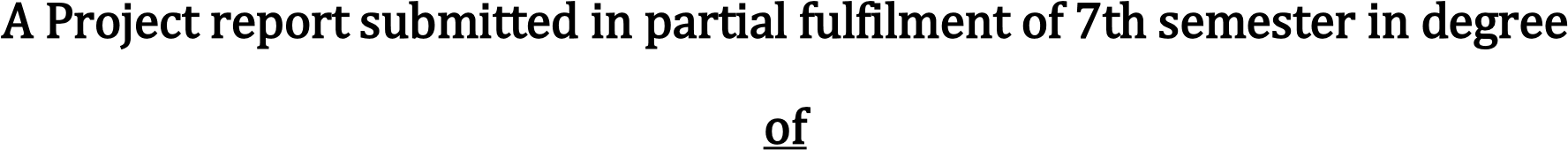
**WEB PHISHING DETECTION**

**Using Machine Learning**



# BACHELOR OF ENGINEERING

## IN

**COMPUTER SCIENCE AND ENGINEERING**

Submitted by

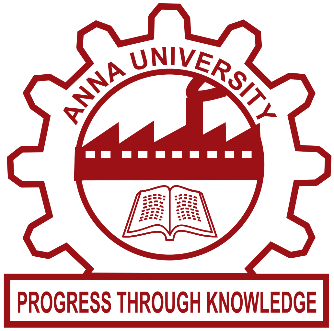
**Team ID: PNT2022TMID22731**

**Team Leader :** Soniya.D (732919CSR104)

**Team Member** : Riya K Mathew (732919CSR085)

**Team Member** : Nivedha.E (732919CSR071)

**Team Member** : Srivarshan.S.N(732919CSR108)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**VELALAR COLLEGE OF ENGINEERING AND**

**TECHNOLOGY**

**ERODE-638012**

## BONAFIDE CERTIFICATE

**Certified that this project report ”WEB PHISHING DETECTION” is the bonafide record work done by SONIYA.D(**(732919CSR104),**RIYA K MATHEW**(732919CSR085),**NIVEDHA.E**(732919CSR071),**SRIVARSHAN.S.N**(732919CSR108) **for IBM-NALAIYATHIRAN in VII semester of B.E., degree course in Computer Science and Engineering branch during the academicyearof2022 - 2023.**

**Staff-In charge Industry mentor**

**MS VISHALAKSHI.R MR SANDESH.P**

**Head of the Department**

**Dr.S. JABEEN BEGUM M.E, Ph.D**

## ACKNOWLEDGEMENT

We are highly grateful to thank our Project coordinator **MS VISHALAKSHI.R** Department of Computer Science and Engineering,

Velalar college of engineering and technology, for the coordinating us throughout this Project. We are very much indebted to thank all the faculty members of Department of Computer Science and Engineering in our Institute, for their excellent moral support and suggestions to complete our Project work successfully.

**SONIYA .D**(732919CSR104)

**RIYA K MATHEW**(732919CSR085)

**NIVEDHA.E**(732919CSR071)

**SRIVARSHAN.S.N**(732919CSR108)

## Project Report Format

1. **INTRODUCTION** 
   1. Project Overview
   2. Purpose
2. **LITERATURE SURVEY**
   1. Existing problem
   2. References
   3. Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
   1. Empathy Map Canvas
   2. Ideation & Brainstorming
   3. Proposed Solution
   4. Problem Solution fit
4. **REQUIREMENT ANALYSIS**
   1. Functional requirement
   2. Non-Functional requirements
5. **PROJECT DESIGN** 
   1. Data Flow Diagrams
   2. Solution & Technical Architecture
   3. User Stories
6. **PROJECT PLANNING & SCHEDULING** 
   1. Sprint Planning & Estimation
   2. Sprint Delivery Schedule
   3. Reports from JIRA
7. **CODING & SOLUTIONING (Explain the features added in the project along with code)** 
   1. Feature 1
   2. Feature 2
   3. Database Schema (if Applicable)
8. **TESTING** 
   1. Test Cases
   2. User Acceptance Testing
9. **RESULTS** 
   1. Performance Metrics
10. **ADVANTAGES & DISADVANTAGES**
11. **CONCLUSION**
12. **FUTURE SCOPE**
13. **APPENDIX**

Source Code

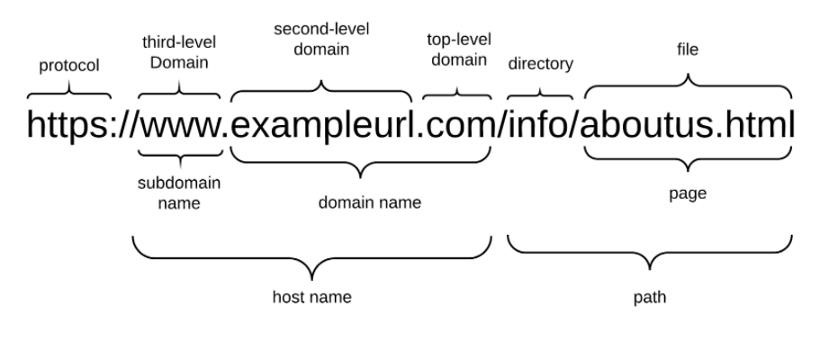
GitHub & Project Demo Link

# 1.INTRODUCTION

Phishing is popular among attackers, since it is easier to trick someone into clicking a malicious link which seems legitimate than trying to break through a computer’s defense systems. The malicious links within the body of the message are designed to make it appear that they go to the spoofed organization using that organization’s logos and other legitimate contents.

In this article I explain: phishing domain (Applied data science with python) characteristics, the features that distinguish them from legitimate domains, why it is important to detect these domains, and how they can be detected using machine learning

Lets check the URL structure for the clear understanding of how attackers think when they create a phishing domain.



## 1.1.Project Overview

**Web Phishing Detection:**

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.

 Large organizations may get trapped in different kinds of scams.

**This Guided Project mainly focuses on applying a machine-learning algorithm to detect Phishing websites.**

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.

**OBJECTIVES :**

The objective of phishing website URLs is to purloin the personal information like user name, passwords and online banking transactions. Phishers use the websites which are visually and semantically similar to those real websites. As technology continues to grow, phishing techniques started to progress rapidly and this needs to be prevented by using antiphishing mechanisms to detect phishing. Machine learning is a powerful tool used to strive against phishing attacks. This paper surveys the features used for detection and detection techniques using machine learning. Phishing becomes a main area of concern for security researchers because it is not difficult to create the fake website which looks so close to legitimate website. Experts can identify fake websites but not all the users can identify the fake website and such users become the victim of phishing attack.

Main aim of the attacker is to steal banks account credentials. Phishing attacks are becoming successful because lack of user awareness. Since phishing attack exploits the weaknesses found in users, it is very difficult to mitigate them but it is very important to enhance phishing detection techniques.

Phishing may be a style of broad extortion that happens once a pernicious web site act sort of a real one memory that the last word objective to accumulate unstable info, as an example, passwords, account focal points, or MasterCard numbers. all the same, the means that there square measure some of contrary to phishing programming

**ALGORITHMS USED :**

Two algorithms have been implemented to check whether a URL is legitimate or fraudulent. Random forest algorithm creates the forest with number of decision trees. High number of tree gives high detection accuracy. Creation of trees is based on bootstrap method. In bootstrap method features and samples of dataset are randomly selected with replacement to construct single tree. Decision tree begins its work by choosing best splitters from the available attributes for classification which is considered as a root of the tree. Algorithm continues to build tree until it finds the leaf node. Decision tree creates training model which is used to predict target value or class in tree representation each internal node of the tree belongs to attribute and each leaf node of the tree belongs to class label.

