WEB PHISHING DETECTION

A PROJECT REPORT

Submitted by

Sruthi S

Ashwini MS

Subhiksha S

Dafni Trisha

Renita V

of

INFORMATION TECHNOLOGY

MADRAS INSTITUTE OF TECHNOLOGY

CHAPTER NO	TITLE	PAGE NO
1.	**INTRODUCTION**	
	1.1 Project Overview	4
	1.2 Purpose	4
2.	**LITERATURE SURVEY**	
	2.1 Existing problem	5
	2.2 References	5
	2.3 Problem Statement Definition	6
3.	**IDEATION & PROPOSED SOLUTION**	
	3.1 Empathy Map Canvas	7
	3.2 Ideation & Brainstorming	8
	3.3 Proposed Solution	9
	3.4 Problem Solution fit	10
4.	**REQUIREMENT ANALYSIS**	
	4.1 Functional requirement	11
	4.2 Non-Functional requirements	11
5.	**PROJECT DESIGN**	
	5.1 Data Flow Diagrams	12
	5.2 Solution & Technical Architecture	13
	5.3 User Stories	13
6.	**PROJECT PLANNING & SCHEDULING**	
	6.1 Sprint Planning & Estimation	14
	6.2 Sprint Delivery Schedule	14
	6.3 Reports from JIRA	15
7.	**CODING & SOLUTIONING**	
	7.1 Feature 1	16
	7.2 Feature 2	26

8.	**TESTING**	
	8.1 Test Cases	36
	8.2 User Acceptance Testing	38
9.	**RESULTS**	
	9.1 Performance Metrics	39
10.	**ADVANTAGES AND DISADVANTAGES**	40
11.	**CONCLUSION**	40
12.	**FUTURE SCOPE**	40
13.	**APPENDIX**	
	13.1 Source Code	41
	13.2 GitHub & Project Demo Link	68

CHAPTER 1
INTRODUCTION

1.1 Project Overview:

Web service is one of the most important internet communication software services. The project mainly focuses on applying a machine-learning algorithm to detect Phishing websites. In order to detect and predict the 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 algorithms applied include Logistic Regression, Support Vector Machine, Decision Tree, Naïve Bayes, Random Forest and Stacking. The 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. When the URL of the website is entered, the machine learning algorithm is used to detect whether the website is a phishing website or not.

1.2 Purpose:

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

Common threats of web phishing:

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

The main purpose of the project is to detect phishing sites to improve the customer’s sense of safety whenever he/she attempts to provide any sensitive information to a site. This awareness will help people to not access the phishing sites, which will reduce the revenue of malicious site owners. This application can be accessed online without paying instead, can be accessed via any browser of the customer’s choice to detect any site with high accuracy.

CHAPTER 2

LITERATURE SURVEY

2.1 Existing Problem:

There are websites online that detect phishing. However, once a usage cap is reached, users are charged. A significant number of them come with a simple foundation of features. Several criteria that could be utilized to recognize a phishing site have been extensively examined and discovered by us. These elements are classified as address bar-based features, domain-based features, HTML-based features, and JavaScript-based features. These features allow us to construct an intelligent system that is highly accurate and effective at detecting phishing sites. Furthermore, it is an open-source website that will be simple for all users to use.

2.2 References:

[1] Alswailem, B. Alabdullah, N. Alrumayh and A. Alsedrani, "Detecting Phishing Websites Using Machine Learning," 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS), 2019, pp. 1-6, doi: 10.1109/CAIS.2019.8769571.

[2] S. Singh, M. P. Singh and R. Pandey, "Phishing Detection from URLs Using Deep Learning Approach," 2020 5th International Conference on Computing, Communication and Security (ICCCS), 2020, pp. 1-4, doi: 10.1109/ICCCS49678.2020.9277459.

[3] M. Abutaha, M. Ababneh, K. Mahmoud and S. A. -H. Baddar, "URL Phishing Detection using Machine Learning Techniques based on URLs Lexical Analysis," 2021 12th International Conference on Information and Communication Systems (ICICS), 2021, pp. 147-152, doi: 10.1109/ICICS52457.2021.9464539.

[4] A. K. Singh and N. Goyal, "Detection of Malicious Webpages Using Deep Learning," 2021 IEEE International Conference on Big Data (Big Data), 2021, pp. 3370-3379, doi: 10.1109/BigData52589.2021.9671622.

[5] C. -Y. Wu, C. -C. Kuo and C. -S. Yang, "A Phishing Detection System based on Machine Learning," 2019 International Conference on Intelligent Computing and its Emerging Applications (ICEA), 2019, pp. 28-32, doi: 10.1109/ICEA.2019.8858325.

［6］A. Lakshmanarao, P. S. P. Rao and M. M. B. Krishna, "Phishing website detection using novel machine learning fusion approach," 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), 2021, pp. 1164-1169, doi: 10.1109/ICAIS50930.2021.9395810.

［7］M. M. Yadollahi, F. Shoeleh, E. Serkani, A. Madani and H. Gharaee, "An Adaptive Machine Learning Based Approach for Phishing Detection Using Hybrid Features," 2019 5th International Conference on Web Research (ICWR), 2019, pp. 281-286, doi: 10.1109/ICWR.2019.8765265.

［8］S. -J. Bu and S. -B. Cho, "Integrating Deep Learning with First-Order Logic Programmed Constraints for Zero-Day Phishing Attack Detection," ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021, pp. 2685-2689, doi: 10.1109/ICASSP39728.2021.9414850.

［9］L. Zhang, P. Zhang, L. Liu and J. Tan, "Multiphish: Multi-Modal Features Fusion Networks for Phishing Detection," ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021, pp. 3520-3524, doi: 10.1109/ICASSP39728.2021.9415016.

［10］S. Yu, C. An, T. Yu, Z. Zhao, T. Li and J. Wang, "Phishing Detection Based on Multi-Feature Neural Network," 2022 IEEE International Performance, Computing, and Communications Conference (IPCCC), 2022, pp. 73-79

2.3 Problem statement definition:

Phishing is a form of social engineering assault that is frequently employed to obtain user information, such as user credentials and credit card data. It happens when an attacker deludes a victim into opening an email, instant message, or text message by disguising themselves as a reliable source. It poses a risk to numerous elements of online security, including the potential for scams and the release of sensitive data. Following are typical hazards posed by web phishing:

● Getting personal information from a person or business.

● Posing as a reliable company to distribute harmful web pages.

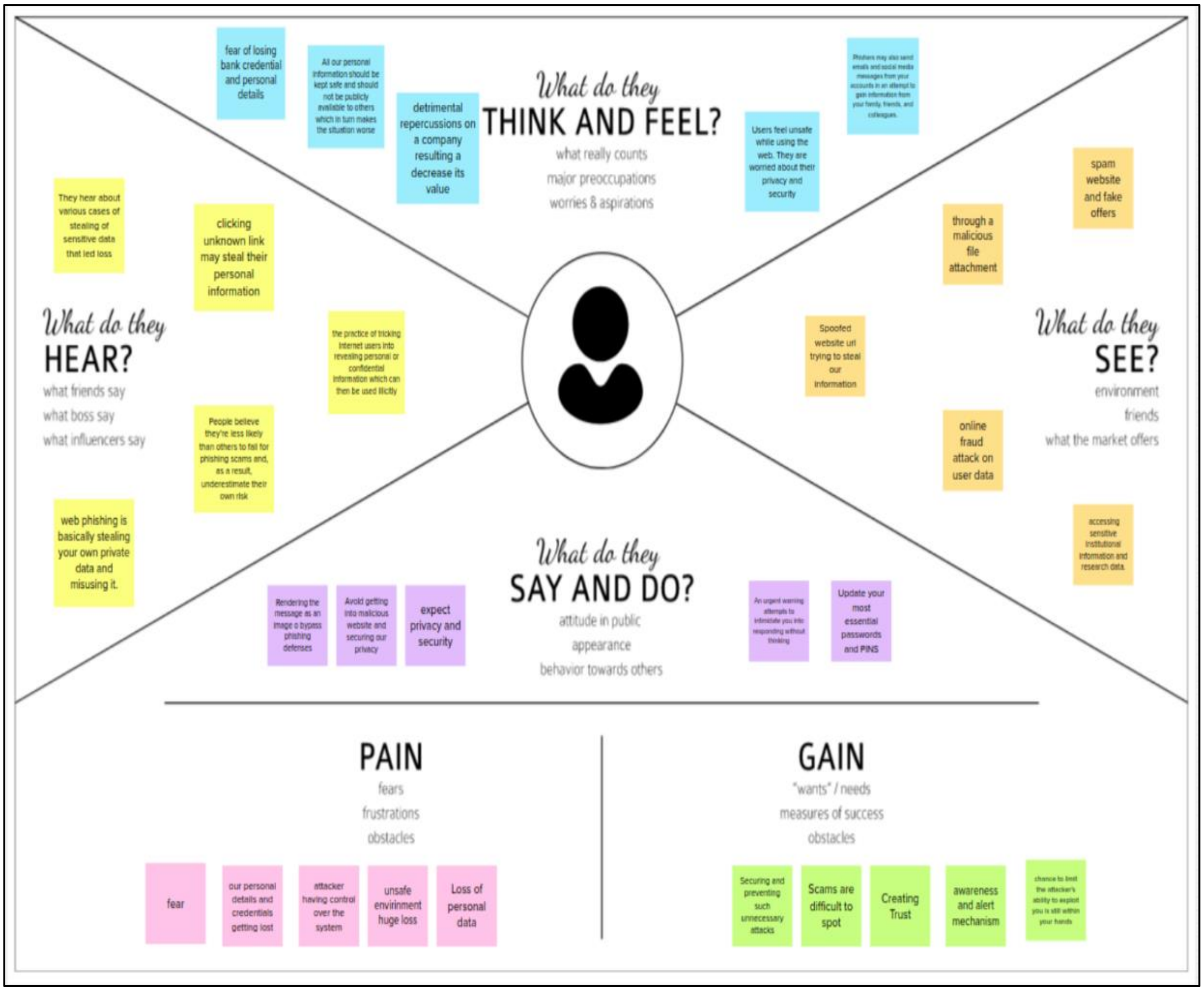
Aim is classification of a phishing website with the aid of various machine learning techniques to achieve maximum accuracy and a concise model. By implementing classification algorithms and approaches to extract the phishing datasets criteria to define their authenticity, we construct an effective and intelligent system to detect such websites in work to circumvent these dangers.

