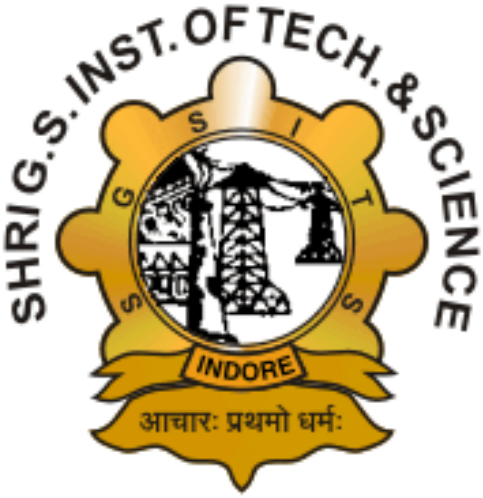
“PHISHING WEBSITES DETECTION SYSTEM”



Final Project Report Submitted to
Rajiv Gandhi Proudyogiki Vishwavidhyalaya, Bhopal
towards partial fulfillment of
the degree of
Bachelor of Engineering in Computer Engineering

Guided By:
Mr. Rajesh Kumar Dhakad
Associate Professor
Department of Computer Engg.

Submitted By:
0801CS171006 - Aditya Patidar
0801CS171027 - Harsh Pastaria
0801CS171042 - Mansoor Rangwala
0801CS171060 - Raghav Parsai
0801CS171068 - Sarvesh Gupta

DEPARTMENT OF COMPUTER ENGINEERING
SHRI G.S. INSTITUTE OF TECHNOLOGY AND SCIENCE, INDORE(M.P.)
2020-2021

SHRI G.S. INSTITUTE OF TECHNOLOGY AND SCIENCE, INDORE(M.P.)

RECOMMENDATION

The project report of Phase-I entitled ***"PHISHING WEBSITES DETECTION SYSTEM"*** submitted by: **0801CS171006 - Aditya Patidar, 0801CS171027 - Harsh Pastaria, 0801CS171042 - Mansoor Rangwala, 0801CS171060 - Raghav Parsai, 0801CS171068 - Sarvesh Gupta**, students of B.E. IV year in the session 2020-2021, towards partial fulfillment of the degree of **Bachelor of Engineering in Computer Engineering** of Rajiv Gandhi Proudyogiki VishwaVidhyalaya, Bhopal is a satisfactory account of their work.

Mr. Rajesh Kumar Dhakad
Project Guide
Department of Computer Engg.

Dr. Urjita Thakar
Head
Department of Computer Engg.

Dean (Academics)
S.G.S.I.T.S. Indore

SHRI G.S. INSTITUTE OF TECHNOLOGY AND SCIENCE, INDORE(M.P.)

CERTIFICATE

The project report of Phase-I entitled ***"PHISHING WEBSITES DETECTION SYSTEM"*** submitted by: **0801CS171006 - Aditya Patidar, 0801CS171027 - Harsh Pastaria, 0801CS171042 - Mansoor Rangwala, 0801CS171060 - Raghav Parsai, 0801CS171068 - Sarvesh Gupta**, students of B.E. IV year in the session 2020-2021, towards partial fulfillment of the degree of **Bachelor of Engineering in Computer Engineering** of Rajiv Gandhi Proudyogiki VishwaVidhyalaya, Bhopal is a satisfactory account of their work.

Internal Examiner **External Examiner**

Date:

ACKNOWLEDGEMENT

With great pleasure and sense of obligation we express our heartfelt gratitude to my esteemed guide **Mr. Rajesh Kumar Dhakad**, Associate Professor, Department of Computer Engineering, S.G.S.I.T.S. Indore, whose constant encouragement enabled us to work enthusiastically. Our project guide, in spite of their heavy work commitments and busy schedule, have been there for their invaluable guidance and ever ready support. Their persistent encouragement, perpetual motivation, everlasting patience and excellent expertise in discussions, during progress of the work, have benefited to an extent which is beyond expression. Their contributions are beyond the purview of the acknowledgement.

We would like to give warm expression of thanks to **Dr. Rakesh Saxena**, Director, S.G.S.I.T.S. Indore, for providing all the facilities and academic environment during the course of study.

We sincerely wish to express, our gratefulness to all the members of staff of Computer Engineering Department who have extended their cooperation at all times and have contributed in their own way in developing the project.

We are thankful to our parents for being a constant source of encouragement in all our endeavors. The support of our friends is worth appreciation and thankfulness.

0801CS171006 - Aditya Patidar _____

0801CS171027 - Harsh Pastaria _____

0801CS171042 - Mansoor Rangwala _____

0801CS171060 - Raghav Parsai _____

0801CS171068 - Sarvesh Gupta _____

ABSTRACT

Phishing is a deceitful attempt for obtaining the sensitive information like credit card details, user names and passwords. It is one of the social engineering methods that gathers personal information through websites such as malicious websites and deceptive e-mail to canvass personal information from a company or an individual by prance as a trustworthy entity or organization. Phishing often attacks email by using as a vehicle and even sending messages by email to users that represent a part of a company or an institution who perform business such as financial institution, banking etc. Phishing is becoming more malicious day by day and its detection is very important. In cyberspace, phishing is motivating the researchers to develop the model through which we can develop more security towards the safe services provided by the web. Here we discuss types of phishing and conflicts due to it.

CONTENTS

Chapter 1

Introduction

In this chapter an overview of the project beginning with the introduction to Phishing Detection have been discussed. Next, a brief introduction is given on model integration. Need of the project and problem statement are given and the solution approach has been explained briefly. Next the project objective is discussed. Summary of the proposed project has been mentioned. At the end, organization of the report has been given.

1.1 Preamble

Phishing is an unfair attempt to obtain sensitive information or data, such as usernames, passwords and credit card details, by pretending to be oneself as a trustworthy entity in an electronic communication. Phishing often directs users to enter personal information at a fake website which matches the look and feel of the legitimate site such fake websites are known as Phishing Websites.
Typically a victim receives a message that appears to have been sent by a known contact or organization. The message contains malicious software targeting the user's computer or has links to direct victims to malicious websites in order to trick them into divulging personal and financial information, such as passwords, account IDs or credit card details.
Phishing is popular among attackers, since it is easier to trick someone into clicking a malicious link which seems legitimate than trying to break through a computer's defense systems. The malicious links within the body of the message are designed to make it appear that they go to the spoofed organization using that organization's logos and other legitimate contents.