**The accuracy of the model :**

Research demonstrates that current phishing detection technologies have an accuracy rate between 70% and 92.52%. The experimental results prove that the accuracy rate of our proposed model can yield up to 95%, which is higher than the current technologies for phishing website detection.

**The Flask framework :**

The advantage of web applications is that they’re platform independent and can be run by anyone who has access to the Internet. Their code is implemented on a back-end server, where the program processes incoming requests and responds through a shared protocol that’s understood by all browsers. Python powers many large web applications and is a common choice as a back-end language. Many Python-driven web applications are planned from the start as web applications and are built using Python web frameworks.

**Project Flow**

**Find below the project flow to be followed while developing the project.**

* Download the dataset.
* Preprocess or clean the data.
* Analyze the pre-processed data.
* Train the machine with preprocessed data using an appropriate machine learning algorithm.
* Save the model and its dependencies.
* Build a Web application using a flask that integrates with the model built.

# Pre-Requisites

**In order to develop this project we need to install the following software/packages:**

**Step 1:**

**Anaconda Navigator :**

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS. Conda is an open-source, crossplatform, package management system. Anaconda comes with great tools like JupyterLab, Jupyter Notebook, QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code.

# Application Building

## Building an Application to integrate the model

After the model is built, we will be integrating it to a web application so that normal users can also use it to know if any website is phishing or safe in a no-code manner.

In the application, the user provides any website URL to check and the corresponding parameter values are generated by analysing the URL using which legitimate websites are detected.

**Features Used for Phishing Domain Detection**

There are a lot of algorithms and a wide variety of data types for phishing detection in the academic literature and commercial products. A phishing URL and the corresponding page have several features which can be differentiated from a malicious URL.

For example; an attacker can register long and confusing domain to hide the actual domain name **(**Cybersquatting, Typosquatting). In some cases attackers can use direct IP addresses instead of using the domain name. This type of event is out of our scope, but it can be used for the same purpose. Attackers can also use short domain names which are irrelevant to legitimate brand names and don’t have any FreeUrl addition. But these type of web sites are also out of our scope, because they are more relevant to fraudulent domains instead of phishing domains.

Beside URL-Based Features, different kinds of features which are used in machine learning algorithms in the detection process of academic studies are used. Features collected from academic studies for the phishing domain detection with machine learning techniques are grouped as given below.

1. URL-Based Features
2. Domain-Based Features
3. Page-Based Features
4. Content-Based Features

## URL-Based Features

URL is the first thing to analyse a website to decide whether it is a phishing or not. As we mentioned before, URLs of phishing domains have some distinctive points. Features which are related to these points are obtained when the URL is processed.

## 1.2.Purpose

Phishing is a major problem, which uses both social engineering and technical deception to get users' important information such as financial data, emails, and other private information. Phishing exploits human vulnerabilities; therefore, most protection protocols cannot prevent the whole phishing attacks

Our approach extracts and analyzes different features of suspected webpages for effective identification of large-scale phishing offenses. The main contribution of this paper is the combined uses of these feature set. For improving the detection accuracy of phishing webpages, we have proposed eight new features. Our proposed features determine the relationship between the URL of the webpage and the webpage content.

## 2.Literature Survey

**Team ID : PNT2022TMID22731**

**Project Name: Web phishing Detection**

**College Name : Velalar college of engineering and technology**

**Department : Computer Science And Engineering**

**Team Leader :** Soniya D

**Team Member** : Riya K Mathew

**Team Member** : Nivedha E **Team Member** : Srivarshan S N Phishing Websites Features

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | Rami M.Mohammad School of Computing and  Engineering University of Huddersfield  Huddersfield, UK. | |  | |  | | --- | | Lee McCluskey  School of Computing and Engineering University of Huddersfield  Huddersfield, UK. | |

**1. Phishing Websites Features:**

One of the challenges faced by our research was the unavailability of reliable training datasets. In fact, this challenge faces any researcher in the field. However, although plenty of articles about predicting phishing websites using data mining techniques have been disseminated these days, no reliable training dataset has been published publically, maybe because there is no agreement in literature on the definitive features that characterize phishing websites, hence it is difficult to shape a dataset that covers all possible features.

In this article, we shed light on the important features that have proved to be sound and effective in predicting phishing websites. In addition, we proposed some new features, experimentally assigned new rules to some well-known features and updated some other features.

**2. Address Bar based Features:**

**Using the IP Address**

If an IP address is used as an alternative of the domain name in the URL, such as

[“http://125.98.3.123/fake.html”](http://125.98.3.123/fake.html), users can be sure that someone is trying to steal their personal information. Sometimes, the IP address is even transformed into hexadecimal code as shown in the following link [“http://0x58.0xCC.0xCA.0x62/2/paypal.ca/index.html”](http://88.204.202.98/2/paypal.ca/index.html).

*Rule*: IF{ Url ip Address>>>phishing otherwise>>>legitime

**Long URL to Hide the Suspicious Part:**

Phishers can use long URL to hide the doubtful part in the address bar.

For example:

[http://federmacedoadv.com.br/3f/aze/ab51e2e319e51502f416dbe46b773a5e/?cmd=\_home&amp;dispatch =11004d58f5b74f8dc1e7c2e8dd4105e811004d58f5b74f8dc1e7c2e8dd4105e8@](http://federmacedoadv.com.br/3f/aze/ab51e2e319e51502f416dbe46b773a5e/?cmd=_home&amp%3Bdispatch=11004d58f5b74f8dc1e7c2e8dd4105e811004d58f5b74f8dc1e7c2e8dd4105e8)phishing.website.html

To ensure accuracy of our study, we calculated the length of URLs in the dataset and produced an average URL length. The results showed that if the length of the URL is greater than or equal 54 characters then the URL classified as phishing. By reviewing our dataset we were able to find 1220 URLs lengths equals to 54 or more which constitute 48.8% of the total dataset size.

**Redirecting using “//”**

The existence of “//” within the URL path means that the user will be redirected to another website. An example of such URL’s is:

“http://www.legitimate.com//[http://www.phishing.com”](http://www.phishing.com/). We examine the location where the “//” appears. We find that if the URL starts with “HTTP”, that means the “//” should appear in the sixth position. However, if the URL employs “HTTPS” then the “//” should appear in seventh position. Rule: IF

**Adding Prefix or Suffix Separated by (-) to the Domain**

The dash symbol is rarely used in legitimate URLs. Phishers tend to add prefixes or suffixes separated by (-) to the domain name so that users feel that they are dealing with a legitimate webpage. For example [http://www.Confirme-paypal.com](http://www.confirme-paypal.com/)

**Sub Domain and Multi Sub Domains**

Let us assume we have the following link: [http://www.hud.ac.uk/students/.](http://www.hud.ac.uk/students/) A domain name might include the country-code top-level domains (ccTLD), which in our example is

“uk”. The “ac” part is shorthand for “academic”, the combined “ac.uk” is called a secondlevel domain (SLD) and “had” is the actual name of the domain. To produce a rule for extracting this feature, we firstly have to omit the (www.) from the URL which is in fact a sub domain in itself. Then, we have to remove the (ccTLD) if it exists. Finally, we count the remaining dots. If the number of dots is greater than one, then the URL is classified as

“Suspicious” since it has one sub domain. However, if the dots are greater than two, it is classified as “Phishing” since it will have multiple sub domains. Otherwise, if the URL has no sub domains, we will assign “Legitimate” to the feature.