CHAPTER 3

IDEATION & PROPOSED SOLUTION

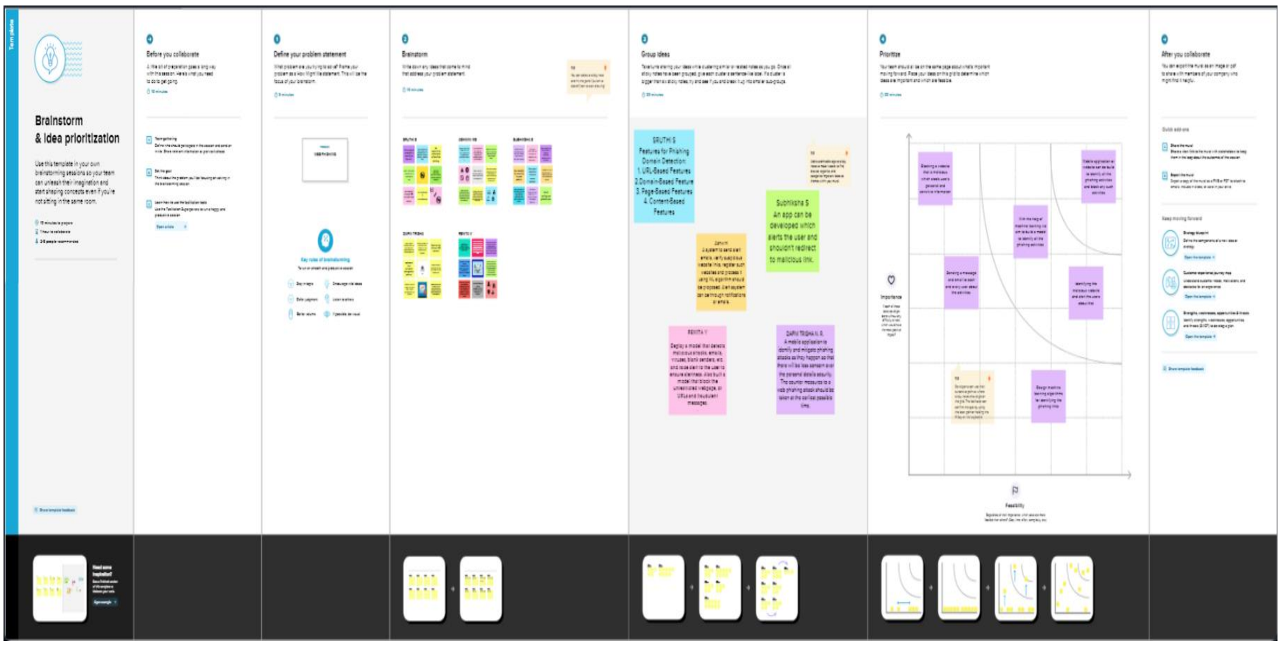
3.1 Empathy Map Canvas:

Empathy maps are useful tool that designers use to not only analyze users' behavior but also to visually convey their results to colleagues, bringing the team together around a common knowledge of the user. In user-centered design, empathy maps are best used from the very beginning of the design process.



3.2 Ideation and Brainstorming:

Ideation is a general term that refers to the process of coming up with and expressing new ideas. It is an imaginative thought that seeks to resolve a dilemma or offer a more effective means of carrying out an action. It includes creating fresh concepts, upgrading existing ones, and figuring out how to put fresh concepts into action. Brainstorming is the most frequently practiced form of ideation. The intention of brainstorming is to leverage the collective thinking of the group, by engaging with each other, listening, and building on other ideas.



3.3 Proposed Solution:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Nowadays, there are many people who are purchasing products and making transaction online through various websites. Sensitive data like user information password, card details are asked by malicious website to steal customer's information. So, this kind of malicious activities should be detected.
2.	Idea / Solution description	Classification data mining algorithm can be used to determine the difference between the legitimate websites and the web phishing websites.
3.	Novelty / Uniqueness	The user will be notified if the website is malicious via SMS / WhatsApp such that their privacy will be ensured and awareness will be created.
4.	Social Impact / Customer Satisfaction	The customer will come to know whether their details are safe/ not and the customer will be restricted from entering into the phishing websites.
5.	Business Model (Revenue Model)	There is a scope for including Advertisements in the website such that revenue can be generated.
6.	Scalability of the Solution	We will deploy the website in the IBM cloud and the end user can make use of it.

3.4 Problem Solution Fit:

Problem-Solution Fit happens when there is proof that customers are interested in particular tasks, challenges, and benefits. You've established that a problem exists and created a value offer that takes into account the tasks, challenges, and gains of your clients at this point.

A problem-solution-fit occurs when such a solution is discovered and a business develops a strategy that, from a variety of angles, offers a game changer for customers.

However, if businesses miss evaluating the Problem-Solution Fit they developed, they face a risk of finding that no one wants their solution, which is unfortunate considering the effort and money invested.

CHAPTER 4

REQUIREMENT ANALYSIS

4.1 Functional Requirements:

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Register by entering details such as name, email, password, phone number, etc.
FR-2	User Login	Login using the registered email id and password.
FR-3	Model Building	Bulid various machine learning model to detect web phishing and compare them.
FR-4	Check URL	Get the URL from user and display if the website is malicious or not
FR-5	Integration	Integrate the frontend and the developed ML model using flask
FR-6	Alert Message	Notify the user through email or phone regarding the malicious website.

4.2 Non – Functional Requirements:

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Any URL must be accepted for detection.
NFR-2	**Security**	Alert message must be sent to the users to enable secure browsing
NFR-3	**Reliability**	The web phishing websites must detected accurately and the result must be reliable.
NFR-4	**Performance**	The performance and interface must be user friendly
NFR-5	**Availability**	Anyone must be able to register and login.
NFR-6	**Scalability**	It must be able to handle increase in the number of users.

CHAPTER 5

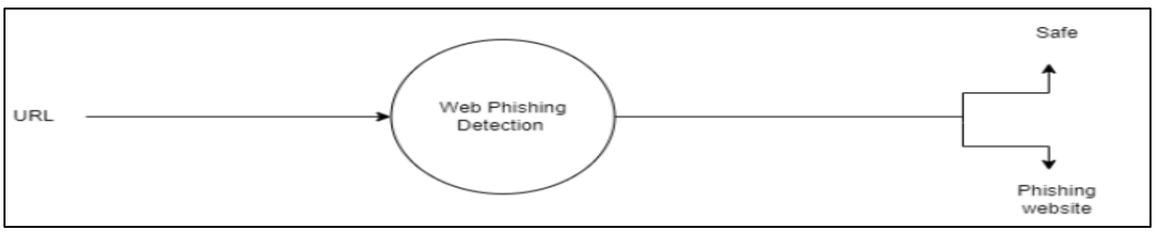
PROJECT DESIGN

5.1 Data Flow Diagram:

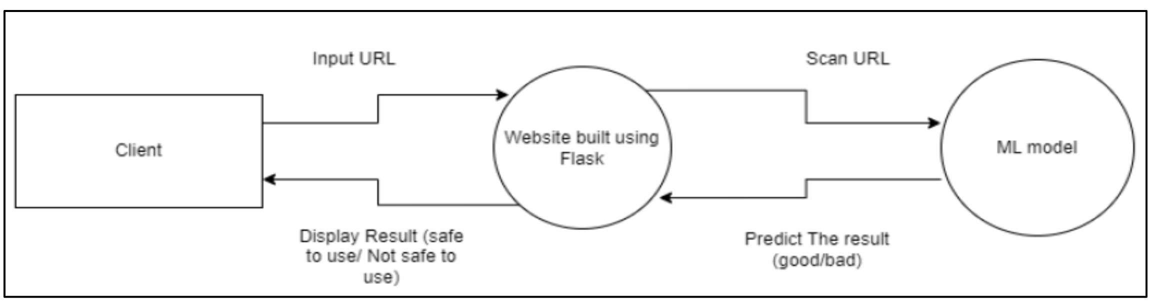
A data flow diagram is a visualization tool used to illustrate the flow of processes in a company or a specific project within it. It highlights the movement of information as well as the sequence of steps or events required to complete a work task.

DFDs can vary in design and complexity, depending on the process it represents. It can be a simple outline of a general system or a more granular sketch of a multi-level procedure

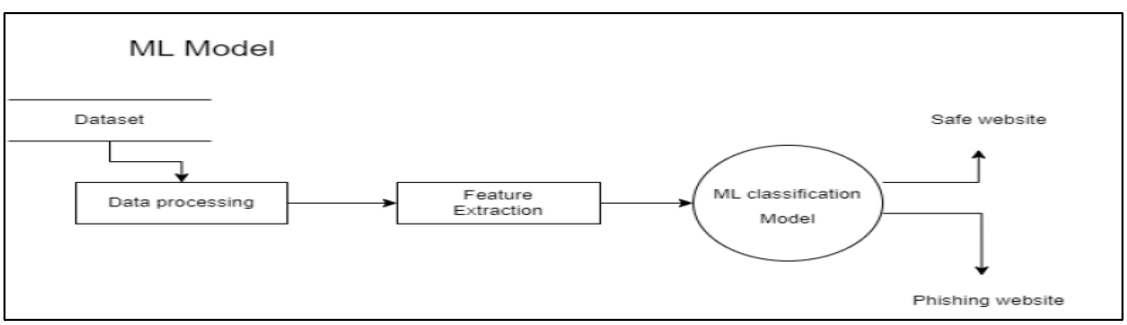
DFD Level 0:



