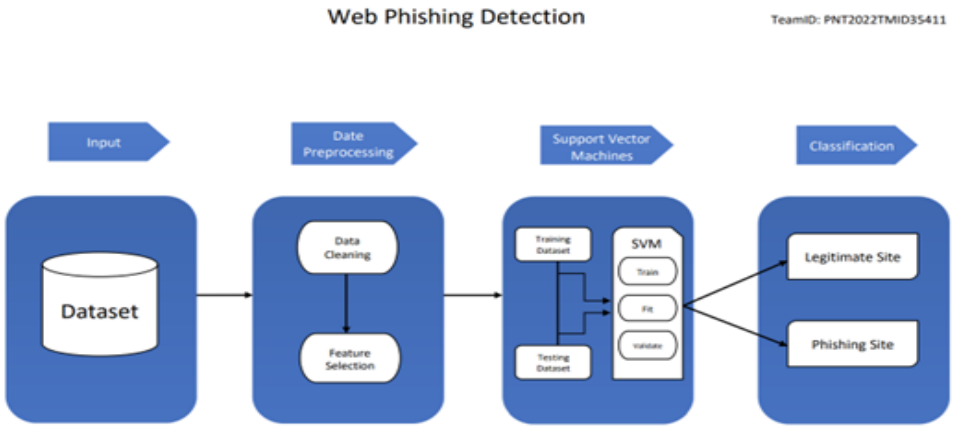
5. PROJECT DESIGN

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.



5.2 Solution and technical architecture



5.3 User Stories

USER TYPE	FUNCTIONAL REQUIREMENTS	USER STORY NUMBER	USER STORY/TASK	ACCEPTANCE CRITERIA	PRIORITY	RELEASE
Customer (web user)	Register	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered to the application	I can receive email confirmation and click "confirm"	High	Sprint-1
		USN-3	As a user, I want to be able to check and see the security of the website.	I can store details in incorrect websites.	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)	Register	USN-5	As a user, I should be able to register with a verification code.	This protects from strangers.	High	Sprint-1
		USN-6	As a user, I should not answer random calls	This is to maintain safety from attacks	High	Sprint-1
Administrator		USN-7	Admin should maintain a database safely	This keeps all in order and can access if needed again	High	Sprint-2
Customer care		USN-8	As a user, if my account is phished	I should be able to complain and report	High	Sprint-1
		USN-9	I should not take others information	I should be punished	Medium	Sprint-2

6. PROJECT PLANNING & SCHEDULING

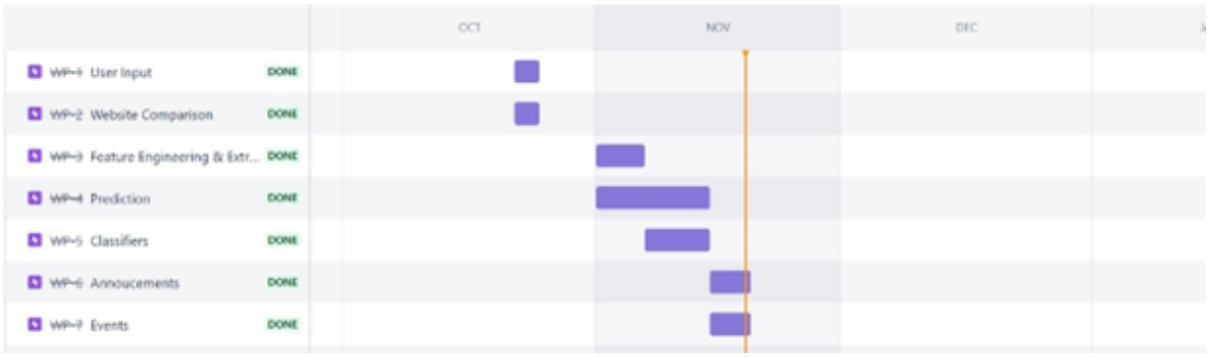
6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (EPIC)	User Story Number	User Story/Task	Story Points	Priority	Team Member
Sprint-1	User Input	USN-1	User inputs a URL in the input field to check for its credibility.	1	Medium	Logesh Suresh
Sprint-1	Website Comparison	USN-2	The built model verifies the credibility using a blacklist and whitelist approach.	1	High	Sudharshan Thirukkumaran
Sprint-2	Feature Extraction	USN-3	If the comparison fails, various feature engineering and extraction methods are employed to extract the URL features.	1	High	Subramania Raja
Sprint-2	Prediction	USN-4	Models like KNN & logsitic regression are used to predict the credibility of the website.	1	High	Jayaant N
Sprint-3	Classifier	USN-5	Models send the generated output to classifier.	1	Medium	Logesh Suresh & Sudharshan Thirukkumaran
Sprint-4	Announcement	USN-6	Models then displays whether it is a phishing website or not.	1	High	Jayaant N & Subramania Raja
Sprint-4	Events	USN-7	This models needs the capability of retrieving and displaying the result.	1	High	Jayaant N

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on planned End Date)	Sprint Release Date (Actual Release Date)
Sprint-1	20	2 Days	22 Oct 2022	24 Oct 2022	20	29 Oct 2022
Sprint-2	20	2 Days	22 Oct 2022	24 Oct 2022	20	05 Nov 2022
Sprint-3	20	14 Days	01 Nov 2022	14 Nov 2022	20	12 Nov 2022
Sprint-4	20	5 Days	15 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3 Reports from JIRA



7. CODING & SOLUTIONING

7.1 Feature

Obtaining these types of features requires active scan to target domain. Page contents are processed for us to detect whether target domain is used for phishing or not. Processed information about pages is given below.

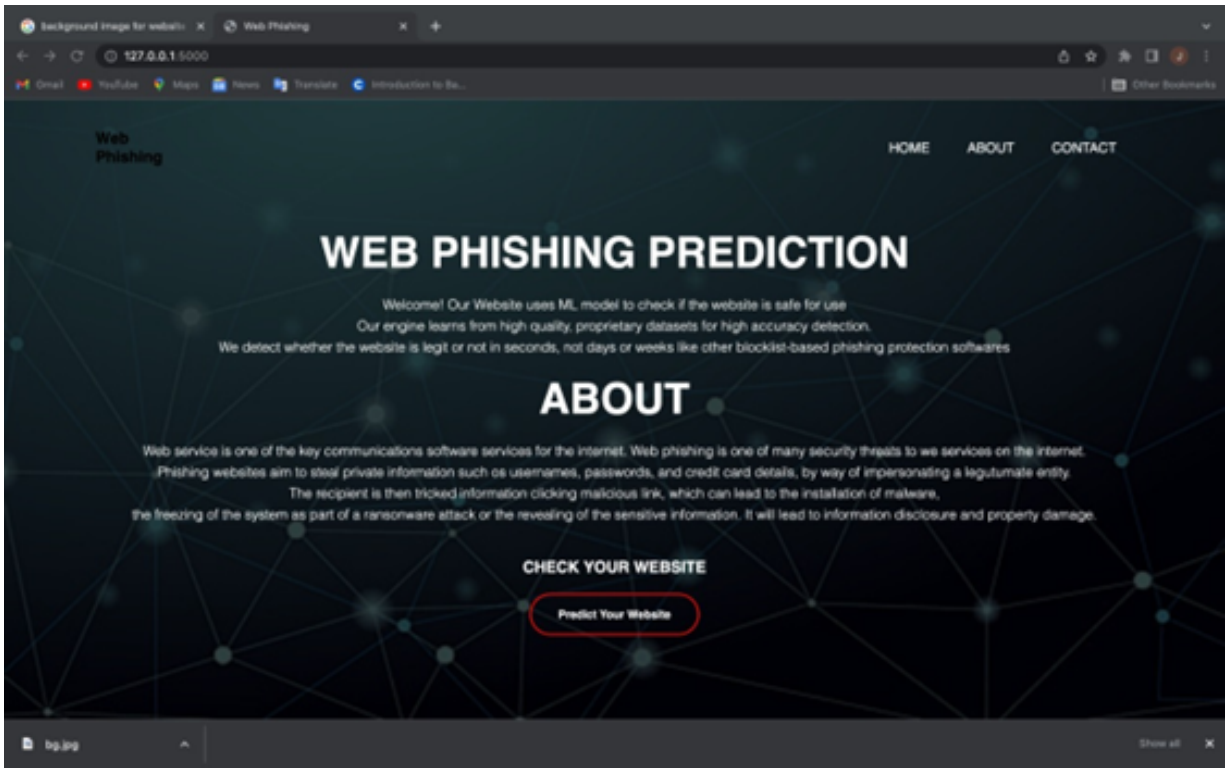
1. Page Titles
2. Meta Tags
3. Hidden Text
4. Text in the Body
5. Images etc.