**HTTPS (Hyper Text Transfer Protocol with Secure Sockets**

**Layer)**

The existence of HTTPS is very important in giving the impression of website legitimacy, but this is clearly not enough. The authors in (Mohammad, Thabtah and McCluskey 2012)(Mohammad, Thabtah and McCluskey 2013) suggest checking the certificate assigned with

HTTPS including the extent of the trust certificate issuer, and the certificate age. Certificate

Authorities that are consistently listed among the top trustworthy names include: “GeoTrust, [GoDaddy,](http://www.godaddy.com/gdshop/ssl/ssl.asp?isc=BESTSSL1) Network Solutions, Thawte, Como- do, Doster and VeriSign”. Furthermore, by testing out our datasets, we find that the minimum age of a reputable certificate is two years.

**Domain Registration Length**

Based on the fact that a phishing website lives for a short period of time, we believe that trustworthy domains are regularly paid for several years in advance. In our dataset, we find that the longest fraudulent domains have been used for one year only.

### Table 1 Common ports to be checked

|  |  |  |  |
| --- | --- | --- | --- |
| PORT | Service | Meaning | Preferred Status |
| 21 | FTP | Transfer files from one host to another | Close |
| 22 | SSH | Secure File Transfer Protocol | Close |
| 23 | Telnet | provide bidirectional interactive text-oriented communication | Close |
| 80 | HTTP | Hyper test transfer protocol | Open |
| 443 | HTTPS | Hypertext transfer protocol secured | Open |
| 445 | SMB | Providing shared access to files, printers, serial ports | Close |
| 1433 | MSSQL | Store and retrieve data as requested by other software applications | Close |
| 1521 | ORACLE | Access oracle database from web. | Close |
| 3306 | MySQL | Access MySQL database from web. | Close |
| 3389 | Remote Desktop | allow remote access and remote collaboration | Close |

# Abstract

The detection of phishing attacks. Phishing attacks target vulnerabilities that exist in systems due to the human factor. Many cyber attacks are spread via mechanisms that exploit weaknesses found in end-users, which makes users the weakest element in the security chain. The phishing problem is broad and no single silver-bullet solution exists to mitigate all the vulnerabilities effectively, thus multiple techniques are often implemented to mitigate specific attacks.

This paper surveys the features used for detection and detection techniques using machine learning. Phishing becomes a main area of concern for security researchers because it is not difficult to create the fake website which looks so close to legitimate website. Experts can identify fake websites but not all the users can identify the fake website and such users become the victim of phishing attack.

The Main aim of the attacker is to steal banks account credentials. Phishing attacks are becoming successful because of lack of user awareness. Since phishing attack exploits the weaknesses found in users, it is very difficult to mitigate them but it is very important to enhance phishing detection techniques.

Phishing maybe a style of broad extortion that happens once a pernicious web site acts sort of a real one memory that the last word objective to accumulate unstable info, as an example, passwords, account focal points, or MasterCard numbers. all the same, the means that there square measure some of contrary to phishing programming

**ALGORITHMS USED :**

Two algorithms have been implemented to check whether a URL is legitimate or fraudulen

Random forest algorithm creates the forest with number of decision trees. A High number of trees gives high detection accuracy. Creation of trees is based on the bootstrap method. In the bootstrap method, features and samples of dataset are randomly selected with replacement to construct a single tree

Decision tree begins its work by choosing the best splitters from the available attributes for classification which is considered as a root of the tree. The algorithm continues to build tree until it finds the leaf node. Decision tree creates training model which is used to predict target value or class in tree representation each internal node of the tree belongs to attribute and each leaf node of the tree belongs to class label.

**The accuracy of the model :**

Research demonstrates that current phishing detection technologies have an accuracy rate **between 70% and 92.52%**. The experimental results prove that the accuracy rate of our proposed model can yield up to 95%, which is higher than the current technologies for phishing website detection.

**Phishing Website Detection Using Machine Learning Algorithms:**

**PROPOSED WORK:**

URLs extracting and analyze Various links by checking with Back listing with the

help of Machine Learning to increase accuracy.

**TOOLS USED/ ALGORITHM:**

* Decision Tree Algorithm
* Random Forest Algorithm  Support Vector Machine Algorithm

**TECHNOLOGY:**

* Machine Learning

**ADVANTAGES/ DISADVANTAGES:**

The disadvantage is that the Characteristics are not guaranteed to always exist in such attacks and the false positive rate in detection is very high.

Advantage is 97.14% detection accuracy using random forest algorithm with lowest false positive rate

**Detecting phishing websites using machine learning technique:**

**PROPOSED WORK:**

URL-based anti-phishing machine learning and method URL Net, a CNNbased deep-neural URL detection network.

**TOOLS USED/ALGORITHM:**

* Support Vector Machine
* K-NN
* Random forest classification  Artificial Neural Network

**TECHNOLOGY:**

* Machine Learning

**ADVANTAGES/ DISADVANTAGES:**

Advantages-Reduces over fitting in decision trees and helps to improve the accuracy.

Disadvantages-Requires a computational power as well as resources as it builds numerous trees to combine their outputs.

**Phishing Website detection using machine learning and deep learning techniques**

**PROPOSED WORK:**

It discusses the machine learning and deep learning algorithms and apply all these algorithms on our dataset and the best algorithm having the best precision and accuracy is selected for the phishing website detection.

**TOOLS USED/ALGORITHM:**

* Regression Techniques
* K nearest neighbor
* Decision Tree
* Random Forest  XG Boost  AdaBoost.

**TECHNOLOGY:**

* Machine Learning  Deep Learning

**ADVANTAGES/ DISADVANTAGES:**

Advantage is to eliminate the cyber threat risk level. Increase user alertness to phishing risks.

Disadvantage is Negative effects on a business, including of money, loss of intellectual property

**Phishing Website Detection Based on URL: PROPOSED WORK:**

To preserve the confidentiality. develop a user-friendly environment and to prevent or mitigate harm or destruction of computer networks,applications, devices, and data.

**TOOLS USED/ALGORITHM:**

* Learning Model Algorithm
* Naive BayesAlgorithm
* Decision tree,
* Support Vector Machine
* Artificial Neural Network
* Sequential Minimal Optimization

**TECHNOLOGY:**

* Machine Learning

**ADVANTAGES/ DISADVANTAGES:**

Advantages–Provide clear idea about the effective level of each classifier on phishingemail detection.

