STRENGTHENING DEFENSES WITH DATA ANALYTICS AGAINST MALICIOUS LINKS

A PROJECT REPORT

Submitted by

GOPALA KRISHNA.J	**712220104016**
SANTHOSH KUMAR.M	**712220104036**
VISAGAN.P	**712220104055**
VISHRAM.R	**712220104056**

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PARK COLLEGE OF ENGINEERING AND TECHNOLOGY

KANIYUR, COIMBATOTE - 641 659

ANNA UNIVERSITY:CHENNAI 600 025

MAY 2024

BONAFIDE CERTIFICATE

Certified that this project report **"STRENGTHENING DEFENSES WITH DATA ANALYTICS AGAINST MALIFICIOUS LINKS"** is the Bonafide work **"GOPALA KRISHNA.J , SANTHOSH KUMAR.M , VISAGAN.P , VISHRAM.R"** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. V.Saranya, M.Tech., Ph.D.	Mrs. S. Princy, M.E.
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Professor	Assistant Professor
Department of Computer Science	Department of Computer Science
and Engineering,	and Engineering,
Park College of Engineering	Park College of Engineering
and Technology, Kaniyur	and Technology, Kaniyur
Coimbatore-641 659.	Coimbatore-641 659.

Submitted for the University Project VIVA -VOCE examination held on_____________

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

First and foremost, we praise and thank God and our parents, from the bottom of our heart for bestowing his benediction on us throughoutthe course of this project.

We wish to express our sincere thanks to our beloved Chairman **Dr. P.V.RAVI Ph.D., MISTE**, for providing an opportunity to work on this project.

We would like to express our deep sense of gratitude to the Chief Executive Officer **Dr. R. ANUSHA B.E., M.S.(USA), Ph.D.,** for her valuable support and encouragement.

We extend sincere thanks to our Principal **Dr. D. LAKSHMANAN M.E., Ph.D.,** for supporting and encouraging to carry out our project.

We are highly indebted and graceful to our beloved Head of the Department **Dr. V. SARANYA M.Tech., Ph.D.,** for her valuable and her constant suggestions and persistent encouragement.

We express our gracious gratitude to our beloved guide **Mrs.S.PRINCY M .E.,** for his guidance and support throughout our project.

We also thank all the faculty members, teaching and non-teaching staff and lab in-charge of our department who helped us to complete our project successfully.

TABLE OF CONTENT

ABSTRACT

Phishing is a type of social engineering hack in which attackers trick victims into providing their login information on a form that sends the information to a malicious server. The fast developing subfield of data science is dependent on the application of machine learning as a central component. Phishing websites pose a significant threat to individuals and network environments by luring users to malicious URLs disguised as legitimate sites to steal private information. Detecting these threats is crucial for cybersecurity. Traditional neural network models can suffer from overfitting due to irrelevant features in training data, hindering effective phishing detection. index called Feature Validity Value to evaluate feature impact on phishing detection. Our approach mitigates overfitting, enhancing the neural network's performance. We implement LSTM and XGBoost algorithms for robust phishing detection. Experimental results demonstrate the accuracy and stability of the model. Furthermore, we deploy the framework as a browser plug-in, providing real-time phishing risk alerts to users during web browsing.

LIST OF FIGURES

LIST OF ABBREVIATIONS

URL	**U**niform **R**esource **A**llocation
CNN	**C**onventional **N**eural **N**etwork
IDS	**I**ntrusion **D**etection **S**ystem
KNN	**K- N**earest **N**eighbour
EDLWADS	**E**nsemble **D**eep **L**earning **b**ased **W**eb **A**ttack **D**etection **S**ystem
XGBoost	e**X**treme **G**radient
UML	**U**nified **M**odeling **L**anguage
DFD	**D**ata **F**low **D**iagram

CHAPTER 1

INTRODUCTION

The introduction discusses the transformative impact of internet services on various sectors like online banking, e-commerce, and social networking, alongside the rising threat of cyberattacks, particularly phishing. It explains the structure of Uniform Resource Locators (URLs) and the challenges of detecting malicious URLs due to their deceptive nature, often disguised in spam emails or through URL shorteners.

The text emphasizes the use of machine learning (ML) to combat cyber threats, specifically highlighting a convolutional neural network (CNN)-based model for malicious URL detection. Additionally, it touches upon predictive analytics and various ML techniques like decision trees, regression, neural networks, LSTM, and XGBoost, which are vital for data analysis and decision-making.

Overall, the introduction underscores the critical role of technology, especially ML, in addressing cybersecurity issues and leveraging data for insightful decision-making in modern business environments.

