DETECTION OF FRAUDULENT WEBSITES USING MACHINE LEARNING

A MINI PROJECT REPORT

Submitted by

A.Vaishnavi	G.Sahruthi	P.Varshini	A.Nikshitha
20RH1A6604	20RH1A6625	20RH1A6651	20RH1A6601

Under the Esteemed Guidance of

Mrs. P. Gayatri

Assistant Professor

in partial fulfillment of the Academic Requirements for the Degree of

BACHELOR OF TECHNOLOGY

CSE(AI & ML)

MALLA REDDY ENGINEERING COLLEGE FOR WOMEN

(Autonomous Institution-UGC, Govt. of India)

Accredited by NBA & NAAC with 'A' Grade, UGC, Govt. of India

NIRF Indian Ranking–2018, Accepted by MHRD, Govt. of India

Permanently Affiliated to JNTUH, Approved by AICTE, ISO 9001:2015 Certified Institution AAAA+

Rated by Digital Learning Magazine, AAA+ Rated by Careers 360 Magazine,

6th Rank CSR, Platinum Rated by AICTE-CII Survey, Top 100 Rank band by ARIIA, MHRD, Govt. of India

National Ranking-Top 100 Rank band by Outlook, National Ranking-Top 100 Rank band by Times Ne ws Magazine

Maisammaguda, Dhullapally, Secunderabad, Kompally-500100

June- 2023

MALLA REDDY ENGINEERING COLLEGE FOR WOMEN

(Autonomous Institution-UGC, Govt. of India)

Accredited by NBA & NAAC with 'A' Grade, UGC, Govt. of India

NIRF Indian Ranking–2018, Accepted by MHRD, Govt. of India
Permanently Affiliated to JNTUH, Approved by AICTE, ISO 9001:2015 Certified Institution
AAAA+ Rated by Digital Learning Magazine, AAA+ Rated by Careers 360 Magazine,
6th Rank CSR, Platinum Rated by AICTE-CII Survey, Top 100 Rank band by ARIIA, MHRD, Govt. of India National Ranking-Top 100 Rank band by Outlook, National Ranking-Top 100 Rank band by Times Ne ws Magazine
Maisammaguda, Dhullapally, Secunderabad, Kompally-500100

DEPARTMENT OF CSE (AI &ML)

CERTIFICATE

This is to certify that the Mini Project work entitled **Detection of Fraudulent Websites using Machine Learning** is carried out by **A.Vaishnavi (20RH1A6604), G.Sahruthi(20RH1A6625), P.Varshini(20RH1A6651), A.Nikshitha(20RH1A6601)** in partial fulfillment for the award of degree of **BACHELOR OF TECHNOLOGY** in CSE(AI & ML), Jawaharlal Nehru Technological University, Hyderabad during the academic year 2022-2023.

Supervisor's Signature

Mrs. P. Gayatri
Assistant Professor

Head of the Department

Dr. G. Kalpana
Associate Professor

External Examiner

MALLA REDDY ENGINEERING COLLEGE FOR WOMEN

(Autonomous Institution –UGC,Govt. of India)

(Permanently Affiliated to JNTU, Hyderabad, Approved by AICTE - ISO 9001:2015 Certified)

Accredited by NBA & NAAC with 'A' Grade

NIRF India Ranking 2018, Accepted by MHRD, Govt. of India

Maisammaguda, Dhulapally (Post Via Hakimpet), Secunderabad – 500100

Department of CSE (AI & ML)

DECLARATION

We hereby declare that the Mini Project entitled **DETECTION OF FRAUDULENT WEBSITES USING MACHINE LEARNING** submitted to Malla Reddy Engineering College For Women affiliated to Jawaharlal Nehru Technological University, Hyderabad (JNTUH) for the award of the Degree of Bachelor of Technology in CSE(AI & ML) is a result of original research work done by us. It is further declared that the Mini Project report or any part there of has not been previously submitted to any University or Institute for the award of Degree.

A.Vaishnavi (20RH1A6604)

G.Sahruthi(20RH1A6625)

P.Varshini(20RH1A6651)

A.Nikshitha(20RH1A6601)

ACKNOWLEDGEMENT

We feel ourselves honored and privileged to place our warm salutation to our college **Malla Reddy Engineering College for Women** and **Department of CSE(AI &ML)** which gave us the opportunity to have expertise in engineering and profound technical knowledge.

We would like to deeply thank our Honorable Minister of Telangana State **Sri**.**Ch. Malla Reddy Garu,** founder chairman MRGI, the largest cluster of institutions in the state of Telangana for providing us with all the resources in the college to make our project success.

We wish to convey gratitude to our **Principal Dr. Y. Madhavee Latha,** for providing us with the environment and mean to enrich our skills and motivating us in our endeavor and helping us to realize our full potential.

We would like to thank **Prof. A. Radha Rani**, Director of Computer Science and Engineering & Information Technology for encouraging us to take up a project on this subject and motivating us towards the Project Work.

We express our sincere gratitude to **Dr. G. Kalpana, Associate Professor, Head of the Department** of **CSE(AI&ML)** for inspiring us to take up a project on this subject and successfully guiding us towards its completion.

We would also like to thank our project coordinator **Mr. B. Srinivasulu, Assistant Professor, Department of CSE(AI&ML)** for his kind encouragement and overall guidance in viewing this program a good asset with profound gratitude.

We would like to thank our internal guide **Mrs.P.Gayatri, Assistant Professor, Department of CSE(AI&ML)** and all the Faculty members for their valuable guidance and encouragement towards the completion of our project work.

With Regards and Gratitude

A.Vaishnavi (20RH1A6604)

G.Sahruthi(20RH1A6625)

P.Varshini(20RH1A6651)

A.Nikshitha(20RH1A6601)

ABSTRACT

Phishing is an internet scam in which an attacker sends out fake messages that look to come from a trusted source. A URL or file will be included in the mail, which when clicked will steal personal information or infect a computer with a virus. Traditionally, phishing attempts were carried out through wide-scale spam campaigns that targeted broad groups of people indiscriminately. The goal was to get as many people to click on a link or open an infected file as possible. There are various approaches to detect this type of attack. One of the approaches is machine learning. The URL's received by the user will be given input to the machine learning model then the algorithm will process the input and display the output whether it is phishing or legitimate. There are various ML algorithms like SVM, Neural Networks, Random Forest, Decision Tree, XG boost etc. that can be used to classify these URLs. The proposed approach deals with the Gradient boost classifier, Random Forest, Decision Tree classifiers and seven more. The proposed approach effectively classified the Phishing and Legitimate URLs with an accuracy of 97.4% for Gradient boost classifier..

LIST OF FIGURES

LIST OF TABLES

INDEX

CHAPTER 1

INTRODUCTION

Nowadays 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. In United States businesses, there is a loss of US$2billion per year because their clients become victim to phishing. In 3rd Microsoft Computing Safer Index Report released in February 2014, it was estimated that the annual worldwide impact of phishing could be as high as $5 billion . 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. The general method to detect phishing websites by updating blacklisted URLs, Internet Protocol (IP) to the antivirus database which is also known as “blacklist" method. To evade blacklists attackers uses creative techniques to fool users by modifying the URL to appear legitimate via obfuscation and many other simple techniques including: fast-flux, in which proxies are automatically generated to host the web-page; algorithmic generation of new URLs; etc.

Major drawback of this method is that, it cannot detect zero-hour phishing attack. Heuristic based detection which includes characteristics that are found to exist in phishing attacks in reality and can detect zero-hour phishing attack, but the characteristics are not guaranteed to always exist in such attacks and false positive rate in detection is very high. To overcome the drawbacks of blacklist and heuristics based method, many security researchers now focused on machine learning techniques.

Machine learning technology consists of a many algorithms which requires past data to make a decision or prediction on future data. Using this technique, algorithm will analyze various blacklisted and legitimate URLs and their features to accurately detect the phishing websites including zero- hour phishing websites.