Disadvantages-Non standard classifier

**Phishing Website Detection Based on Deep Convolution Neural Network and**

## Random Forest Ensemble Learning

**PROPOSED WORK:**

It proposes an integrated phishing website detection method based on convolution neural networks (CNN) and random forest (RF)

**TOOLS USED/ALGORITHM:**

* Linear Regression
* K nearest neighbor
* Support Vector Machine
* Random Forest
* XG Boost
* Naïve Bayes
* RNN Model  CNN Model

**TECHNOLOGY:**

* Machine Learning  Deep Learning

**ADVANTAGES/ DISADVANTAGES:**

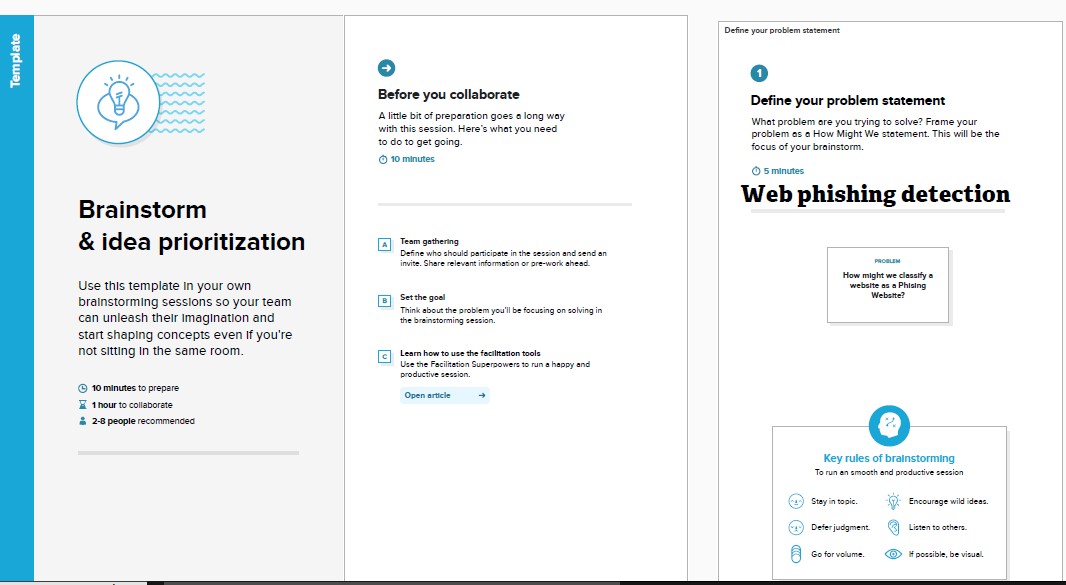
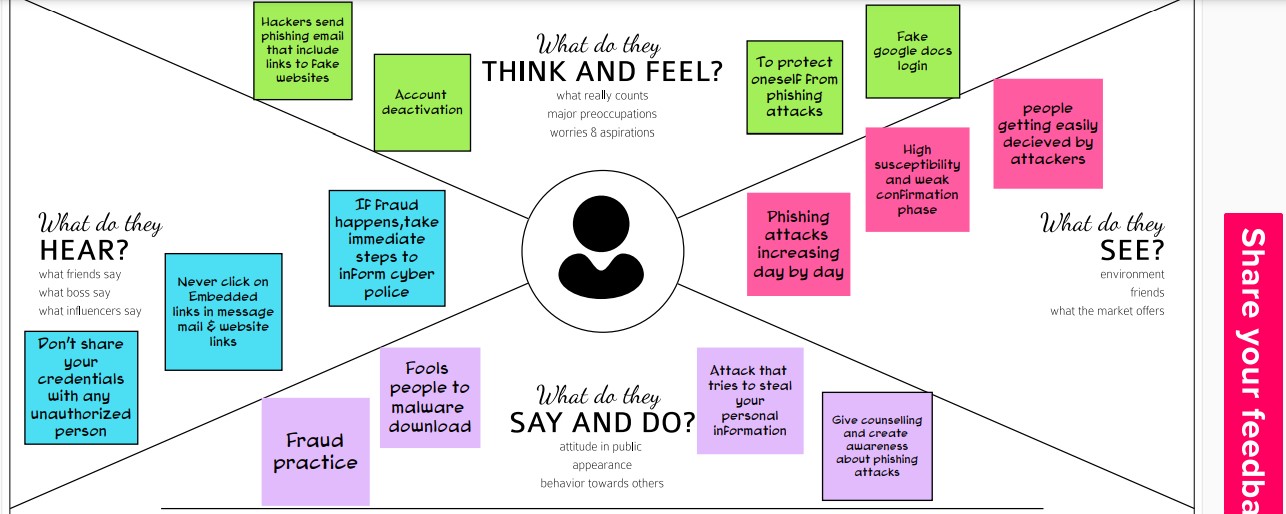
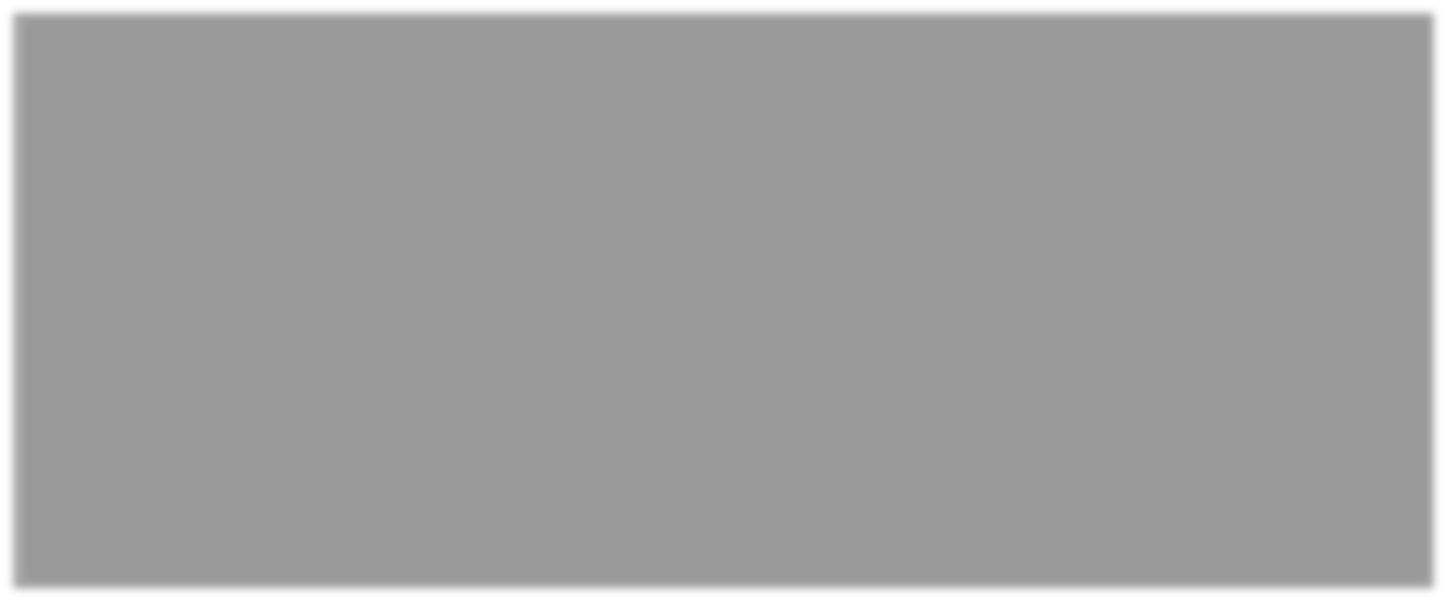
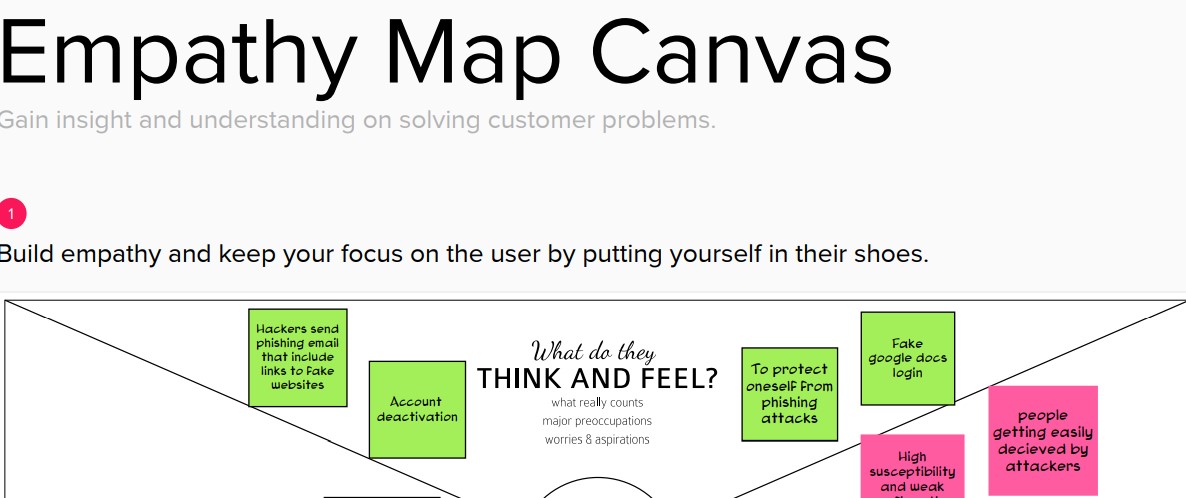
The disadvantage is that the model cannot determine whether the URL is active or not, so it is necessary to test whether the URL is active or not before detection. Advantage is that the third-party service is independent.