1.2 Need of the Project

Web phishing is one of many security threats to web services on the Internet. Web phishing aims to steal private information, such as usernames, passwords, and credit card details, by way of impersonating a legitimate entity. It will lead to information disclosure and property damage.
Unfortunately, many of the existing phishing-detection tools, especially those that depend on an existing blacklist, suffer limitations such as low detection accuracy and high false alarm
Therefore to safeguard online users from becoming victims of online fraud, divulging confidential

information to an attacker among other effective uses of phishing as an attacker's tool, phishing detection tools play a vital role in ensuring a secure online experience for users.

1.3 Problem Statement

Phishing Websites are duplicate Web pages created to mimic real Websites to deceive people to get their personal information. Because of the adaptability of their tactics with little cost Detecting and identifying Phishing Websites is really a complex and dynamic problem. Phishing attacks can lead to huge financial losses for customers of banking and financial services. Phishing detection techniques do suffer low detection accuracy and high false alarm especially when novel phishing approaches are introduced. Today Researchers have come up with machine learning frameworks to detect phishing sites, but they are not in a state to be used by individuals having no technical knowledge. To make sure that these tools are accessible to every individual, we have improvised and introduced detection methods as the browser plugin.

1.4 Objectives

- The goal of our project is to implement a machine learning solution to the problem of detecting phishing and malicious web links.
- The phishing website can be detected based on some important characteristics like URL and Domain Identity, and security and encryption criteria in the final phishing detection rate.
- We implemented classification algorithm and techniques to extract the phishing data sets criteria to classify their legitimacy.
- The end result of our project will be a software product which uses machine learning algorithm to detect malicious URLs.

1.5 Proposed Approach

Anti-phishing modules look-up the URL on dozens of phishing symptoms, such as domain names, frame usage, shortening service, double slash redirection and input encryption usage, and compare them with other indications. Heuristic analysis helps recognize a phishing URL even if it's not yet featured in available databases , but prediction based on URL alone is not reliable and can give lots of false alarm as newly registered website's URL also have some features which can be classified as phishing. To overcome this we will use another filtering mechanism in which the structure and the content of the webpage is compared with the legitimate webpage of similar domain names.

1.6 Organization of the Report

This section provides a brief summary about the different sections of report. Other sections of this report are categorized as follows:

- Chapter 1 introduces us to the project, its need and our approach to resolving the problem.
- Chapter 2 is devoted to the Background and Tools / Technologies used in this project.
- Chapter 3 covers Literature Survey about the project.
- Chapter 4 elaborates on the design of the project. It explains the architecture of the system and various services, interfaces, database design for the project.
- Chapter 5 presents the Implementation and Code Snippets of all the modules.
- Chapter 6 shows the results of testing done on the individual module and the integrated project.
- Chapter 7 concludes the whole project and shows the accuracy of it.

Chapter 2

Background

2.1 Tools and Technologies

Our backend of the project is created in python and below are the major libraries used in our project.

Module 1:

- python-whois
- Ipaddress
- requests
- numpy
- Scikit learn
- Pandas

Module 2:

- gensim
- google
- bs4

Integration Module:

- http.server
- ssl
- json
- urllib.parse

Chrome browser plugin development using HTML,CSS & JavaScript

Chapter 3

Literature Review

3.1 Summary of reviewed Literature

A. Google Safe Browsing API:
It maintains a detected list of phishing URLs which is frequently udated by the Google. It compares the URL entered by a user and compares with the list. If a match is found, it gives the phishing alert. Google Safe Browsing API is used by web browsers such as Mozilla, FireFox, Google Chrome etc. This protocol is still in experimental stage only. It's drawback is that the phishing sites do not exist for a long time. The attackers continuously change the URL's and IP addresses or may delete the phishing site from their server. Hence, maintaining a huge black list phishing URLs and frequently updating is not easy.

B. CANTINA:
It is based on a TF-IDF information retrieval algorithm. TF stands for the 'term frequency' which is the total number of times a particular word appears in a document. IDF stands for the 'inverse document frequency' which is the number of times a particular word is found in the group of documents. These are often used in information retrieval and text mining operations. It's method is to search the google with 5 words having maximum TF-IDF values. If the first n results contains the URL then it is treated as a legitimate site, otherwise as a phishing site. It's drawback is that unregistered legitimate sites also get blocked and shown as phishing sites.

C. Analysing Snapshots and web content of web pages:
Hara, M,Yamada A, and Miyake Y suggested a method based on the comparison of browser rendered screen shots for visual similarity.
Eric Medvet, and Engin Kilda suggested a method based on the comparison of the web page signatures. The signatures are generated by using text, image and the overall browser rendered web page image. Text similarity, image similarity and overall similarity of the web pages are considered in this. Its drawback is that due to the dynamic nature of the websites, web contents are changing continuously over the time. Hence it cant detect the changes in the web site.

Best software's available in the market for phishing detection:

INKY Phish Fence: It is an affordable cloud-based email security platform designed to be far more than artificially intelligent. She understands email, searches for signs of fraud, and can spot imposters by a pixel.

BitDam: BitDam offers phishing detection and prevention as part of its comprehensive Advanced Threat Protection solution for business collaboration platforms which includes protection for email, cloud drives, and Instant Messaging – covering threats of any type hidden in files and links.

Cofense: Cofense (formerly PhishMe) provides organizations with the ability to improve their employees' resilience towards spear phishing, malware, and drive-by attacks and further, facilitate employee-sourced detection of such attacks.

Chapter 4

Analysis & Design

4.1 Detailed Problem Statement

Phishing detection techniques do suffer low detection accuracy and high false alarm especially when novel phishing approaches are introduced. Besides, the most common technique used, blacklist-based method is inefficient in responding to emanating phishing attacks since registering new domain has become easier, no comprehensive blacklist can ensure a perfect up-to-date database. There has been a 3x - 4x increase in phishing over mass scale since December and noticed more in Tier 2 and Tier 3 cities.Today Researchers have come up with machine learning frameworks to detect phishing sites, but they are not in a state to be used by individuals having no technical knowledge. To make sure that these tools are accessible to every individual, we have improvised and introduced detection methods as the browser plugin.

