Chapter 1

Introduction

1.1 Introduction

Phishing refers to a malicious cyber-security attack whereby deceptive messages are sent by malicious actors impersonating trusted entities. These messages skillfully manipulate unsuspecting users, inducing them to take detrimental actions, such as unwittingly downloading or clicking on malicious files, or divulging sensitive information. The crafty nature of phishing attacks involves the utilization of various techniques, including link manipulation, filter evasion, website forgery, and covert redirects. The escalating frequency of phishing attacks has emerged as a grave concern in recent times. A common modus operandi employed by attackers entails the creation of deceptive websites that closely mimic the names and appearances of legitimate counterparts, thereby enticing users to access these fraudulent platforms.

Phishing attacks have become a significant concern owing to an increase in their numbers. A typical phishing attack technique involves using a phishing website, where the attacker lures users to access fake websites by imitating the names and appearances of legitimate websites, such as eBay, Facebook, and Amazon. It is difficult for the average person to distinguish phishing websites from normal websites because phishing websites appear similar to the websites they imitate. In many cases, users do not check the entire website URL, and, once they visit a phishing website, the attacker can access sensitive and personal information. With the growth in the field of e-commerce, phishing attack and cybercrimes are rapidly growing. Attackers use websites, emails, and malware to conduct phishing attacks.

Regrettably, majority of the phishing crimes go unreported, with the true extent of the problem often underestimated. Scholars and experts actively engage in debates and put forth a range of potential measures, encompassing both human-centric educational initiatives and technological solutions. Among the notable anti-phishing approaches are strategies focused on prevention, user training, and detection. Presently, Machine Learning models exhibit considerable promise as effective tools for detecting phishing attempts, showcasing their potential in the ongoing battle against this pervasive threat.

This project proposes developing a web browser plugin using machine learning approach to detect phishing websites with a combination of well trained ensemble stacking model of advanced classifiers such as Cat boost & Light GBM and Random Forest to detect phishing website URLs & develop another defense layer wherein the website content analysis is performed to finally classify it as either original or a fake copy with analysis report and to catch malicious figures and contents that are found even on legitimate websites. This can effectively shield the users from malicious attackers. This design also makes use of a NoSQL Database namely MongoDB to store classified phishy website URLs so that when the same is encountered again, the response time required would be reduced.

1.2 Problem Description

Phishing attacks have become a significant concern in the rapidly growing field of e-commerce and cybercrime. Users often struggle to distinguish phishing websites from legitimate ones, resulting in compromised security and the potential exposure of sensitive information. Current anti-phishing solutions rely on human-based education and technical approaches, but there is a need for more effective detection mechanisms. This project aims to develop a web browser plugin that utilizes a machine learning approach to detect phishing websites. By leveraging a combination of well-trained ensemble stacking models, including CatBoost, Light GBM, and Random Forest, the plugin will analyze website URLs to identify potential phishing attempts. Additionally, the project will implement a website content analysis module, providing users with a comprehensive analysis report. The plugin will also incorporate MongoDB database, to store and efficiently retrieve classified phishing website URLs, reducing response time when encountering previously identified threats. Ultimately, this project seeks to provide an effective defense layer against phishing attacks, safeguarding users from malicious actors and enhancing their overall online security.

Chapter 2

Literature Review

2.1 Literature Survey

Researchers have applied phishing website characteristics to undertake substantial anti-phishing research. Black and Whitelist approach, heuristics, visual similarity and the most modern approach of machine learning are some of the phishing detection techniques.

In a paper titled “Phishing website detection based on multidimensional features driven by deep learning” by the trio Peng Yang, G. Zhao & Peng Zeng [1] – IEEE 2019, they proposed a model where in first step, character sequence features of the given URL are extracted and used for quick classification by deep learning model of Convolutional Neural Network and Long Short Term Memory (CNN + LSTM model). In the next step, URL statistical features, webpage code features & classification result of deep learning are combined into multidimensional features.

In a paper titled “Phishpedia: A hybrid Deep Learning approach to visually identify phishing webpages” [2] Yun Lin, Ruofan Liu & Dinil Mon Divakaran – IEEE 2021, they proposed a model to visually identify Phishing Webpages called Phishpedia. The model takes as input a URL and a target brand list describing legitimate brand logos and their web domains. It then generates a phishing target as output. It used faster-RCNN model and PyTorch framework.

In a paper titled “Phishing website detection using diverse ML algorithms” [3] A. Zamir & N. Yousaf – IEEE 2021, they proposed an approach where the features of phishing data set are to be analyzed. Then, diverse machine learning algorithms like Random Forest, Neural Network, Bagging, K-Nearest Neighbor are applied on features. Afterwards, two stacking models: Stacking1 (RF þ NN þ Bagging) and Stacking2 (kNN þ RF þ Bagging) are applied by combining highest scoring classifiers to improve the classification accuracy.

In a paper titled “Phishing website detection based on Convolutional Neural Network (CNN) & Random Forest (RF) ensemble learning” [4] R. Yang, K. Zheng & Bin Wu – IEEE 2021, they proposed phishing website detection method based on character embedding, CNN, and Random Forest. Here the URL data is transformed into a character

vector. Then CNN network is trained using the transformed URL data. After the model is trained, URL features are extracted & then the features extracted from different network layers are classified using Random Forest.

In a paper titled "An effective & secure mechanism for phishing attack detection using ML approach" [5] by J. Visumathi, Miroslav Mahdal and Jose Anand – IEEE 2022, they presented detailed comparative analysis using ML classifiers. The algorithms used for ML approach were Support Vector Machine, Random Forest & Neural Network separately for classification of URL features. Neural Network was found to be having 95.18% accuracy & is selected as best method.

2.2 Comparative Analysis of the Related Work

The table 2.1 discusses the comparative analysis of the current systems in light of the suggested proposal.