CHAPTER 2

LITERATURE SURVEY

2.1 TITLE: Malicious URL prediction based on community detection

AUTHORS: Zheng Li-xiong, Xu Xiao-lin, Li Jia, Zhang Lu and Pan Xuan-chen.

YEAR: 2020

DISCRIPTION :

Traditional Anti-virus technology is primarily based on static analysis and dynamic monitoring. However, both technologies are heavily depended on application files, which increase the risk of being attacked, wasting of time and network bandwidth. In this study, we propose a new graph-based method, through which we can preliminary detect malicious URL without application file. First, the relationship between URLs can be found through the relationship between people and URLs. Then the association rules can be mined with confidence of each frequent URLs. Secondly, the networks of URLs was built through the association rules. When the networks of URLs were finished, we clustered the date with modularity to detect communities and every community represents different types of URLs.

2.2 TITLE:Detecting malicious URLs using machine learning techniques

AUTHORS: Frank Vanhoenshoven, Gonzalo Nápoles, Rafael Falcon.

YEAR: 2016

DISCRIPTION:

The World Wide Web supports a wide range of criminal activities such as spam-advertised e-commerce, financial fraud and malware dissemination. Although the precise motivations behind these schemes may differ, the common denominator lies in the fact that unsuspecting users visit their sites. These visits can be driven by email, web search results or links from other web pages. In all cases, however, the user is required to take some action, such as clicking on a desired Uniform Resource Locator (URL). In order to identify these malicious sites, the web security community has developed blacklisting services. These blacklists are in turn constructed by an array of techniques including manual reporting, honeypots, and web crawlers combined with site analysis heuristics. Inevitably, many malicious sites are not blacklisted either because they are too recent or evaluated.

2.3 TITLE: Detecting Malware, Malicious URLs and Virus Using Machine Learning and Signature Matching

AUTHORS: Jatin Acharya, Anshul Chaudhary, Anish Chhabria

YEAR: 2019

Nowadays most of our data is stored on an electronic device. The risk of that device getting infected by Viruses, Malware, Worms, Trojan, Ransomware, or any unwanted invader has increased a lot these days. This is mainly because of easy access to the internet. Viruses and malware have evolved over time so identification of these files has become difficult. Not only by viruses and malware your device can be attacked by a click on forged URLs. Our proposed solution for this problem uses machine learning techniques and signature matching techniques. The main aim of our solution is to identify the malicious programs/URLs and act upon them. The core idea in identifying the malware is selecting the key features from the Portable Executable file headers using these features we trained a random forest model.

2.4 TITLE:A Comparative Analysis of Machine Learning Algorithms on Malicious URL Prediction.

AUTHORS: Tianlong Liu, Yu Qi, Liang Shi and Jianan Yan.

YEAR: 2019

Phishing is a form of fraudulent behavior where an entity or a person mimics to be a valid user. Phishing has become popular in cyber space and it is used as a tool for deceiving the users. Most of the phishing messages are difficult to interpret. In the existing literature, there are many schemes have addressed the phishing issue. Yet, there is no concrete solution to thwart such attacks. Taking this into consideration, to detect phishing threats, a machine learning-based prediction scheme is proposed in this article. From the experimental analysis, it is identified that logistic regression outperforms the other schemes in terms of accuracy and error rate. The accuracy on URL prediction with logistic regression is 97%.

2.5 TITLE: Detecting Malicious URLs Using Machine Learning Techniques Review and Research Directions

AUTHORS: Mohan Li, Yanbin Sun; Hui Lu, and Sabita Maharjan.

YEAR: 2022

In recent years, the digital world has advanced significantly, particularly on the Internet, which is critical given that many of our activities are now conducted online. As a result of attackers' inventive techniques, the risk of a cyberattack is rising rapidly. One of the most critical attacks is the malicious URL intended to extract unsolicited information by mainly tricking inexperienced end users, resulting in compromising the user's system and causing losses of billions of dollars each year. As a result, securing websites is becoming more critical. In this paper, we provide an extensive literature review highlighting the main techniques used to detect malicious URLs that are based on machine learning models, taking into consideration the limitations in the literature, detection technologies, feature types, and the datasets used. Moreover, due to the lack of studies related to malicious Arabic website detection, we highlight the directions of studies in this context

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

In existing system supervised learning algorithm has been implemented Machine learning based algorithm has been implemented KNN (k nearest neighbor) has been implemented.

Web pages not by their content but using their URLs, which is much faster as no delays are incurred in fetching the page content or parsing the text. The URL was segmented into multiple tokens from which classification features were extracted. The features modeled sequential dependencies between tokens.