4.2 Requirement Analysis

4.2.1 Functional Requirements

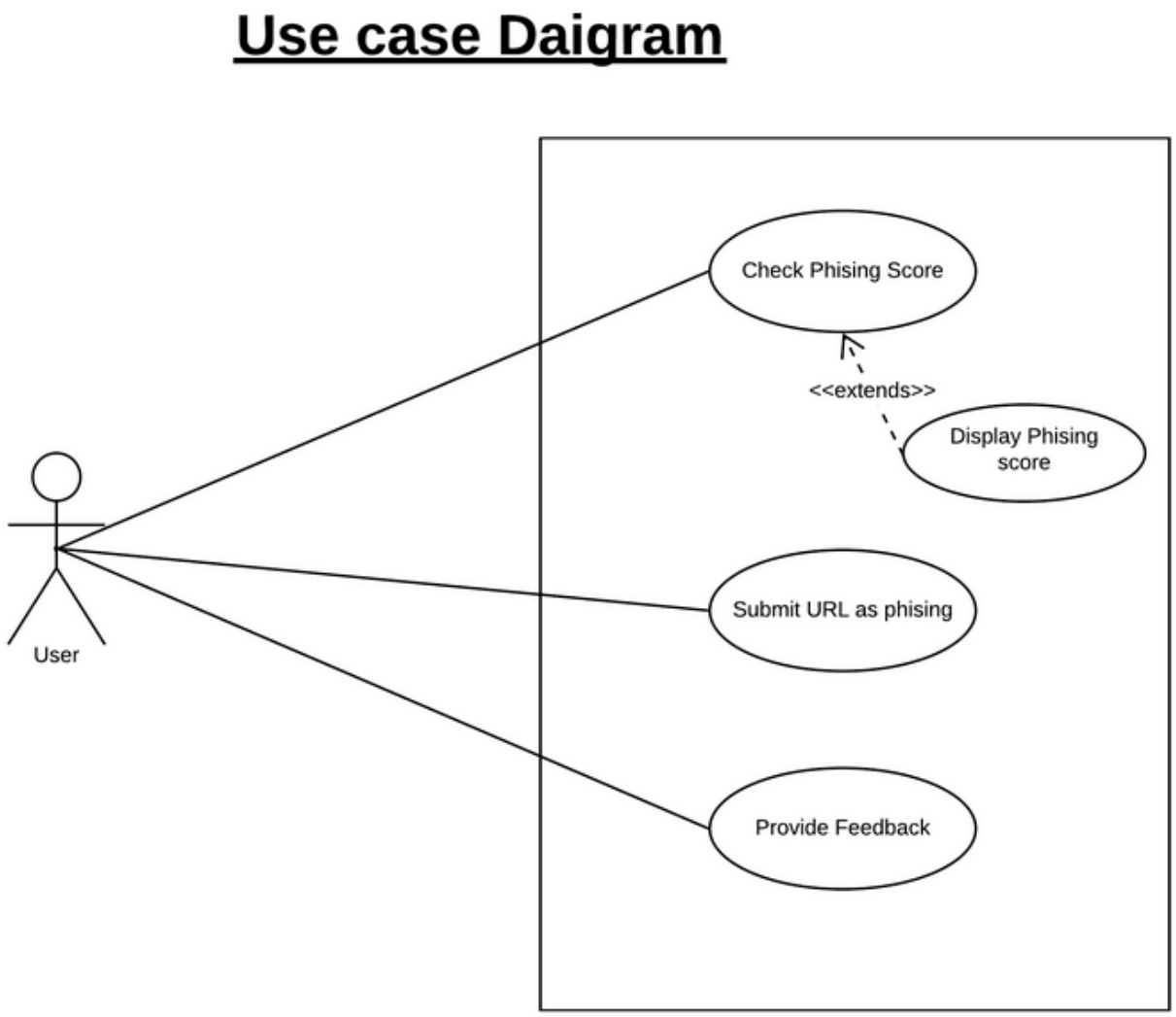
- **Typo squatting detection** in domain name also called URL hijacking, is a form of cybersquatting which relies on mistakes such as typographical errors made by Internet users when inputting a website address into a web browser or based on typographical errors. Ex google.com and gooogle.com

- **Cybersquatting or Domain Squatting detection:** It is registering, trafficking in, or using a domain name with bad faith intent to profit from the goodwill of a trademark belonging to someone else. Ex flipkart.com and flipkart.co

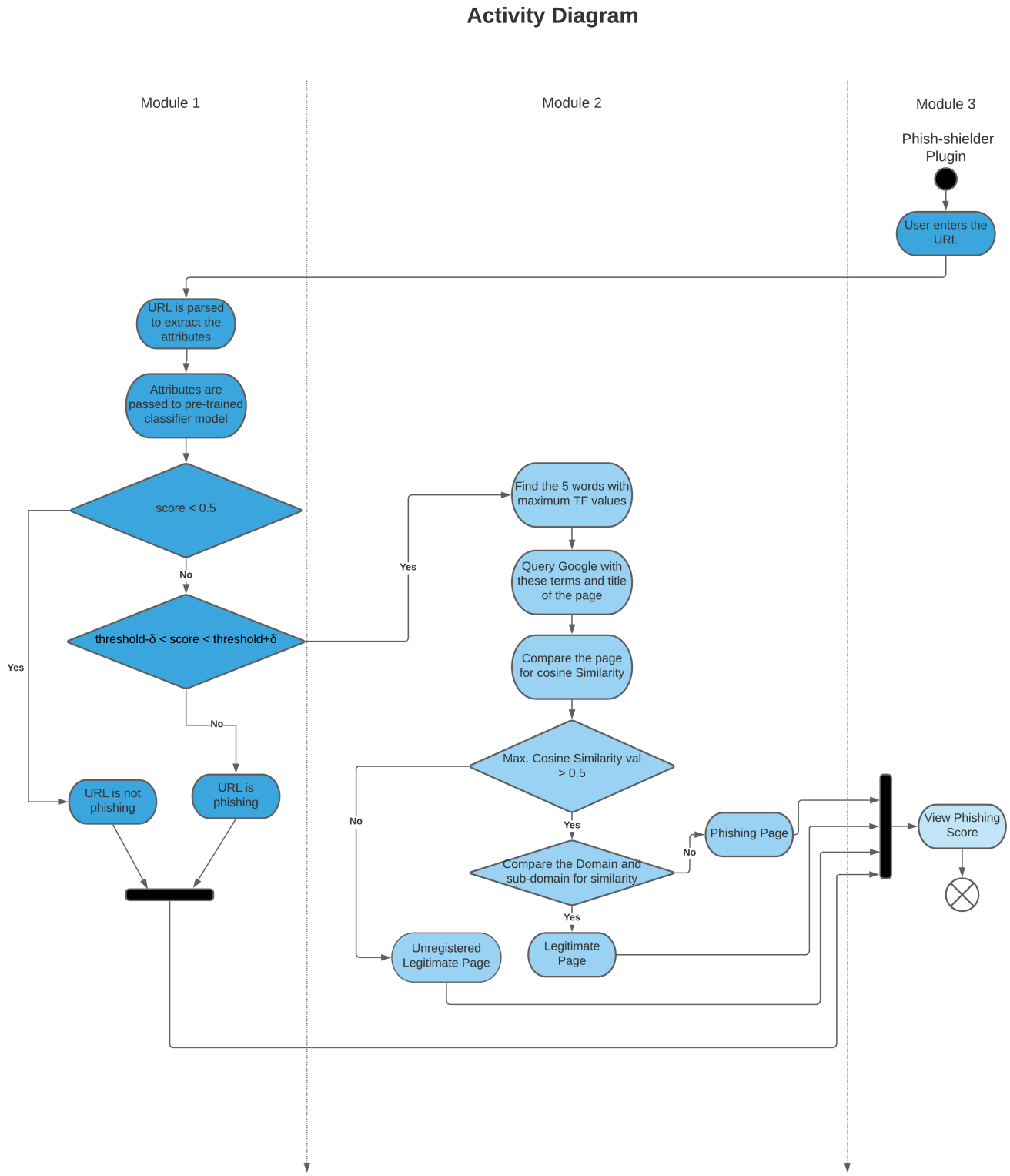
- **Anomaly in URLs** i.e. URL containing lots of sub-domains, very long URL's, URL's using URL shortener etc. Phishing URLs often hide the real URL-destination. Sub-domains and usernames are inserted in the URL to simulate a legitimate destination and to confuse the user. Phishing Check removes these irrelevant parts of the phishing URL.