CHAPTER 2

LITERATURE SURVEY

2.1.Existing System:

S.no	Authors	Contribution	Limitations
1.	Jain A.K., and Gupta B.B (2017)	A combination of the NB and SVM algorithms were used to find the dangerous websites.	SVM and NB are both slow learners and do not keep track of previous outcomes in memory. As a result, the URLdetector's effectiveness can be diminished.
2.	Purbay M., and Kumar D (2022)	URL classification was done using a variety of ML techniques.	The effectiveness of several ML method types was compared. The algorithms' retrieval capabilities, however, were not discussed.
3.	Gandotra E., and Gupta D. (2020)	a variety of categorization algorithms were used to find dangerous URLs.	The results of the studies showed that the system's performance was superior to that of other ML techniques. It cannot handle bigger amounts of data, though.
4.	Hung Le et al. (2018)	a deep learning-based URL detector was proposed. According to authors, the approach can generate insights from URL.	Deep learning techniques require more time to complete a task. Additionally, it parses the URL and compares it to the library to produce a result.

5.	Hong J. et al., (2021)	created a crawler to harvest URLs from data storage systems. Lexical features were used to locate the phishing websites.	A dataset collected via a crawler was used to evaluate performance. As a result, there is no guarantee regarding the URL detector's performance with real-time URLs.
6.	Kumar J. et al. (2020)	created a URL detector using a collection of blacklisted websites. In order to distinguish between dangerous and trustworthy websites, a lexical feature technique was also used.	The performance of the detector using real-time URLs can be affected by the authors' use of an older dataset.
7.	Hassan Y.A. and Abdelfettah B. (2021)	For classifying webpages and identifying phishing websites, a URL detector has been proposed. The performance was enhanced using GA method.	Although the GA-based URL detector performed better, prediction time was quite long for complex sets of URLs.
8.	Rao RS and Pais AR. (2021)	The page characteristics used by authors include the logo, favicon, scripts, and styles.	Because a server was used in the approach to update the page properties, the detecting system performed less well.
9.	Aljofey A et al. (2022)	a phishing page detection system based on CNN. To discover URLs, a sequential pattern is employed.	According to the available studies, CNN works better at fetching images than words.

10.	AlEroud A and Karabatis G (2020)	The research uses a generative adversarial network to get around a detection mechanism.	By learning about the environment around it, a neural network-based detection system can recognise the impression of an unfavourable network.

Table 2.1-Literature Review

2.2 Proposed System:

The goal of the "Detection of Fraudulent Websites Using ML" project is to identify fraudulent or phishing websites and offer users with a safety % indicating the amount of danger associated with the website. Based on the safety percentage displayed, an individual can proceed to the website with one click. We can accomplish precise identification of fraudulent websites by training machine learning algorithms, making them an effective and efficient method for detecting and flagging potentially misleading online platforms.

One of the most popular and effective kinds of machine learning is supervised machine learning. Supervised learning is utilised anytime a given collection of characteristics is used to predict a certain outcome/label. Creating a machine learning model using our training set of features-label pairs in order to produce accurate predictions for fresh, never-before-seen data.

CHAPTER 3

SOFTWARE REQUIREMENT ANALYSIS

3.1 Problem Definition:

Phishing websites are phoney websites designed to deceive visitors into providing sensitive information such as login credentials, financial information, or personal information. These websites are intended to seem like official websites, frequently copying well-known businesses, organisations, or financial institutions. Phishing websites often utilise a variety of false approaches to persuade visitors to disclose sensitive information.

We propose a project called Phishing website detection to detect these websites, which entails the process of identifying and stopping people from accessing phoney websites meant to fool and steal critical information. The goal is to identify and save people from phishing websites, hence safeguarding individuals and organisations from phishing assaults.

Phishing website detection entails using many techniques, technologies, and analysis methodologies to differentiate between legal and malicious websites; the most popular way is machine learning.

3.2 Modules and Functionalities:

1)Data Collection: Gather a dataset of labeled URLs, including both legitimate and phishing URLs. This dataset will serve as the basis for training your machine learning model.

2)Feature Extraction: Extract relevant features from the URLs that can help distinguish between legitimate and phishing URLs. Common features include domain information, URL length, presence of suspicious keywords, and use of special characters.

3)Preprocessing: Clean and preprocess the extracted features to ensure they are in a suitable format for training the machine learning model. This may involve tasks such as normalization, encoding categorical variables, and handling missing data.

4)Machine Learning Algorithms: Choose and implement appropriate machine learning algorithms for classification, as phishing URL detection is essentially a binary classification problem. Commonly used algorithms include decision trees, random forests, support vector machines (SVM), and neural networks.

5)Training: Split your dataset into training and testing sets. Use the training set to train your machine learning model on the labeled examples. The model learns to identify patterns and characteristics that differentiate between legitimate and phishing URLs.

6)Evaluation: Evaluate the performance of your trained model using the testing set. Common evaluation metrics include accuracy, precision, recall, and F1 score. This step helps assess how well your model generalizes to new, unseen data.

7)Deployment: Once you are satisfied with the performance of your model, deploy it in a production environment where it can be used to classify new URLs in real-time. This may involve integrating the model into an existing system or building a new application around it.

3.3.1 Software Requirements:

Operating system	:	Windows, Ultimate, Linux, Mac.
Front-End	:	Python, JavaScript, HTML, CSS.
Coding Language	:	Python.
Software Environment	:	Jupyter Notebook.

3.3.2 Hardware Requirements:

System	:	Intel i Processor.
RAM	:	4GB.

CHAPTER 4

SOFTWARE DESIGN

4.1 System Architecture:

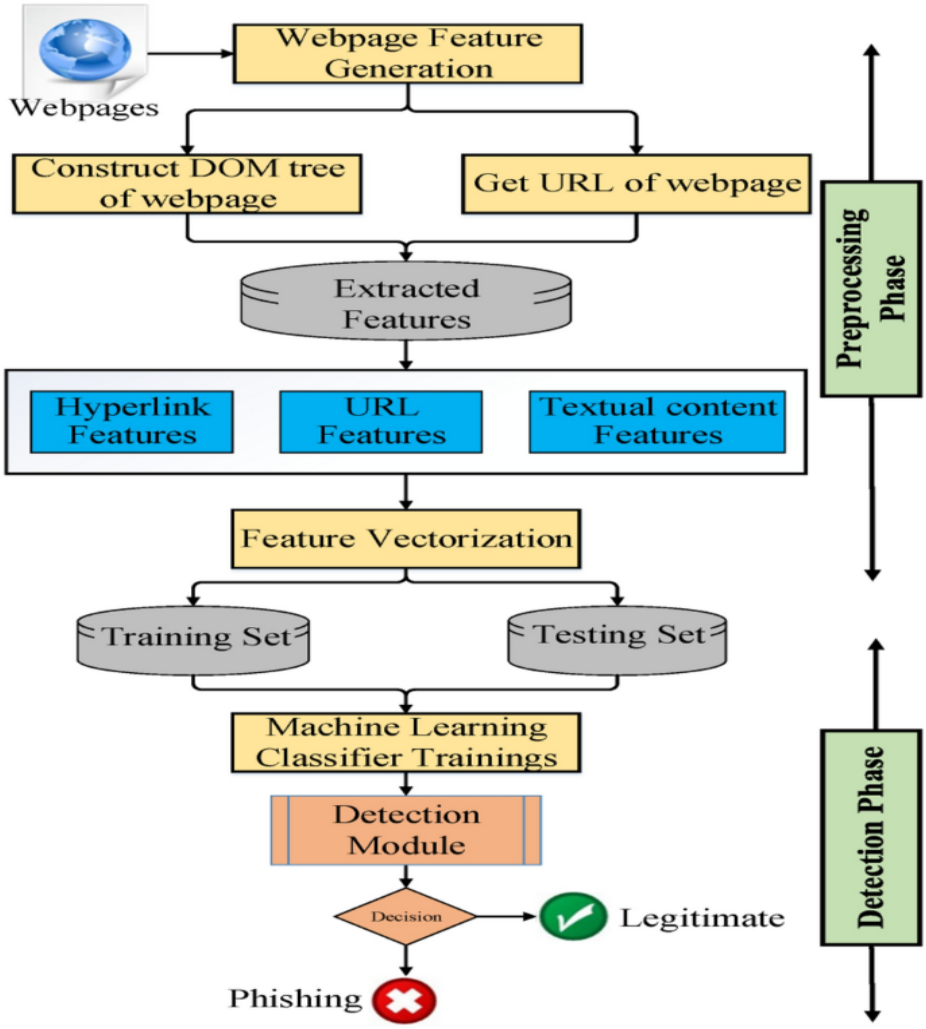


Fig.4.1-System Architecture

We have implemented python program to extract features from URL. Below are the features that we have extracted for detection of phishing URLs.