To the fact, that the combination of high-quality URL segmentation and feature extraction improved the classification rate over several baseline techniques. Pursue a similar objective: topic classification from URLs. They trained separate binary classifiers for each topic (student, faculty, course and project) and were able to improve over the best reported F-measure.

In existing system different kind web phishing detection has been implemented using KNN, random forest, logistic regression algorithm has been implemented. Multi- Layer Perceptron’s (MLP) has been implemented for detection of web phishing web. Random forest has been implemented in existing model algorithm.

3.2 PROPOSED SYSTEM:

The proposed Ensemble Deep Learning based Web Attack Detection System (EDL-WADS) consists of four modules aimed at intelligent rule-based phishing website classification using a hybrid LSTM and XGBoost approach. The system entails:

3.2.1 Dataset Collection and Feature Extraction:

Gathering a comprehensive dataset of URLs, encompassing both phishing and legitimate websites. Extracting relevant features from URLs, such as domain length, presence of special characters, and URL length, crucial for distinguishing between phishing and legitimate URLs.

3.2.2 Intelligent Rule-based Classification:

Defining heuristic rules based on domain knowledge to swiftly classify URLs as phishing or legitimate. Applying rules like checking for homograph attacks and known malicious keywords to quickly classify URLs without complex machine learning models.

3.2.3 LSTM-based Classification:

Training LSTM models on URL datasets to capture sequential patterns and identify phishing or legitimate URLs based on learned features.

3.2.4 XGBoost-based Classification:

Utilizing XGBoost, a gradient boosting algorithm, to handle complex feature relationships and classify URLs based on non-linear patterns extracted from URL features.

3.2.5 Ensemble Approach:

Combining rule-based, LSTM-based, and XGBoost-based classifications through ensemble methods like voting or stacking to enhance overall classification accuracy and performance.

3.2.6 Model Evaluation and Optimization:

Evaluating trained models using performance metrics such as accuracy, precision, recall, F1-score, and area under the ROC curve. Iteratively fine-tuning and optimizing models to achieve the highest possible performance against evolving phishing techniques.

3.2.7 Deployment:

Integrating the optimized system into real-time URL scanning or deploying it as an API for URL classification in web browsers or applications, ensuring real-time protection against phishing attacks. definition, and meticulous model optimization. Regular updates and maintenance are vital to keep the system effective amid evolving cyber threats. The hybrid LSTM and XGBoost model offers a powerful solution by leveraging LSTM's sequence modeling capabilities and XGBoost's ability to handle complex feature interactions, potentially enhancing predictive performance in web attack detection. Thorough validation of the system's performance on specific datasets and tasks is crucial before deployment in a production environment.

CHAPTER 4

MODULE IMPLEMENTATION

4.1 MODULE LIST

- Data set collection
- Preprocessing
- Feature Extraction
- Training of feature Extraction

4.2 MODULE DESCRIPTION

4.2.1 Data set collection:

- Created at: Date that the response was sent data set is collected is based on reviews collection
- The data base creation is done based on positive and negative reviews based on collection of tweets we created the website on this system.
- The dataset for our project has been collected from the Third party website called Kaggle.
- The provided dataset includes 11430 URLs with 87 extracted features. The dataset is designed to be used as benchmarks for machine learning-based phishing detection systems. Features are from three different classes: 56 extracted from the structure and syntax of URLs, 24 extracted from the content of their correspondent pages, and 7 are extracted by querying external services. The dataset is balanced, it contains exactly 50% phishing and 50% legitimate URLs.

4.2.2 Preprocessing

The preprocessing steps are done based on the tokenization and stop words

- Tokenization - Separating each word from a sentence.
- Stop-words - Removing meaning less word from the sentence. One of the most important aspects of analyzing data is to ensure that our data is being understood by machines. Machines do not understand text, images, or videos, they can comprehend only 1’s and 0’s. To be able to provide an input consisting of 1’s and 0’s is a multistep process.= Pre-processing the data is an absolute necessity and calls for a technique called data cleaning which involves transforming raw data into a machine-understandable format we are delete the rows and columns of data from data set .
- Null values processing, Normalization are the process using in the module.
- We preprocessed of data null the values for reduce the data .

4.2.3 Feature extraction

- After collection of data set we are applied stemming tokenization based on sentimental words collection we are build our model.
- We build the model based on sentimental collection of positive and negative tweets.