Table 2.1 Comparative Analysis

Sl. No	Author(s)	Algorithms/Techniques	Performance Measures
1.	Peng Yang, G. Zhao and Peng Zeng	Convolutional Neural Network and Long Short Term Memory Model	Accuracy
2.	Yun Lin, Ruofan Liu and Dinil Mon Divakaran	faster-RCNN model and PyTorch framework	Accuracy
3.	A. Zamir and N. Yousaf	Random Forest, Neural Network, Bagging, K-Nearest Neighbor	Accuracy
4.	R. Yang, K. Zheng and Bin Wu	Convolutional Neural Network and Random Forest	Accuracy
5.	J. Visumathi, Miroslav Mahdal and Jose Anand	Support Vector Machine, Random Forest & Neural Network	Accuracy

2.3 Summary

These were the research papers that we studied to gain a better understanding of the problem. Machine learning classification algorithms are more accurate compared to the traditional techniques when it comes to detecting phishing websites. Hence, we chose to analyze advanced classifiers like Random Forest, LightGBM, Cat Boost algorithms and form their ensemble model and implement the prediction model using the best and most efficient algorithm which is the ensemble model in the project.

Chapter 3

Problem Formulation

3.1 Problem Statement

Phishing, a formidable cyber threat, inflicts substantial financial losses amounting to hundreds of millions of dollars annually while also causing numerous data breaches. The escalation in the prevalence of phishing attacks has sparked significant concern. These attacks are not only widely utilized but also exceptionally effective and damaging. Perpetrators employ sophisticated tactics to deceive users into unwittingly disclosing their confidential information, including passwords and credit card details. A typical method involves the creation of phishing websites that meticulously imitate the names and visual identities of legitimate online platforms like eBay, Facebook, and Amazon. Regrettably, discerning individuals find it arduous to differentiate between genuine websites and their fraudulent counterparts due to the striking similarities meticulously engineered by cyber criminals.

In the recent years, cyber criminals delivered a wave of cyber attacks that were not just highly coordinated, but far more frequent and advanced than ever before seen. Simple endpoint attacks became complex, multi-stage operations. Ransomware attacks hit small businesses and huge corporations alike. Cryptomining malware attacks gave cyber criminals an easy foothold into company networks. 2022 was a year of massive data leaks, expensive ransomware payouts, and a vast, new, complicated threat landscape. And it was a year that saw cyber criminals up their threat game in a big way. Despite the widespread occurrence and severe consequences of phishing attacks, they often go underreported. This lack of reporting further complicates efforts to combat this form of cybercrime effectively.

Therefore, there is an urgent need to develop robust and efficient methods for detecting phishing websites, enabling proactive defense against this pervasive threat. To address this problem, a comprehensive approach is required to accurately identify and distinguish phishing websites from legitimate ones. This entails leveraging advanced technologies, such as machine learning, data analytics, and website analysis, to detect subtle indicators

and patterns that differentiate phishing sites from genuine platforms. By developing effective detection mechanisms, individuals and organizations can better protect themselves against phishing attacks, mitigating the financial and reputational damages associated with such incidents

3.2 Objectives of the Present Study

The objectives of the proposed project are as follows:

1. To train the ML Classifier models like Random Forest, Cat Boost, Light GBM and their ensemble model using the datasets of phishy & legitimate websites..
2. To run comparative analysis by calculating accuracy of each training model to know the best suitable model for phishing website detection.
3. To create a phishing detection system using the best suitable model.
4. To develop an interactive web browser plug-in for the real-time detection & blocking of Phishing website.

3.3 Summary

Utilizing machine learning techniques represents the optimal solution for detection of phishing websites. These advanced classification algorithms outperform traditional methods of identifying fraudulent sites. The development of a robust system for more accurate detection holds tremendous value for both IT professionals and individuals at risk. By promptly alerting users to the presence of phishing websites, this technology empowers them to take immediate preventive measures, such as avoiding interaction with malicious content and safeguarding their sensitive information. Such proactive actions can effectively impede the propagation of cyber threats and enhance overall cybersecurity.

Chapter 4

Requirements and Methodology

4.1 Hardware Requirements

The hardware requirements for the proposed project are depicted in Table 4.1.

Table 4.1: Hardware requirements

Sl. No	Hardware/Equipment	Specification
1.	Graphics Card	Intel 621 Graphics card or 2GB
2.	RAM	4GB or above

4.2 Software Requirements

The software requirements for the proposed project are depicted in Table 4.2.

Table 4.2: Software requirements

Sl. No	Software	Specification
1.	Anaconda	Anaconda 64 bit
2.	Python	Python 3 and above
3.	Framework	Flask
4.	MongoDB Database	MongoDB 5 and above
5.	Google Chrome web browser	Version 108 and above

4.3 Methodology Used

The proposed phishing detection system is implemented using the following steps:

1) **Data Acquisition & Preprocessing:** This step involves collection of datasets of phishy and legitimate websites from open-source platforms like Kaggle and then preprocessing of data.

2) **Database Creation:** This process follows blacklist methodology of phishing detection in which database containing blacklist of illegal websites is created. Any phishy website detected by our model will be added to blacklist database so that when the user encounters same website, the site is immediately blocked with faster response time.

3) **Training of URL classifier model:** This involves selecting various classifier algorithms and training them using URL Features, running a comparative analysis of performance and choosing best suited model. We test various algorithms like Random Forest, CatBoost, Light GBM and their ensemble model, train and analyze the performance. Finally the best suited model which is the ensemble model is chosen.

4) **Developing Website content analysis module:** Even after classification of website as legitimate, there is a possibility that a legitimate website may be housing unwanted images, icons and pop-ups. Hence a website content analysis is which provides user with information regarding presence of suspicious elements.

5) **Extraction of URL Features & Output Prediction:** The user provides URL of the required website as input. Relevant features are extracted from this URL and a dataframe is created. The ensemble model takes the extracted features of the given website to predict whether the URL is suspicious or not. If the URL is found to be suspicious then the user is warned and the blacklist database will be updated automatically.

Chapter 5

System Design

5.1 Architecture of the Proposed System

Figure 5.1 shows the architecture of the proposed system.