1) Presence of IP address in URL: If IP address present in URL then the feature is set to 1 else set to 0. Most of the benign sites do not use IP address as an URL to download a webpage. Use of IP address in URL indicates that attacker is trying to steal sensitive information.

2) Presence of @ symbol in URL: If @ symbol present in URL then the feature is set to 1 else set to 0. Phishers add special symbol @ in the URL leads the browser to ignore everything preceding the "@" symbol and the real address often follows the "@" symbol [4].

3) Number of dots in Hostname: Phishing URLs have many dots in URL. For example http://shop.fun.amazon.phishing.com, in this URL phishing.com is an actual domain name, whereas use of "amazon" word is to trick users to click on it. Average number of dots in benign URLs is 3. If the number of dots in URLs is more than 3 then the feature is set to 1 else to 0.

4) Prefix or Suffix separated by (-) to domain: If domain name separated by dash (-) symbol then feature is set to 1 else to 0. The dash symbol is rarely used in legitimate URLs. Phishers add dash symbol (-) to the domain name so that users feel that they are dealing with a legitimate webpage. For example Actual site is http://www.onlineamazon.com but phisher can create another fake website like http://www.online-amazon.com to confuse the innocent users.

5) URL redirection: If "//" present in URL path then feature is set to 1 else to 0. The existence of "//" within the URL path means that the user will be redirected to another website [4].

6) HTTPS token in URL: If HTTPS token present in International Journal of Computer Applications (0975 – 8887) Volume 181 – No. 23, October 2018 46 URL then the feature is set to 1 else to 0. Phishers may add the "HTTPS" token to the domain part of a URL in order to trick users. For example, http://https-wwwpaypal-it-mpp-home.soft-hair.com [4].

7) Information submission to Email: Phisher might use "mail()" or "mailto:" functions to redirect the user's information to his personal email[4]. If such functions are present in the URL then feature is set to 1 else to 0.

8) URL Shortening Services "TinyURL": TinyURL service allows phisher to hide long phishing URL by making it short. The goal is to redirect user to phishing websites. If the URL is crafted using shortening services (like bit.ly) then feature is set to 1 else 0.

9) Length of Host name: Average length of the benign URLs is found to be a 25, If URL's length is greater than 25 then the feature is set to 1 else to 0

10) Presence of sensitive words in URL: Phishing sites use sensitive words in its URL so that users feel that they are dealing with a legitimate webpage. Below are the words that found in many phishing URLs :- 'confirm', 'account', 'banking', 'secure', 'ebyisapi', 'webscr', 'signin', 'mail', 'install', 'toolbar', 'backup', 'paypal', 'password', 'username', etc;

11) Number of slash in URL: The number of slashes in benign URLs is found to be a 5; if number of slashes in URL is greater than 5 then the feature is set to 1 else to 0.

12) Presence of Unicode in URL: Phishers can make a use of Unicode characters in URL to trick users to click on it. For example the domain "xn--80ak6aa92e.com" is equivalent to "apple.com". Visible URL to user is "apple.com" but after clicking on this URL, user will visit to "xn--80ak6aa92e.com" which is a phishing site.

13) Age of SSL Certificate: The existence of HTTPS is very important in giving the impression of website legitimacy [4]. But minimum age of the SSL certificate of benign website is between 1 year to 2 year.

14) URL of Anchor: We have extracted this feature by crawling the source code oh the URL. URL of the anchor is defined by tag. If the tag has a maximum number of hyperlinks which are from the other domain then the feature is set to 1 else to 0.

15) IFRAME: We have extracted this feature by crawling the source code of the URL. This tag is used to add another web page into existing main webpage. Phishers can make use of the "iframe" tag and make it invisible i.e. without frame borders. Since border of inserted webpage is invisible, user seems that the inserted web page is also the part of the main web page and can enter sensitive information.

16) Website Rank: We extracted the rank of websites and compare it with the first One hundred thousand websites of Alexa database. If rank of the website is greater than 10,0000 then feature is set to 1 else to 0.