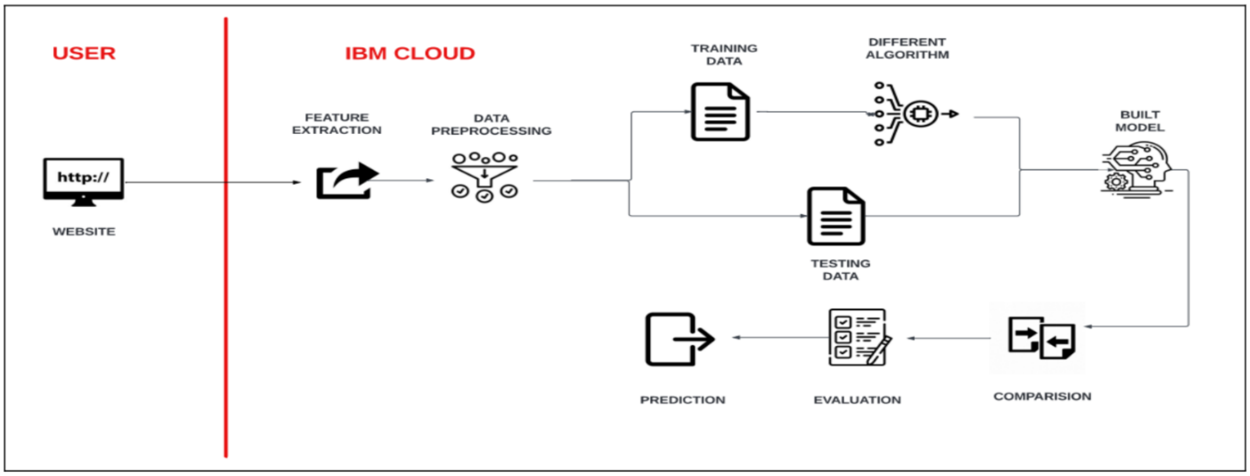
DFD Level 1:



DFD Level 2:



5.2 Solution and Technical Architecture



5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Web user)	Website	USN-1	As a user, I need a website to check whether the URL is safe to enter or not.	Website should be user-friendly and responsive	High	Sprint-3
	Alert Notification	USN-2	If I enter into some Malicious Link , Notification has to be sent to me	Receive notification in mobile or to my mail id	Low	Sprint-3
	Blocking	USN-3	If the link is not safe to enter, It should block me to use that site.		High	Sprint-2
	Allowing	USN-4	If I wish to use that website then ,iT should also allow me to enter into that website		Medium	Sprint-2
	Accurate Prediction	USN-5	As a User, I need a correct result. There shouldn't be any anomaly	The phishing website has to be determined correctly.	High	Sprint-1

CHAPTER 6

PROJECT PLANNING & SCHEDULING

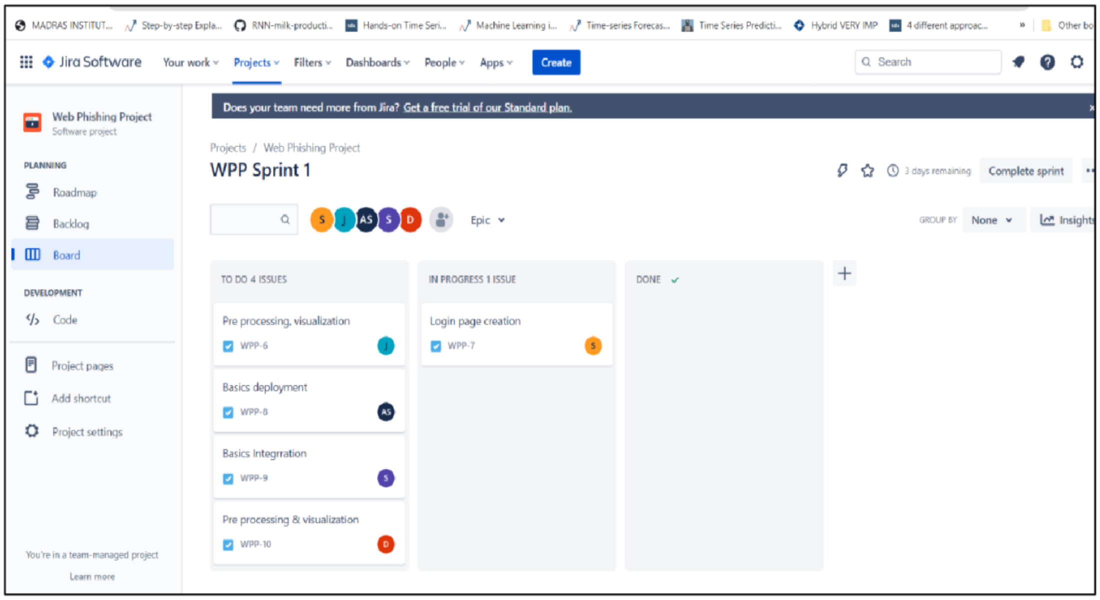
6.1 Sprint planning and estimation

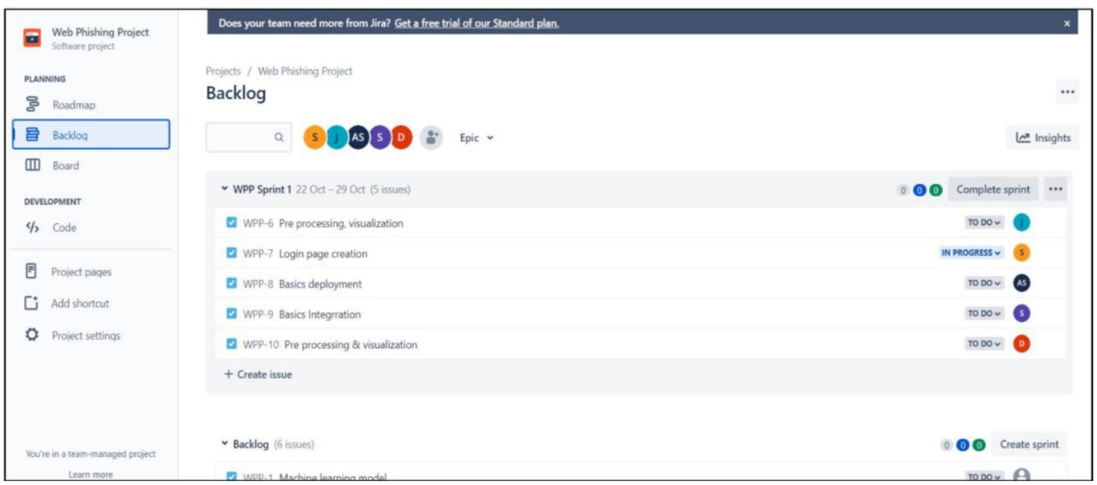
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirmingmy password.	2	High	SRUTHI S
Sprint-2	Registration	USN-3	As a user, I can register for the applicationthrough Facebook	2	Low	SRUTHI S
Sprint-2	Registration	USN-4	As a user, I can register for the applicationthrough Gmail	2	Medium	SRUTHI S
Sprint-2	Login	USN-5	As a user, I can log into the application byentering email & password	1	High	SUBHIKSHA S
Sprint-2	Dashboard	USN-6	Once the user is registered and have logged in,he will be able to access the dashboard over the browser.	1	Medium	ASHWINI MS
Sprint-3	Model Building	USN-7	Using various machine learning techniques, amodel has to be built.	2	High	DAFNI TRISHA, RENITA V
Sprint-3	Model Testing	USN-8	Built model have to be checked for accuracy and other performance metrics to correctlyclassify.	2	High	DAFNI TRISHA, RENITA V
Sprint-4	Integration	USN-9	Integrate the frontend and the developed MLmodel using flask and deploy in the cloud.	2	High	ASHWINI MS, SUBHIKSH A S

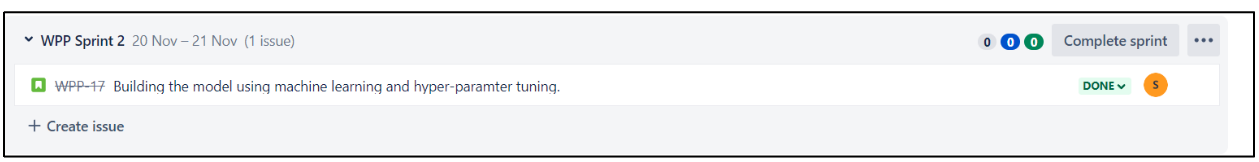
6.2 Sprint delivery Schedule

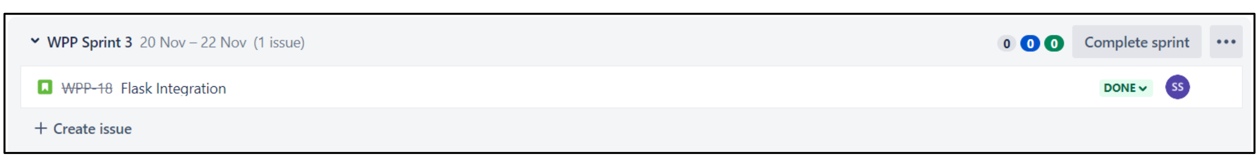
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date(Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022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

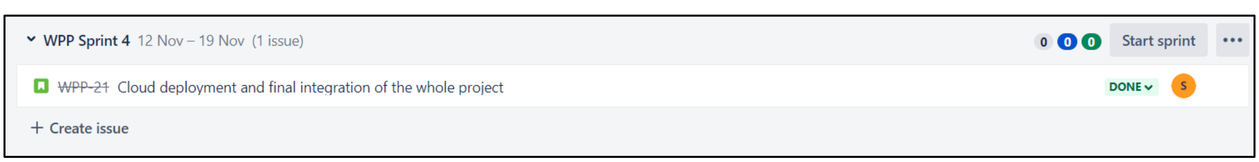
6.3 Reports from JIRA











CHAPTER 7

CODING & SOLUTION

7.1 Model Building

7.1.1 Data Collection & Exploratory Data Analysis

The dataset contains 32 features and 11055 records. All the data columns are of the type int64. EDA is the process of performing initial investigation on the dataset. The features that are present in the data set include:

- IP Address in URL
- Length of URL
- Using URL Shortening Services
- "@" Symbol in URL
- Redirection "//" in URL
- Prefix or Suffix "-" in Domain
- Having Sub Domain
- Length of Domain Registration
- Favicon
- Port Number
- HTTPS Token
- Request URL
- URL of Anchor
- Links in Tags
- SFH
- Email Submission
- Abnormal URL
- Status Bar Customization (on mouse over)
- Disabling Right Click
- Presence of Popup Window
- IFrame Redirection
- Age of Domain
- DNS Record
- Web Traffic
- Page Rank
- Google Index
- Links pointing to the page
- Statistical Report
- Result

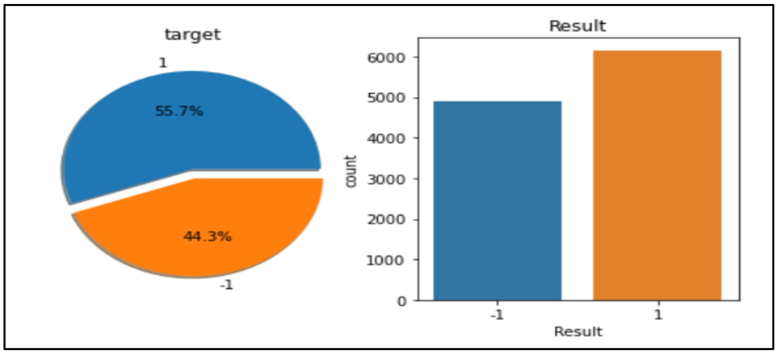
```
Index(['index', 'having_IPhaving_IP_Address', 'URLURL_Length',
       'Shortining_Service', 'having_At_Symbol', 'double_slash_redirecting',
       'Prefix_Suffix', 'having_Sub_Domain', 'SSLfinal_State',
       'Domain_registeration_length', 'Favicon', 'port', 'HTTPS_token',
       'Request_URL', 'URL_of_Anchor', 'Links_in_tags', 'SFH',
       'Submitting_to_email', 'Abnormal_URL', 'Redirect', 'on_mouseover',
       'RightClick', 'popUpWidnow', 'Iframe', 'age_of_domain', 'DNSRecord',
       'web_traffic', 'Page_Rank', 'Google_Index', 'Links_pointing_to_page',
       'Statistical_report', 'Result'],
      dtype='object')
```

7.1.2 Data Visualization

Data visualization helps to understand the data and also explain the data to others. Histogram, box plot, correlation matrix plot, scatter matrix plot, pair plot has been plotted.

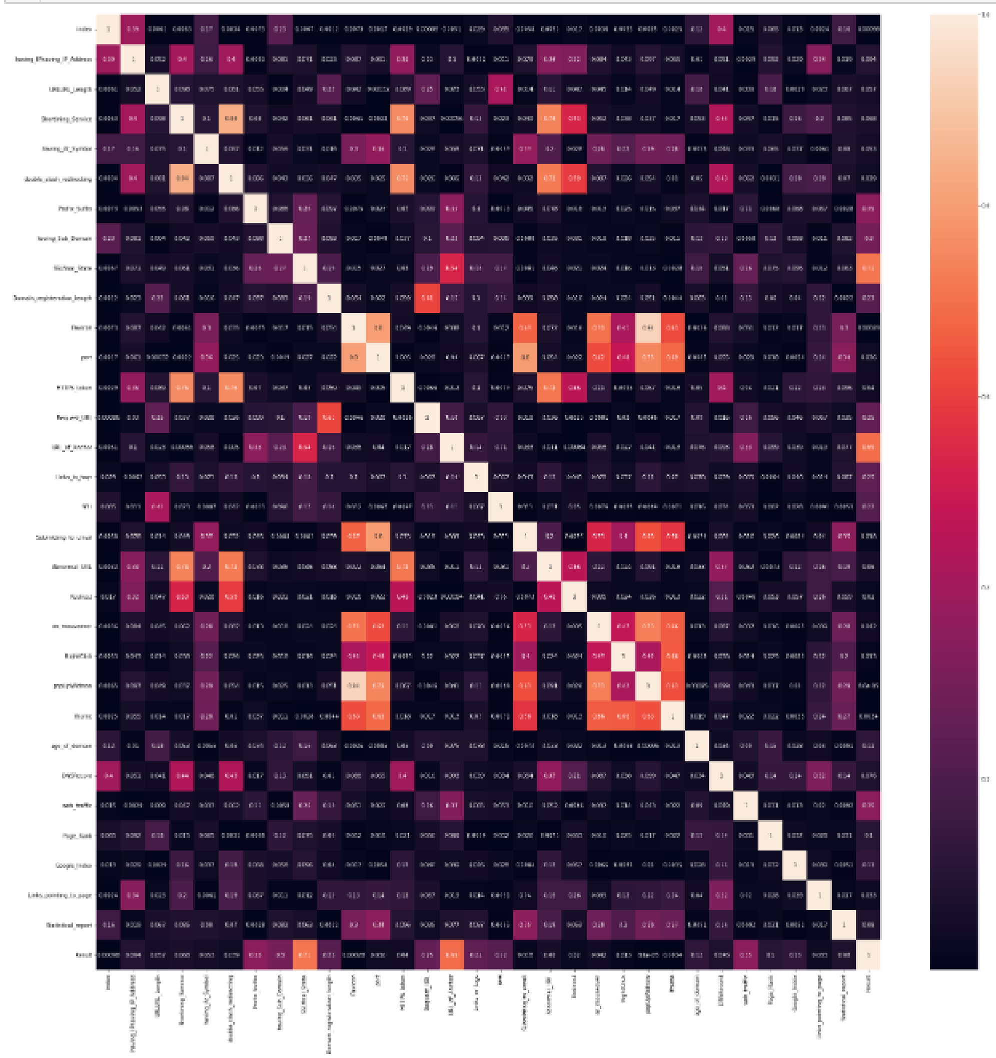
Univariate analysis: Univariate analysis provides an understanding in the characteristics of each feature in the data set. Different characteristics are computed for numerical and categorical data. For the numerical features characteristics are standard deviation, skewness, kurtosis, percentile, interquartile range (IQR) and range. For the categorical features characteristics are count, cardinality, list of unique values, top and freq.

	count	mean	std	min	25%	50%	75%	max
index	11055.0	5528.000000	3191.447947	1.0	2764.5	5528.0	8291.5	11055.0
having_IPhaving_IP_Address	11055.0	0.313795	0.949534	-1.0	-1.0	1.0	1.0	1.0
URLURL_Length	11055.0	-0.633198	0.766095	-1.0	-1.0	-1.0	-1.0	1.0
Shortining_Service	11055.0	0.738761	0.673998	-1.0	1.0	1.0	1.0	1.0
having_At_Symbol	11055.0	0.700588	0.713598	-1.0	1.0	1.0	1.0	1.0
double_slash_redirecting	11055.0	0.741474	0.671011	-1.0	1.0	1.0	1.0	1.0
Prefix_Suffix	11055.0	-0.734962	0.678139	-1.0	-1.0	-1.0	-1.0	1.0
having_Sub_Domain	11055.0	0.063953	0.817518	-1.0	-1.0	0.0	1.0	1.0
SSLfinal_State	11055.0	0.250927	0.911892	-1.0	-1.0	1.0	1.0	1.0
Domain_registeration_length	11055.0	-0.336771	0.941629	-1.0	-1.0	-1.0	1.0	1.0
Favicon	11055.0	0.628584	0.777777	-1.0	1.0	1.0	1.0	1.0
port	11055.0	0.728268	0.685324	-1.0	1.0	1.0	1.0	1.0
HTTPS_token	11055.0	0.675079	0.737779	-1.0	1.0	1.0	1.0	1.0
Request_URL	11055.0	0.186793	0.982444	-1.0	-1.0	1.0	1.0	1.0
URL_of_Anchor	11055.0	-0.076526	0.715138	-1.0	-1.0	0.0	0.0	1.0
Links_in_tags	11055.0	-0.118137	0.763973	-1.0	-1.0	0.0	0.0	1.0
SFH	11055.0	-0.595749	0.759143	-1.0	-1.0	-1.0	-1.0	1.0
Submitting_to_email	11055.0	0.635640	0.772021	-1.0	1.0	1.0	1.0	1.0
Abnormal_URL	11055.0	0.705292	0.708949	-1.0	1.0	1.0	1.0	1.0
Redirect	11055.0	0.115694	0.319872	0.0	0.0	0.0	0.0	1.0
on_mouseover	11055.0	0.762099	0.647490	-1.0	1.0	1.0	1.0	1.0
RightClick	11055.0	0.913885	0.405991	-1.0	1.0	1.0	1.0	1.0
popUpWidnow	11055.0	0.613388	0.789818	-1.0	1.0	1.0	1.0	1.0
Iframe	11055.0	0.816915	0.576784	-1.0	1.0	1.0	1.0	1.0
age_of_domain	11055.0	0.061239	0.998168	-1.0	-1.0	1.0	1.0	1.0
DNSRecord	11055.0	0.377114	0.926209	-1.0	-1.0	1.0	1.0	1.0
web_traffic	11055.0	0.287291	0.827733	-1.0	0.0	1.0	1.0	1.0
Page_Rank	11055.0	-0.483673	0.875289	-1.0	-1.0	-1.0	1.0	1.0
Google_Index	11055.0	0.721574	0.692369	-1.0	1.0	1.0	1.0	1.0
Links_pointing_to_page	11055.0	0.344007	0.569944	-1.0	0.0	0.0	1.0	1.0
Statistical_report	11055.0	0.719584	0.694437	-1.0	1.0	1.0	1.0	1.0
Result	11055.0	0.113885	0.993539	-1.0	-1.0	1.0	1.0	1.0



Correlation Matrix:

```
plt.figure(figsize=(32,32))
sns.heatmap(np.abs(df.corr()), annot=True);
```