By analysing this information, we can gather information such as;

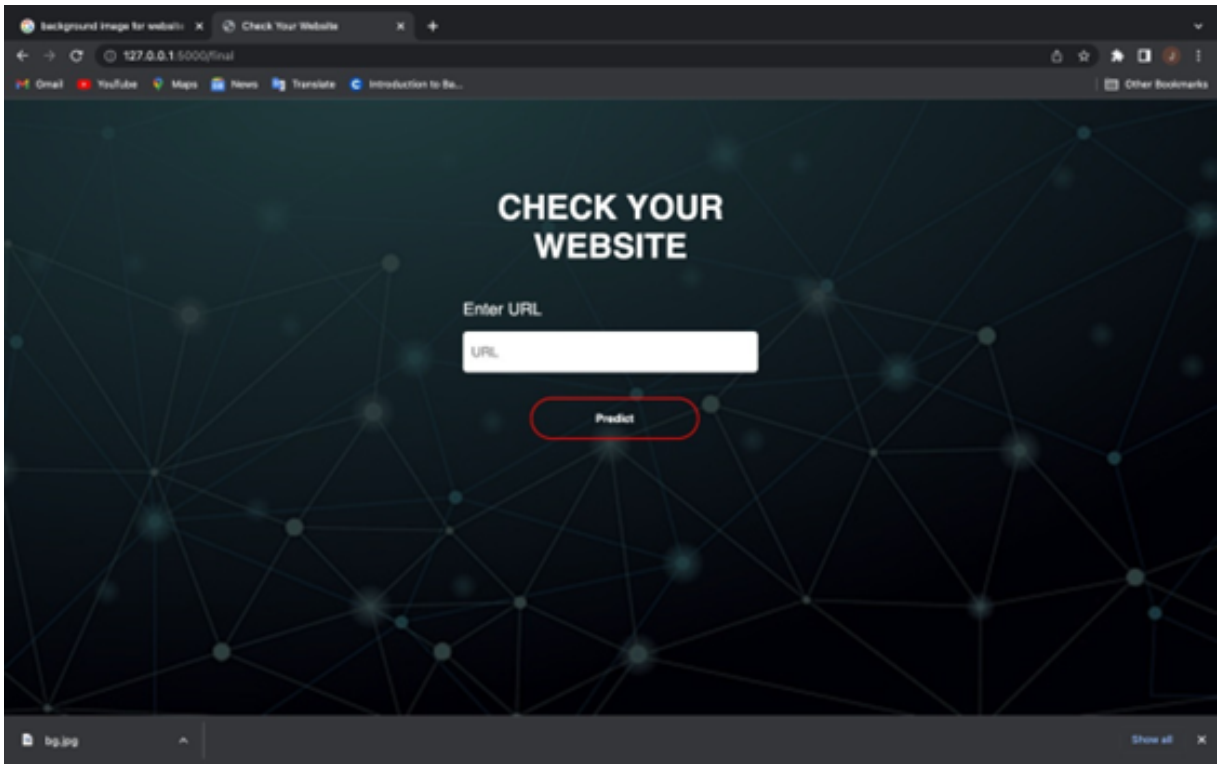
- Is it required to login to website
- Website category
- Information about audience profile etc.

All of features explained above are useful for phishing domain detection. In some cases, it may not be useful to use some of these, so there are some limitations for using these features. For example, it may not be logical to use some of the features such as Content-Based Features for the developing fast detection mechanism which is able to analyze the number of domains between 100.000 and 200.000. Another example would be, if we want to analyze new registered domains Page-Based Features is not very useful. Therefore, the features that will be used by the detection mechanism depends on the purpose of the detection mechanism. Which features to use in the detection mechanism should be selected carefully.