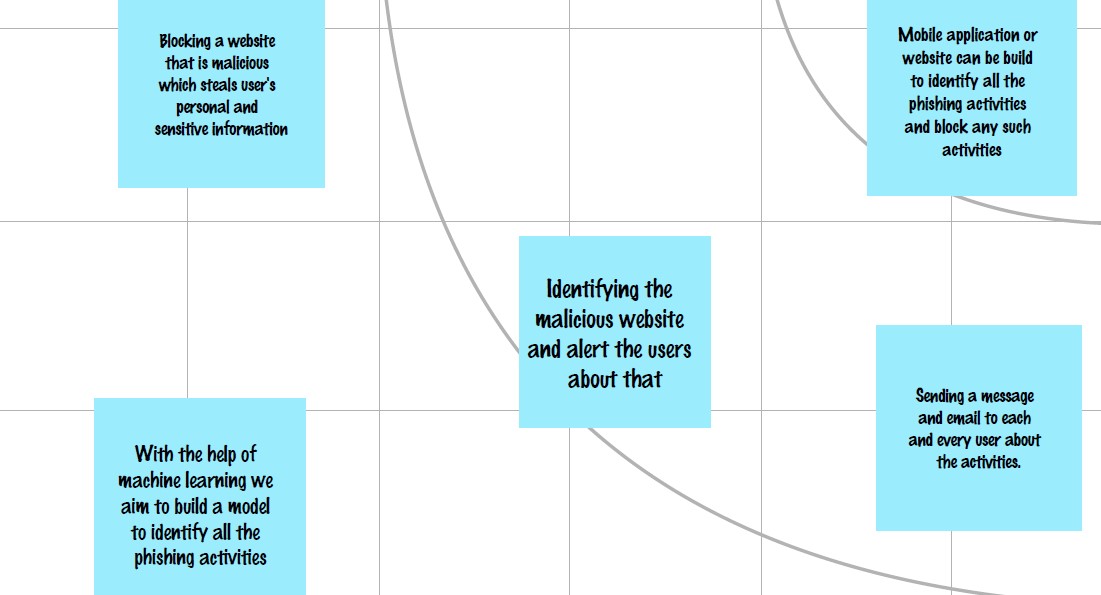
**3.IDEATION & PROPOSED SOLUTION**

**3.1**

**Empathy Map Canvas**

**:**





* 1. **Proposed Solution**

**Proposed Solution Template:**

Project team shall fill the following information in proposed solution template.

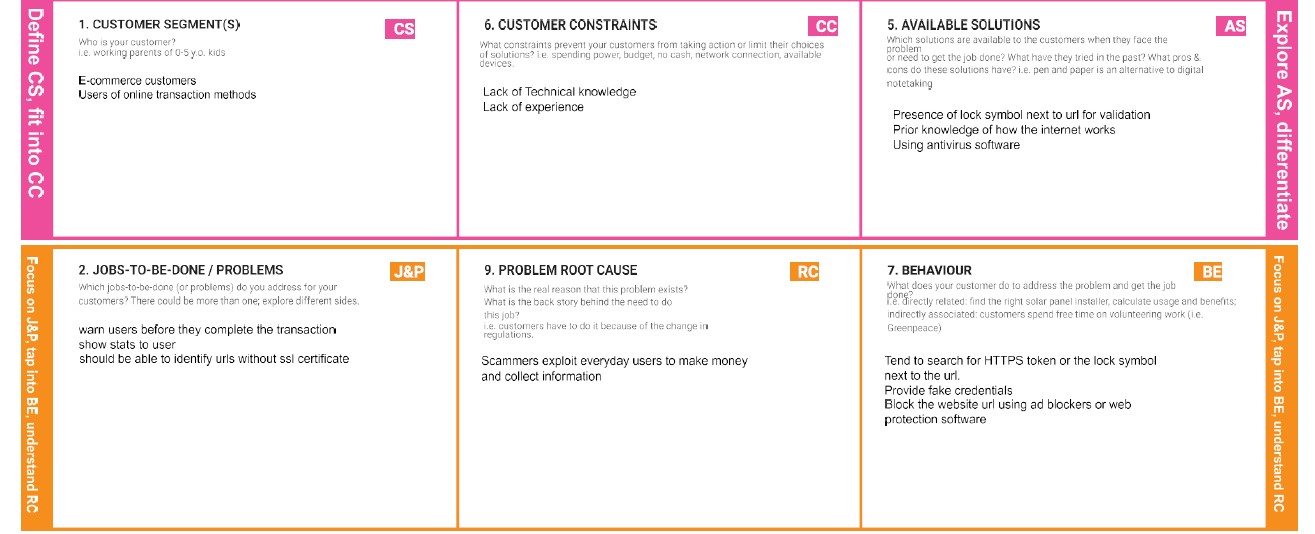
|  |  |  |
| --- | --- | --- |
| **S.No.** | **Parameter** | **Description** |
| 1. | **Problem Statement (Problem to besolved)** | Web phishing tends to steal a lots of information from the user during online transaction like username, password, important documents that has been attached to that websites. One of the many security risks to web services on the Internet is web phishing. There are Multiple Types of Attacks happens here every day, but there is no auto detection Process through Machine Learning is achieved |
| 2. | **Idea / Solution description** | Through ML and data mining techniques like classification algorithm user can able to attain a warning signal to notify these phishing websites which helps the user to safeguard their identities and their login credentials etc. python is the language that helps to enable these techniques for the online users Using the results obtained saf websites for online transactions are acquired which would then be made accessible to the user through an online application. |