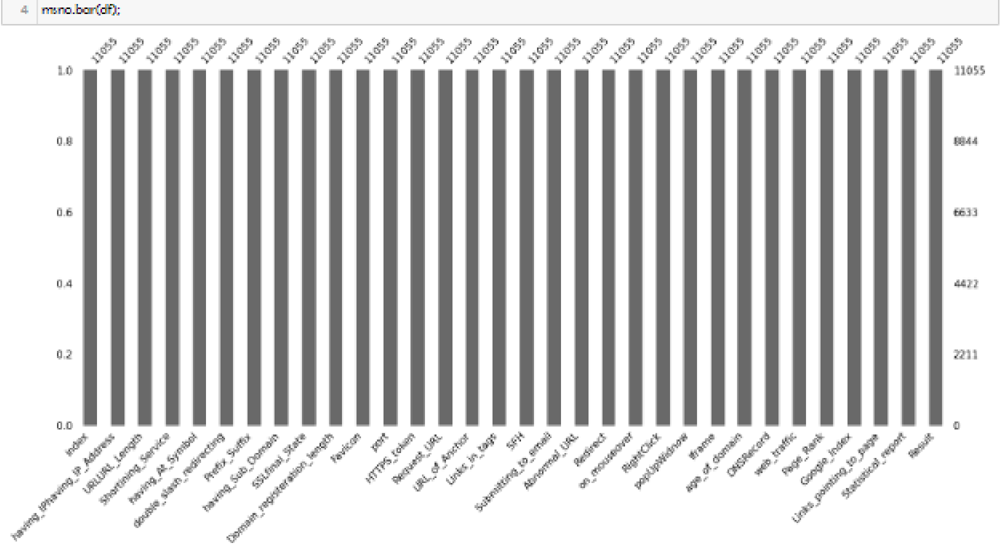


Histogram:

```
df.hist(bins=10, edgecolor = "black", figsize=(15,15))
plt.show()
```

Missing Values:

```
# Have been visualized using the missingno library for the visualization of missing observations.
# Plotting
import missingno as msno
msno.bar(df);
```



7.1.3 Data Preprocessing & Splitting the dataset

In the first step data preprocessing is done. Preprocessing is the method by which we perform data cleaning i.e., raw dataset is converted into cleaned dataset. There are no missing values in the dataset. The dataset is divided into 80:20 ratio where 80% is for training data and 20% for testing data.

```
# Sepratating & assigning features and target columns to X & y
X=df.iloc[:,1:31]
y=df.iloc[:,31]
X.shape, y.shape
```

```
((11055, 30), (11055,))
```

```
X_train, X_test, y_train, y_test = train_test_split(X.values, y.values, test_size = 0.2, random_state = 12)
```

7.1.4 Model Building and Hyper parameter tuning

From the dataset above, it is clear that this is a supervised machine learning task. There are two major types of supervised machine learning problems, called classification and regression. This data set comes under classification problem, as the input URL is classified as phishing (-1) or legitimate (1). In this we have done 5 supervised classification algorithm namely Logistic regression, support vector machine, Random Forest, Decision tree, Naïve bayes.

Hyper parameter tuning is the process through which we choose a set of optimal hyper parameters for learning algorithm. Hyper parameters are model argument whose value is set before the learning process begins. Not all hyper parameters are equally important. GridSearchCV is a method to find the best set of optimal hyper parameters from a grid and this method will go through all the possible intermediate combinations. As a result, accuracy can be improved.

Evaluation is done using classification accuracy. Confusion matrix and classification report is also plotted.

Logistic Regression

```
param_grid_LR = {
    'solver' : ['newton-cg', 'lbfgs', 'liblinear','sag','saga'],
    'penalty' : ['l1', 'l2', 'elasticnet'],
    'C' : [100, 10, 1.0, 0.1, 0.01]
}
grid_LR = GridSearchCV(LogisticRegression(),param_grid_LR, scoring='accuracy', n_jobs=-1,
cv=5).fit(X_train,y_train)
print('Best Score: %s' % grid_LR.best_score_)
print('Best Hyperparameters: %s' % grid_LR.best_params_)
Tuned_LR_model = LogisticRegression(**grid_LR.best_params_).fit(X_train, y_train)
Tuned_LR_y_prediction = Tuned_LR_model.predict(X_test)



HP_accuracy1_LR = Tuned_LR_model.score(X_train, y_train)
print("Accuracy of train data = ", HP_accuracy1_LR * 100, "%")
HP_accuracy2_LR = Tuned_LR_model.score(X_test, y_test)
print("Accuracy of validation data = ", HP_accuracy2_LR * 100, "%")
```

```
Accuracy of train data =  92.79737675260064 %
Accuracy of validation data =  93.441881501583 %
```

	precision	recall	f1-score	support
-1	0.93	0.92	0.93	1001
1	0.93	0.95	0.94	1210
accuracy			0.93	2211
macro avg	0.93	0.93	0.93	2211
weighted avg	0.93	0.93	0.93	2211

Support Vector Machine

```
param_grid_SVM = {'C': [0.01,0.1, 1, 10, 100, 1000],
            'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
            'kernel': ['rbf','sigmoid']}
grid_SVM = GridSearchCV(SVC(), param_grid_SVM, refit = True,return_train_score=True,
n_jobs=-1,cv=5).fit(X_train, y_train)


# print best parameter after tuning
print('Best Score: %s' % grid_SVM.best_score_)
print("BEST PARAMETER:",grid_SVM.best_params_)


Tuned_SVM_model = SVC(**grid_SVM.best_params_).fit(X_train, y_train)
Tuned_SVM_y_prediction= Tuned_SVM_model.predict(X_test)


HP_accuracy1_SVM = Tuned_SVM_model.score(X_train, y_train)
print("Accuracy of train data = ", HP_accuracy1_SVM * 100, "%")
HP_accuracy2_SVM = Tuned_SVM_model.score(X_test, y_test)
print("Accuracy of validation data = ", HP_accuracy2_SVM * 100, "%")
```

```
Accuracy of train data =  98.4283129805518 %
Accuracy of validation data =  96.96969696969697 %
```

	precision	recall	f1-score	support
-1	0.98	0.96	0.97	1001
1	0.96	0.98	0.97	1210
accuracy			0.97	2211
macro avg	0.97	0.97	0.97	2211
weighted avg	0.97	0.97	0.97	2211

Decision Tree

```
param_grid_DT = {'criterion': ["gini", "entropy"],
            'splitter': ["best", "random"],
            'max_depth': [10,15,35,30],
            'min_samples_split': [2, 3, 4,5,6]}
grid_DT = GridSearchCV(DecisionTreeClassifier(), param_grid_DT, cv=5, n_jobs=-1,scoring = 'accuracy' ).fit(X_train,y_train)


Tuned_DT_model = DecisionTreeClassifier(**grid_DT.best_params_).fit(X_train,y_train)
Tuned_DT_y_prediction= Tuned_DT_model.predict(X_test)


HP_accuracy1_DT = Tuned_DT_model.score(X_train, y_train)
print("Accuracy of train data = ", HP_accuracy1_DT * 100, "%")
HP_accuracy2_DT = Tuned_DT_model.score(X_test, y_test)
print("Accuracy of validation data = ", HP_accuracy2_DT * 100, "%")_LR * 100, "%")
```

```
Accuracy of train data =  98.93713251922208 %
Accuracy of validation data =  96.8340117593849 %
```

	precision	recall	f1-score	support
-1	0.97	0.96	0.96	1001
1	0.97	0.98	0.97	1210
accuracy			0.97	2211
macro avg	0.97	0.97	0.97	2211
weighted avg	0.97	0.97	0.97	2211

Random Forest

```
param_grid_RF = {'bootstrap': [True],
    'max_depth': [80, 90, 100,110],
    'criterion' : ('gini','entropy'),
    'max_features': [2, 3],
    'n_estimators': [ 200, 300, 1000]}
grid_RF = GridSearchCV(RandomForestClassifier(), param_grid_RF, cv=5, n_jobs=-1,scoring
= 'accuracy' ).fit(X_train,y_train)


Tuned_RF_model = RandomForestClassifier(**grid_RF.best_params_).fit(X_train,y_train)
Tuned_RF_y_prediction= Tuned_RF_model.predict(X_test)


HP_accuracy1_RF = Tuned_RF_model.score(X_train, y_train)
print("Accuracy of train data = ", HP_accuracy1_RF * 100, "%")
HP_accuracy2_RF = Tuned_RF_model.score(X_test, y_test)
print("Accuracy of validation data = ", HP_accuracy2_RF * 100, "%")
```

```
Accuracy of train data =  98.93713251922208 %
Accuracy of test data =  97.46720940750791 %
```

	precision	recall	f1-score	support
-1	0.98	0.97	0.97	1001
1	0.97	0.98	0.98	1210
accuracy			0.97	2211
macro avg	0.98	0.97	0.97	2211
weighted avg	0.97	0.97	0.97	2211

Naïve Bayes

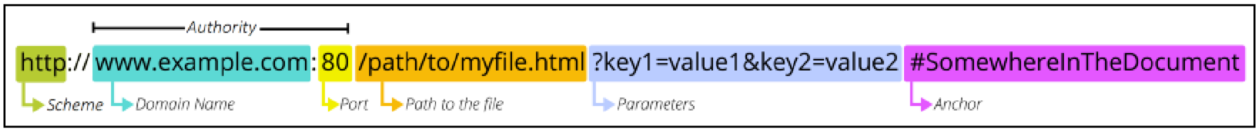
```
params_NB = {'var_smoothing': np.logspace(0,-9, num=100)}

grid_NB = GridSearchCV(GaussianNB(), param_grid=params_NB, cv=5,
scoring='accuracy').fit(X_train,y_train)



Tuned_NB_model = GaussianNB(**grid_NB.best_params_).fit(X_train,y_train)

Tuned_NB_y_prediction= Tuned_NB_model.predict(X_test)

HP_accuracy1_NB = Tuned_NB_model.score(X_train, y_train)

print("Accuracy of train data = ", HP_accuracy1_NB * 100, "%")

HP_accuracy2_NB = Tuned_NB_model.score(X_test, y_test)

print("Accuracy of validation data = ", HP_accuracy2_NB * 100, "%")
```

```
Accuracy of train data =  90.79601990049751 %
Accuracy of validation data =  91.85888738127545 %
```

	precision	recall	f1-score	support
-1	0.90	0.92	0.91	1001
1	0.93	0.92	0.92	1210
accuracy			0.92	2211
macro avg	0.92	0.92	0.92	2211
weighted avg	0.92	0.92	0.92	2211

7.2 Feature Extraction from URL

The address of a specific unique resource on the Web is all that is contained in a URL, also known as a uniform resource locator. Theoretically, every legitimate URL leads to a different resource.