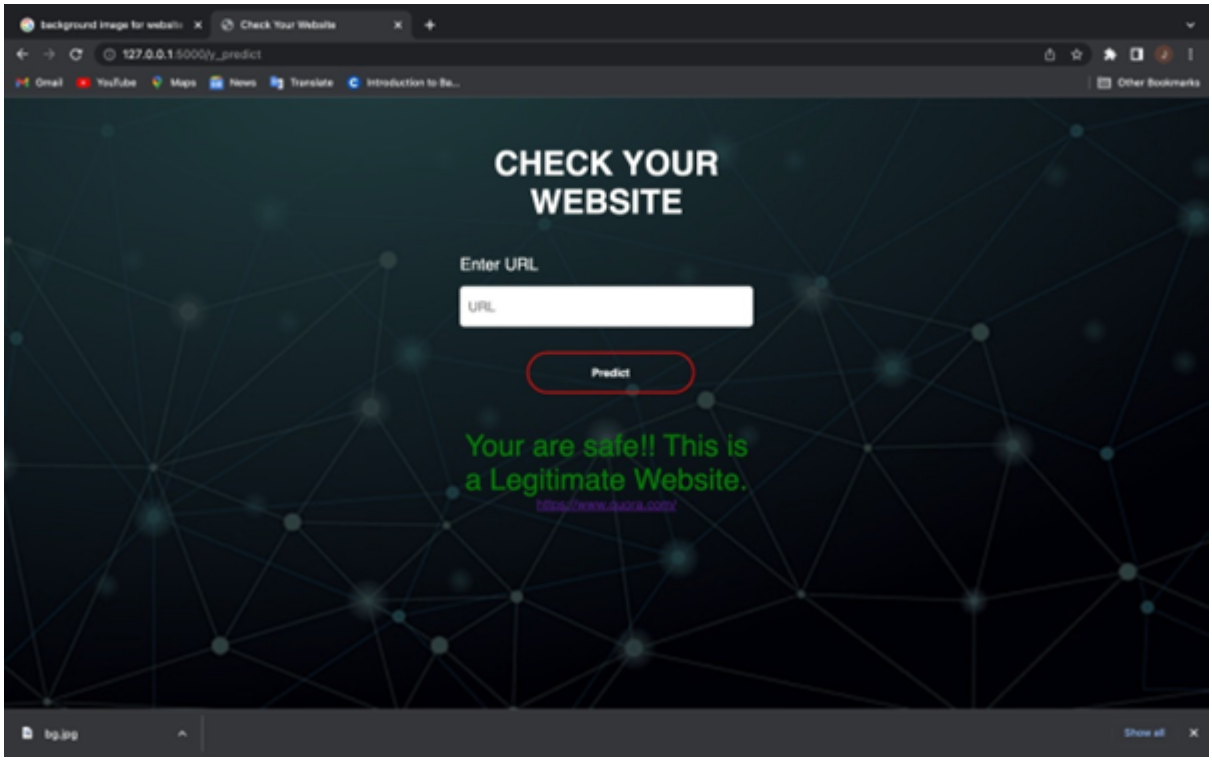
7.2 Outputs



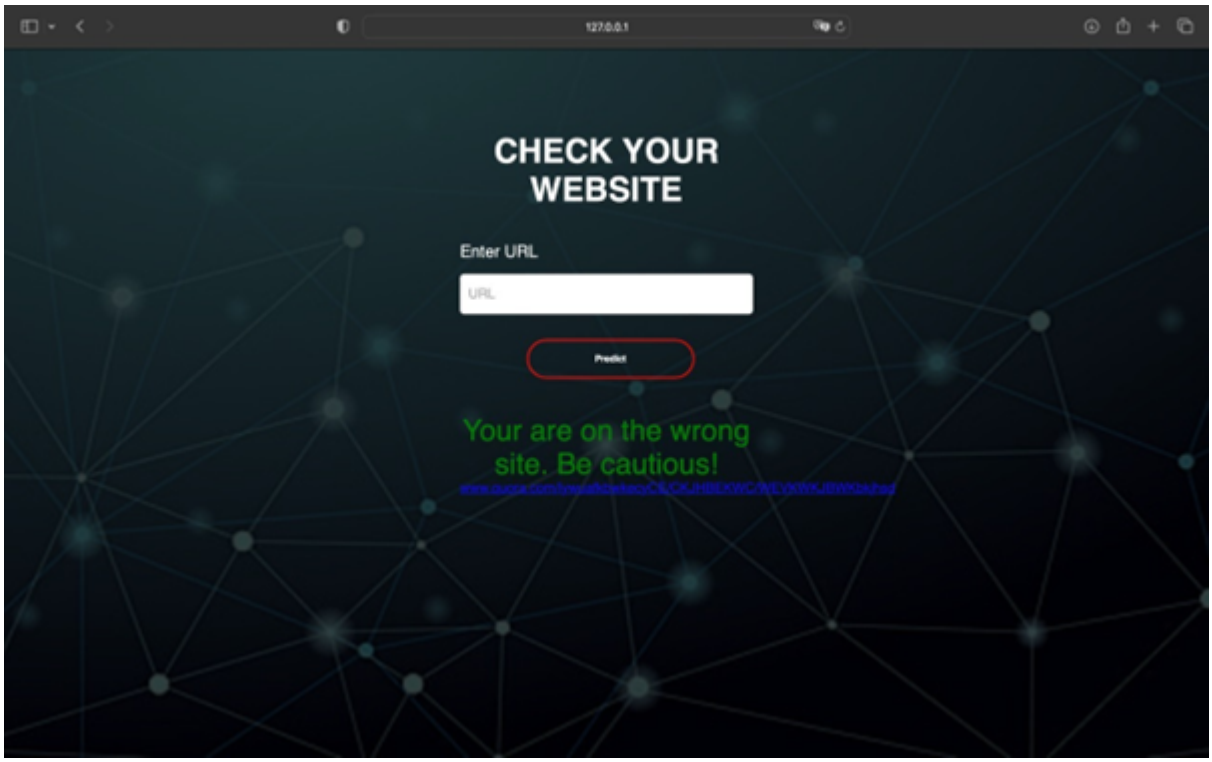
Home Page



URL of the website to be entered here



If the URL entered is safe



If the website entered is illegitimate or supicious of phishing

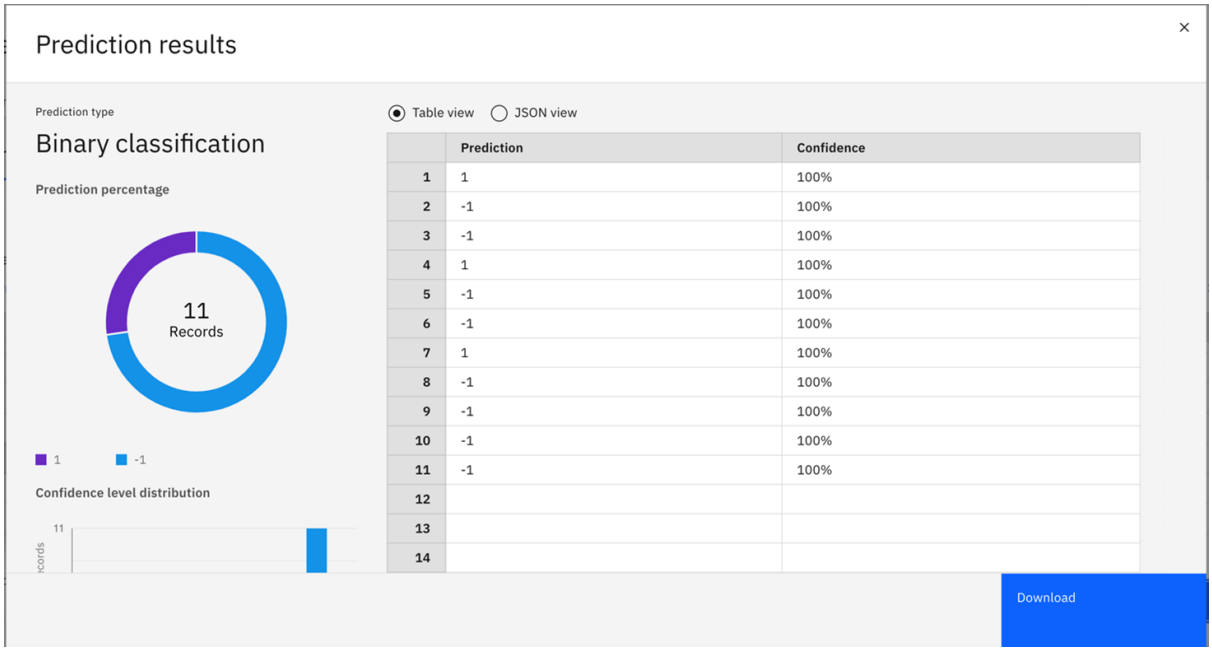
7.3 Database Schema (if Applicable)

Predictive research is chiefly concerned with forecasting (predicting) outcomes, consequences, costs, or effects. This type of research tries to extrapolate from the analysis of existing phenomena, policies, or other entities in order to predict something that has not been tried, tested, or proposed before.
Machine learning model predictions allow businesses to make highly accurate guesses as to the likely outcomes of a question based on historical data, which can be about all kinds of things – customer churn likelihood, possible fraudulent activity, and more.
These techniques can provide managers and executives with decision-making tools to influence upselling, sales and revenue forecasting, manufacturing optimization, and even new product development.

8. TESTING

8.1 Test Cases

The above decision tree classifier model has been trained with a validation accuracy of 96% . A given validation set of 11 entries had been given as the input and the output obtained for the same has been submitted below.



	Prediction	Confidence
1	1	100%
2	-1	100%
3	-1	100%
4	1	100%
5	-1	100%
6	-1	100%
7	1	100%
8	-1	100%
9	-1	100%
10	-1	100%
11	-1	100%
12		
13		
14		

8.2 User Acceptance Testing

Acceptance Testing
UAT Execution & Report Submission

Date	17 November 2022
Team ID	PNT2022TMID35411
Project Name	Project – Web Phishing Detection
Maximum Marks	4 Marks

1. 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).