- **Providing REST APIs** for easy integration into the browser plugin or with a proxy server.

4.2.2 Use Cases



4.2.3 Activity Diagram

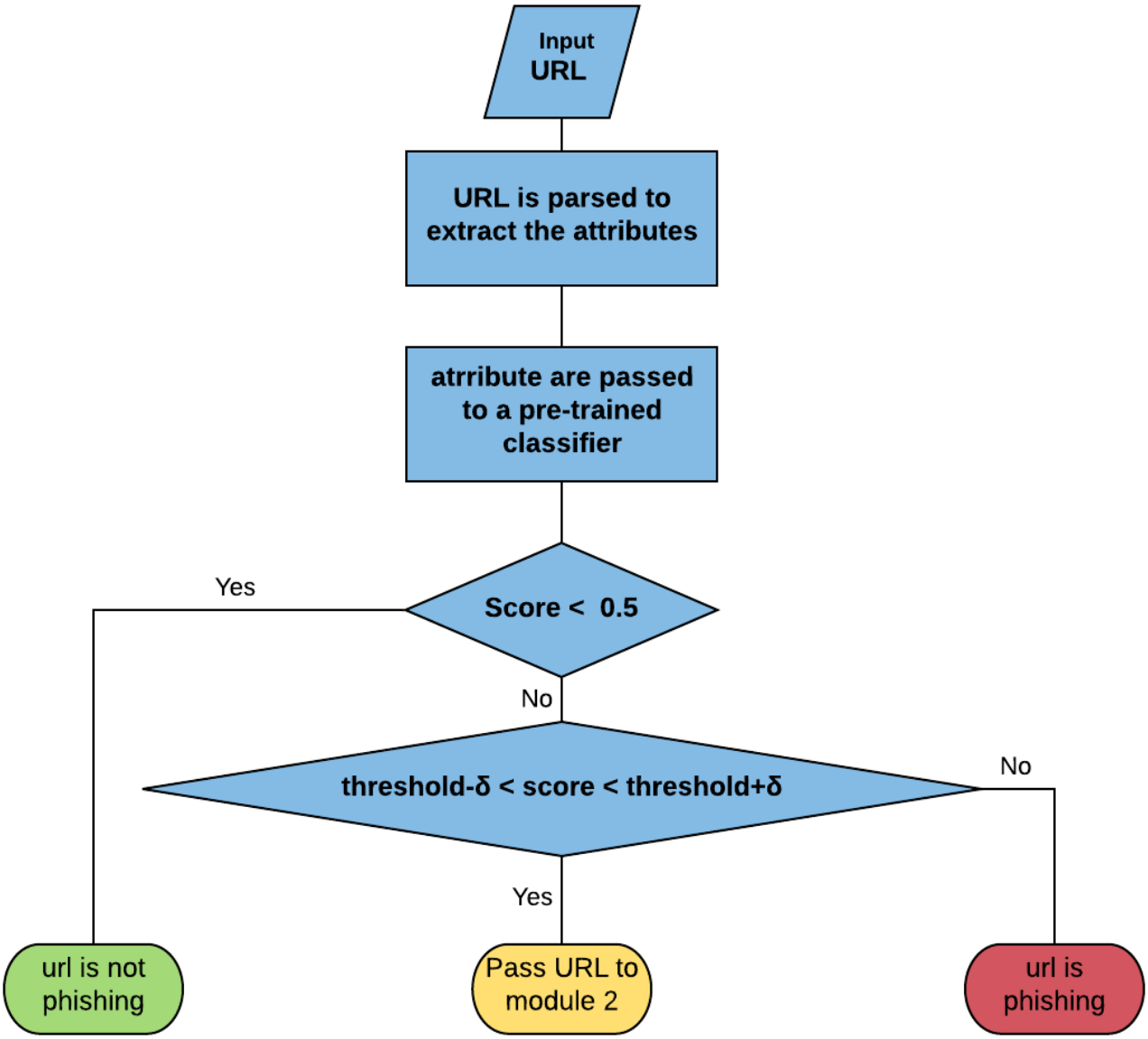


4.3 Non Functional Requirements

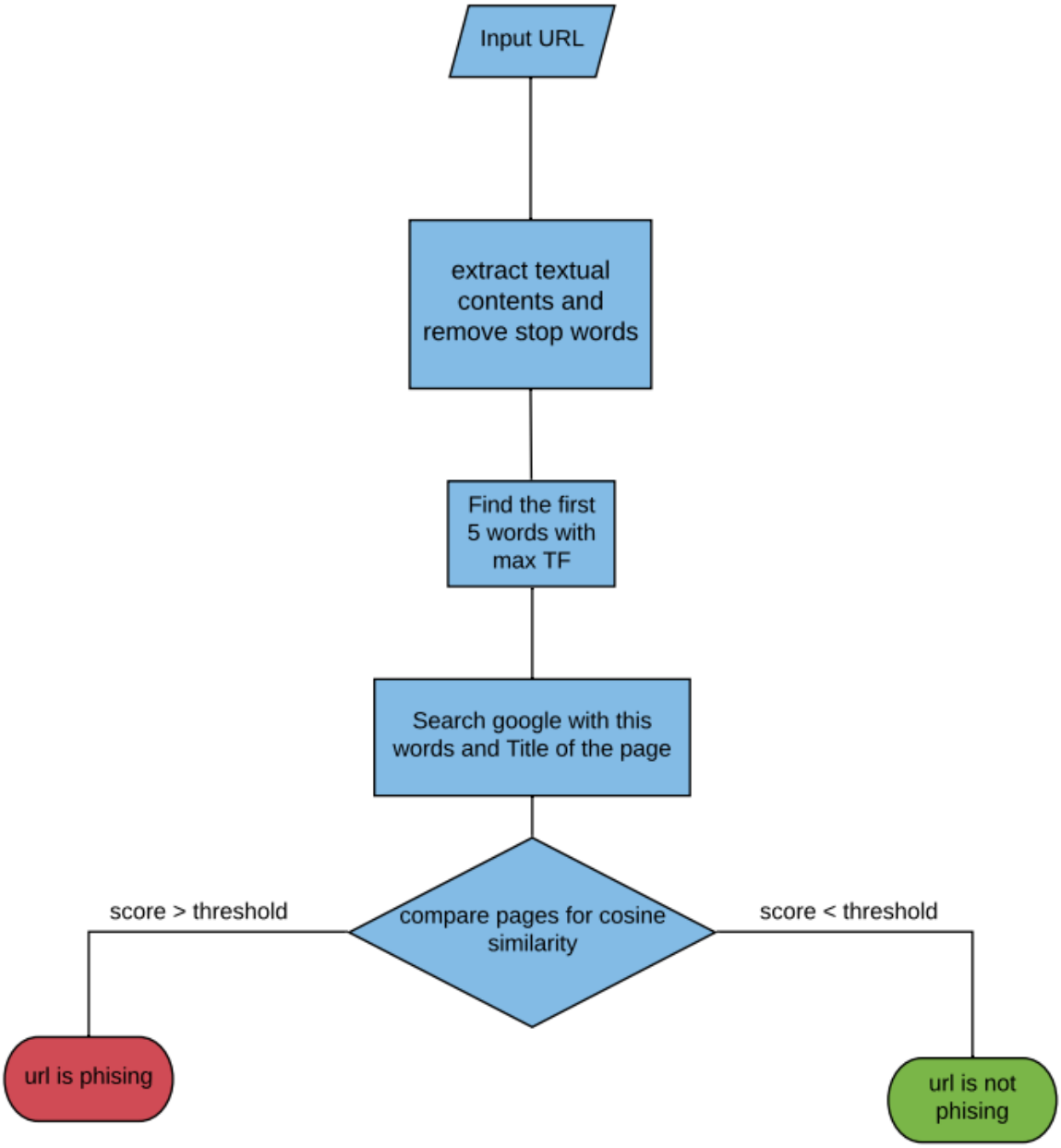
- User should be able to submit **feedback** regarding particular domain if it is prone to phishing & particular URL will be marked as phishing if it is reported by several users.
- **Higher accuracy** in detecting whether a website is phishing or not.
- **Lower detection time:** Project should be able to detect whether a URL is phishing or not in less amount of time.