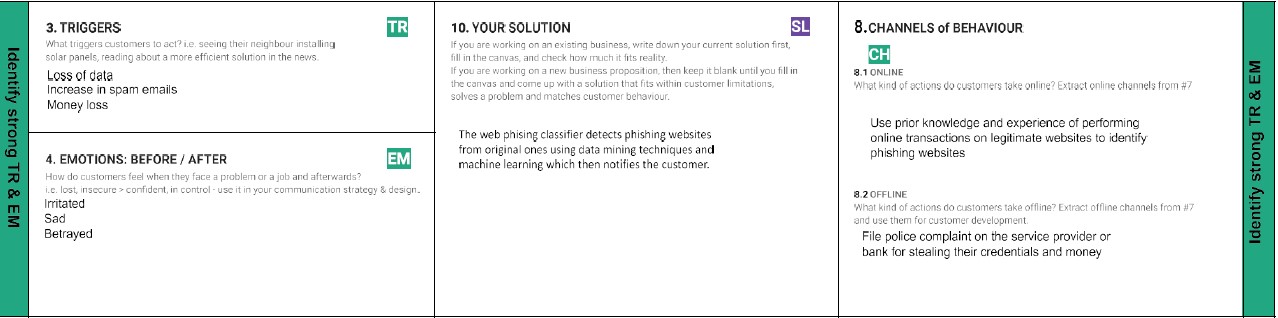
e s

|  |  |  |
| --- | --- | --- |
| 3. | **Novelty / Uniqueness** | This project not only able to identify the malicious websites it also has the ability to automatically block these kind of websites completely in the future when it has been identified and also blocks some various mails  /ads from these malicious websites |
| 4. | **Social Impact / Customer Satisfaction** | This web phishing detection project attains the customer satisfaction by discarding various kinds of malicious websites to protect their privacy. This project is not only capable of using by an single individual ,a large social community and a organization can use this web phishing detection to protect their privacy. This project helps to block various malicious websites simultaneously. |
| 5. | **Business Model (Revenue Model)** | This developed model can be used as an enterprise applications by organizations which handles sensitive information and also can be sold to government agencies to prevent the loss of potential important data.Based on membership levels, different levels of security strictness and multiple volumes of secure e-commerce websites would be offered. |

|  |  |  |
| --- | --- | --- |
| 6  . | **Scalability of the Solution** | This project’s performance rate will be high and it also provide many capabilities to the user without reducing its efficiency to detect the malicious websites. thus scalability of this project will be high .It could be further extended for other security concerns such as audio recording, video recording, location tracking, virus attacks and more.with safer and more effective solutions. |

* 1. **Problem Solution fit:**





1. **REQUIREMENT ANALYSIS:**

**Functional Requirements:**

Following are the functional requirements of the proposed solution.

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Functional Requirement (Epic)** | **Sub Requirement (Story / Sub-Task)** |
| FR-1 | User Registration | Registration through Form Registration through Gmail |
| FR-2 | User Confirmation | Confirmation via Email |
| FR-3 | Website identification | Model Detects the malicious website using blacklist And give alert message |
| FR-4 | Prediction | Model predicts the URL using Machine Learning algorithms such as Regression, KNN |
| FR-5 | Classifier | Model predicts all the output to classifier and produces the final result. |
| FR-6 | Results | Model predict the website and give pop- up to the user before they enter any confidential details |

**Non-functional Requirements:**

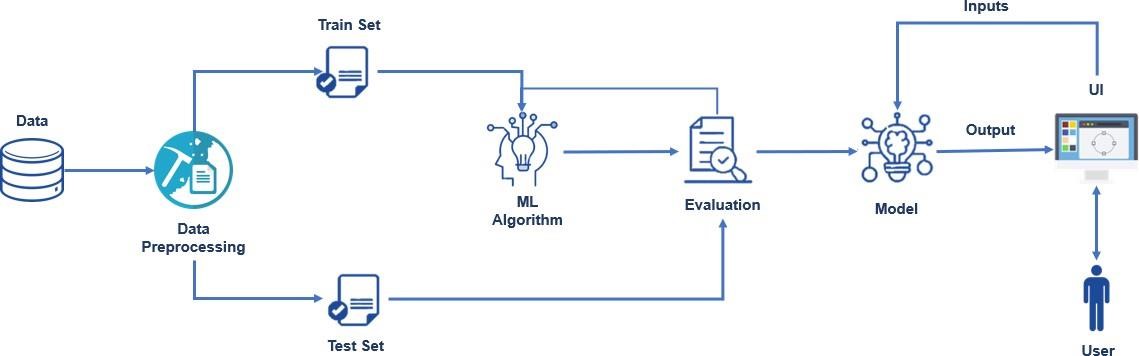
Following are the non-functional requirements of the proposed solution.

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Non-Functional Requirement** | **Description** |
| NFR-1 | **Usability** | User can access to several website easily using web phishing detection without loosing any data |
| NFR-2 | **Security** | User can check whether the websites are secure or not by getting pop-up message |
| NFR-3 | **Reliability** | User should feel it reliable while using any website |
| NFR-4 | **Performance** | The performance should be faster and user friendly for the effective performance |
| NFR-5 | **Availability** | The users should get availability to access the resources must be valid and reliable |
| NFR-6 | **Scalability** | The performance of the website should be efficient to handle the increasing user and loads without any disturbance |