2. 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	11	4	2	2	19
Duplicate	1	0	4	0	5
External	2	0	0	1	3
Fixed	11	3	2	14	30
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	2	3
Won't Fix	0	1	2	1	5
Totals	25	8	12	20	66

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	11	4	2	2	19
Duplicate	1	0	4	0	5
External	2	0	0	1	3
Fixed	11	3	2	14	30
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	2	3
Won't Fix	0	1	2	1	5
Totals	25	8	12	20	66

3. 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	15	0	0	15
Client Application	42	2	0	40
Security	2	0	0	2
Outsource Shipping	3	0	0	3

Exception Reporting	5	0	0	5
Final Report Output	10	0	0	10
Version Control	2	0	0	2

9. RESULTS

9.1 Performance Metrics

```python
# importing the libraries
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import confusion_matrix,accuracy_score
```

Python

```python
#reading the dataset
ds = pd.read_csv("../Flask/dataset_website.csv")
```

Python

```python
ds.head()
```

Python

	index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain	SSLfinal_State	Domain_registeration_leng
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 × 32 columns

```python
#splitting data into independent and dependent variables
x,y = ds.iloc[:,1:31].values,ds.iloc[:,-1].values
print(x,y)
```

```
[[-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]] [-1 -1 -1 ... -1 -1 -1]
```

```python
#splitting data into test set and train set
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)
```