4.4 Software Design

4.4.1 Flow chart Diagram(Module 1)



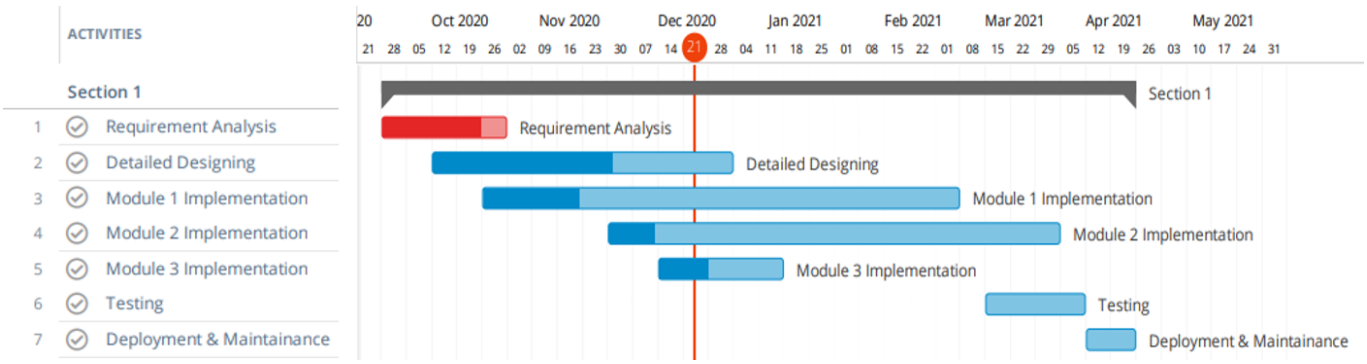
4.4.2 Flow chart Diagram(Module 2)



4.5 Planning & Scheduling

4.5.1 Gantt Chart

	ACTIVITIES	ASSIGNEE	EH	START	DUE	%
	Section 1		-	01/Oct	22/Apr	29%
1	Requirement Analysis	Everyone	-	01/Oct	30/Oct	80%
2	Detailed Designing	Everyone	-	15/Oct	30/Dec	60%
3	Module 1 Implementation	AP, RP	-	01/Nov	01/Mar	20%
4	Module 2 Implementation	HP, MR, SG	-	01/Dec	31/Mar	10%
5	Module 3 Implementation	Mansoor Rangwala	-	15/Dec	15/Jan	40%
6	Testing	Everyone	-	15/Mar	08/Apr	0%
7	Deployment & Maintainance	Everyone	-	15/Apr	22/Apr	0%