4.2 UML Diagrams:

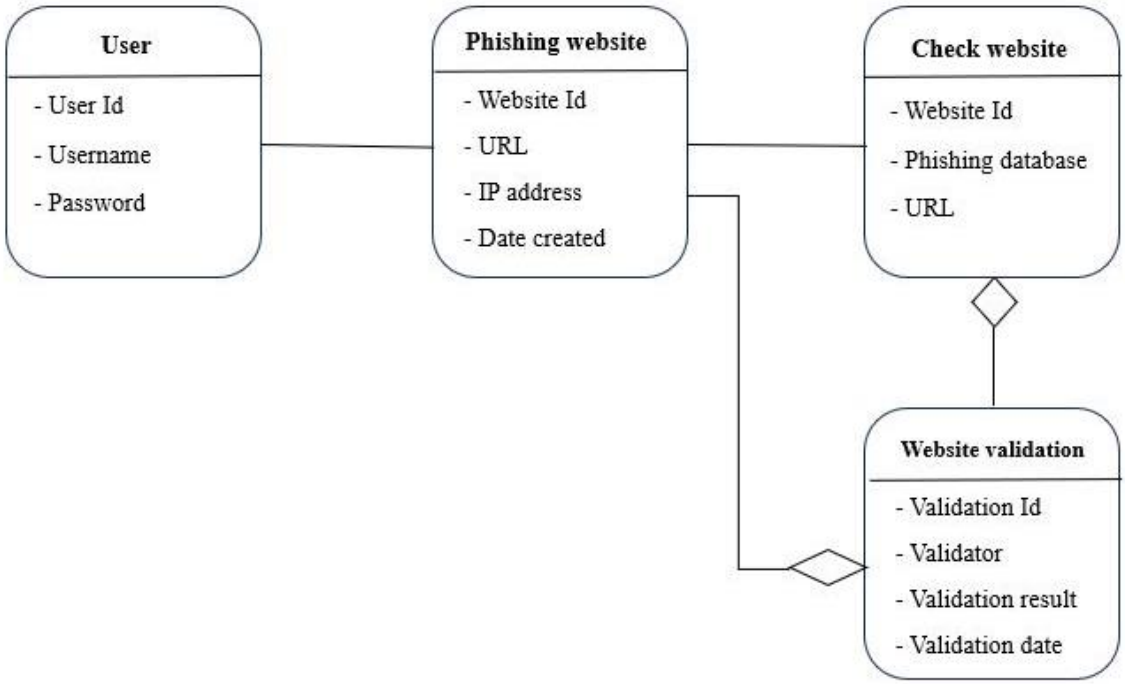


Fig.4.2.1-Class diagram

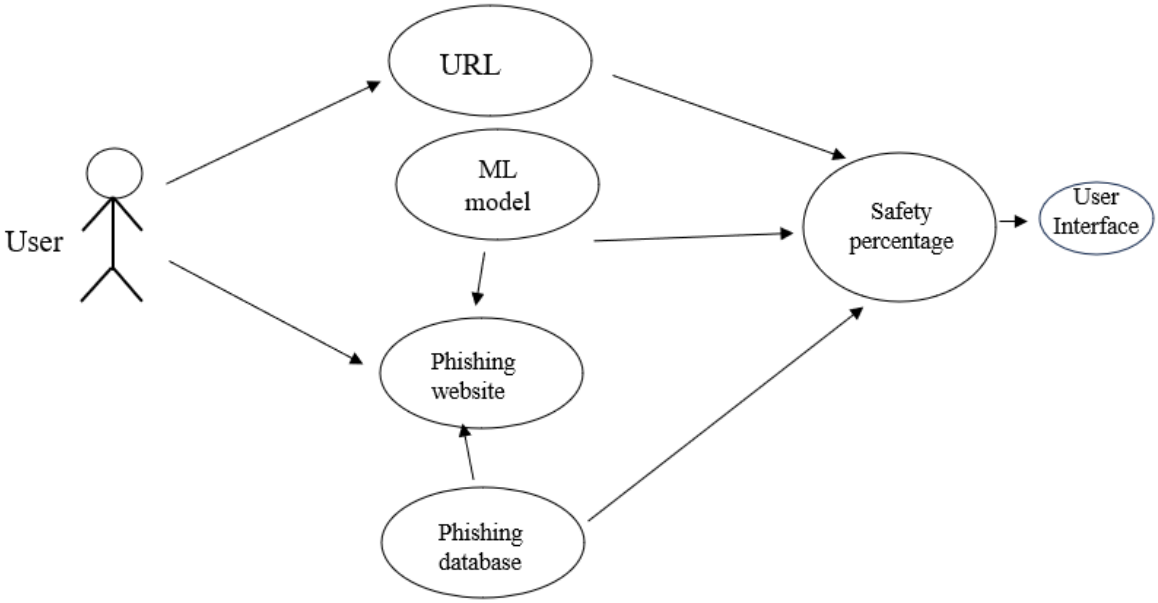


Fig.4.2.2-Use case diagram

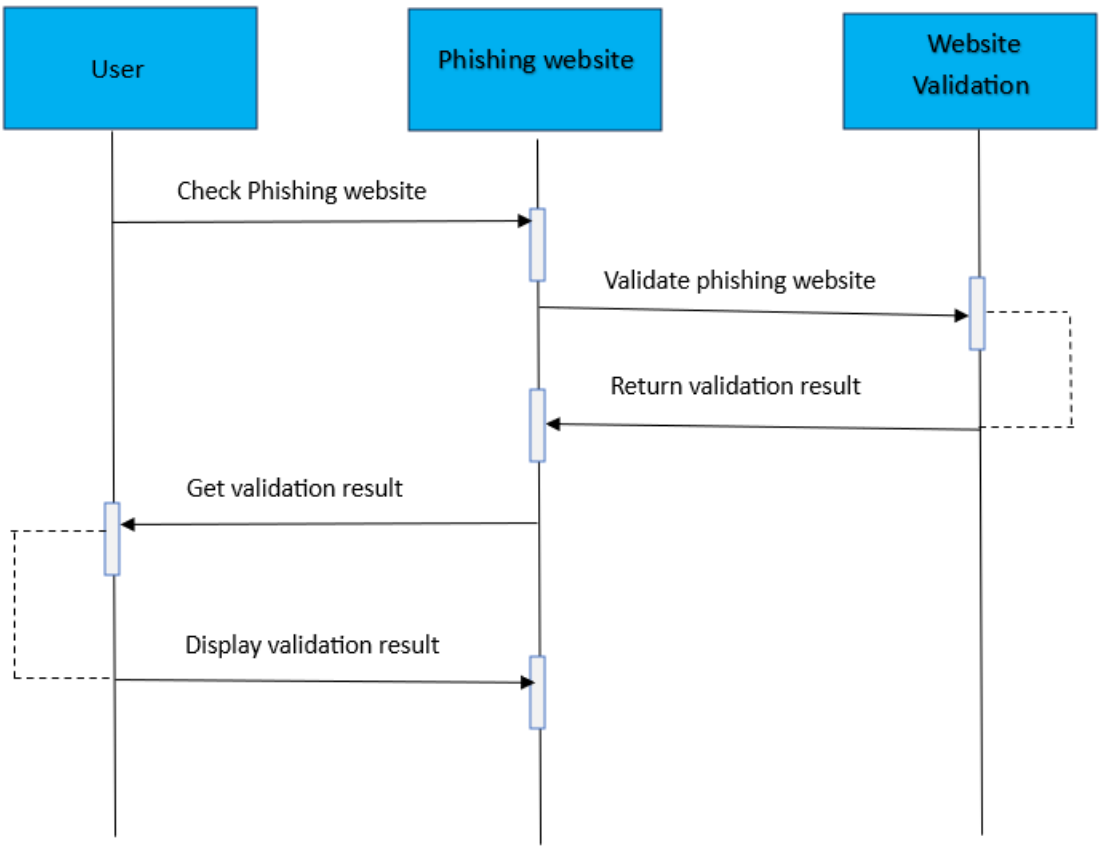


Fig.4.2.3-Sequence diagram

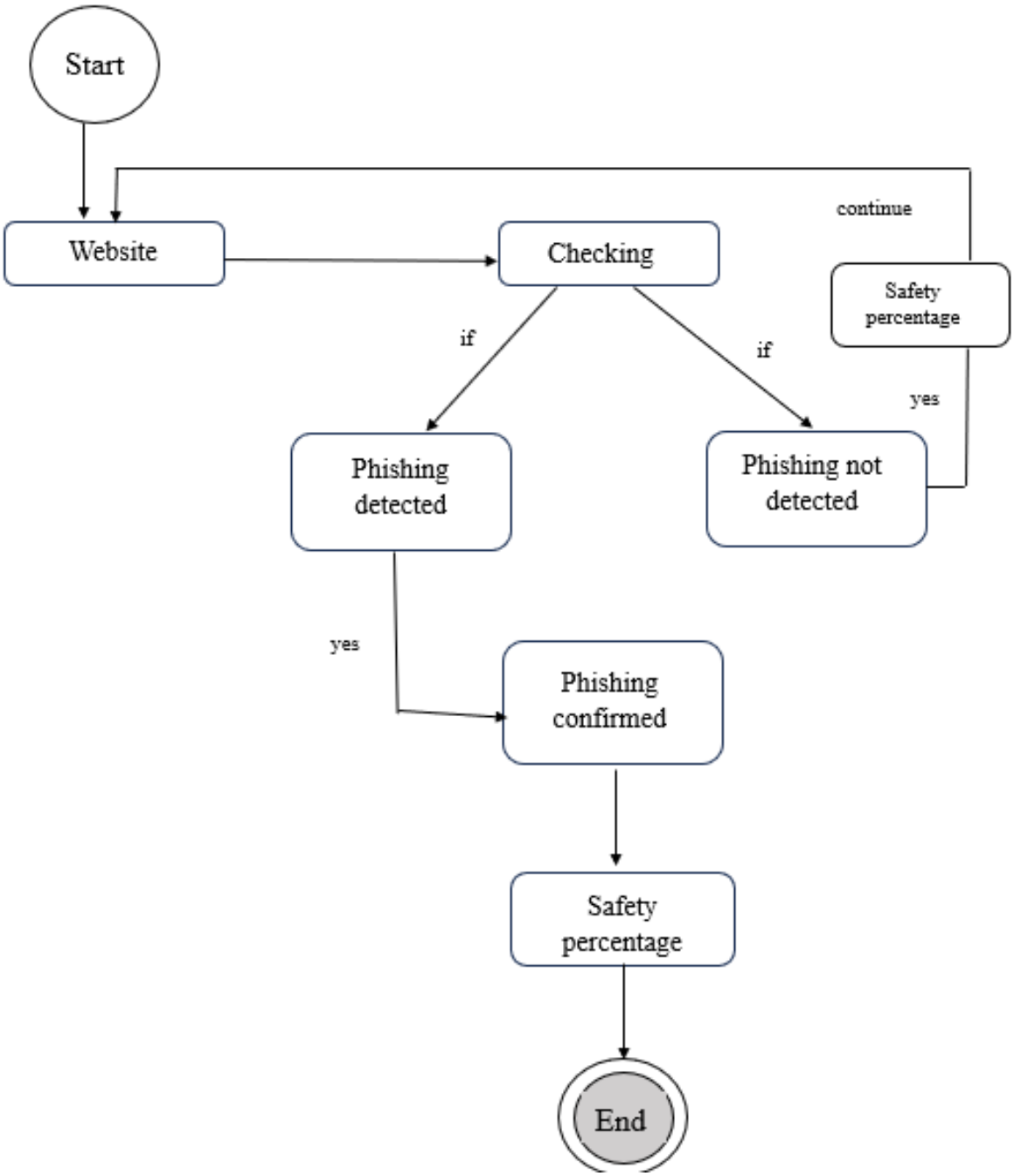


Fig.4.2.4-State Machine Chart

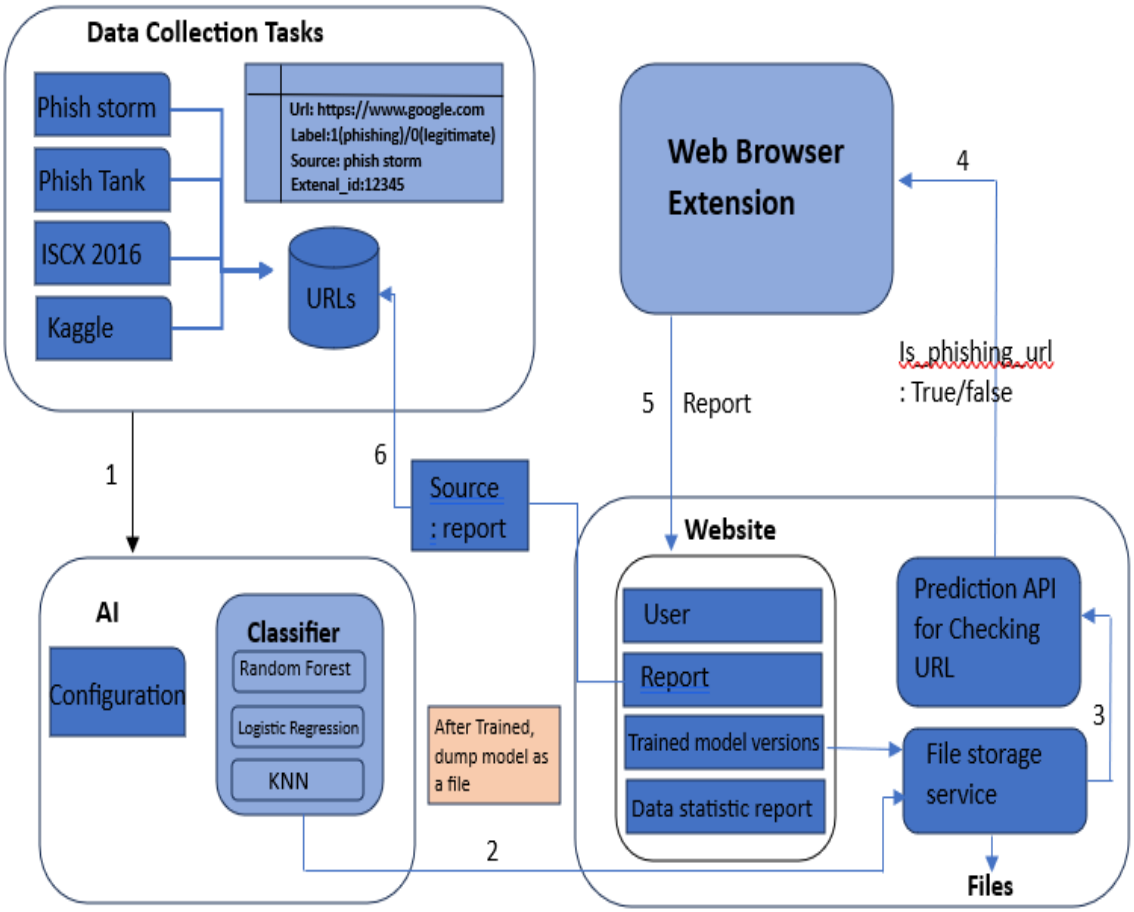


Fig.4.2.5-Deployment chart

CHAPTER 5

IMPLEMENTATION

5.1 Libraries

1.beautifulsoup4==4.9.3-BeautifulSoup: A Python library for parsing HTML and XML documents. It helps extract relevant information from webpages, such as form fields or specific elements, which can aid in identifying phishing elements.

2.scikit_learn==1.0.1- A popular machine learning library in Python that provides various algorithms for classification tasks. It can be used to build machine learning models to detect phishing URLs based on features extracted from URLs.

3.Flask==2.0.2-Flask is a popular web framework for building Python web applications. However, Flask itself does not provide built-in functionality for detecting phishing URLs or malicious websites.

Flask can be used as a framework to build a web application that incorporates these techniques for phishing URL detection. However, the actual implementation would involve using other libraries or services to perform the detection, while Flask would handle the web application's routing, request handling, and response generation.

It's important to note that phishing detection is a complex problem, and no single approach can guarantee 100% accuracy. It's always recommended to combine multiple techniques and stay updated with the latest security practices to enhance the effectiveness of phishing URL detection.

4.numpy:numpy library is not directly used for phishing URL detection, it can be used as a supporting tool for various data manipulation tasks that are part of the overall phishing URL detection process. Here are a few examples of how numpy can be utilized:

5.pandas:The pandas library is a powerful tool for data manipulation and analysis in Python. Although it is not directly used for phishing URL detection, it can be helpful in several aspects of the process

6.gunicorn:Gunicorn (Green Unicorn) is a Python WSGI (Web Server Gateway Interface) HTTP server commonly used to deploy web applications. While Gunicorn itself is not directly used for phishing URL detection, it plays a role in the deployment and hosting of web applications that may include phishing URL detection functionality.