4.2.4Training of feature extraction

- After apply the machine learning algorithm for extracted features
- We split the dataset train and test dataset for splitted the model for training and testing dataset

CHAPTER 5

SYSTEM SPECIFICATION

5.1 HARDWARE REQUIREMENTS

- Processor : Pentium - IV
- RAM : 4 GB (min)
- Hard Disk : 20 GB

5.2 SOFTWARE REQUIREMENTS

- Operating System : Windows 7 or 8
- Front End : python Idle version 3.7

CHAPTER-6

SOFTWARE DESCRIPTION

6.1 Python Technology

Python is an interpreter, high-level, general-purpose programming language. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. **Python** is often described as a "batteries included" language due to its comprehensive standard library.

6.2 Python Programing Language

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by Meta programming and met objects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python packages with a wide range of functionality, including:

- Easy to Learn and Use
- Expressive Language
- Interpreted Language
- Cross-platform Language
- Free and Open Source
- Object-Oriented Language
- Extensible
- Large Standard Library
- GUI Programming Support
- Integrated

6.3 Features

- **Easy to Learn and Use:**

  Python's syntax is straightforward and expressive.

- **Interpreted and Cross-platform:**

  Python code can run on various platforms without modifications.

- **Open Source and Extensible:**

  The language is freely available and extensible via modules.

- **Large Standard Library:**

  Python comes with a comprehensive standard library.

- **Support for GUI Programming:**

  There are libraries like Tkinter for building graphical user interfaces.

- **Dynamic Typing:**

  Python uses dynamic typing and supports duck typing.

- **Readable and Concise Syntax:**

  Python emphasizes readability and simplicity.

6.4 Python's Application Areas

- **CPython:**

  The reference implementation written in C.

- **PyPy:**

  A fast, compliant interpreter with a just-in-time compiler.

- **Jython and IronPython:**

  Enable Python to interact with Java and .NET libraries respectively.

- **MicroPython and CircuitPython:**

  Optimized for microcontrollers.

- **Other Compilers:**

  Cython, Pyjs, Numba, and others compile Python to different languages.

6.5 Python Libraries and Tools

- **Pandas:**

Used for data manipulation and analysis.

- **Sphinx, Epydoc:**

Tools for generating API documentation.

- **IDEs:**

Python supports various development environments like IDLE, IPython, and PyCharm.