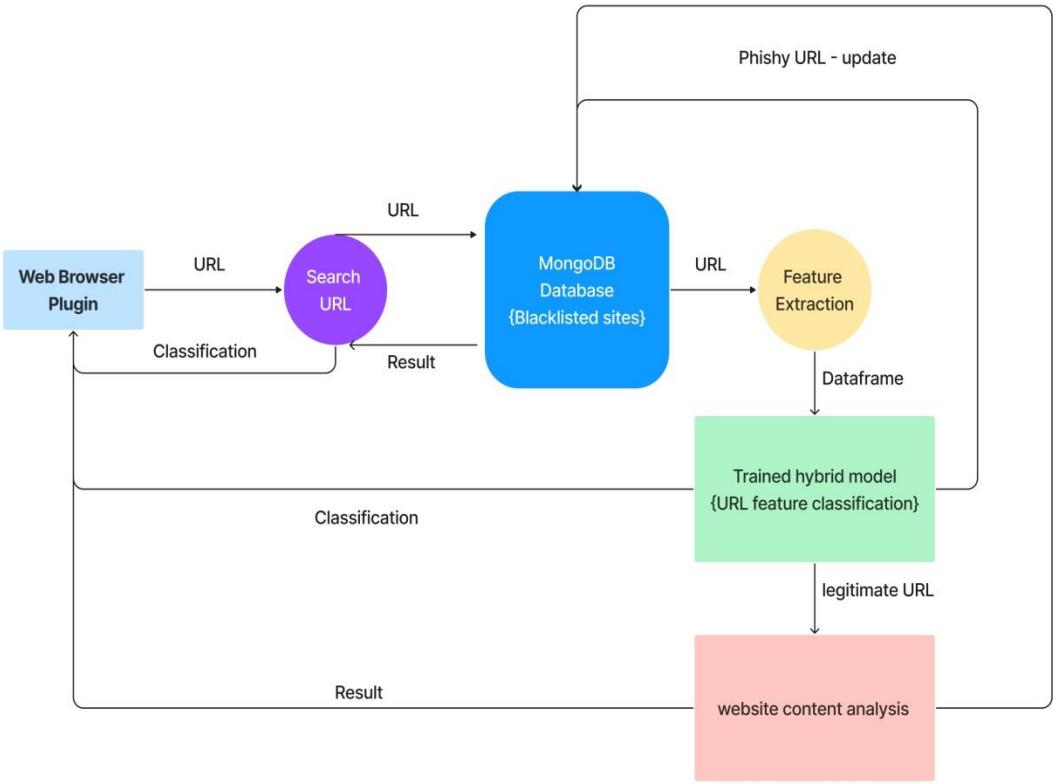

Figure 5.1: Architecture of the proposed system

The first step is to collect, clean and pre-process the data set. The data set is then trained using the ML classification algorithms Random Forest, Cat Boost, LightGBM and their ensemble stacking model. The accuracy is calculated for each model and the most accurate model will be used to implement the web browser plugin. A MongoDB blacklist consisting of suspicious URLs is created. A feature extraction code to acquire feature values corresponding to the dataset used is developed using selected model. Similarly a website content analyzer code is developed. After integrating the various components of the project – Prediction model, website content analyzer and blacklist, it is deployed as the final plugin application.

5.2 System Flowchart

A system flowchart is a way of depicting how data flows in a system and how decisions are made to control events. Figure 5.2 depicts the system flowchart.

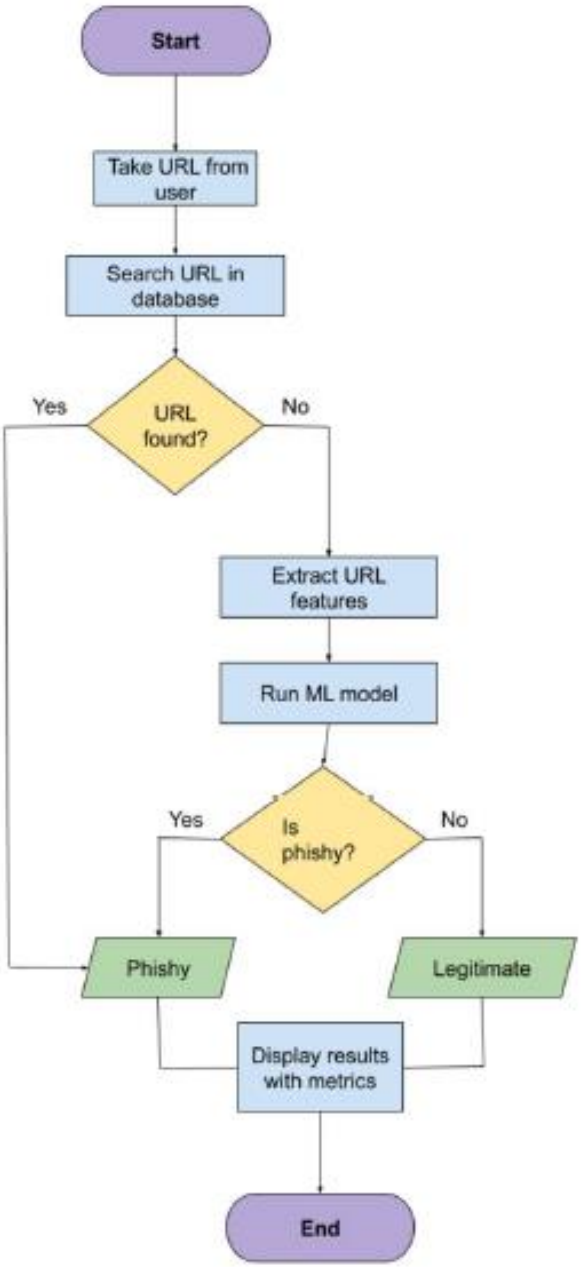

Figure 5.2: System Flowchart

The raw dataset must be loaded, cleaned, and preprocessed. The dataframe is created with the selected features. The prediction model is created using Ensemble stacking model of Random Forest, Cat Boost and Light GBM. This model classifies the website into Phishy and legitimate.

Chapter 6

Implementation

6.1 Pseudocode

Algorithm: Ensemble Stacking Model for Phishing Website Detection.

Input: URL of websites which could be phishy or legitimate.

Output: A prediction model predicting legitimacy of website.

1. Load the dataset and extract the features into the variable x. Extract the target feature into the variable y.
2. Split the data into training and testing sets. Assign 80% of the data to the training set and 20% to the testing set. Set the random_state parameter to 42 for reproducibility.
3. Create the base models:
    a. Initialize a random forest and assign it to the variable rf.
    b. Initialize a CatBoost classifier and assign it to the variable cat.
    c. Initialize an LGBM classifier and assign it to the variable lgbm.
4. Create the voting classifier:
    a. Create a list of tuples called estimators, where each tuple consists of a string identifier and a corresponding base model.
    b. Initialize a stacking classifier with the estimators and the final_estimator set to lgbm. Assign it to the variable stack_model.
5. Fit the stack_model and use it for predicting legitimacy of websites by taking URL input and extracting URL features, subsequently classifying the URL into phishy or legitimate.