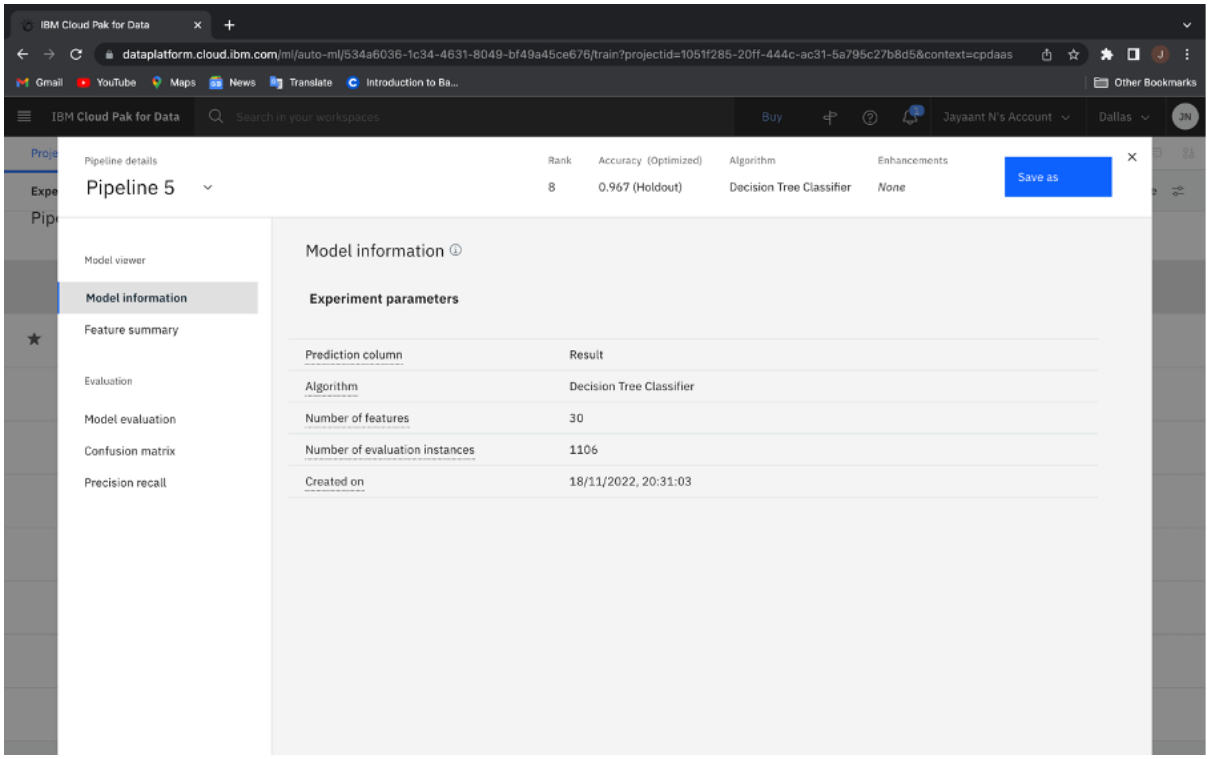
```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

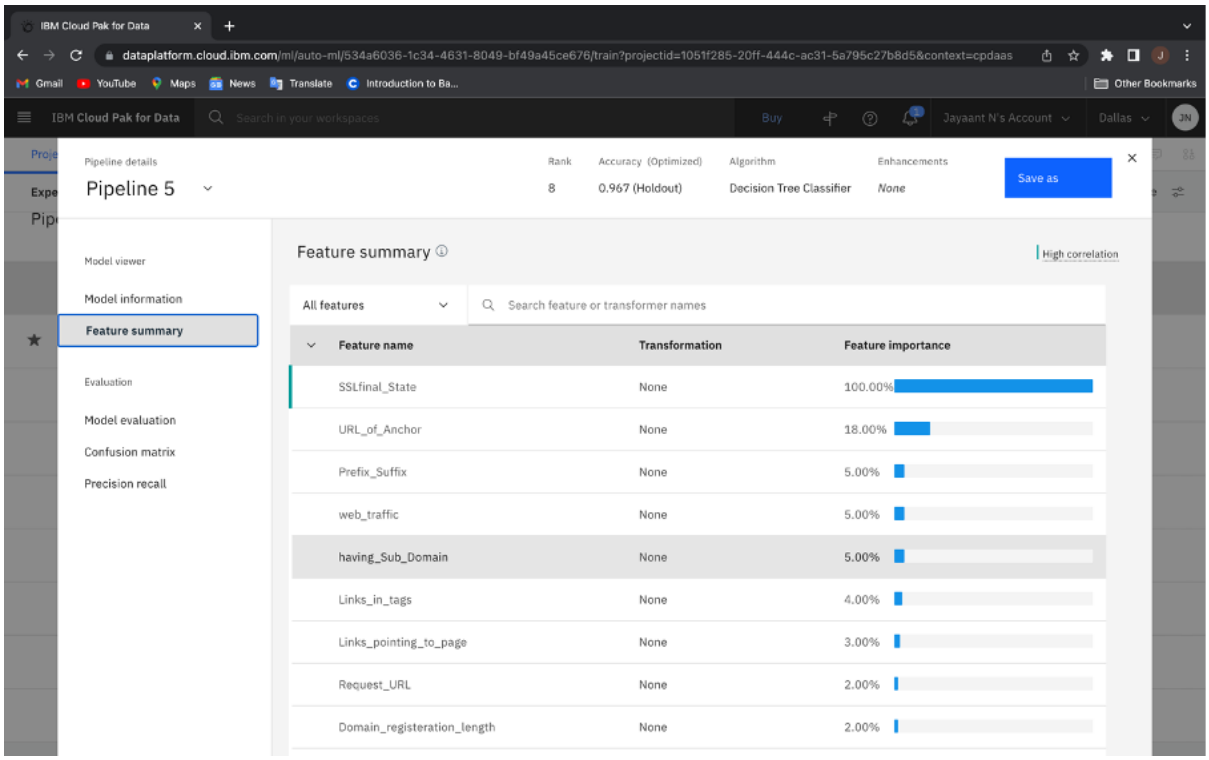
```
y_pred5=dt.predict(x_test)
from sklearn.metrics import accuracy_score
dec_tree=accuracy_score(y_test,y_pred5)
print(dec_tree)
```

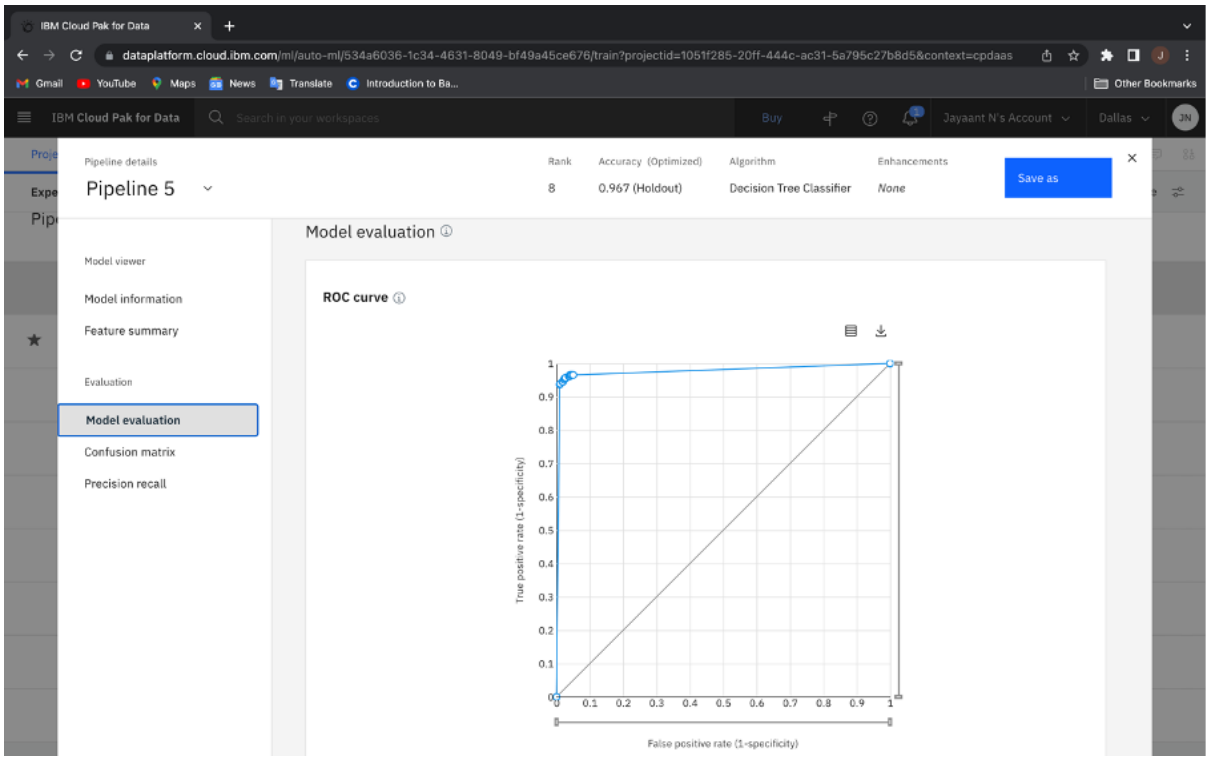
```
0.9611035730438715
```

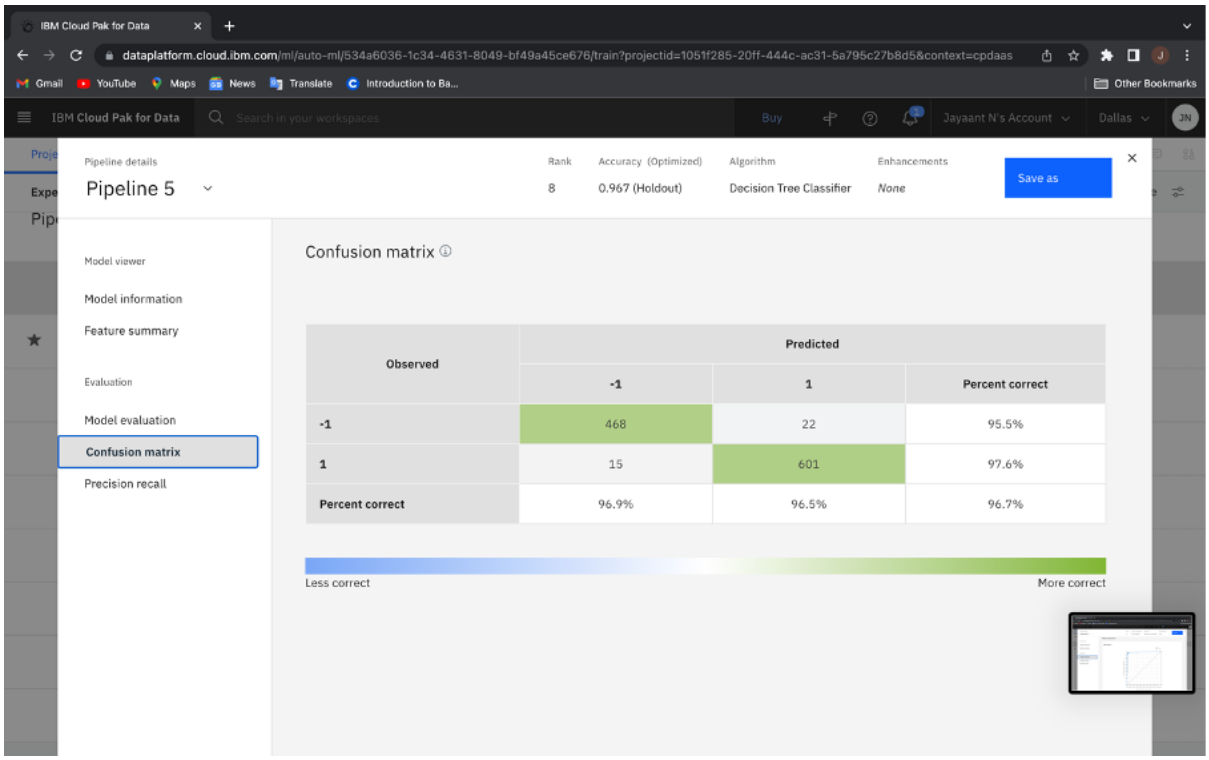
Model Accuracy : 96.11 %

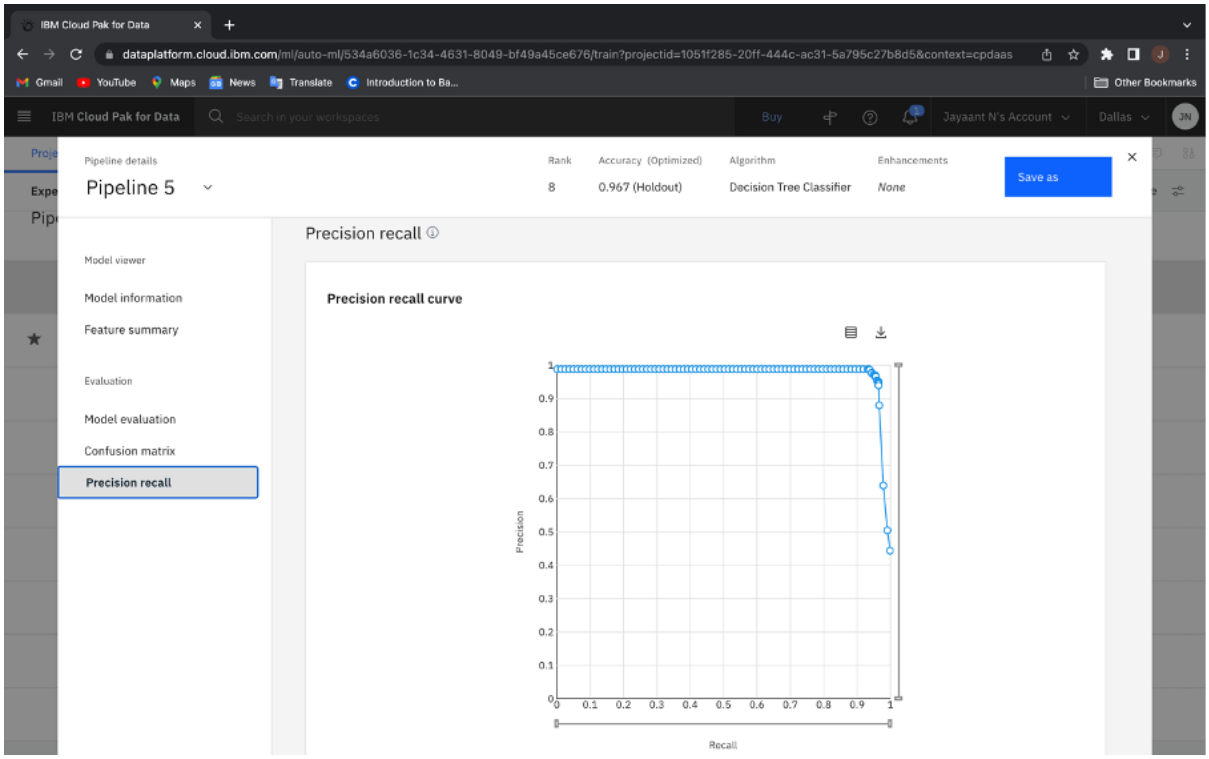
The above designed model has been uploaded on IBM cloud and the performance metrics has been analyzed.




IBM Cloud Pak for Data
dataplatform.cloud.ibm.com/ml/auto-ml/534a6036-1c34-4631-8049-bf49a45ce676/train?projectid=1051f285-20ff-444c-ac31-5a795c27b8d5&context=cpdaas
IBM Cloud Pak for Data
Pipeline details
Pipeline 5
Rank
8
Accuracy (Optimized)
0.967 (Holdout)
Algorithm
Decision Tree Classifier
Enhancements
None
Save as
Model viewer
Model information
Feature summary
Evaluation
Model evaluation
Confusion matrix
Precision recall
Feature summary
High correlation
All features
Search feature or transformer names
Feature name
Transformation
Feature importance
SSLfinal_State None 100.00%
URL_of_Anchor None 18.00%
Prefix_Suffix None 5.00%
web_traffic None 5.00%
having_Sub_Domain None 5.00%
Links_in_tags None 4.00%
Links_pointing_to_page None 3.00%
Request_URL None 2.00%
Domain_registeration_length None 2.00%


IBM Cloud Pak for Data
dataplatform.cloud.ibm.com/ml/auto-ml/534a6036-1c34-4631-8049-bf49a45ce676/train?projectid=1051f285-20ff-444c-ac31-5a795c27b8d5&context=cpdaas
IBM Cloud Pak for Data
Pipeline details
Pipeline 5
Rank
8
Accuracy (Optimized)
0.967 (Holdout)
Algorithm
Decision Tree Classifier
Enhancements
None
Save as
Model viewer
Model information
Feature summary
Evaluation
Model evaluation
Confusion matrix
Precision recall
Model evaluation
ROC curve
True positive rate (1-specificity)
False positive rate (1-specificity)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


IBM Cloud Pak for Data
dataplatform.cloud.ibm.com/ml/auto-ml/534a6036-1c34-4631-8049-bf49a45ce676/train?projectid=1051f285-20ff-444c-ac31-5a795c27b8d5&context=cpdaas
IBM Cloud Pak for Data
Pipeline details
Pipeline 5
Rank
8
Accuracy (Optimized)
0.967 (Holdout)
Algorithm
Decision Tree Classifier
Enhancements
None
Save as
Model viewer
Model information
Feature summary
Evaluation
Model evaluation
Confusion matrix
Precision recall
Confusion matrix
Observed
Predicted
-1
1
Percent correct
-1
468
22
95.5%
1
15
601
97.6%
Percent correct
96.9%
96.5%
96.7%
Less correct
More correct


IBM Cloud Pak for Data
dataplatform.cloud.ibm.com/ml/auto-ml/534a6036-1c34-4631-8049-bf49a45ce676/train?projectid=1051f285-20ff-444c-ac31-5a795c27b8d5&context=cpdaas
IBM Cloud Pak for Data
Pipeline details
Pipeline 5
Rank
8
Accuracy (Optimized)