URL Features

Deep learning techniques offer a predictive strategy that is independent of prior knowledge of well-known signatures and generalization across platforms. ML approaches will extract features of well-known good and bad URLs and generalize these features to identify new and previously undiscovered good or bad URLs given a sample of legitimate and malicious malware samples.

URL having IP

From URL, we are checking whether IP address is present or not. If URL has IP address, then there is a chance that the URL has some malicious link.

Code Snippet:

```
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)
```

URL Length

If Length of the URL is less than 54 , then the website can be phishing website.Malicious URLs are generally shorter in length than benign URLs.

Code Snippet:

```
def url_length(url):
    length=len(url)
    if(length<54):
        return -1
    elif(54<=length<=75):
        return 0
    else:
        return 1
```

Having @ symbol

If an website contain @ symbol then the website may be malicious.

Code Snippet:

```
def having_at_symbol(url):
    symbol=regex.findall(r'@',url)
    if(len(symbol)==0):
        return -1
    else:
        return 1
```

Extract prefix-suffix

If domain of the website contain '-' ,then the website is malicious. For instance,"https://www.binance-co.com/", this website contain hypen in domain , and it is classified as malicious link.

Code Snippet:

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

Extract subdomain

If there is more than one subdomain present in the URL , then the website may be malicious. For instance,the URL ‘amazon.com’ do not look suspicious, however, the same sub-string looks malicious in ‘amazon.com.support.info’.

Code Snippet:

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

SSL final state

A secure connection can be established over the internet with the aid of the Secure socket layer (SSL) or Transport level security (TLS) protocols. But it does more than just collect information. Its purpose is to securely verify the identities of the websites. Check whether website has http connection in secure way by https.

Code Snippet:

```
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','Trustwave
','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
```

Domain registration

From the url , website domain registration date is found, if it is less than 365 days then the website is phishing website.

Code Snippet:

```
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
```

HTTP Token

If sometimes , the attacker can include https part in domain of the website to make that website look like secure one.

Code Snippet:

```
def https_token(url):
    subDomain, domain, suffix = extract(url)
    host =subDomain +'.' + domain + '.' + suffix
if(host.count('https')):
    return 1
    else:
        return -1
```

Url of Anchor

When one hits a link (anchor tag) on a web page, and it opens in a new browser tab, there are chances that a hacker might have taken control over your original tab web page.while the link is opening in another tab, the attacker can redirect the original tab's URL location to a phishing page in the background, designed to look like the real original page, asking for login credentials

Code Snippet:

```
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
```

Links in tags

Code Snippet:

```
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
```

Email submits

Extracting the webpage html file and check whether the webpage is try to send mail to any other website, then the website is malicious.

Code Snippet:

```
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
```

Age of domain

The WHOIS database can be used to extract this characteristic. The majority of phishing websites are only active for a little time. For this initiative, a legal domain must have a minimum age of 12 months. Age in this context simply refers to the interval between creation and expiration times.

Code Snippet:

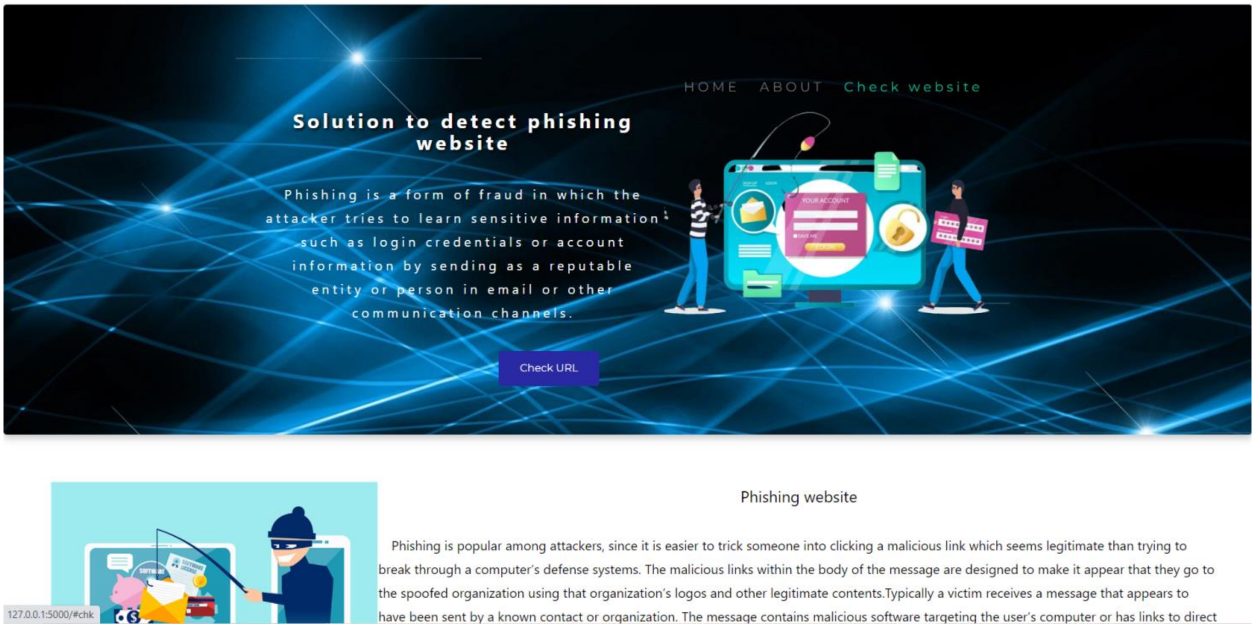
```
def age_of_domain(url):
    try:
        w = whois.whois(url)
        start_date = w.creation_date
        current_date = datetime.datetime.now()
        age =(current_date-start_date[0]).days
```

```
        if(age>=180):
            return -1
        else:
            return 1
    except Exception as e:
        print(e)
        return 0
```

7.2.1 Phishing Website

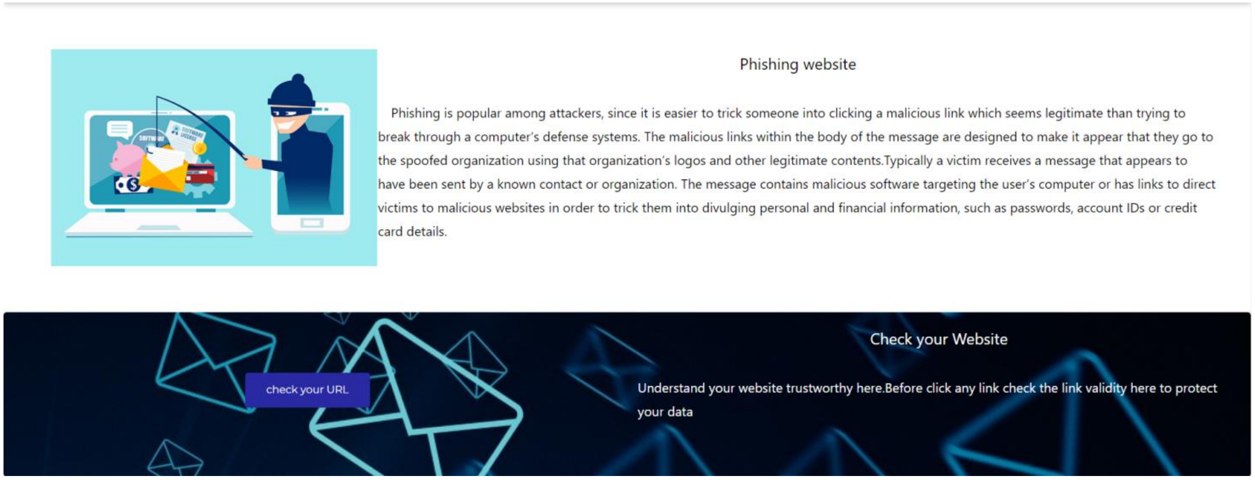
Home page

Landing page of the phishing detection page where user can find check url button to check whether the URL is good or not.



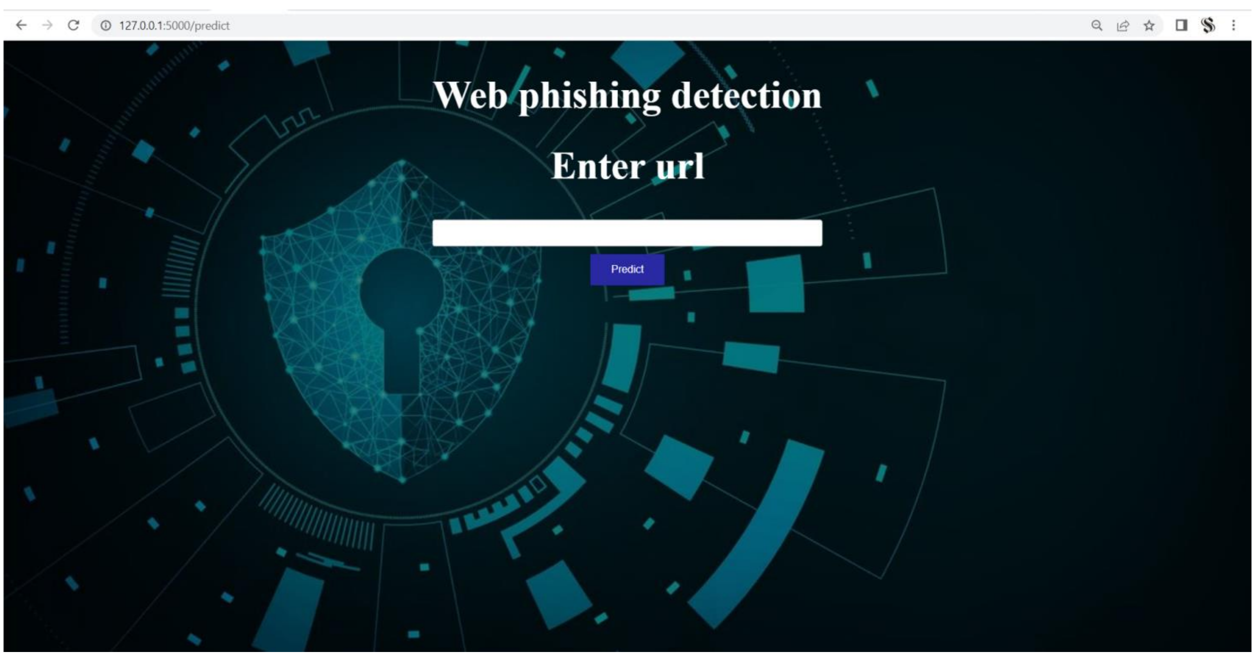
About and Check URL Section:

About Section of the page where user can know about what is phishing site and how harmful it is.