7.dateutil:The python_dateutil library is not specifically designed for phishing URL detection in machine learning (ML). However, it is a powerful Python library commonly used for parsing, manipulating, and working with dates and times in various formats.

8.requests: The requests library is a popular Python library used for making HTTP requests to interact with web resources. While it doesn't directly specialize in phishing URL detection, it can be utilized as part of the data collection and retrieval process in machine learning-based phishing URL detection systems.

9.whois:The whois command-line utility and protocol are used to retrieve registration information about domain names, IP addresses, and autonomous system numbers from the Internet's domain name registrars and registries. While whois itself is not a Python library, there are Python packages available, such as python-whois and dnspython, that provide a Pythonic interface to query and retrieve WHOIS information.

5.2 Algorithms

The algorithms used are:

5.2.1 Random Forest:

Random forest algorithm is one of the most powerful algorithms in machine learning technology and it is based on concept of decision tree algorithm. Random forest algorithm creates the forest with number of decision trees. High number of tree gives high detection accuracy. Creation of trees are 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 the classification and like decision tree algorithm; Random forest algorithm also uses gini index and information gain methods to find the best splitter. This process will get continue until random forest creates n number of trees. Each tree in forest predicts the target value and then algorithm will calculate the votes for each predicted target. Finally random forest algorithm considers high voted predicted target as a final prediction

5.2.2 Logistic Regression:

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.Logistic regression predicts the output of a categorical dependent variable. This type of statistical model (also known as *logit model*) is often used for classification and predictive analytics. Logistic regression estimates the probability of an event occurring, such as voted or didn’t vote, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1. In logistic regression, a logit transformation is applied on the odds—that is, the probability of success divided by the probability of failure.