0.967 (Holdout)
Algorithm
Decision Tree Classifier
Enhancements
None
Save as
Model viewer
Model information
Feature summary
Evaluation
Model evaluation
Confusion matrix
Precision recall
Precision recall
Precision recall curve
Precision
Recall
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

10. ADVANTAGES

1. This system can be used by many E-Commerce or other website in order to have a good customer relationship.
2. User can make online payment securely.
3. Data mining algorithm used in this system provides better performance as compared to other traditional classification algorithms.
4. Creates and increases user awareness against the phishing attacks.
5. The important information and credentials can be protected and saved before it's too late.

DISADVANTAGES

In spite of its advantages, there will always be cases where old-fashioned manual reviews will be preferable.

False positives: If a legitimate action is marked as fraud and you don't realize it, it will influence the whole system negatively. In that sense, a badly calibrated machine learning engine can create a negative loop where the more false positives aren't flagged, the less precise your results will be in the future.

No human understanding: If you're trying to get to the bottom of understanding why a user action is suspicious, it's hard to beat good old psychology.

Detection is not completely accurate: Though the phishing website is built with random forest boosting and bagging, there will always be certain websites that cannot be detected by the program.

11. CONCLUSION

It is remarkable that a good anti-phishing tool should be able to predict attacks in a reasonable amount of time. The main goal of this project was to build an effective model to detect phishing websites. A good anti-phishing tool must be available in a timely manner in order to increase the level of anticipatory awareness. Consistent retraining should be used to continuously improve this model. In fact, having access to clean, modern training data that we can obtain using our own equipment will enable us to continuously retrain our model and handle any changes to the features that are significant in determining the site class. Despite the fact that neural networks can handle a wide range of classification problems, finding the ideal structure is very challenging, and it is frequently determined through experimentation. Our model addresses this issue by computerising the procedure for organising a neural system; thus, if we build anadversary of phishing model and need to refresh it for any reason, our model will encourage this procedure, that is, because our model will mechanise the organising procedure and will request few client-defined parameters.

12. SOURCE CODE

Index.html