CHAPTER 7

SYSTEM DESIGN

7.1 SYSTEM ARCHITECTURE

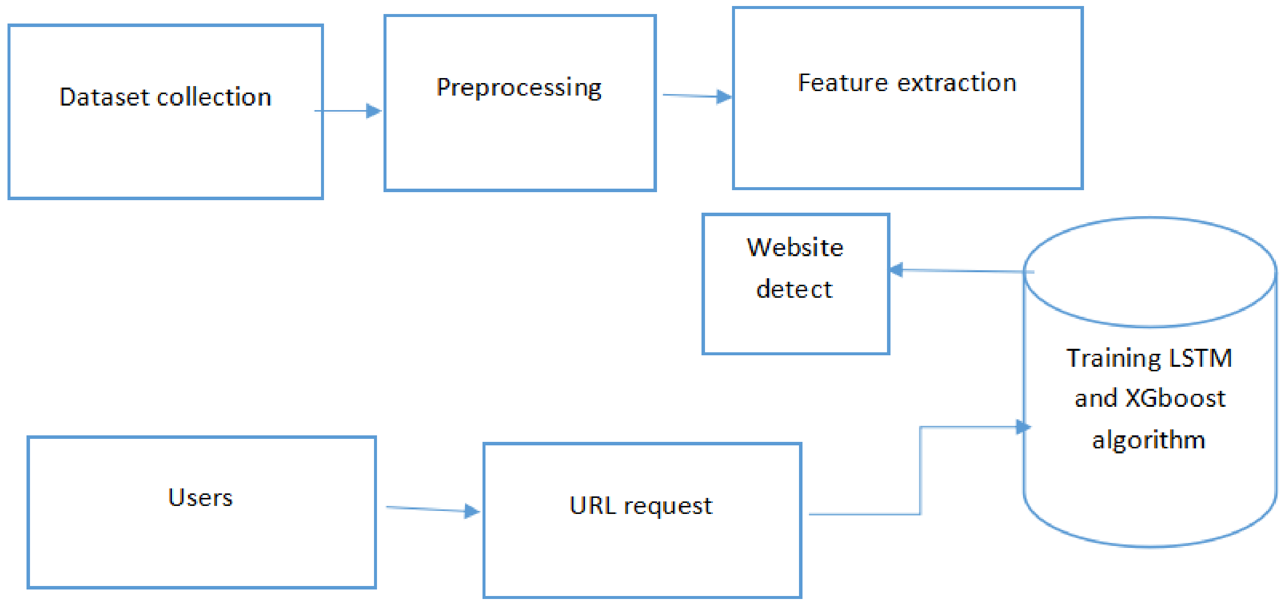

Fig 7.1 System Architecture

CHAPTER 8

ALGORITHM EXPLANATION

8.1 LONG SHORT-TERM MEMORY (LSTM)

The Long Short-Term Memory (LSTM) algorithm is a type of recurrent neural network (RNN) architecture designed to effectively model and learn long-range dependencies in sequential data. It overcomes the vanishing gradient problem associated with traditional RNNs, making it particularly suitable for tasks involving sequences of data such as time series prediction, natural language processing (NLP), speech recognition, and more.

8.1.1 LSTM ALGORITHM

```
Initialize:

Input sequence: X = [x^(1), x^(2), ..., x^(T)]

Initialize cell state c and hidden state h to zero vectors (c = 0, h = 0)
For each time step t from 1 to T
    forget_gate = sigmoid(W_f * [h, x^(t)] + b_f)
    input_gate = sigmoid(W_i * [h, x^(t)] + b_i)
    candidate_cell_state = tanh(W_c * [h, x^(t)] + b_c)
    output_gate = sigmoid(W_o * [h, x^(t)] + b_o)
    c = forget_gate * c + input_gate * candidate_cell_state
    h = output_gate * tanh(c)
    End For
Output: Final hidden state h
```

8.2 EXTREME GRADENT BOOSTNG(XGBOOST)

XGBoost is a popular open-source gradient boosting machine learning library that is designed to optimize and boost the performance of decision tree-based models. It was developed by Tianqi Chen and Carlos Guestrin and is written in C++ with bindings available for several programming languages, including Python, R, Java, and Scala.

XGBoost is known for its efficiency and accuracy, and it has been widely used in various machine learning competitions and real-world applications. It uses an ensemble technique called boosting, which sequentially adds weak models (typically decision trees) to the ensemble in a way that each subsequent model corrects the errors of the previous models, leading to improved overall performance

8.2.1 XGBOOST ALGORITHM

Input:

Training dataset (D_train) consisting of features (X) and target values (y)

Hyperparameters such as learning rate (eta), number of trees (num_trees), maximum depth of trees (max_depth), etc.

Output:

Trained ensemble of boosted trees

Procedure:

STEP 1 : Initialize the model:

Initialize the ensemble of trees with a base prediction.

STEP 2 : For t = 1 to num_trees:

a.Compute the gradient (g_i) and hessian (h_i) for each instance:

For each training instance i:

g_i = dL(y_i, F_{t-1}(x_i)) / dF_{t-1}(x_i)

h_i = d^2L(y_i, F_{t-1}(x_i)) / (dF_{t-1}(x_i))^2

where L is the loss function, F_{t-1}(x_i) is the prediction from previous trees.

b. Fit a regression tree to the negative gradient:

Train a regression tree on the dataset (X, -g) with the following properties:

- Max depth = max_depth
- Use hessian (h) to calculate split scores
- Regularization terms (e.g., min_child_weight) can be incorporated

c. Update the model:

Update the ensemble:

F_t(x) = F_{t-1}(x) + eta * tree(x)

STEP 3: Output the trained ensemble of trees.

Prediction (given a new instance x):

Compute the prediction F(x) using the trained ensemble of trees:

F(x) = sum(eta * tree(x))

Return F(x) as the final prediction.

CHAPTER 9

CONCLUSION AND FUTURE WORK

9.1 CONCLUSION:

The project "Strengthening defenses with data analytics against malicious links " marks a significant stride in the realm of cybersecurity. By harnessing the power of data analytics, this endeavor has illuminated novel avenues for identifying and thwarting online threats posed by malicious URLs. Through meticulous analysis of diverse features encompassing URL structure, content, and historical behavior, the project has unveiled patterns and anomalies indicative of malicious intent. The successful implementation of machine learning algorithms, coupled with robust feature engineering and model optimization, has yielded promising results in accurately classifying URLs as either benign or malicious. This not only bolsters the efficacy of traditional cybersecurity measures but also empowers defenders with proactive tools to stay ahead of evolving threats.

Furthermore, the project underscores the importance of interdisciplinary collaboration, drawing insights from fields such as data science, cybersecurity, and behavioral analysis. By amalgamating expertise from these domains, a holistic approach to threat detection and mitigation has been realized, capable of addressing the multifaceted nature of cyber threats.