5.2.3 XG Boost Classifier

The XGBoost (eXtreme Gradient Boosting) is a popular and efficient open-source implementation of the gradient boosted trees algorithm. Gradient boosting is a supervised learning algorithm that attempts to accurately predict a target variable by combining an ensemble of estimates from a set of simpler and weaker models. The XGBoost algorithm performs well in machine learning competitions because of its robust handling of a variety of data types, relationships, distributions, and the variety of hyperparameters that you can fine-tune. You can use XGBoost for regression, classification (binary and multiclass), and ranking problems.XGBoost is an optimized distributed gradient boosting library designed to be highly efficient**,** flexible and portable**.** It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way.

5.2.4 Support Vector Machine Algorithm

Support vector machine is another powerful algorithm in machine learning technology. In support vector machine algorithm each data item is plotted as a point in n-dimensional space and support vector machine algorithm constructs separating line for classification of two classes, this separating line is well known as hyperplane. Support vector machine seeks for the closest points called as support vectors and once it finds the closest point it draws a line connecting to them. Support vector machine then construct separating line which bisects and perpendicular to the connecting line. In order to classify data perfectly the margin should be maximum. Here the margin is a distance between hyperplane and support vectors. In real scenario it is not possible to separate complex and non linear data, to solve this problem support vector machine uses kernel trick which transforms lower dimensional space to higher dimensional space.

5.2.5 K Nearest Neighbours:

k-nearest neighbors algorithm (k-NN) is a non-parametric supervised learning . It is used for classification and regression. In both cases, the input consists of the *k* closest training examples in a data set. The output depends on whether *k*-NN is used for classification or regression.K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

CHAPTER 6

SOURCE CODE

Importing the libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn import metrics
import warnings
warnings.filterwarnings('ignore')
```

1.Loading the dataset:

```
data = pd.read_csv("phishing.csv")
data.head()
```

2.Familarizing with data & EDA:

```
data.shape
data.columns
data.info()
data.nunique()
data = data.drop(['Index'],axis = 1)
data.describe().T
```

3.Visualizing the data:

```
plt.figure(figsize=(15,15))
sns.heatmap(data.corr(), annot=True)
plt.show()
df = data[['PrefixSuffix-', 'SubDomains', 'HTTPS','AnchorURL','WebsiteTraffic','class']]
sns.pairplot(data = df,hue="class",corner=True);
data['class'].value_counts().plot(kind='pie',autopct='%1.2f%%')
plt.title("Phishing Count")
plt.show()
```

4.Splitting the data:

```
X = data.drop(["class"],axis =1)
y = data["class"]
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

Model Building & Training:

```
ML_Model = []
accuracy = []
f1_score = []
recall = []
precision = []
def storeResults(model, a,b,c,d):
  ML_Model.append(model)
  accuracy.append(round(a, 3))
  f1_score.append(round(b, 3))
  recall.append(round(c, 3))
  precision.append(round(d, 3))
from sklearn.linear_model import LogisticRegression
log = LogisticRegression()
log.fit(X_train,y_train)
y_train_log = log.predict(X_train)
y_test_log = log.predict(X_test)
acc_train_log = metrics.accuracy_score(y_train,y_train_log)
acc_test_log = metrics.accuracy_score(y_test,y_test_log)
print("Logistic Regression : Accuracy on training Data: {:.3f}".format(acc_train_log))
print("Logistic Regression : Accuracy on test Data: {:.3f}".format(acc_test_log))
print()
f1_score_train_log = metrics.f1_score(y_train,y_train_log)
f1_score_test_log = metrics.f1_score(y_test,y_test_log)
print("Logistic Regression : f1_score on training Data: {:.3f}".format(f1_score_train_log))
print("Logistic Regression : f1_score on test Data: {:.3f}".format(f1_score_test_log))
print()
f1_score_train_knn = metrics.f1_score(y_train,y_train_knn)
f1_score_test_knn = metrics.f1_score(y_test,y_test_knn)
print("K-Nearest Neighbors : f1_score on training Data: {:.3f}".format(f1_score_train_knn))
```

```
print("K-Nearest Neighbors : f1_score on test Data: {:.3f}".format(f1_score_test_knn))
print()
precision_score_train_knn = metrics.precision_score(y_train,y_train_knn)
precision_score_test_knn = metrics.precision_score(y_test,y_test_knn)
print("K-Nearest Neighbors : precision on training Data:
{:.3f}".format(precision_score_train_knn))
print("K-Nearest Neighbors : precision on test Data:
{:.3f}".format(precision_score_test_knn))
print(metrics.classification_report(y_test, y_test_knn))
training_accuracy = []
test_accuracy = []
for n in depth:
    knn = KNeighborsClassifier(n_neighbors=n)
    knn.fit(X_train, y_train)
    training_accuracy.append(knn.score(X_train, y_train))
    test_accuracy.append(knn.score(X_test, y_test))
plt.plot(depth, training_accuracy, label="training accuracy")
plt.plot(depth, test_accuracy, label="test accuracy")
plt.ylabel("Accuracy")
plt.xlabel("n_neighbors")
plt.legend();
acc_train_svc = metrics.accuracy_score(y_train,y_train_svc)
acc_test_svc = metrics.accuracy_score(y_test,y_test_svc)
print("Support Vector Machine : Accuracy on training Data: {:.3f}".format(acc_train_svc))
print("Support Vector Machine : Accuracy on test Data: {:.3f}".format(acc_test_svc))
print()
f1_score_train_svc = metrics.f1_score(y_train,y_train_svc)
f1_score_test_svc = metrics.f1_score(y_test,y_test_svc)
print("Support Vector Machine : f1_score on training Data:
{:.3f}".format(f1_score_train_svc))
print("Support Vector Machine : f1_score on test Data: {:.3f}".format(f1_score_test_svc))
print()
tree = DecisionTreeClassifier(max_depth=30)
y_train_tree = tree.predict(X_train)
```

```
y_test_tree = tree.predict(X_test)
acc_train_tree = metrics.accuracy_score(y_train,y_train_tree)
acc_test_tree = metrics.accuracy_score(y_test,y_test_tree)
print("Decision Tree : Accuracy on training Data: {:.3f}".format(acc_train_tree))
print("Decision Tree : Accuracy on test Data: {:.3f}".format(acc_test_tree))
print()
f1_score_train_tree = metrics.f1_score(y_train,y_train_tree)
f1_score_test_tree = metrics.f1_score(y_test,y_test_tree)
print("Decision Tree : f1_score on training Data: {:.3f}".format(f1_score_train_tree))
print("Decision Tree : f1_score on test Data: {:.3f}".format(f1_score_test_tree))
print()
recall_score_train_tree = metrics.recall_score(y_train,y_train_tree)
recall_score_test_tree = metrics.recall_score(y_test,y_test_tree)
print("Decision Tree : Recall on training Data: {:.3f}".format(recall_score_train_tree))
print("Decision Tree : Recall on test Data: {:.3f}".format(recall_score_test_tree))
print()
precision_score_train_tree = metrics.precision_score(y_train,y_train_tree)
precision_score_test_tree = metrics.precision_score(y_test,y_test_tree)
print("Decision Tree : precision on training Data: {:.3f}".format(precision_score_train_tree))
print("Decision Tree : precision on test Data: {:.3f}".format(precision_score_test_tree))
print(metrics.classification_report(y_test, y_test_tree))
training_accuracy = []
test_accuracy = []
# try max_depth from 1 to 30
depth = range(1,30)
for n in depth:
    tree_test = DecisionTreeClassifier(max_depth=n)
    tree_test.fit(X_train, y_train)
plt.plot(depth, training_accuracy, label="training accuracy")
plt.plot(depth, test_accuracy, label="test accuracy")
plt.ylabel("Accuracy")
plt.xlabel("max_depth")
plt.legend();
storeResults('Decision Tree',acc_test_tree,f1_score_test_tree,
```

```
        recall_score_train_tree,precision_score_train_tree)
y_train_forest = forest.predict(X_train)
y_test_forest = forest.predict(X_test)
acc_train_forest = metrics.accuracy_score(y_train,y_train_forest)
acc_test_forest = metrics.accuracy_score(y_test,y_test_forest)
print("Random Forest : Accuracy on training Data: {:.3f}".format(acc_train_forest))
print("Random Forest : Accuracy on test Data: {:.3f}".format(acc_test_forest))
print()
f1_score_train_forest = metrics.f1_score(y_train,y_train_forest)
f1_score_test_forest = metrics.f1_score(y_test,y_test_forest)
print("Random Forest : f1_score on training Data: {:.3f}".format(f1_score_train_forest))
print("Random Forest : f1_score on test Data: {:.3f}".format(f1_score_test_forest))
print()
recall_score_train_forest = metrics.recall_score(y_train,y_train_forest)
recall_score_test_forest = metrics.recall_score(y_test,y_test_forest)
print("Random Forest : Recall on training Data: {:.3f}".format(recall_score_train_forest))
print("Random Forest : Recall on test Data: {:.3f}".format(recall_score_test_forest))
print()
plt.figure(figsize=None)
plt.plot(depth, training_accuracy, label="training accuracy")
plt.plot(depth, test_accuracy, label="test accuracy")
plt.ylabel("Accuracy")
plt.xlabel("n_estimators")
plt.legend();
storeResults('Random Forest',acc_test_forest,f1_score_test_forest,
        recall_score_train_forest,precision_score_train_forest)
```