```
<html>

  <head>

    <title>Web Phishing</title>

    <link rel="stylesheet" href="../static/css/style.css">

    <script>

      function myFunction()

      {
```

```
        window.location.href="final";
      }
    </script>


</head>
<body>
    <div class="banner">
      <div class="navbar">
        <h3 class="logo">Web Phishing</h3>
        <ul>
          <li><a href="#">Home</a></li>
          <li><a href="#">About</a></li>
          <li><a href="#">Contact</a></li>
        </ul>
      </div>
      <div class="content">
        <h1 style="font-size:50px;">WEB PHISHING PREDICTION</h1>
        <p>Welcome! Our Website uses ML model to mimic how a person would
          look at, understand, and draw a verdict on a suspicious website.
          <br>Our engine learns from high quality, proprietary datasets containing
           millions of  samples for high accuracy detection.
           <br>We detect whether the website is legit or not in seconds, not days
           or weeks like other blocklist-based phishing protection softwares</p>
```

```
        <h1 style="font-size:50px;">ABOUT</h1>

        <P>Web service is one of the key communications software services for the
internet.

        Web phishing is one of many security threats to we services on the internet.<br>

        Web phishing aims to steal private information such os usernames, passwords, and
credit card details,

        by way of impersonating a legutumate entity.<br> The recipient is then tricked
information clicking

        malicious link, which can lead to the installation of malware,<br> the freezing of the
system as part of

        a ransonware attack or the revealing of the sensitive information. It will lead to
information disclosure and property damage.

        </P><br>

        <h3> CHECK YOUR WEBSITE</h3>

        <P>Understanding if the website is a valid one or not is important and plays a vital
role in securing the data.<br>

            To know if the URL is a valid one or your information is at risk check your website.

        </P>

        <div>

            <button type="button" onclick="myFunction()"><span></span>Predict Your
Website</button>

        </div>

      </div>

    </div>

  </body>

</html>
```

Final.html

```
<html>
  <head>
    <title>Check Your Website</title>
    <link rel="stylesheet" href="../static/css/style.css">
  </head>
  <body>
    <div class="predict-page">
      <h1>Check Your Website</h1>
      <form action="{{ url_for('y_predict') }}" method="post">
        <p>Enter URL</p>
        <input type="text" name="URL" placeholder="URL">
        <a href= {{ url }} >
        <button style="right: -20%; top: 10px;" type="submit"><span></span> Predict
</button>
      </a>
      </form>
      <div style="text-align: center ;">
        <div id='result', style="color: green;padding-top: 2rem;font-size: 2.2rem;" font-
size:30px;>{{ prediction_text }}</div>
          <a href=" {{ url }} "> {{ url }} </a>
        </div>
```

```
      </div>
   </body>
</html>
```

App.py

```
from flask import Flask, render_template,request,jsonify
import pickle
import inputScript
import numpy as np
from pathlib import Path


HERE = Path(__file__).parent




app = Flask(__name__)
loaded_model = pickle.load(open(HERE/"Phishing_2Website.pkl", 'rb'))
@app.route('/')
def home():
   return render_template('index.html')


@app.route('/final')
def  final():
```

```
    return render_template("final.html")


@app.route('/y_predict',methods=['POST'])
def y_predict():



    url = request.form['URL']
    checkprediction = inputScript.main(url)
    prediction =loaded_model.predict(checkprediction)
    print(prediction)
    output=prediction[0]
    if(output==1):
        pred="Your are safe!!  This is a Legitimate Website."
    else:
        pred="Your are on the wrong site. Be cautious!"
    return render_template('final.html', prediction_text='{}'.format(pred),url=url)


@app.route('/predict_api',methods=['POST'])
def predict_api():
    '''
    For direct API calls trought request
    '''
    data = request.get_json(force=True)
```

```
    prediction = loaded_model.y_predict([np.array(list(data.values()))])


    output = prediction[0]
    return jsonify(output)
if __name__ == '__main__':
   app.run()


   '''
import numpy as np
from flask import Flask,request,jsonify,render_template
import pickle
#importing the inputScript file used to analyze the URL
import inputScript
#load model
app = Flask(__name__)
model=pickle.load(open('Phishing_Websites.pkl','rb'))
#fetches the URL given by the URL and passes to inputScript
#Takes the input parameters fetched from the URL by inputsScript and returns the
if __name__ =='__main__':
    app.run(host='0.0.0.0', debug=True)
'''
```

Getting the pickle file

```
!pip install pandas

!pip install numpy

!pip install scikit-learn


# importing the libraries

import pandas as pd

import numpy as np

from sklearn.preprocessing import MinMaxScaler

from sklearn.metrics import confusion_matrix,accuracy_score


#reading the dataset

ds = pd.read_csv("../Sprint 1/dataset_website.csv")


ds.head()


#checking for Null values in a dataset and handling if any

ds.isnull().any()


ds.info()


#splitting data into independent and dependent variables

x,y = ds.iloc[:,1:31].values,ds.iloc[:,-1].values

print(x,y)
```

```
#splitting data into test set and train set
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)

# Creating a logistic regression model
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(x_train,y_train)

# checking the metrics of the model
ypred=lr.predict(x_test)
from sklearn.metrics import accuracy_score
log_reg=accuracy_score(y_test,ypred)
print(ypred,log_reg)

# Creating a K Nearest Neighbour model
from sklearn.neighbors import KNeighborsClassifier
knl=KNeighborsClassifier(n_neighbors=5,metric='minkowski',p=2)

knl.fit(x_train,y_train)

y_pred2=knl.predict(x_test)
```

```
from sklearn.metrics import accuracy_score

knn_euc=accuracy_score(y_test,y_pred2)

print(knn_euc)


# Creating a KNN-manhattan distance model

kn2=KNeighborsClassifier(n_neighbors=5,metric='minkowski',p=1)

kn2.fit(x_train,y_train)


y_pred3=kn2.predict(x_test)

from sklearn.metrics import accuracy_score

knn_man=accuracy_score(y_test,y_pred3)

print(knn_man)


# Creating a support vector machine - sigmoid model

from sklearn.svm import SVC

svm=SVC(kernel='sigmoid')

svm.fit(x_train,y_train)


# Creating a Decision Tree model

from sklearn.tree import DecisionTreeClassifier

dt=DecisionTreeClassifier()

dt.fit(x_train,y_train)
```

```
y_pred5=dt.predict(x_test)

from sklearn.metrics import accuracy_score

dec_tree=accuracy_score(y_test,y_pred5)

print(dec_tree)


# Creating a Random Forest model

from sklearn.ensemble import RandomForestRegressor

Rf=RandomForestRegressor(n_estimators=10,random_state=0,n_jobs=-1)

Rf.fit(x_train,y_train)


y_pred6=Rf.predict(x_test)

from sklearn.metrics import accuracy_score

rf=accuracy_score(y_test,y_pred6.round())

print(rf)


# Creating a Naive-Bayes model

from sklearn.naive_bayes import GaussianNB

gb=GaussianNB()

gb.fit(x_train,y_train)


y_pred7=gb.predict(x_test)

from sklearn.metrics import accuracy_score

nb=accuracy_score(y_test,y_pred7)
```

```
print(nb)


import pickle

pickle.dump(lr,open('Phishing_Website.pkl','wb'))


pickle.dump(dt,open('Phishing_Website_with_dt.pkl','wb'))
```

InputScript.py

```
import regex

from tldextract import extract

import ssl

import socket

from bs4 import BeautifulSoup

import urllib.request

import whois

import datetime



def url_having_ip(url):

#using regular function

  symbol = regex.findall(r'(http((s)?)://)((((\d)+).)*)((\w)+)(/((\w)+))?',url)

  if(len(symbol)!=0):
```

```
        having_ip = 1 #phishing
    else:
        having_ip = -1 #legitimate
    return(having_ip)
    # return 0



def url_length(url):
    length=len(url)
    if(length<54):
        return -1
    elif(54<=length<=75):
        return -1
    else:
        return 1



def url_short(url):
    #ongoing
    return 0


def having_at_symbol(url):
    symbol=regex.findall(r'@',url)
```

```
    if(len(symbol)==0):
        return -1
    else:
        return 1

def doubleSlash(url):
    #ongoing
    return 0

def prefix_suffix(url):
    subDomain, domain, suffix = extract(url)
    if(domain.count('-')):
        return -1
    else:
        return 1

def sub_domain(url):
    subDomain, domain, suffix = extract(url)
    if(subDomain.count('.')==0):
        return -1
    elif(subDomain.count('.')==1):
        return 0
    else:
```