Dataset Creation using feature importance graph

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
#Reading the csv file and storing it in a dataframe
data = pd.read_csv("dataset_phishing.csv")
print(data)
```

```
#Viewing the statistical details of the data
print(data.describe())
#Determining the datatypes of data in each column
print(data.dtypes)
#Total null values in the dataset
print(data.isnull().sum())
sns.heatmap(data.isnull())
plt.show()

# Creating a new dataframe with selected columns
df =
pd.DataFrame(data,columns=['url','length_url','ip','nb_dots','nb_hyphens','nb_at','nb_q
m','nb_and','nb_eq','nb_percent','nb_slash','nb_colon','nb_semicolumn','nb_www','nb_
com','nb_dslash','https_token','prefix_suffix','phish_hints','shortening_service','whois_
registered_domain','domain_age','status'])
print(df)

#Replacing the categorical values
df["status"].replace({"legitimate":0,"phishing":1},inplace=True)
print(df)
#Convert dataframe into csv file
df.to_csv('phishing_dataset_latest.csv',index=False)
```

Training algorithms like Random Forest, Cat Boost, LightGBM & their ensemble model using the dataset created

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier
from catboost import CatBoostClassifier
from lightgbm import LGBMClassifier
from sklearn.ensemble import StackingClassifier
from sklearn.metrics import accuracy_score
```

```
# Load dataset
data = pd.read_csv("phishing_dataset_latest.csv")
X = data.iloc[:,:-1].drop(['url'], axis=1)
y = data['status']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Create the base models
rf = RandomForestClassifier(n_estimators=10)
cat = CatBoostClassifier()
lgbm = LGBMClassifier(num_leaves=31,max_depth=-
1,learning_rate=0.1,n_estimators=100)

# Create the voting classifier
estimators = [('rf', rf), ('cat', cat)]
stack_model = StackingClassifier(estimators=estimators, final_estimator=lgbm)

# Fit the voting model on the training data
stack_model.fit(X_train, y_train)
# Make predictions on the test data
y_pred = stack_model.predict(X_test)
# Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))

cnf_matrix=metrics.confusion_matrix(y_test,y_pred)
print(cnf_matrix)
sns.heatmap(pd.DataFrame(cnf_matrix),annot=True,fmt='g')
plt.show()

# Save ML model to disk in pickle file
model_filename = "StackingModel.sav"
saved_model = pickle.dump(stack_model, open(model_filename,'wb'))
```

Pseudo code for URL Feature Extraction

```
#various feature extraction functions like url_length, domain_age implemented.
#Function to extract features
def featureExtraction(url):
  parsed = urlparse(url)
  scheme = parsed.scheme
  domain = urlparse(url).netloc
  words_raw = re.split("\-|\.|\/|\?|\=|\@|\&|\%|\:|\_", domain.lower())
  features = []
  features.append(url_length(url))
  features.append(having_ip_address(url))
  features.append(count_dots(url))
  features.append(count_hyphens(url))
  features.append(count_at(url))
  features.append(count_qm(url))
  features.append(count_and(url))
  features.append(count_equal(url))
  features.append(count_percentage(url))
  features.append(count_slash(url))
  features.append(count_colon(url))
  features.append(count_semicolumn(url))
  features.append(check_www(words_raw))
  features.append(check_com(url))
  features.append(count_double_slash(url))
  features.append(https_token(scheme))
  features.append(prefix_suffix(url))
  features.append(phish_hints(url))
  features.append(shortening_service(url))
  features.append(whois_registered_domain(url))
  features.append(domainAge(url))
  features.append(get_google_index_count(url))
  return features
```

MongoDB Database creation

```
from genericpath import exists
from pydoc import cli
from numpy import true_divide
import pymongo
client =
pymongo.MongoClient('mongodb+srv://<username>:<password>@cluster0.xwat7j
a.mongodb.net/phishing-websites?retryWrites=true&w=majority')
# connect to DB
db = client['phishing-websites']

# connect to collection in DB
collection = db['blacklist']

def addURL_MongoDB(url): # add new URL to DB
    data = {'url' : url}
    collection.insert_one(data)

def search_URL(url):   # search user input URL in database
    found = collection.find({'url' : url})
    exists = True
    if found:
        for result in found:
            return exists
        exists = False
    return exists
```

Prediction and updating Database

```
import feature_extraction as fe
import database as db
import pickle
model_filename = "StackingModel.sav"
# Load model to predict new data
with open(model_filename, 'rb') as f:
    rfc = pickle.load(f)
```

```
# Classify URL input
def classifyURL(url):
    if db.search_URL(url):
        return "Phishy"
    else:
        df = fe.featureExtraction(url)
        res = int(rfc.predict([df]))
        if res:
            db.addURL_MongoDB(url)
            return "Phishy"
        return "Legitimate"
```

Plugin rendering Manifest file

```
{
  "manifest_version": 3,
  "name": "Citadel",
  "version": "1.0.0",
  "description": "Web Advisor",
"icons":
{
"128":"icon.png"
},
  "permissions": [
    "activeTab",
    "webNavigation"
  ],
  "background": {
    "service_worker": "background.js"
  }
}
```

```
#Background.js
const visitedURLs = new Set();
chrome.webNavigation.onBeforeNavigate.addListener(details => {
```

```
  const { url, tabId, parentFrameId } = details;
  // Ignore if the navigation is a subframe of an existing tab
  if (parentFrameId !== -1) {
    return;
  }
  // Ignore if URL starts with "chrome", "file", or "chrome-extension"
  if (url.startsWith('chrome') || url.startsWith('file') || url.startsWith('chrome-
extension')) {
    return;
  }
// Ignore if URL starts with Google search query
  if (url.startsWith('https://www.google.com/search?')) {
    return;
  }
  // Check for infinite loop
  if (visitedURLs.has(url)) {
    return;
  }
  // Ignore localhost URLs except for form submission to /results
  if ((url.startsWith('http://localhost') || url.startsWith('http://127.0.0.1:5000/')) &&
!url.includes('localhost/results')) {
    return;
  }
  visitedURLs.add(url);
  // Redirect to localhost/results with URL as query parameter
  const redirectURL = `http://localhost:5000/results?url=${url}`;
  if (!url.includes('localhost/results')) {
    chrome.tabs.update(tabId, { url: redirectURL });
  }
});
```