Chapter 5

Implementation

5.1 Module 1

Dataset Used: https://archive.ics.uci.edu/ml/datasets/phishing+websites

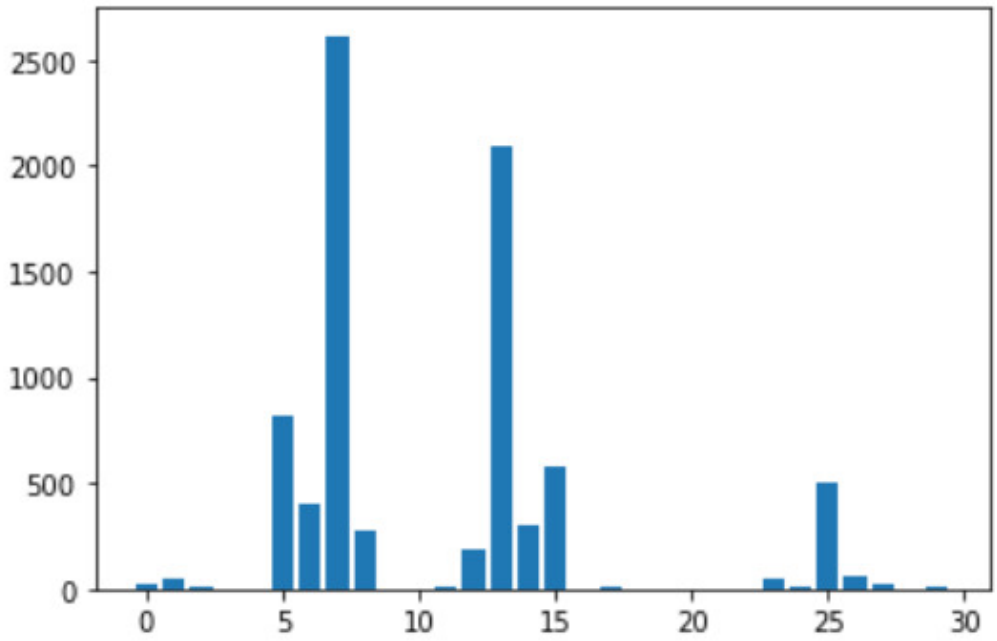
Feature selection:
Feature selection is also known as attribute selection is a process of extracting the most relevant features from the dataset and then applying machine learning algorithms for the better performance of the model. A large number of irrelevant features increases the training time exponentially and increase the risk of overfitting.

Chi-square Test for Feature Extraction: Chi-square test is used for categorical features in a dataset. We calculate Chi-square between each feature and the target and select the desired number of features with best Chi-square scores. It determines if the association between two categorical variables of the sample would reflect their real association in the population. Chi- square score is given by :

$$\chi^2 = \Sigma(Oi–Ei)^2/Ei$$

where Oi is the observed value and Ei is the expected value.

```
Feature having_IPhaving_IP_Address:                         20.937124
Feature URLURL_Length:                          46.652553
Feature Shortining_Service:                          6.589220
Feature having_At_Symbol:                         2.672576
Feature double_slash_redirecting:                              2.890637
Feature Prefix_Suffix:                          813.926512
Feature having_Sub_Domain:                          405.781999
Feature SSLfinal_State:                          2614.819638
Feature Domain_registeration_length:                              278.004305
Feature Favicon:                             0.000303
Feature port:                           1.329776
Feature HTTPS_token:                             3.805655
Feature Request_URL:                             191.550858
Feature URL_of_Anchor:                             2099.780464
Feature Links_in_tags:                             299.759786
Feature SFH:                           579.478214
Feature Submitting_to_email:                               0.464428
Feature Abnormal_URL:                            6.913206
Feature Redirect:                           0.898417
Feature on_mouseover:                            1.597113
Feature RightClick:                           0.024673
Feature popUpWidnow:                           0.003222
Feature Iframe:                         0.013806
Feature age_of_domain:                             49.736819
Feature DNSRecord:                           12.681512
Feature web_traffic:                             508.471263
Feature Page_Rank:                           64.742709
Feature Google_Index:                             17.918736
Feature Links_pointing_to_page:                                1.723448
Feature Statistical_report:                              8.319721
```



Final attributes that we will consider for training our model after applying feature selection:

- contains_ip_address
- long_url
- url_shortner_used
- has_at
- has_double_redirection
- has_multidomain
- new_domain
- has_https
- findRequestURL
- findSFH
- redirects
- about_to_expire
- dns_record
- detectWebTraffic
- pageRank
- google_indexed

URL Parsing: It is used to fetch attributes from the URL.

```
def google_indexed(self,url):
    query = {'q': 'site:'+url}
    google = "https://www.google.com/search?" + urlencode(query)
    user_agent = 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2272.101 Safari/537.36'
    headers = { 'User-Agent' : user_agent }
    try:
        data = requests.get(google,headers=headers)
        data.encoding = 'ISO-8859-1'
        soup = BeautifulSoup(data.content, "html.parser")
        firstLink = soup.find('div',class_='BNeawe UPmit AP7Wnd').text.split(" ")[0]
        if(firstLink in url):
            return 1
        else:
            return -1
    except:
        return -1
```

```
def detectWebTraffic(self,url):
      print("=> Fetching Website rank from Alexa")
    #Website Rank<100,000 => legitimate => returns 1
    #Otherwise            => Phishing   => return -1

    alexa_api_endpoint="https://data.alexa.com/data?cli=10&url="+quote(url)
    #print(alexa_api_endpoint)

    try:
        r = requests.get(alexa_api_endpoint)
        source_code = r.content

        #Parsing XML Response
        res_xml=ET.fromstring(source_code)
        website_rank=int(res_xml.find('SD').find('REACH').attrib['RANK'])

          print("Website Rank of {} : {}".format(url,website_rank))
    except:
        website_rank=999999999
          print("Something went wrong in detectWebTraffic Module")
    if website_rank < 1000000 :
          print("Status: Non-phishing URL")
        return(1)
    else :
          print("Status: phishing URL")
        return(-1)
```

```
def findSFH(self,url):
      print("=> Fetching SFHs")
    #r = requests.get(url)
    #source_code = str(r.content)
    source_code = self.fetchSourceCode(url)
      source_code = self.source_code
    if not source_code:
        return 0

    count_sfh=0                 #stores count of sfh which point to empty,or external domains
    count_total_sfh=0
    #Using BeautifulSoup:
    tree = BeautifulSoup(source_code, 'html.parser')
    #form_handler = [s.get('action') for s in tree.find_all('form') if s.get('action')]
    for s in tree.find_all('form'):
        count_total_sfh+=1
        #print(s.get('action'))
        if s.get('action'):
            if urlparse(s.get('action')).netloc == urlparse(url).netloc:
                continue
            elif re.findall('^/',s.get('action')):
                continue
        count_sfh+=1

      print("Total SFH: {}".format(count_total_sfh))
    if count_sfh !=0:
          print("Contains suspicious SFH: {}".format(count_sfh))
          print("Status: Phishing")
        return -1
    else:
          print("Status: Legitimate")
        return 1
```

```
def new_domain(self,url):
    try:
        whois_info = self.query_whois(url)
        if(whois_info.domain_name==None):
            return -1
        if(type(whois_info.expiration_date)==list):
            crt = whois_info.creation_date[0]
        else:
            crt = whois_info.creation_date
        ans = int(str(datetime.datetime.now()-crt).split(" ")[0])
        if(ans < 180):
            return -1
        else:
            return 1
    except:
        return -1
```

FINDING DELTA/RANGE FOR PASSING TO MODULE 2

We will plot graph of false negative (i.e module detected legitimate but actually phishing) and find the range in which maximum false negative results are their. The results which are falling in this range will be passed to module 2 for further analysis.