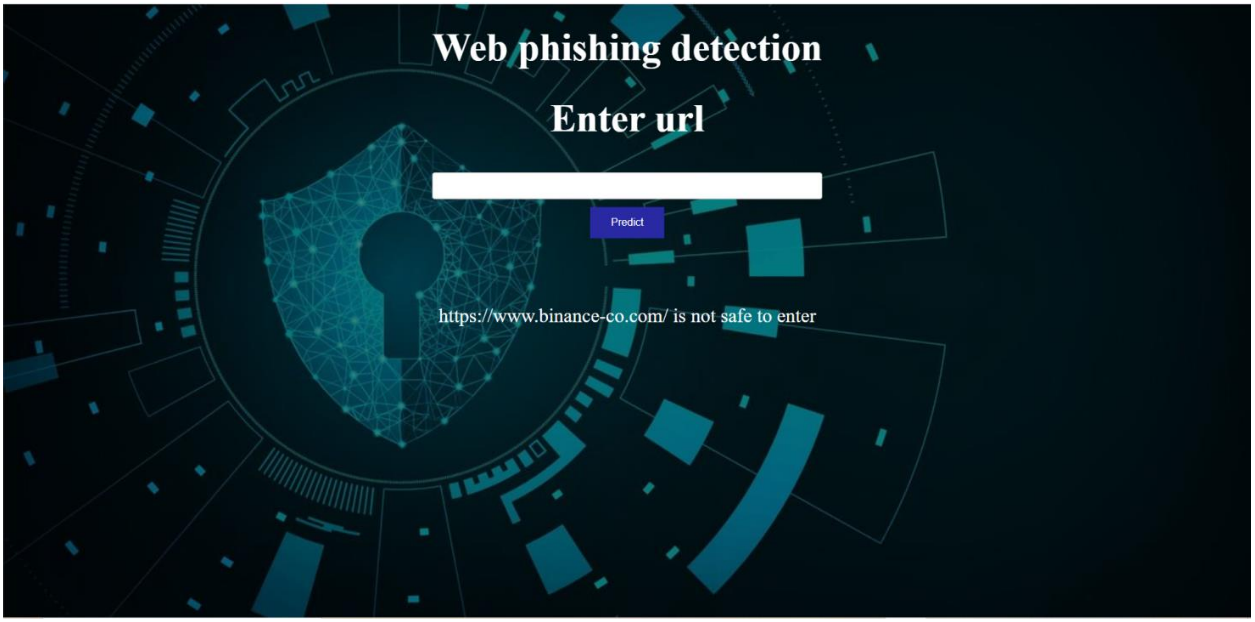


Prediction Page

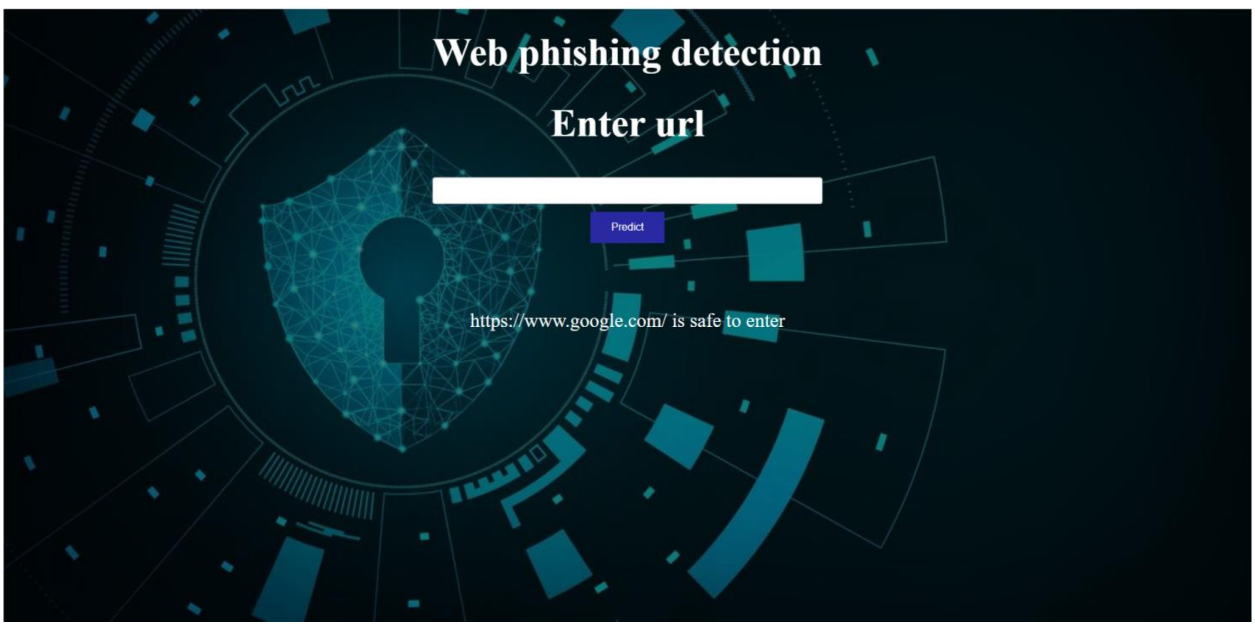
When User clicks the Check URL button , User will redirect to phishing detection page where user can enter any URL.



Output for detection for phishing URL



Output for Safe URL Prediction



Chapter 8

TESTING

8.1 Test Case

Test case ID	Feature Type & Component	Test Scenario	Steps To Execute	Test Data	Expected Result	Actual Result	Status
HomePage_TC_OO1	Functional & Home Page	Verify user is able to see check URL button to predict the trustworthiness of the URL	1.Enter URL and click go 2.User can see the landing page of the website 3.In header section , user can view various section. 4.In About section, user can know about phishing website 5.In Check section , user can view the check URL button. 6.Redirects to the phishing detection page.	https://www.amazon.com/	Application should predict whether the URL is good or bad	Working as expected	Pass
HomePage_TC_OO2	UI & Home Page	Verify the UI elements in home page	1.Enter URL and click go 2.Click on check url button 3. It redirects to check section in home page where user can again click check url a. The above fuction redirects the user to phishing detection page.	https://www.amazon.com/	Application should show below UI elements: a. User will see the result text below the predict button b. The URL Field will be	Working as expected	Pass

			b. In phishing detection page , user can enter url to predict c. Once predict button is clicked , the predicted result will be shown to user d. Again user can enter other url to check whether the provided url is good or bad.		empty c. Predict button to view the result		
Phishing Page_TC _OO3	Functional & Phishing page	Verify user is able to enter url and check whether url is good or bad	1.Enter URL(https://subhiksha.pythonanywhere.com/) and click go 2.Click on check url in check Section 3.Redirects user to phishing detection page 4.Enter url in input field 5.Click on predict button	https://www.binance-co.com/	User should get result the url is not safe to enter	Working as expected	Pass
Phishing Page_TC _OO4	Functional & Phishing page	Verify user is not getting result when url is not entered	1.Enter URL and click go 2.Click on check URL button 3.Redirects to the phishing page 4.Click on predict button without entering anything to the input field 6. The message is displayed to enter URL in input field		Application should show 'Please fill out Enter URL Field'	Working as expected	Pass

8.2 User Acceptance Testing

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	8	3	2	2	15
Duplicate	1	0	1	0	2
External	2	2	0	0	4
Fixed	9	2	3	13	27
Not Reproduced	0	0	1	0	1
Skipped	0	0	0	1	1
Won't Fix	0	4	1	1	6
Totals	20	11	8	17	56

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	9	0	0	9
Client Application	45	0	0	45
Security	2	0	0	2
Outsource Shipping	2	0	0	2
Exception Reporting	10	0	0	10
Final Report Output	3	0	0	3
Version Control	2	0	0	2

CHAPTER 9

RESULTS

9.1 Performance Metrics

Accuracy	**Random Forest** Training Accuracy - 98.93% Validation Accuracy -97.46% **Support Vector Machine** Training Accuracy – 98.42% Validation Accuracy -96.96% **Logistic Regression** Training Accuracy -92.79 % Validation Accuracy -93.44% **Decision Tree** Training Accuracy – 98.93% Validation Accuracy -96.83% **Naïve Bayes** Training Accuracy – 90.79% Validation Accuracy -91.85%	**Random Forest** Accuracy of train data = 98.93713251922208 % Accuracy of test data = 97.46720940750791 % **Support Vector Machine** Accuracy of train data = 98.4283129805518 % Accuracy of validation data = 96.96969696969697 % **Logistic Regression** Accuracy of train data = 92.79737675260064 % Accuracy of validation data = 93.441881501583 % **Decision Tree** Accuracy of train data = 98.93713251922208 % Accuracy of validation data = 96.8340117593849 % **Naïve Bayes** Accuracy of train data = 90.79601990049751 % Accuracy of validation data = 91.85888738127545 %

CHAPTER 10

ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- Identify the phishing URL's
- Differentiate legitimate and phishing links
- User friendly

DISADVANTAGES:

- Not a generalized model
- Huge number of rules

CHAPTER 11

CONCLUSION

Due to its importance in preserving privacy and ensuring security, experts are currently very interested in the detection of phishing. There are numerous ways to detect phishing. By applying machine learning, our technology seeks to improve the detection process for phishing websites. We were successful in achieving a high detection accuracy, and the findings demonstrate that the classifiers work better as we use more training data. Maximum accuracy of 97.46% is achieved for testing data with random forest classifier and we used that to deploy in the cloud. Ensemble methods like stacking also have been tried to improve the accuracy of weak learners along with hyper parameter tuning to improve the classification accuracy for both testing and training data.