Chapter 7
System Testing, Results and Discussion

7.1 System Testing

Table 7.1: Unit test cases

Test case number	Input	Stage	Expected behavior	Observed behavior	Status P=Pass F=Fail
1	Enter input values from the test set	Catch Phish page/ search bar in browser	The result should appear as predicted by Phishing Detection Plugin	As expected	P
2	Enter input values from the test set	Catch Phish page/ search bar in browser	The Phishing Detection Plugin result should change accordingly	As expected	P

7.2 Result Analysis

The main aim of the project was to predict the legitimacy of any website using machine learning algorithms. Table 7.2 shows the analysis that was performed on the four models with different training and testing sizes. It was found that ensemble model was the most accurate in all the cases.

Table 7.2: Analysis of the four algorithms

Training Size	Testing Size	Accuracy			
		RF	CB	LGBM	Ens
80%	20%	0.9481	0.9488	0.9501	0.9514
70%	30%	0.9457	0.9442	0.9454	0.9475

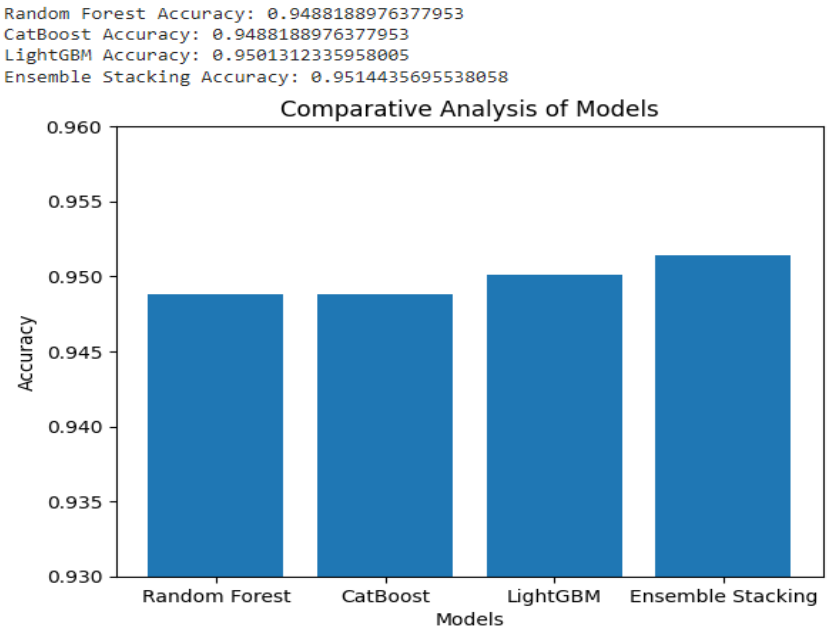

Figure 7.1: Graph analysis of the first set

Figure 7.1 shows the bar graph for the accuracy of the four algorithms where the train set size was 80% and the test set size was 20**%.**

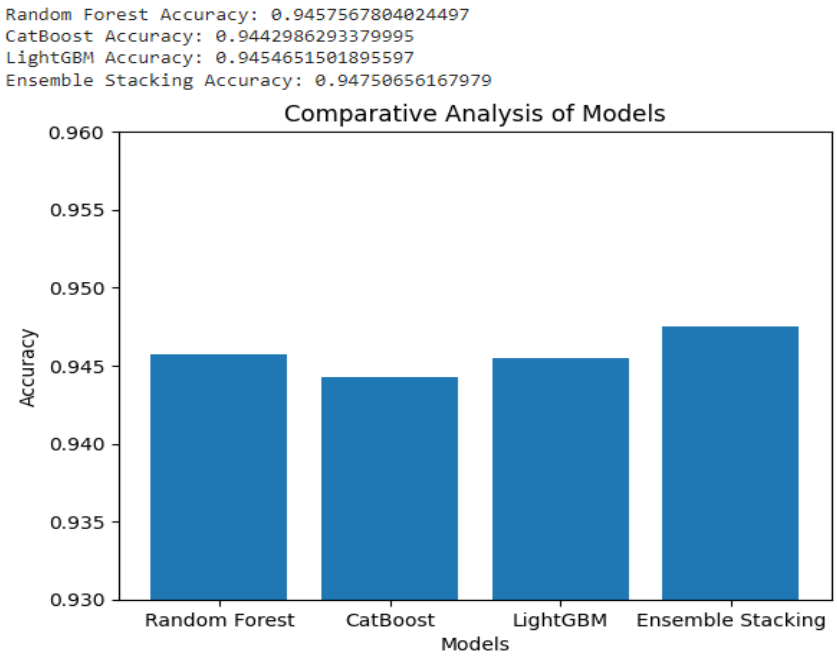

Figure 7.2: Graph analysis of the second set

Figure 7.2 shows the bar graph for the accuracy of the four algorithms where the train set size was 70% and the test set size was 30%.

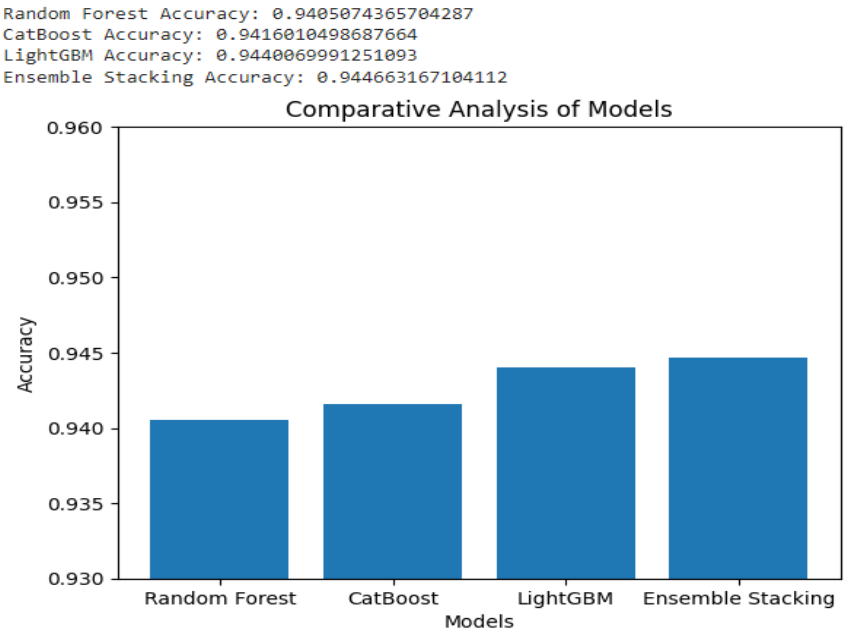

Figure 7.3: Graph analysis of the third set

Figure 7.3 shows the bar graph for the accuracy of the four algorithms where the train set size was 60% and the test set size was 40%.