Looking ahead, the findings and methodologies elucidated in this project hold profound implications for the cybersecurity landscape. Integration into real-world systems, such as web browsers, firewalls, and threat intelligence platforms, can enhance their capabilities in detecting and neutralizing malicious URLs in real-time. Moreover, ongoing research and development efforts can further refine and extend the efficacy of data analytics in combating emerging cyber threats.

9.2 FUTURE WORK:

Future work in this area could include exploring additional URL-based features for improved classification accuracy, such as the presence of certain keywords or patterns in the URL. Additionally, the use of machine learning algorithms such as deep learning could be explored to further enhance the accuracy and efficiency of the classification process. Another area for future work could involve the development of a real-time phishing website classification system that could be integrated with web browsers or email clients. Such a system could provide users with immediate feedback on the trustworthiness of websites or email links, thereby enhancing their online security and reducing the risk of phishing attacks.

The proposed approach offers a promising direction for improving the detection and classification of phishing websites, with potential applications in a variety of domains, including cyber security, e-commerce, and online banking. Further research and development in this area could have significant implications for enhancing online security and protecting users from malicious online activity.

CHAPTER 10

APPENDIX

10.1 SOURCE CODE

```
import ipaddress

import re

import urllib.request

from bs4 import BeautifulSoup

import socket

import requests

from googlesearch import search

import whois

from datetime import date, datetime

import time

from dateutil.parser import parse as date_parse

from urllib.parse import urlparse


class FeatureExtraction:

    features = []

    def _init_(self,url):

        self.features = []

        self.url = url

        self.domain = ""

        self.whois_response = ""
```

```
self.urlparse = ""

self.response = ""

self.soup = ""


try:

    self.response = requests.get(url)

    self.soup = BeautifulSoup(response.text, 'html.parser')

except:

    pass

 try:

    self.urlparse = urlparse(url)

    self.domain = self.urlparse.netloc

except:

    pass

try:

    self.whois_response = whois.whois(self.domain)

except:

    pass

self.features.append(self.UsingIp())

self.features.append(self.longUrl())

self.features.append(self.shortUrl())

self.features.append(self.symbol())

self.features.append(self.redirecting())

self.features.append(self.prefixSuffix())
```

```
self.features.append(self.SubDomains())
self.features.append(self.Hppts())
self.features.append(self.DomainRegLen())
self.features.append(self.Favicon())

self.features.append(self.NonStdPort())
self.features.append(self.HTTPSDomainURL())
self.features.append(self.RequestURL())
self.features.append(self.AnchorURL())
self.features.append(self.LinksInScriptTags())
self.features.append(self.ServerFormHandler())
self.features.append(self.InfoEmail())
 self.features.append(self.AbnormalURL())
self.features.append(self.WebsiteForwarding())
self.features.append(self.StatusBarCust())

self.features.append(self.DisableRightClick())
self.features.append(self.UsingPopupWindow())
self.features.append(self.IframeRedirection())
self.features.append(self.AgeofDomain())
self.features.append(self.DNSRecording())
self.features.append(self.WebsiteTraffic())
self.features.append(self.PageRank())
self.features.append(self.GoogleIndex())
```

```
        self.features.append(self.LinksPointingToPage())

        self.features.append(self.StatsReport())

    def UsingIp(self):

        try:

            ipaddress.ip_address(self.url)

            return -1

        except:

            return 1

    def longUrl(self):

        if len(self.url) < 54:

            return 1

        if len(self.url) >= 54 and len(self.url) <= 75:

            return 0

        return -1

    def shortUrl(self):

        match                                                                                           =
re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|'

'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|'

'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|'

'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|'

'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|'

'q\.gs|is\.gd|po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|'
```

```
'x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.m
e|v\.gd|tr\.im|link\.zip\.net', self.url)
        if match:
            return -1
        return 1
    def symbol(self):
        if re.findall("@",self.url):
            return -1
        return ;
    def redirecting(self):
        if self.url.rfind('//')>6:
            return -1
        return 1
    def prefixSuffix(self):
        try:
            match = re.findall('\-', self.domain)
            if match:
     return -1
            return 1
        except:
            return -1
    def SubDomains(self):
        dot_count = len(re.findall("\.", self.url))
        if dot_count == 1:
```

```
        return 1
    elif dot_count == 2:
        return 0
    return -1
```

HTTPS

```
def Hppts(self):
    try:
        https = self.urlparse.scheme
        if 'https' in https:
            return 1
        return -1
    except:
        return 1
```