10 Fold Cross-validation:
Cross-validation is a technique to evaluate predictive models by partitioning the original sample into a training set to train the model, and a test set to evaluate it.
In k-fold cross-validation, the original sample is randomly partitioned into k equal size subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k-1 subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds can then be averaged (or otherwise combined) to produce a single estimation. The advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once.
For classification problems, one typically uses stratified k-fold cross-validation, in which the folds are selected so that each fold contains roughly the same proportions of class labels.
In repeated cross-validation, the cross-validation procedure is repeated n times, yielding n random partitions of the original sample. The n results are again averaged (or otherwise combined) to produce a single estimation.

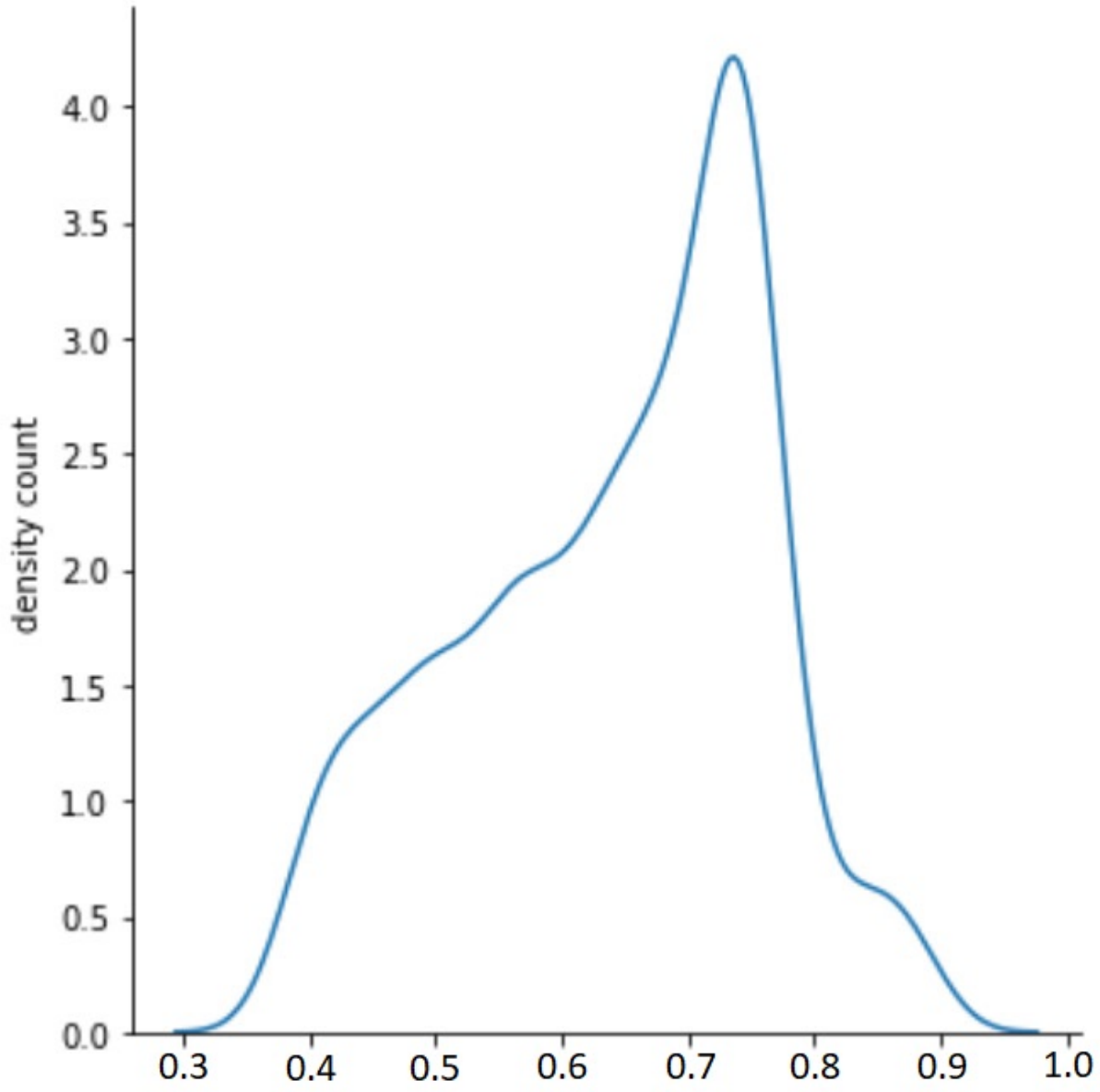
```
from sklearn.model_selection import KFold
```

```
cv = KFold(n_splits=10)
```

```
from sklearn.svm import SVC
```

```
false_negatives = []
svc=SVC(probability=True)
for train_index,test_index in cv.split(X):
    X_train, X_test, y_train, y_test = X.iloc[train_index], X.iloc[test_index], Y.iloc[train_index], Y.iloc[test_index]
    model = svc.fit(X_train,np.array(y_train['Result']))
    prediction = model.predict(X_test)
    prediction_prob = model.predict_proba(X_test)[:,1]
    y_test_array = np.array(y_test['Result'])
    for i in range(len(prediction)):
        if(prediction[i]==1 and y_test_array[i]==-1):
            false_negatives.append(prediction_prob[i])
```