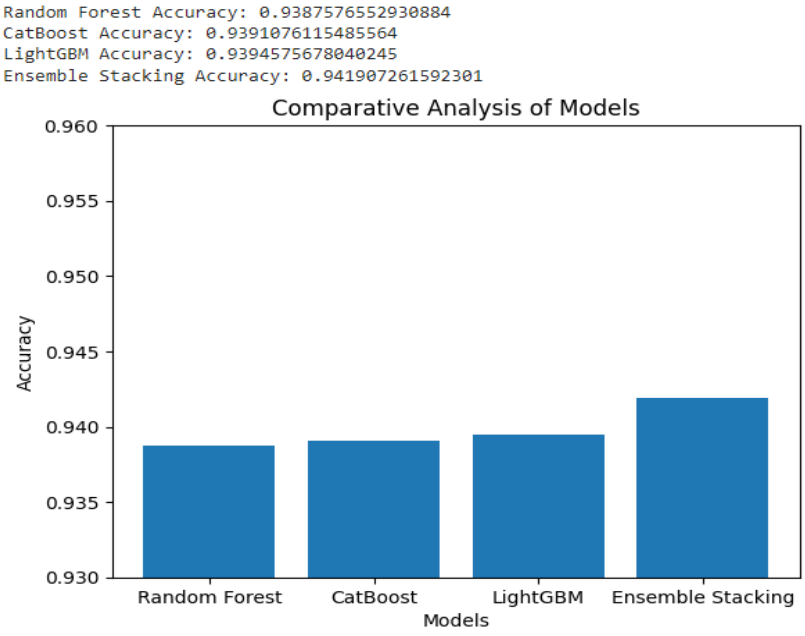

Figure 7.4: Graph analysis of the fourth set

Figure 7.4 shows the bar graph for the accuracy of the four algorithms where the train set size was 50% and the test set size was 50%.

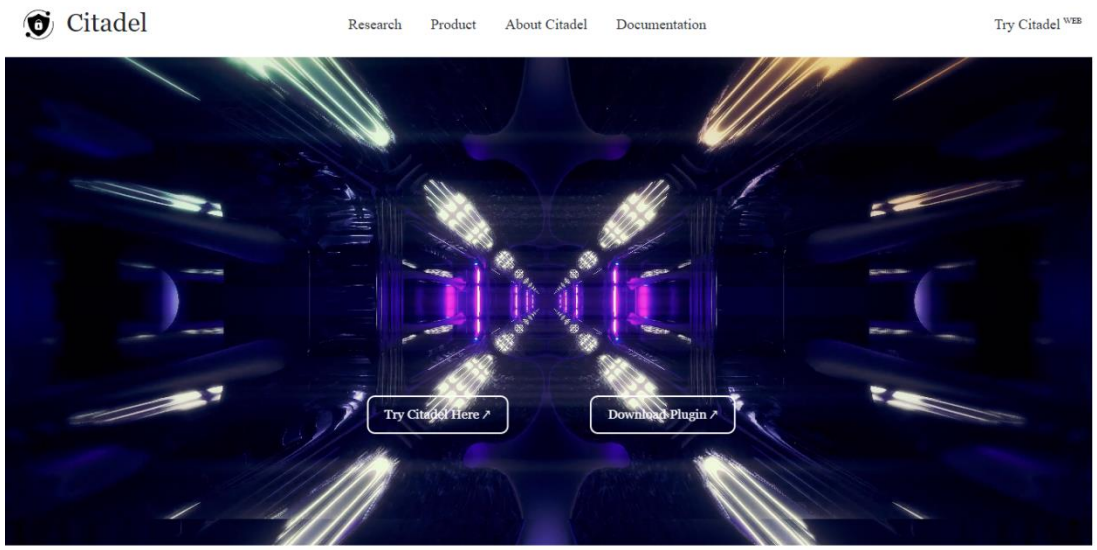

Figure 7.5: Home Page

Figure 7.5 is the home page for the users who use this application in the web version.

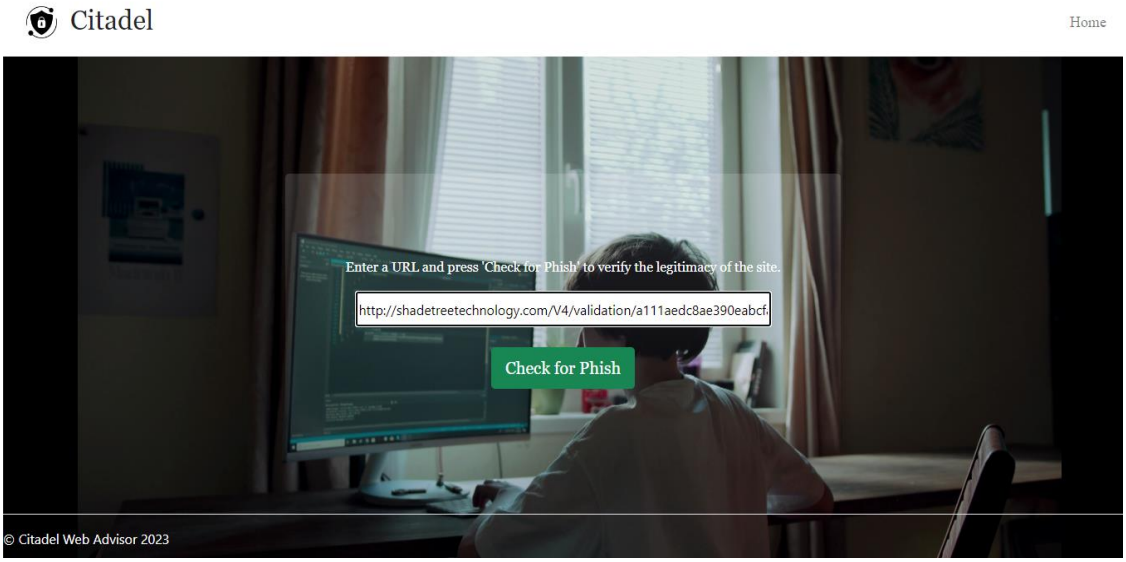

Figure 7.6 Catch Phish page

Figure 7.6 is the Catch Phish page. Here, the user will enter the URL. URL features will be extracted.These features are the ones that are responsible for the result.