CHAPTER 12

FUTURE SCOPE

We plan to develop system add-ons in the future, and if we can obtain a structured dataset of phishing, we will be able to detect it much more quickly than with any other method. In the future work a web extension can be made so that the working will be much simplified for the end users / customers.

Chapter 13

APPENDIX

13.1 Source Code

Flask Integration with scoring end point

```
from flask import Flask, render_template

from flask import request

import pickle

import inputScript

import requests

app = Flask(__name__)


@app.route('/')

def home():

    return render_template('phishing.html')


@app.route('/predict',methods=['POST'])

def predict():

    return render_template('home.html')


@app.route('/result',methods=['POST'])

def result():
```

```
#For rendering results on HTML GUI
int_features = request.form['url']
print(int_features)
checkprediction = inputScript.main(int_features)
API_KEY = "iCvI0Alk0K9_Cp4834eaop53ZhOOtjQ29ylS-6craV-p"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":
 API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]

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

#int_features = "https://www.binance-co.com/"
checkprediction = inputScript.main(int_features)

# NOTE: manually define and pass the array(s) of values to be scored in the next line
payload_scoring = {"input_data": [{"field":
[["index","having_IPhaving_IP_Address","URLURL_Length","Shortining_Service",

"having_At_Symbol","double_slash_redirecting","Prefix_Suffix","having_Sub_Domain",

"SSLfinal_State","Domain_registeration_length","Favicon","port","HTTPS_token",

"Request_URL","URL_of_Anchor","Links_in_tags","SFH","Submitting_to_email",
```

```
"Abnormal_URL","Redirect","on_mouseover","RightClick","popUpWidnow","Iframe",

"age_of_domain","DNSRecord","web_traffic","Page_Rank","Google_Index",
                                        "Links_pointing_to_page","Statistical_report"]],
                              "values": checkprediction}]}
   response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/30400c66-72c5-4e4b-8234-a746f751e5e4/predictions?version=2022-11-18',
                              json=payload_scoring, headers={'Authorization': 'Bearer ' + mltoken})
   print("Scoring response")
   predictions=response_scoring.json()
   print(predictions)
   pred=predictions['predictions'][0]['values'][0][0]
   print(pred)
   res=""
   if(pred==1):
      res=int_features+" is not safe to enter"
   elif(pred==-1):
      res=int_features+" is safe to enter"
   print(res)
   return render_template('home.html', prediction_text= res)
if __name__ == '__main__':
   app.run()
```

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 0
    else:
        return 1



def url_short(url):
    return 0


def having_at_symbol(url):
    symbol=regex.findall(r'@',url)
    if(len(symbol)==0):
        return -1
    else:
        return 1


def doubleSlash(url):

    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','Trustwave
','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):
    return 0

def port(url):
    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):
    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):
    return 0


def redirect(url):
    return 0
```

```
def on_mouseover(url):
    return 0


def rightClick(url):
    return 0


def popup(url):
    return 0


def iframe(url):
    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):
    return 0


def web_traffic(url):
    return 0


def page_rank(url):
    return 0


def google_index(url):
    return 0




def links_pointing(url):
    return 0


def statistical(url):
    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("length:",len(check))
    print("final op:",check)
    return check
```

Phishing.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
```

```
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
  <link href="https://fonts.googleapis.com/css?family=Montserrat" rel="stylesheet">
  <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
  <style>
  body {
    font: 20px Montserrat, sans-serif;
    line-height: 1.8;
    color: #f5f6f7;
  }
  p {font-size: 16px;}
  .margin {margin-bottom: 45px;}
  /*.bg-1 {
    background-color: #1abc9c; /* Green */
    color: #ffffff;
  }*/
  .bg-2 {
    background-color: #474e5d; /* Dark Blue */
    color: #ffffff;
  }
  .bg-3 {
```

```
  background-color: #ffffff; /* White */
  color: #555555;
}
.bg-4 {
  background-color: #2f2f2f; /* Black Gray */
  color: #fff;
}
.container-fluid {
  padding-top: 70px;
  padding-bottom: 70px;

}
.navbar {
 background-color: #1abc9c;
  border: 0;
  border-radius: 0;
  margin-bottom: 0;
  font-size: 20px;
  letter-spacing: 5px;

}
.navbar-nav  li a:hover {
  color: #1abc9c !important;
```

```
 }
 #sliding{

       box-shadow: 0 4px 8px 0 rgba(0,0,0,0.2);
 transition: 0.8s;
 width: 100%;
 border-radius: 5px;
 color: white;
 align:center
 background-color: #1abc9c;
 background-image:url("/static/images/bg1.jpg");
 padding-top:30px;

 }
 #sliding:hover {
 box-shadow: 0 8px 16px 0 rgba(0,0,0,0.2);
}
.about{
padding-top: 70px;
padding-left:70px;
padding-right:50px;
   padding-bottom: 70px;
color:black;
```

```
align:center;
height:400px;
}
.check{
box-shadow: 0 4px 8px 0 rgba(0,0,0,0.2);
  transition: 0.8s;
  width: 100%;
  border-radius: 5px;
  color: white;
  align:center;
  background-image:url("/static/images/bg6.jpeg");
  background-size: 100% 100%;
  background-repeat: no-repeat;
  background-size: cover;
  display: flex;
}
.column{
 flex: 40%;
 padding: 16px;
              height: 250px;}
.btn{
background-color: #2929a3;
border: none;
```

```
  color: white;
  padding: 15px 32px;
  text-align: center;
  align:left;
  text-decoration: none;
  display: inline;
  font-size: 16px;
  margin: 4px 2px;
  cursor: pointer;
}

  </style>
</head>
<body>

<!-- Navbar -->


<center>
<nav class="navbar navbar-default" id="sliding">
  <div class="container">
        <div class="container-fluid bg-1 text-center" >
    <div class="collapse navbar-collapse" id="myNavbar">
```

```html
      <ul class="nav navbar-nav navbar-right">
        <li><a href="#sliding">HOME</a></li>
        <li><a href="#abt">ABOUT</a></li>
                        <li><a href="#chk">Check website</a></li>
      </ul>
    </div>

  <img src="/static/images/phish.png" class="img-responsive" align="right" alt="Bird"
width="500" height="500">
  <h3 class="margin" style="color:white; text-shadow:2px 2px 4px #000000;font-size:
30px;font-weight: bold;">Solution to detect phishing website</h3>
  <h4 class="margin" style="text-align:left;display:inline">Phishing is a form of fraud in which
the attacker tries to learn sensitive information such as login credentials or account information
by sending as a reputable entity or person in email or other communication channels.</h4>
  <br>
  <br>
  <a href="#chk"><button class="btn" style="position: relative;right:120px;">Check
URL</button></a>
  </div>
  </div>
</nav>
</center>
```

```html
<div class="about" id="abt">

<img src="/static/images/about.png" class="img-responsive" align="left" alt="Bird"
width="500" height="500">

  <h3 class="margin" style="text-align:center">Phishing website</h3>

  <h4 class="margin" style="text-align:right;display:inline;padding-right:20px;padding-
left:20px;">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.Typically a victim receives a message that appears to have been sent by a known
contact or organization. The message contains malicious software targeting the user’s computer
or has links to direct victims to malicious websites in order to trick them into divulging personal
and financial information, such as passwords, account IDs or credit card details.</h4>

</div>

<br>

<br>


<div class="check"id="chk">

		<div class="column">

		<br>

		<br>

		 <form action="/predict" method="post">

<a href="#chk"><button class="btn" style="position: relative;left:350px;">check your
URL</button></a>
```

```
</form>

<br>

<br>

              </div>

              <div class="column">

               <h3 class="margin" style="text-align:center">Check your Website</h3>

<h4 class="margin" style="text-align:right;display:inline;">Understand your website trustworthy here.Before click any link check the link validity here to protect your data</h4>

              </div>

</div>

</body>

</html>
```

Home.html

```
<html>

<style>

body{

background-image: linear-gradient(rgba(0, 0, 0, 0.5), rgba(0, 0, 0, 0.5)), url("/static/images/urlbg2.jpg");

  height: 50%;

  background-position: center;

  background-repeat: no-repeat;

  background-size: cover;
```

```
  position: relative;
}
.container{
color:white;
font-size:30px;
text-align: center;
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
}
input[type=text], select {
  width: 100%;
  padding: 12px 20px;
  margin: 8px 0;
  display: inline-block;
  border: 1px solid #ccc;
  border-radius: 4px;
  box-sizing: border-box;
}
btn:hover {
  background-color: #31499F;
}
```

```
.btn{

background-color: #2929a3;

border: none;

  color: white;

  padding: 15px 32px;

  text-align: center;

  align:left;

  text-decoration: none;

  display: inline;

  font-size: 16px;

  margin: 4px 2px;

  cursor: pointer;

}


</style>

<body>

<div class="container">

<h1>Web phishing detection</h1>

<h1>Enter url</h1>

<form action="/result" method="post">

<input type="text" name="url" required="required" />

<button type="submit" class="btn">Predict</button>

</form>
```


{{ prediction_text }}

</div>

</body>

</html>

GitHub and project link

GitHub link - IBM-EPBL/IBM-Project-2617-1658478323: Web Phishing Detection (github.com)

Demo link -
https://drive.google.com/file/d/1re8S3V3zvKgTsrXMBpOhW6JCXCjUa4I1/view?usp=sharing

Website link- https://subhiksha.pythonanywhere.com