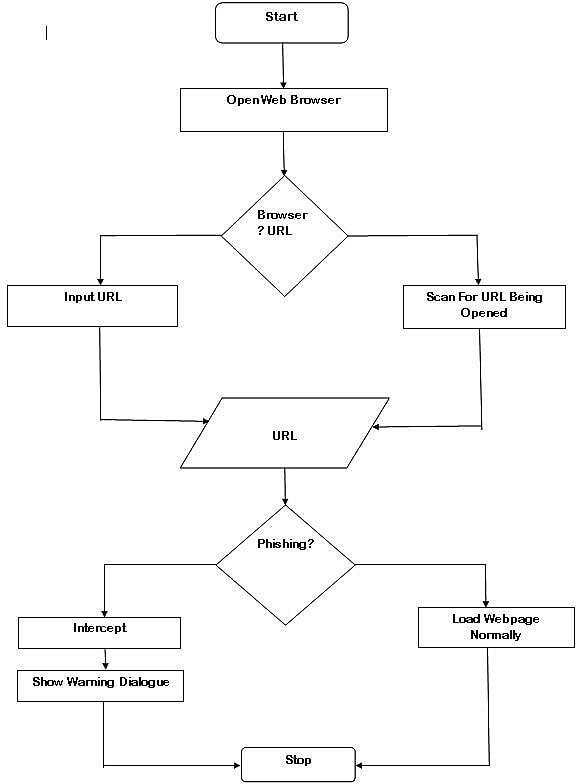
1. **PROJECT DESIGN:**

**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.



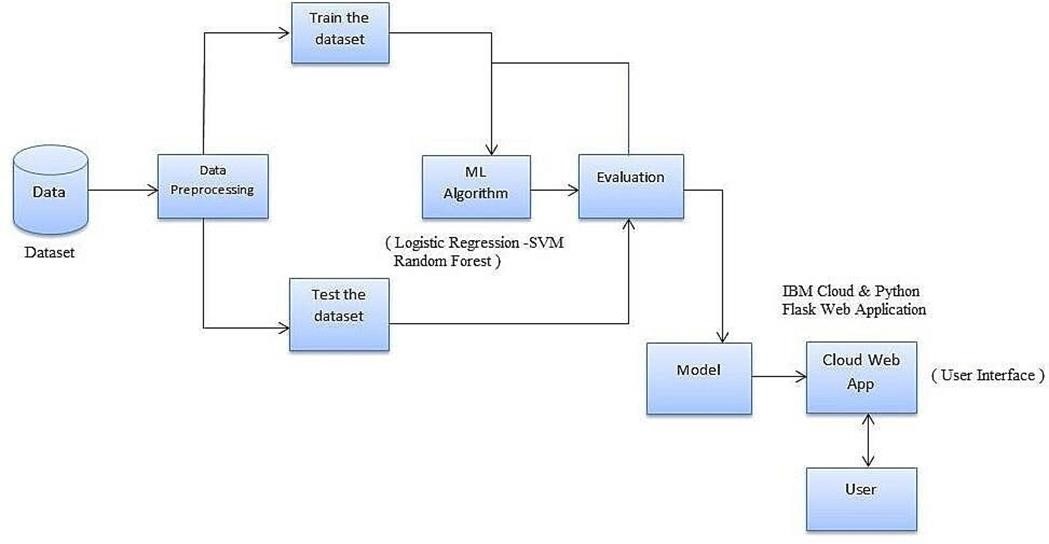
**DFD Level 0:**



* 1. **Solution & Technical Architecture**

**Technical Architecture:**

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2



* 1. **User Stories**

Use the below template to list all the user stories for the product.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **User Type** | **Functional Requireme nt(Epic)** | **User**  **Story**  **Number** | **User Story / Task** | **Acceptance criteria** | **Priority** | **Release** |
| Customer (Mobile user) | Contact | USN-1 | As a user, I can contact to the administration system to report about the performance. | I can contact directly by writing in the contact form. | High | Sprint-1 |
|  |  | USN-2 | As a user, I will engage with admin to report about query regarding phishing website. | I can have a chat aboutthe query. | High | Sprint-1 |
|  | About | USN-3 | As a user, I can read about the phishing and be aware of the harmful sides of the websites. |  | High | Sprint-1 |
|  | Dashboard |  |  |  |  |  |
| Customer (Web user) | User input | USN-1 | As a user I can input the particular URL in the Search field & get the prediction of website. | I can access the website by knowing about website security. | High | Sprint-1 |
| Customer  Care  Executive | Feature extraction | USN-1 | After I compare in case if none found on comparison then we can extract feature using other various approach. | As a User I can have comparison between websites for security. | High | Sprint-1 |
| Administrator | Prediction | USN-1 | Here the Model will predict the URL websites using  Machine Learning algorithms such as Logistic Regression, KNN. | I can have correct prediction on the particular algorithms | High | Sprint-1 |
|  | Classifier | USN-2 | Here I will predict the URL to give output in order toproduce final result by using classifier model. | In this I will find the correct classifier for predicting the result. | Medium | Sprint-2 |

## 6.1 PROJECT PLANNING & SCHEDULING

**1.1 Sprint Planning & Estimation**

|  |  |  |
| --- | --- | --- |
| TITLE | DESCRIPTION | DATE |
| Prepare Empathy Map | Prepare Empathy Map Canvas to capture the user Pains & Gains, Prepare list of problem statements | 10 Sept 2022 |
| Ideation Brain Storming | List the by organizing the brainstorming session and prioritize the top 3 ideas based on the feasibility & importance. | 15 Sept 2022 |
| Proposed Solution | Prepare the proposed solution document, which includes the novelty, feasibility of idea, business model, social impact, scalability of solution, etc. | 23 Sept 2022 |
| Problem Solution Fit | Prepare problem solution fit document | 28 Sept 2022 |
| Solution Architecture | Prepare solution architecture document. | 5 October 2022 |
| Customer Journey | Prepare the customer journey maps to understand the user interactions & experiences with the application (entry to exit). | 7 October 2022 |
| Solution Requirement | Prepare the functional requirement document | 8 October 2022 |

|  |  |  |
| --- | --- | --- |
| Data Flow Diagrams | Draw the data flow diagramsand submit for review. | 10 October 2022 |
| Technology Architecture | Prepare the technology architecture diagram. | 15 October 2022 |
| Prepare Milestone &Activity  List | Prepare the milestones & activity list of the project. | 26 October 2022 |
| Project Development -  Delivery of Sprint-1, 2, 3& 4 | Develop & submit the developed code by testing it. | 14 November 2022 |

**6.2 Sprint Delivery Schedule:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Functional Re**  **qui re**  **me**  **nt**  **(Ep**  **ic)** | **User**  **Story**  **Numbe**  **r** | **User Story / Task** | **Story Points** | **Priority** | **Team**  **Members** |
| Sprint-1 | Homepage | USN-1 | As a user, I can explore the resources of thehomepage for the functioning | 10 | Low | SONIYA.D, RIYA K MATHEW ,NIVEDHA.E,SRIVARSHAN.S.N |
| Sprint-1 | Url detection | USN-2 | As a user, I can learn about the various sides ofthe web phishing and be aware of the scams | 5 | High | SONIYA.D, RIYA K MATHEW, NIVEDHA.E, SRIVARSHAN.S.N |
| Sprint-2 | Final page | USN-3 | As a user, I can explore the resources of thefinal page for the functioning | 15 | Low | SONIYA.D ,RIYA K MATHEW,NIVEDHA.E,SRIVARSHAN.S.N |
| Sprint-3 | Prediction | USN-4 | As a user, I can predict the URL easily for detecting whether the website is legitimate ornot | 10 | High | SONIYA.D,RIYA K MATHEW,NIVEDHA.E,SRIVARSHAN.S.N |
| Sprint-4 | Chat | USN-5 | As a user, I can share the experience or contactt he admin for the support | 10 | High | SONIYA.D, RIYA K MATHEW, NIVEDHA.E,SRIVARSHAN.S.N |
| Sprint-1 | Homepage | USN-6 | As a admin, we can design interface andm  aintain the functioning of the website | 5 | High | SONIYA.D,RIYA K MATHEW,NIVEDHA.E,SRIVARSHAN.S.N |
| Sprint-2 | Final page | USN-7 | As a admin, we can design the complexity oft he website for making it userfriendly | 5 | Medium | SONIYA.D,RIYA K MATHEW,NIVEDHA.E,SRIVARSHAN.S.N |
| Sprint-3 | Prediction | USN-8 | As a admin, we can use various ML classifier model for the accurate result for the detection ofURL | 10 | High | SONIYA.D,RIYA K MATHEW,NIVEDHA.E,SRIVARSHAN.S.N |
| Sprint-4 | The final step | USN-9 | As a admin, we can response to the usermessage for improvement of the website | 10 | Medium | SONIYA.D,RIYA K MATHEW,NIVEDHA.E,SRIVARSHAN.S.N |

**7.CODING & SOLUTIONING:**

**7.1 Feature 1:**

|  |
| --- |
| #importing required libraries  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") phishing = pickle.load(file) file.close()    app = Flask(\_\_name\_\_)  @app.route('/')  @app.route('/web detection.html') Def Home():  return render\_template("web detection.html")  @app.route('/')  @app.route('/About.html') def About():  return render\_template("About.html")    @app.route('/Getstarted.html') def Getstarted():  return render\_template("Getstarted.html")  @app.route('/Contact.html') def Contact():  return render\_template("Contact.html")    @app.route("/predict", 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 =phishing.predict(x)[0]  #1 is safe #-1 is unsafe y\_pro\_phishing = phishing.predict\_proba(x)[0,0] y\_pro\_non\_phishing = phishing.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('predict.html',xx =["It is {0:.2f} % Safe to go  ".format(y\_pro\_non\_phishing\*100), "It is {0:.2f} % Unsafe to go ".format(y\_pro\_phishing\*100)],url=url)  # else:  # return render\_template("predict.html", xx ="Your are on the wrong site. Be cautious!")    if \_\_name\_\_ == "\_\_main\_\_":  app.run(debug=True,port=5000 |