DomainRegLen

```
def DomainRegLen(self):
    try:
        expiration_date = self.whois_response.expiration_date
        creation_date = self.whois_response.creation_date
        try:
            if(len(expiration_date)):
                expiration_date = expiration_date[0]
        except:
```

```
            pass
        try:
            if(len(creation_date)):
                creation_date = creation_date[0]
        except:
       pass

        age = (expiration_date.year-creation_date.year)*12+ (expiration_date.month-creation_date.month)
        if age >=12:
            return 1
        return -1
    except:
        return -1
```

Favicon

```
 def Favicon(self):
    try:
        for head in self.soup.find_all('head'):
            for head.link in self.soup.find_all('link', href=True):
                dots = [x.start(0) for x in re.finditer('\.', head.link['href'])]
                if self.url in head.link['href'] or len(dots) == 1 or domain in head.link['href']:
                    return 1
        return -1
    except:
```

```
        return -1
```

DNSRecording

```
  def DNSRecording(self):
     try:
        creation_date = self.whois_response.creation_date
        try:
           if(len(creation_date)):
              creation_date = creation_date[0]
        except:
           pass

        today  = date.today()
        age = (today.year-creation_date.year)*12+(today.month-creation_date.month)
        if age >=6:
           return 1
   return -1
     except:
        return -1
```

WebsiteTraffic

```
  def WebsiteTraffic(self):
     try:
        rank                                                                    =
BeautifulSoup(urllib.request.urlopen("http://data.alexa.com/data?cli=10&dat=s&url="
+ url).read(), "xml").find("REACH")['RANK']
        if (int(rank) < 100000):
```

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

PageRank

```
def PageRank(self):
    try:
        prank_checker_response = requests.post("https://www.checkpagerank.net/index.php", {"name": self.domain})

        global_rank = int(re.findall(r"Global Rank: ([0-9]+)", rank_checker_response.text)[0])
        if global_rank > 0 and global_rank < 100000:
            return 1
        return -1
    except:
        return -1
```

GoogleIndex

```
def GoogleIndex(self):
    try:
        site = search(self.url, 5)
        if site:
return 1
        else:
            return -1
```

```
    except:
        return 1
```

LinksPointingToPage

```
def LinksPointingToPage(self):
    try:
        number_of_links = len(re.findall(r"<a href=", self.response.text))
        if number_of_links == 0:
            return 1
        elif number_of_links <= 2:
            return 0
        else:
            return -1
    except:
        return -1
```

```
# 30. StatsReport
def StatsReport(self):
    try:
        url_match = re.search(
'at\.ua|usa\.cc|baltazarpresentes\.com\.br|pe\.hu|esy\.es|hol\.es|sweddy\.com|myjino\.ru|96\.lt|ow\.ly', url)
        ip_address = socket.gethostbyname(self.domain)
        ip_match =
```

```
re.search('146\.112\.61\.108|213\.174\.157\.151|121\.50\.168\.88|192\.185\.217\.116|78\.46\.211\.158|181\.174\.165\.13|46\.242\.145\.103|121\.50\.168\.40|83\.125\.22\.219|46\.242\.145\.98|'

'107\.151\.148\.44|107\.151\.148\.107|64\.70\.19\.203|199\.184\.144\.27|107\.151\.148\.108|107\.151\.148\.109|119\.28\.52\.61|54\.83\.43\.69|52\.69\.166\.231|216\.58\.192\.225|'

'118\.184\.25\.86|67\.208\.74\.71|23\.253\.126\.58|104\.239\.157\.210|175\.126\.123\.219|141\.8\.224\.221|10\.10\.10\.10|43\.229\.108\.32|103\.232\.215\.140|69\.172\.201\.153|'

'216\.218\.185\.162|54\.225\.104\.146|103\.243\.24\.98|199\.59\.243\.120|31\.170\.160\.61|213\.19\.128\.77|62\.113\.226\.131|208\.100\.26\.234|195\.16\.127\.102|195\.16\.127\.157|'

'34\.196\.13\.28|103\.224\.212\.222|172\.217\.4\.225|54\.72\.9\.51|192\.64\.147\.141|198\.200\.56\.183|23\.253\.164\.103|52\.48\.191\.26|52\.214\.197\.72|87\.98\.255\.18|209\.99\.17\.27|'

'216\.38\.62\.18|104\.130\.124\.96|47\.89\.58\.141|78\.46\.211\.158|54\.86\.225\.156|54\.82\.156\.19|37\.157\.192\.102|204\.11\.56\.48|110\.34\.231\.42', ip_address)

            if url_match:

                return -1

            elif ip_match:

                return -1

            return 1

        except:

            return 1

    def getFeaturesList(self):

        return self.featur
```

10.2 SCREENSHOT

Fig 10.2.1 Home Screen

Fig 10.2.2 Output Screen

CHAPTER-11

REFERENCES

[1] Symantec, “2016 Internet security threat report,” https://www.symantec. Com/security-center/threat-report, 2016, [Online; accessed 11-Aug2016].