6.Comparison of Models

```
#creating dataframe
result = pd.DataFrame({ 'ML Model' : ML_Model,
                        'Accuracy' : accuracy,
                        'f1_score' : f1_score,
                        'Recall'   : recall,
                        'Precision': precision,
                       })
```

```
# dispalying total result
result
#Sorting the datafram on accuracy
sorted_result=result.sort_values(by=['Accuracy',
'f1_score'],ascending=False).reset_index(drop=True)
# dispalying total result
sorted_result
```

Storing the Model:

```
#  XGBoost Classifier Model
from xgboost import XGBClassifier
# instantiate the model
gbc = GradientBoostingClassifier(max_depth=4,learning_rate=0.7)
# fit the model
gbc.fit(X_train,y_train)
import pickle
# dump information to that file
pickle.dump(gbc, open('pickle/model.pkl', 'wb'))
#checking the feature improtance in the model
plt.figure(figsize=(9,7))
n_features = X_train.shape[1]
plt.barh(range(n_features), gbc.feature_importances_, align='center')
plt.yticks(np.arange(n_features), X_train.columns)
plt.title("Feature importances using permutation on full model")
plt.xlabel("Feature importance")
plt.ylabel("Feature")
plt.show()
```

CHAPTER 7

TESTING

Black box Testing:

Functional testing: Functional testing in black box testing focuses on validating the functional requirements of a system without considering its internal structure or implementation details. Testers treat the system as a black box and verify its behavior by providing different inputs and examining the corresponding outputs. The objective is to ensure that the system functions correctly and produces the expected results based on the specified requirements. Functional testing aims to validate the system's features, functionality, and interactions with users or other system components, ensuring that it meets the intended purpose and behaves as intended from an external perspective.

Non functional testing: Non-functional testing in black box testing focuses on evaluating the attributes and characteristics of the system that are not directly related to its functional requirements. It involves assessing factors such as performance, scalability, reliability, usability, security, and compatibility. Testers conduct black box testing by considering the system as a whole without knowledge of its internal workings, and they verify if it meets the specified non-functional requirements, ensuring that the system performs optimally, is user-friendly, secure, and compatible with various platforms and environments. This type of testing helps ensure that the system not only functions correctly but also meets the desired non-functional aspects that contribute to its overall quality and user satisfaction.

Regression: Regression testing in black box testing is a technique used to ensure that changes or modifications made to the system do not introduce new defects or impact existing functionality. It involves retesting the previously tested areas of the system using a black box approach, focusing on the system's inputs, outputs, and external behavior. The objective of regression testing is to detect any unintended side effects or regression issues that may arise due to code changes, configuration updates, or system enhancements. By retesting the system as a whole without considering the internal implementation, regression testing in black box testing helps ensure that the modified system continues to function correctly and meets the specified requirements while maintaining its existing functionality.

White box testing:

White box testing is a software testing approach that focuses on examining the internal structure, code, and implementation details of the system under test (SUT). Testers with knowledge of the internal workings of the system conduct white box testing, using techniques such as code coverage analysis, path testing, and code review. The objective is to assess the logic, control flow, and data flow within the system, identify potential errors or vulnerabilities, and verify that the system operates as intended. White box testing helps uncover defects, validate code quality, optimize performance, and ensure the robustness and reliability of the SUT by directly examining its internal components and leveraging knowledge of the underlying architecture and codebase.

TESTING TABLES:

s.no	Test scenario	Test steps	Test data	Expected Result	Success/ Failure
1	Check with valid phishing websites	Test with phishing dataset	URL 1	Successful in displaying the result	Success
2	Check with legitimate websites	Test without phishing dataset	URL 2	Successful in displaying the result	Success

Table 7.1- Testing the dataset with phishing and safe websites

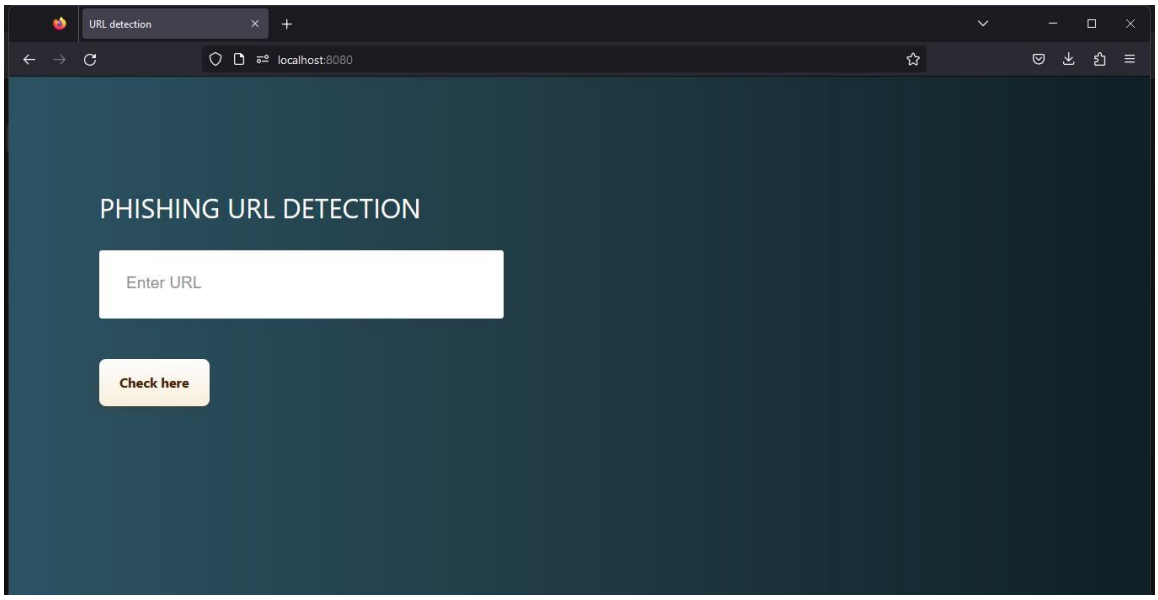
s.no	Test scenario	Test steps	Test data	Expected Result	Success/ Failure
1	Check with valid phishing websites	Test entering URL to user interface	URL 3	Successful in displaying the result	Success
2	Check with legitimate websites	Test entering URL to user interface	URL 4	Successful in displaying the result	Success