**7.2 Feature 2:**

|  |
| --- |
| import ipaddress import re from tracemalloc import DomainFilter from urllib import response 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\.c om|fic\.kr|loopt\.us|'  'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.d o|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|cut t\.us|u\.bb|yourls\.org|'  'x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.ne t|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.monthcreation\_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 DomainFilter 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 =

|  |
| --- |
| '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]+)", 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): |

BeautifulSoup(urllib.request.urlopen("http://data.alexa.com/data?cli=10&dat=s&url=" + try:

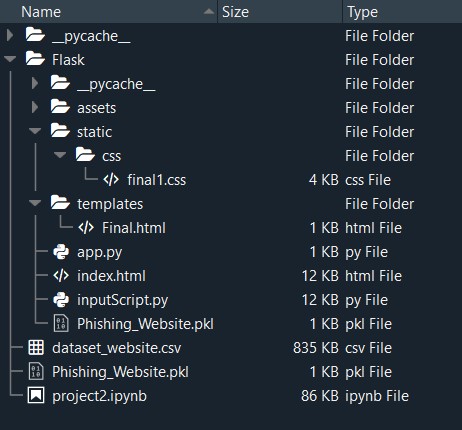
url\_match = re.search( 'at\.ua|usa\.cc|baltazarpresentes\.com\.br|pe\.hu|esy\.es|hol\.es|sweddy\.com|myj

|  |
| --- |
| ino\.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\.4 6\.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\.6  9\.166\.231|216\.58\.192\.225|'  '118\.184\.25\.86|67\.208\.74\.71|23\.253\.126\.58|104\.2 39\.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|19 9\.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\.23  1\.42', ip\_address) if url\_match: return -1 elif ip\_match: return -1 return 1 except:  return 1  def getFeaturesList(self): return self.features |

**7.3 Database Schema (if Applicable):**

* **"Flask"** folder contains two sub-folders static and templates. Static folder contains the style sheet used.
* **"Templates"** folder has the HTML pages.
* **"app.py"**  is the python script for server side computing.
* **"index.html"**  is the home page which should be used for initiating the application.
* **"inputScript.py"** has all the parameters of evaluation for a URL.
* **"Phishing\_Website.pkl"**  is the model file which you have to build.  **"dataset\_website.csv"** is the dataset
* **"project2.ipynb"** is the training notebook.

Refer below image for the same:



**1. ALGORITHMS USED:**

Two algorithms have been implemented to check whether a URL is legitimate or fraudulent.

Random forest algorithm creates the forest with number of decision trees. High number of tree gives high detection accuracy. Creation of trees is based on bootstrap method. In bootstrap method features and samples of dataset are randomly selected with replacement to construct single tree. Among randomly selected features, random forest algorithm will choose best splitter for classification.

Decision tree begins its work by choosing best splitter from the available attributes for classification which is considered as a root of the tree. Algorithm continues to build tree until it finds the leaf node. Decision tree creates training model which is used to predict target value or class in tree representation each internal node of the tree belongs to attribute and each leaf node of the tree belongs to class label.

Fig.1. Decision Tree Algorithm working

Fig.2. Random Forest Algorithm working

**2. PROJECT REQUIREMENTS Hardware Requirements:-**

* 2GB RAM (minimum)
* 100GB HDD (minimum)
* Intel 1.66 GHz Processor Pentium 4 (minimum)
* Internet Connectivity

### 3. Software Requirements:-

* WINDOWS 7 or higher
* Python 3.6.0 or higher
* Visual Studio Code
* FLASK framework
* HTML
* Dataset of Phishing Websites

**8. TESTING:**

**TESTCASES REPORT**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | | Date | 15-Nov-22 |  | | |  |  |  |  |  |
| Team ID | PNT2022TMID41066 |
| Project Name | Project - Web Phishing Detection |
| Maximum Marks | 4 marks |
| **Test case ID** | **Feature Type** | **Componen t** | **Test Scenario** | **Pre-Requisite** | **Steps To Execute** | **Test Data** | **Expected Result** | **Actual Result** | **Statu s** | **Comments** | **TC for Automation(Y/N)** | **BU**  **G**  **ID** | **Executed By** |
| LoginPage\_TC\_O  O 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 pr ocessing or not. | https://phishing- shield.herokuapp.com/ | Should Display the Webpage | Working as expected | Pass |  | N |  | Soniya.D |
| LoginPage\_TC\_O  O 2 | UI | Home Page | Verify the UI eleme nts is  Responsive |  | 1. Enter URL and clic k 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 |  | **Riya K Mathew** |
| LoginPage\_TC\_O  O 3 | Functional | Home page | Verify whether the link is  legitimate ornot |  | 1. Enter URL and clic k 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 ornot. | Working as expected | Pass |  | N |  | Nivedha .E |
| LoginPage\_TC\_O  O 4 | Functional | Home Page | Verify user is able to access the    legitimate website ornot |  | 1. Enter URL and clic k 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 |  | Srivarshan,S.N |
| LoginPage\_TC\_O  O 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   1. totalpad.com 2. http[s://www.klnce.edu](http://www.klnce.edu/) 4. salescript.info   5.  http[s://www.google.com/ 6.](http://www.google.com/)  delgets.com | User can able to identify the websites whether it is secure or not | Working as expected | Pass |  | N |  | Soniya .D |

**9.RESULT**

* Scikit-learn tool has been used to import Machine learning algorithms. Each classifier is trained using training set and testing set is used to evaluate performance of classifiers.
* Performance of classifiers has been evaluated by calculating classifier's accuracy score.

improve the accuracy of our models with better feature extraction.

**Fig.3. Accuracy with Random Forest Algorithm**

**Fig.4. Accuracy with Decision Tree Algorithm**

1. **ADVANTAGES:**

* + Measure the degrees of corporate and employee vulnerability.
  + Eliminate the cyber threat risk level.
  + Increase user alertness to phishing risks.
  + Instill a cyber security culture and create cyber security heroes.

**10.2 DISADVANTAGES:**

Phishing has a list of negative effects on a business, including loss of

money, loss of intellectual property, damage to reputation, and disruption of operational activities. These effects work together to cause loss of company value, sometimes with irreparable repercussions

**11.CONCLUSION:**

Thus to summarize, we have seen how phishing is a huge threat to the security and safety of the web and how phishing detection is an important problem domain. We have reviewed some of the traditional approaches to phishing detection; namely blacklist and heuristic evaluation methods, and their drawbacks. We have tested two machine learning algorithms on the Phishing Websites Dataset and reviewed their results.

We then selected the best algorithm based on its performance and built a Chrome extension for detecting phishing web pages. The extension allows easy deployment of our phishing detection model to end users. We have detected phishing websites using Random Forest algorithm with and accuracy of 97.31%. For future enhancements, we intend to build the phishing detection system as a scalable web service which will incorporate online learning so that new phishing attack patterns can easily be learned

**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 web service 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.

**13.APPENDIX:**

**GitHub:**

https://github.com/IBM-EPBL/IBM-Project-34587-1660239001