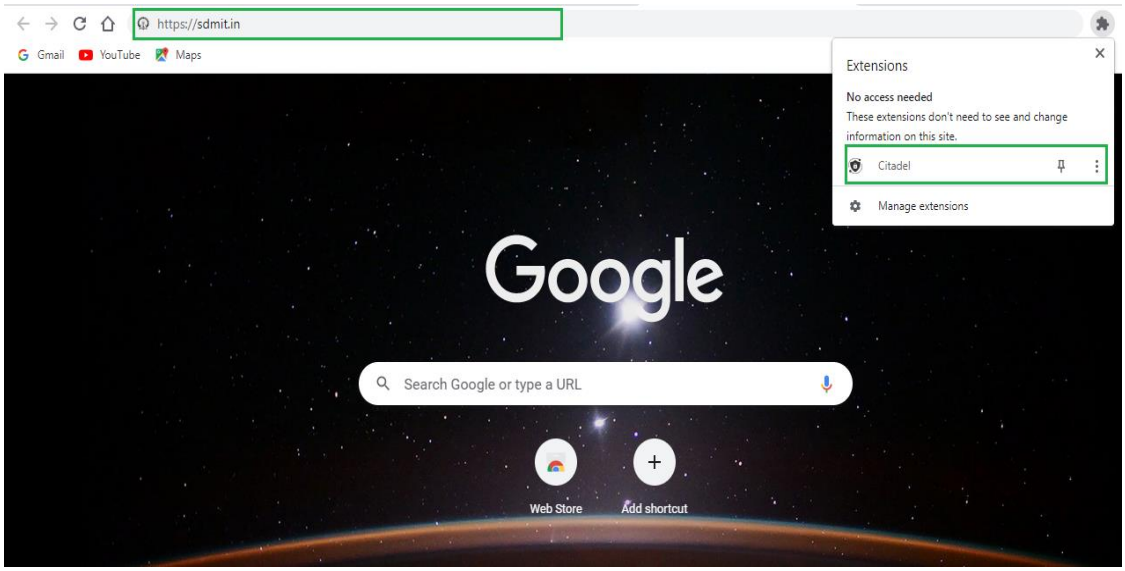

Figure 7.7: Browser search bar with plugin enabled

Figure 7.7 is the page where user can type URL directly in their browser with plugin enabled.

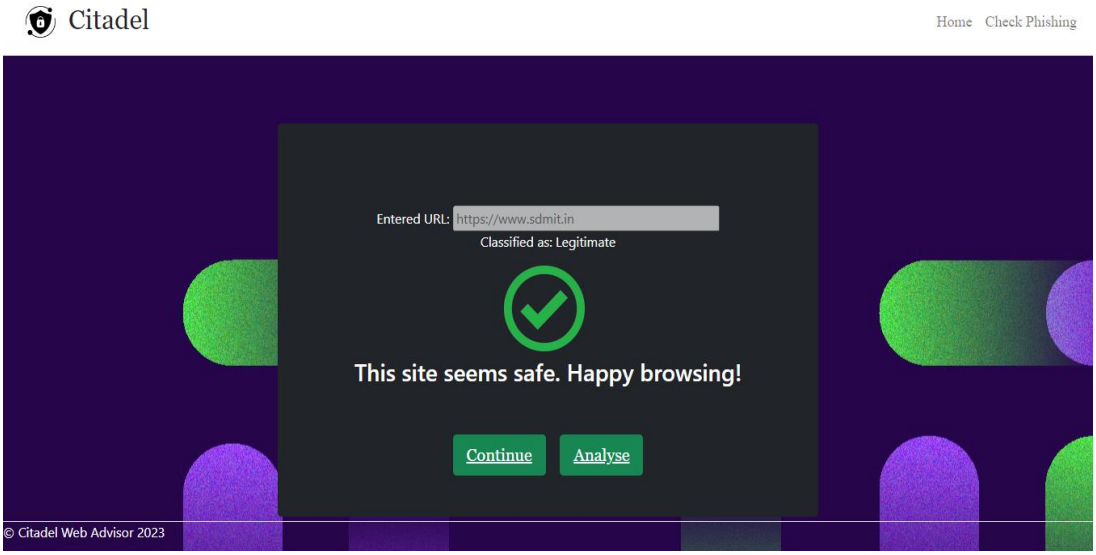

Figure 7.8 Prediction of site legitimacy

Figure 7.8 is the result page where the prediction is being carried out.