Table 7.2 - Testing with website URL in the user interface

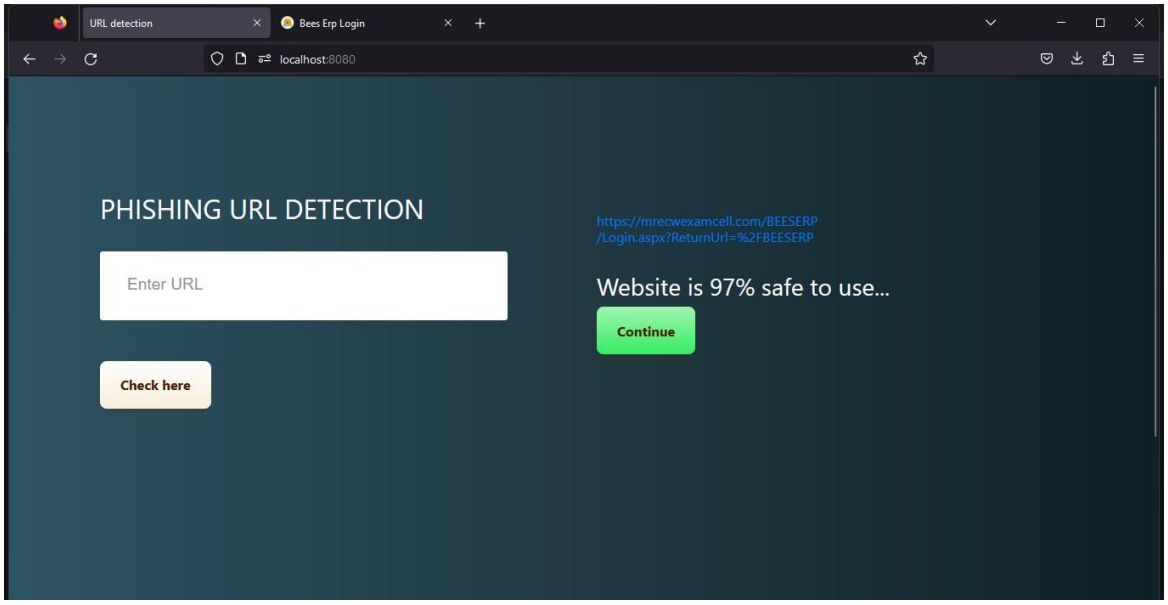
CHAPTER 8

RESULTS

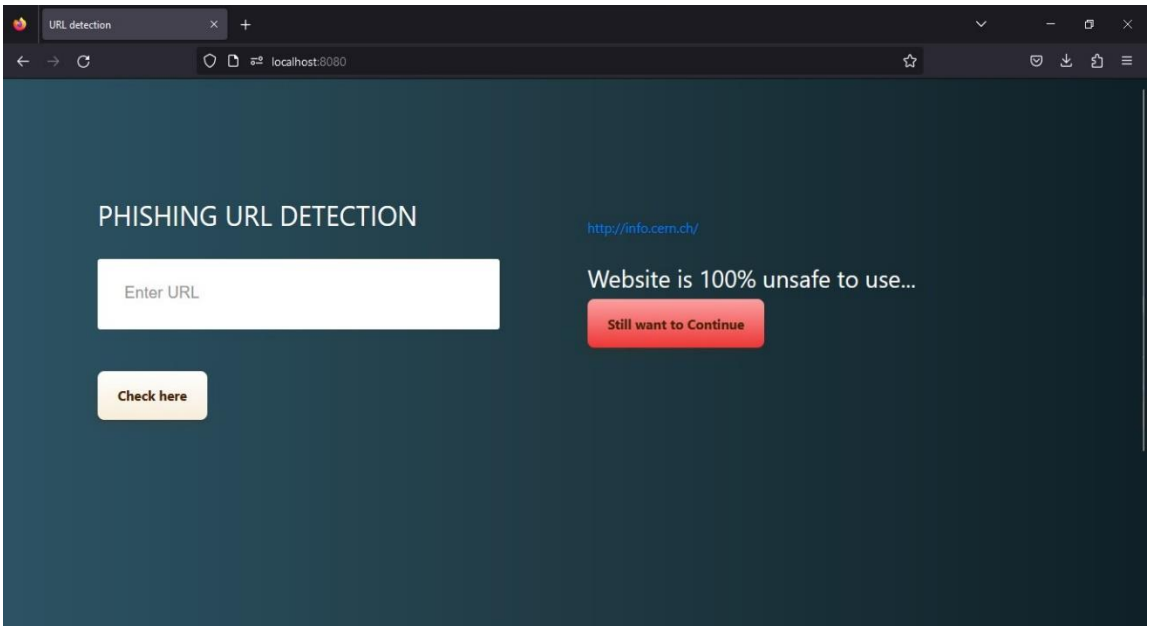
The User Interface:



Test case 1:



Test case-2:



	ML Model	Accuracy	f1_score	Recall	Precision
0	Gradient Boosting Classifier	0.974	0.977	0.994	0.986
1	CatBoost Classifier	0.972	0.975	0.994	0.989
2	XGBoost Classifier	0.969	0.973	0.993	0.984
3	Multi-layer Perceptron	0.969	0.973	0.995	0.981
4	Random Forest	0.967	0.971	0.993	0.990
5	Support Vector Machine	0.964	0.968	0.980	0.965
6	Decision Tree	0.960	0.964	0.991	0.993
7	K-Nearest Neighbors	0.956	0.961	0.991	0.989
8	Logistic Regression	0.934	0.941	0.943	0.927
9	Naive Bayes Classifier	0.605	0.454	0.292	0.997

Table 8.1- Accuracy Table

CHAPTER 9

CONCLUSION

In conclusion, this system is designed for resources are used as intended, prevents from valuable information from leaks out, produce better control mechanism and alerts the user to keep their private information safe. Like any other programs, there are improvements which could be made into this system. Based on the capabilities which the current system processes, text message integration would a great recommendation that could be made to improve the program in the future. The future version of the application could also implement an option to directly notify the blacklisted website with a text message. The program could be made to access the list as an attachment. This text message integration function would further the usability of the application.

This project aims to enhance detection method to detect phishing websites using machine learning technology. We achieved 97.14% detection accuracy using random forest algorithm with lowest false positive rate. Also result shows that classifiers give better performance when we used more data as training data.

CHAPTER 10

FUTURE SCOPE

In future if we get structured dataset of phishing we can perform phishing detection much more faster than any other technique.In future we can use a combination of any other two or more classifier to get maximum accuracy. We also plan to explore various phishing techniques that uses Lexical features, Network based features,Content based features, Webpage based features and HTML and JavaScript features of web pages which can improve the performance of the system. In particular, we extract features from URLs and pass it through the various classifiers.

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.

REFERENCES

1)Gunter Ollmann, “The Phishing Guide Understanding & Preventing Phishing Attacks”, IBMInternet Security Systems, 2007.

2) Mohammad R., Thabtah F. McCluskey L., (2015) Phishing websites dataset. Available: https://archive.ics.uci.edu/ml/datasets/Phishing+Websites Accessed January 2016

3) Mahmoud Khonji, Youssef Iraqi, "Phishing Detection: A Literature Survey IEEE, and Andrew Jones, 2013

4)Matthew Dunlop, Stephen Groat, David Shelly (2010) " GoldPhish: Using Images for Content-Based Phishing Analysis”

5) Rishikesh Mahajan (2018) “Phishing Website Detection using Machine Learning Algorithms”

6)Purvi Pujara, M. B.Chaudhari (2018) “Phishing Website Detection using Machine Learning : A Review”

7) D., Akila. (2019). Phishing Websites Detection Using Machine Learning. 8. 111-114. 10.35940/ijrte.B1018.0982S1119.

8) Parekh, S., Parikh, D., Kotak, S., Sankhe, S.: A new method for detection of phishing websites: URL detection. In: IEEE 2nd International Conference on Inventive Communication and Computational Technologies (ICICCT), April 2018