```
        return 1


def SSLfinal_State(url):
    try:
#check wheather contains https
        if(regex.search('^https',url)):
            usehttps = 1
        else:
            usehttps = 0
#getting the certificate issuer to later compare with trusted issuer
        #getting host name
        subDomain, domain, suffix = extract(url)
        host_name = domain + "." + suffix
        context = ssl.create_default_context()
        sct = context.wrap_socket(socket.socket(), server_hostname = host_name)
        sct.connect((host_name, 443))
        certificate = sct.getpeercert()
        issuer = dict(x[0] for x in certificate['issuer'])
        certificate_Auth = str(issuer['commonName'])
        certificate_Auth = certificate_Auth.split()
        if(certificate_Auth[0] == "Network" or certificate_Auth == "Deutsche"):
            certificate_Auth = certificate_Auth[0] + " " + certificate_Auth[1]
        else:
```

```
            certificate_Auth = certificate_Auth[0]

        trusted_Auth =
['Comodo','Symantec','GoDaddy','GlobalSign','DigiCert','StartCom','Entrust','Verizon','Trustw
ave','Unizeto','Buypass','QuoVadis','Deutsche Telekom','Network
Solutions','SwissSign','IdenTrust','Secom','TWCA','GeoTrust','Thawte','Doster','VeriSign']

#getting age of certificate

        startingDate = str(certificate['notBefore'])

        endingDate = str(certificate['notAfter'])

        startingYear = int(startingDate.split()[3])

        endingYear = int(endingDate.split()[3])

        Age_of_certificate = endingYear-startingYear


#checking final conditions

        if((usehttps==1) and (certificate_Auth in trusted_Auth) and (Age_of_certificate>=1) ):

            return -1 #legitimate

        elif((usehttps==1) and (certificate_Auth not in trusted_Auth)):

            return 0 #suspicious

        else:

            return 1 #phishing


    except Exception as e:


        return 1
```

```
def domain_registration(url):
    try:
        w = whois.whois(url)
        updated = w.updated_date
        exp = w.expiration_date
        length = (exp[0]-updated[0]).days
        if(length<=365):
            return 1
        else:
            return -1
    except:
        return 0


def favicon(url):
    #ongoing
    return -1


def port(url):
    #ongoing
    return 0


def https_token(url):
    subDomain, domain, suffix = extract(url)
```

```
    host =subDomain +'.' + domain + '.' + suffix
    if(host.count('https')): #attacker can trick by putting https in domain part
        return 1
    else:
        return -1


def request_url(url):
    try:
        subDomain, domain, suffix = extract(url)
        websiteDomain = domain


        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        imgs = soup.findAll('img', src=True)
        total = len(imgs)


        linked_to_same = 0
        avg =0
        for image in imgs:
            subDomain, domain, suffix = extract(image['src'])
            imageDomain = domain
            if(websiteDomain==imageDomain or imageDomain==''):
                linked_to_same = linked_to_same + 1
```

```
        vids = soup.findAll('video', src=True)
        total = total + len(vids)

        for video in vids:
            subDomain, domain, suffix = extract(video['src'])
            vidDomain = domain
            if(websiteDomain==vidDomain or vidDomain==''):
                linked_to_same = linked_to_same + 1
        linked_outside = total-linked_to_same
        if(total!=0):
            avg = linked_outside/total

        if(avg<0.22):
            return -1
        elif(0.22<=avg<=0.61):
            return 0
        else:
            return 1
    except:
        return 0


def url_of_anchor(url):
```

```
try:
    subDomain, domain, suffix = extract(url)
    websiteDomain = domain


    opener = urllib.request.urlopen(url).read()
    soup = BeautifulSoup(opener, 'lxml')
    anchors = soup.findAll('a', href=True)
    total = len(anchors)
    linked_to_same = 0
    avg = 0
    for anchor in anchors:
        subDomain, domain, suffix = extract(anchor['href'])
        anchorDomain = domain
        if(websiteDomain==anchorDomain or anchorDomain==''):
            linked_to_same = linked_to_same + 1
    linked_outside = total-linked_to_same
    if(total!=0):
        avg = linked_outside/total


    if(avg<0.31):
        return -1
    elif(0.31<=avg<=0.67):
        return 0
```

```
        else:
            return 1
    except:
        return 0


def Links_in_tags(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')


        no_of_meta =0
        no_of_link =0
        no_of_script =0
        anchors=0
        avg =0
        for meta in soup.find_all('meta'):
            no_of_meta = no_of_meta+1
        for link in soup.find_all('link'):
            no_of_link = no_of_link +1
        for script in soup.find_all('script'):
            no_of_script = no_of_script+1
        for anchor in soup.find_all('a'):
            anchors = anchors+1
```

```
        total = no_of_meta + no_of_link + no_of_script+anchors
        tags = no_of_meta + no_of_link + no_of_script
        if(total!=0):
            avg = tags/total

        if(avg<0.25):
            return -1
        elif(0.25<=avg<=0.81):
            return 0
        else:
            return 1
    except:
        return 0

def sfh(url):
    #ongoing
    return 0

def email_submit(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        if(soup.find('mailto:')):
```

```
            return 1
        else:
            return -1
    except:
        return 0


def abnormal_url(url):
    #ongoing
    return 1


def redirect(url):
    #ongoing
    return 0


def on_mouseover(url):
    #ongoing
    return -1


def rightClick(url):
    #ongoing
    return -1


def popup(url):
```

```
    #ongoing
    return 1


def iframe(url):
    #ongoing
    return 0


def age_of_domain(url):
    try:
        w = whois.whois(url)
        start_date = w.creation_date
        current_date = datetime.datetime.now()
        age =(current_date-start_date[0]).days
        if(age>=180):
            return -1
        else:
            return 1
    except Exception as e:
        print(e)
        return 0


def dns(url):
    #ongoing
```

```
    return -1


def web_traffic(url):
    #ongoing
    return 0


def page_rank(url):
    #ongoing
    return 0


def google_index(url):
    #ongoing
    return -1



def links_pointing(url):
    #ongoing
    return 0


def statistical(url):
    #ongoing
    return 0
```

```
def main(url):




    check = [[url_having_ip(url),url_length(url),url_short(url),having_at_symbol(url),
            doubleSlash(url),prefix_suffix(url),sub_domain(url),SSLfinal_State(url),
             domain_registration(url),favicon(url),port(url),https_token(url),request_url(url),
             url_of_anchor(url),Links_in_tags(url),sfh(url),email_submit(url),abnormal_url(url),
             redirect(url),on_mouseover(url),rightClick(url),popup(url),iframe(url),
             age_of_domain(url),dns(url),web_traffic(url),page_rank(url),google_index(url),
             links_pointing(url),statistical(url)]]



    print(check)
    return check
```

TEAM GITHUB LINK :

https://github.com/IBM-EPBL/IBM-Project-1639-1658406552

PROJECT DEMO LINK

https://www.dropbox.com/s/sbuy6pmb66moqom/Web%20Phishing%20TeamID-PNT2022TMID35411.mov?dl=0