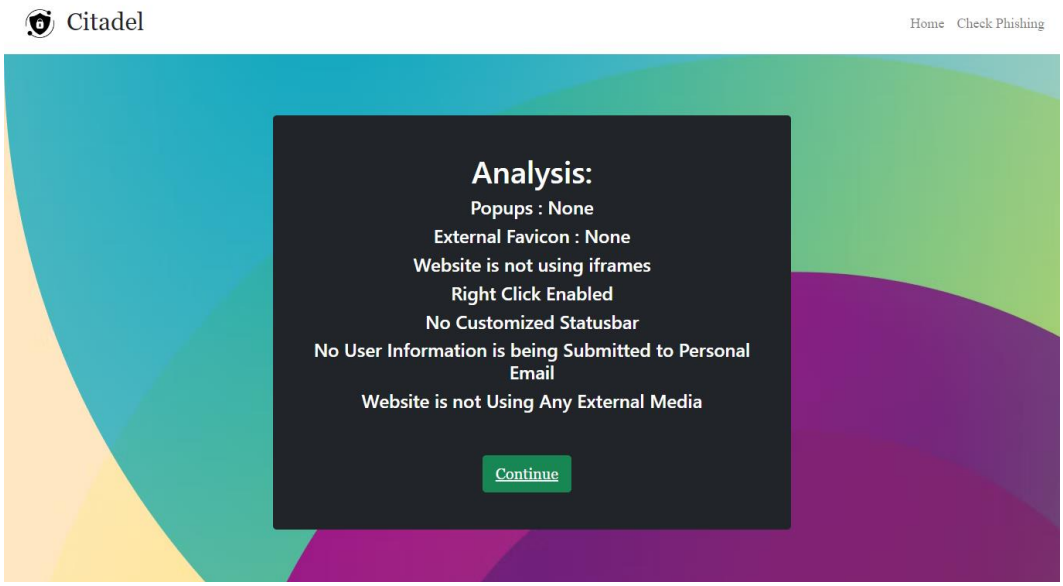

Figure 7.9: Website analysis

Figure 7.8 is the result page where the website analysis is being carried out.

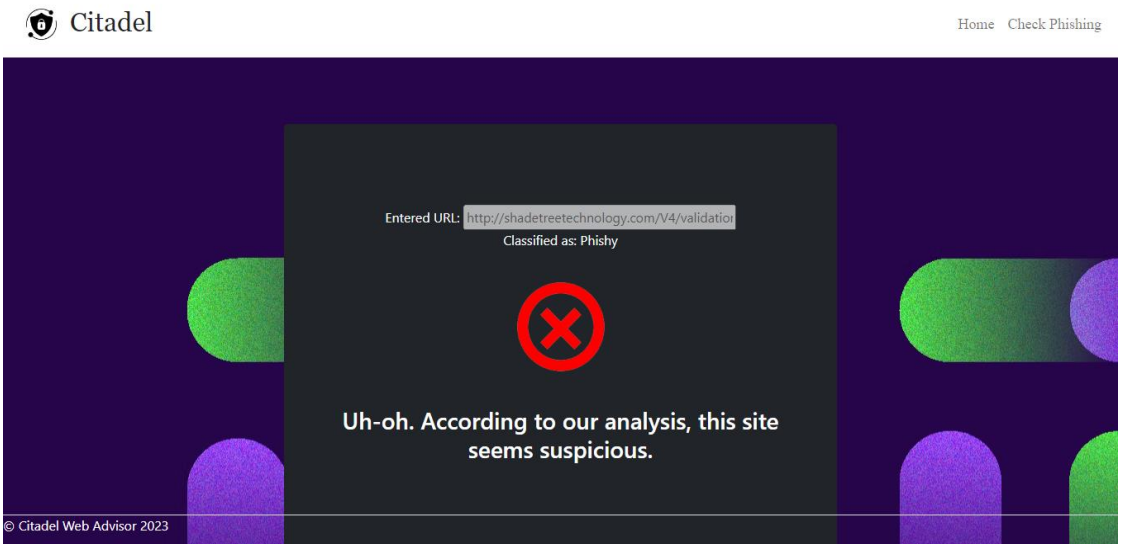

Figure 7.10: Phishy site blocking

Figure 7.9 shows the blocking of phishy website by the phishing detection plugin.

7.3 Summary

The application was developed using the Flask framework. The Python programminglanguages is used alongside basic HTML. The figures in the previous section showed the snapshots of various pages of the application. Since Ensemble stacking model of Random Forest, LightGBM and CatBoost was found to be the most accurate among the four algorithms, the prediction model was created using it.

Chapter 8

Conclusion and Scope for Future Work

8.1 Conclusion

The project proposes an efficient method for improvising current phishing detection techniques by combining both blacklist approach and machine learning approach. The use of blacklist approach reduces the response the since the URL will be searched in the database before going through feature extraction and being passed to the classifier for prediction. This application is aimed at providing both higher accuracy and speed so that anyone who uses the phishing detection plugin finds it synchronized to real-time providing a seemingly effortless and ease of use. It can be used by every surfer to effectively shield from malicious attackers. The simple design helps even the novice user to understand the usage and easily access the application. Hence this can be helpful in significantly reducing phishing crimes when deployed and used on larger scale.

8.2 Scope for Future Work

The project can be further improvised by developing a desktop application which could be downloaded to user's system. This can help user get relevant messages and warnings and also changes in any features could be notified immediately. The project can also be incorporated with an API service for developers to use the model for requesting result of determining the legitimacy of websites in their applications.

References

[1] Peng Yang, G. Zhao & Peng Zeng “Phishing website detection based on multidimensional features driven by deep learning”, IEEE 2019 (For research/ technical papers)

[2] Yun Lin, Ruofan Liu & Dinil Mon Divakaran “Phishpedia: A hybrid Deep Learning approach to visually identify phishing webpages”, IEEE 2021 (For research/ technical papers)

[3] A. Zamir & N. Yousaf “Phishing website detection using diverse ML algorithms”, IEEE 2021 (For research/ technical papers)

[4] R. Yang, K. Zheng & Bin Wu “Phishing website detection based on CNN & RandomForest ensemble learning”, IEEE 2021 (For research/ technical papers)

[5] J. Visumathi, Miroslav Mahdal and Jose Anand “An effective & secure mechanism for phishing attack detection using ML approach”, IEEE 2022 (For research/ technical papers)

[6] Safa Alrefaai, Ghina Özdemir "Detecting Phishing Websites Using Machine Learning", IEEE 2022 (For research/ technical papers)

[7] Bryan Espinoza, Jéssica Simba, Walter Fuertes, "Phishing Attack Detection: A Solution Based on the Typical Machine Learning Modeling Cycle", IEEE 2019 (For research/ technical papers)

[8] Andrii Mykytiuk, Victoria Vysotska, Solomiia Albota, "Spam Filtration System with the Use of Machine Learning Technology", IEEE 2021 (For research/ technical papers)

[9] Yongjie Huang, Qiping Yang, Jinghui Qin, Wushao Wen, "Phishing URL Detection via CNN and Attention-Based Hierarchical RNN", IEEE 2019 (For research/ technical papers)

[10] Ashit Kumar Dutta, "Detecting phishing websites using machine learning technique", Research Gate 2021 (For research/ technical papers)