[2] May Almousa, Tianyang Zhang, Abdolhossein Sarrafzadeh and Mohd Anwar, "Phishing website detection: How effective are deep learning-based models and hyperparameter optimization?", *Security and Privacy*, pp. e256, 2022.

[3] P. de las Cuevas, Z. Chelly, A. Mora, J. Merelo, and A. EsparciaAlcazar, “An improved decision system for URL accesses based on a ´ rough feature selection technique,” in Recent Advances in Computational Intelligence in Defense and Security. Springer, 2016, pp. 139–167.

[4] A. Mora, P. De las Cuevas, and J. Merelo, “Going a step beyond the black and white lists for URL accesses in the enterprise by means of categorical classifiers,” ECTA, pp. 125–134, 2014.

[5] M.-Y. Kan and H. O. N. Thi, “Fast webpage classification using url features,” in Proceedings of the 14th ACM international conference on Information and knowledge management. ACM, 2005, pp. 325–326.

[6] E. Baykan, M. Henzinger, L. Marian, and I. Weber, “Purely URL-based topic classification,” in Proceedings of the 18th international conference on World wide web. ACM, 2009, pp. 1109–1110.

[7] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, “Beyond blacklists: learning to detect malicious web sites from suspicious urls,” in Proceedings of the 15th ACM

SIGKDD international conference on Knowledge discovery and data mining. ACM, 2009, pp. 1245–1254.

[8] May Almousa and Mohd Anwar, "Detecting exploit websites using browser-based predictive analytics", *2019 17th International Conference on Privacy Security and Trust (PST)*, pp. 1-3, 2019..

[9] P. Zhao and S. C. Hoi, “Cost-sensitive online active learning with application to malicious URL detection,” in Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2013, pp. 919–927.

[10] Ali Aljofey, Qingshan Jiang, Abdur Rasool, Hui Chen, Wenyin Liu, Qiang Qu, et al., "An effective detection approach for phishing websites using url and html features", *Scientific Reports*, vol. 12, no. 1, pp. 1-19, 2022.

[11] R. O. Duda, P. E. Hart, and D. G. Stork, Pattern classification, 2nd ed. John Wiley & Sons, 2012.

[12] S. Jagadeesan, A. Chaturvedi and S Kumar, "URL Phishing Analysis using Random Forest", *Int. J. Pure Appl. Math*, vol. 118, no. 20, pp. 4159-4163, 2018.

[13] V. Lopez, A. Fern ´ andez, S. Garc ´ ´ıa, V. Palade, and F. Herrera, “An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics,” Information Sciences, vol. 250, pp. 113–141, 2013.

[14] B. Frenay and M. Verleysen, “Classification in the presence of label ´ noise: a survey,” Neural Networks and Learning Systems, IEEE Transactions on, vol. 25, no. 5, pp. 845–869, 2014.

[15] S.N. Shivappriya, M. Jasmine Pemeena Priyadarsini, Andrzej Stateczny, C. Puttamadappa and B. D. Parameshachari, "Cascade object detection and remote sensing object detection method based on trainable activation function", *Remote Sensing*, vol. 13, no. 2, pp. 200, 2021.

[16] I. Triguero, S. Garc´ıa, and F. Herrera, "Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study," Knowledge and Information Systems, vol. 42, no. 2, pp. 245–284, 2015.

[17] G. Tsoumakas and I. Katakis, "Multi-label classification: an overview," International Journal of Data Warehousing and Mining, vol. 3, no. 3, pp. 1–13, 2007.

[18] M. Fernandez-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do ´ we Need Hundreds of Classifiers to Solve Real World Classification Problems?" Journal of Machine Learning Research, vol. 15, pp. 3133– 3181, 2014.

[19] M. Wainberg, B. Alipanahi, and B. J. Frey, "Are random forests truly the best classifiers?" Journal of Machine Learning Research, vol. 17, no. 110, pp. 1–5, 2016.

[20] Ping Yi, Yuxiang Guan, Futai Zou, Yao Yao, Wei Wang and Ting Zhu, "Web Phishing DetectionUsing a Deep Learning Framework", *Wireless Communications and Mobile Computing*, vol. 2018, pp. 1-9, 2018