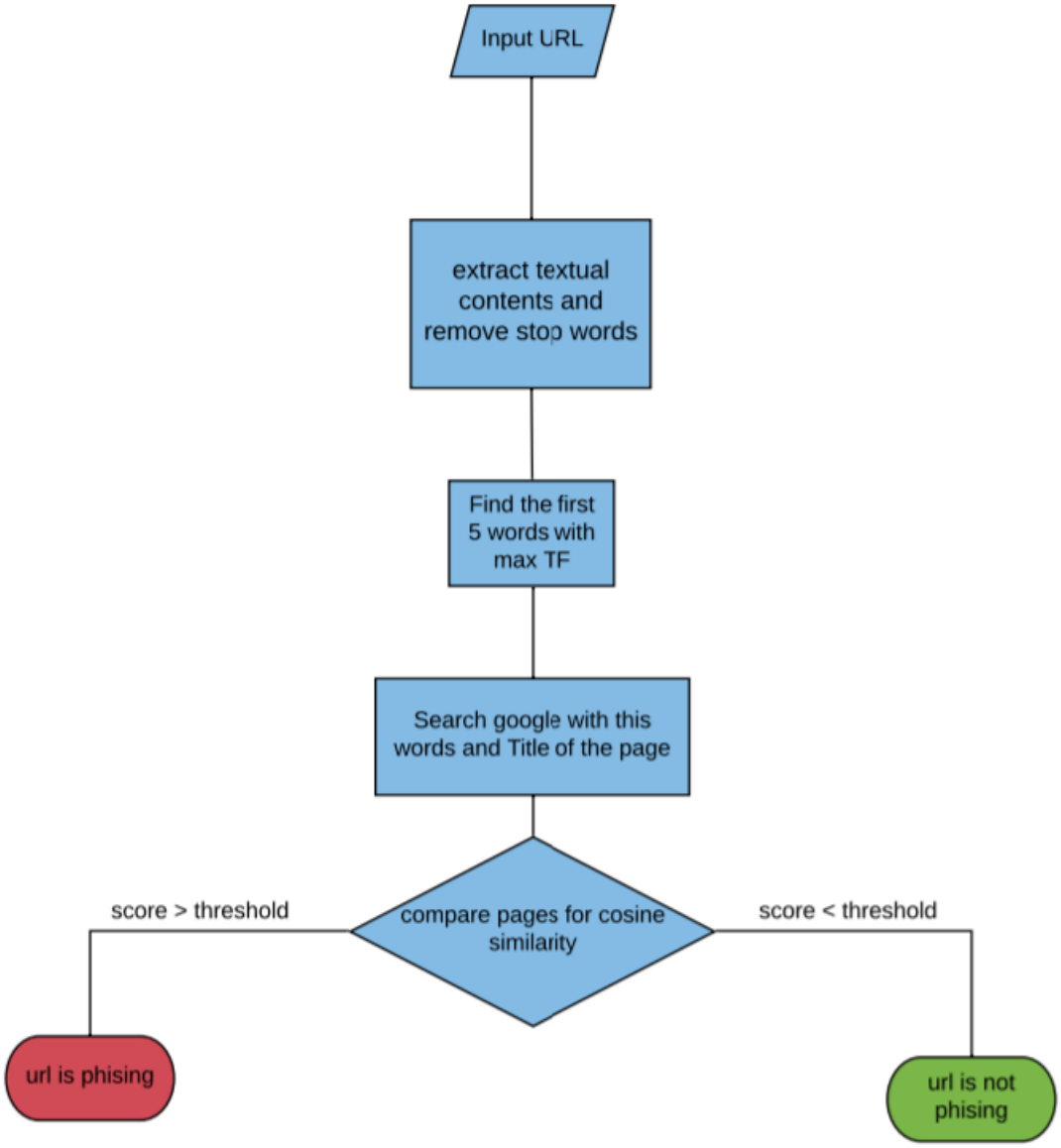
False Negative Graph for Support Vector Machine:



Total Number of False Negative Output: 1086 (After 10 fold test)
Range Having Maximum Number of False Negative Density: [0.6507983 , 0.81118389].
Hence values of these range are passed to Module 2.

5.2 Module 2

Process flow of module 2:



Detailed Explanation of each function used in this module:

- Suspicious URL of module 1 is passed to the **main function** of Module 2.

```
if __name__ == '__main__':
    url = str(input())
    # url = sys.argv[1]
    driver_func(url)
```

- **Driver function** is called in main function of module 2, and it will perform all the operations and check all the required conditions.

```
def driver_func(url):

    temporary = fetch_wordlist(url)
    clean_list = clean_wordlist(temporary)
    top = create_dictionary(clean_list)
    queryResult(top,url)
    print(urlList)
    contentList = []
    #original website
    contentList.append(' '.join([str(elem) for elem in clean_list]))
    #top 5 results
    for urls in urlList:
        temp_word_list = fetch_wordlist(urls)
        temp_clean_list = clean_wordlist(temp_word_list)
        contentList.append(' '.join([str(elem) for elem in temp_clean_list]))

    #print(contentList)
    #print("Cosine similarity index : ")
    index = check_similarity(contentList)
    print(index)
    phish_status = ""
    # print(index,end='')
    if index>0.5:
        phish_status = "True"
    elif index == -1:
        phish_status = ""
        print("Empty list passed !!!")
    else :
        #print("Non phishing...")
        phish_status = "False"
    print("Phish_status: {}".format(phish_status))
    print("Cosine Similarity Score: {}".format(index))
    return phish_status,index
```

- **Fetch Word List** extracts the complete text of the URL given as an input. And then the extracted text passed to next function for further computation.

```
def fetch_wordlist(url):
    wordlist = []
    source_code = requests.get(url,verify =False).text

    if not source_code:
        exit
    soup = BeautifulSoup(source_code, 'html.parser')
    #words = set(nltk.corpus.words.words())
    #print(words)
    lis = ['body']
    for temp in lis:
        for each_text in soup.findAll(temp):
            content = each_text.text
            words = content.lower().split()
            for each_word in words:
                wordlist.append(each_word)

    return wordlist
```

- **Clean Word List** is used to remove all the tags, and stop words from the above extracted text.

```
def clean_wordlist(wordlist):

    clean_list =[]
    for word in wordlist:
        symbols = '!@#$%^&*()_-+\\-=\'{[}]|\;:"<>?/.,→↵○I '

        for i in range (0, len(symbols)):
            word = word.replace(symbols[i], '')

        if len(word) > 0:
            clean_list.append(word)
    str = listToString(clean_list)
    filtered_sentence = remove_stopwords(str)
    clean_list = filtered_sentence.split()
    return clean_list
```

- **Create Dictionary** is used to compute word count of each word, and with the help of this we get our most frequent word. Now these most frequent words are passed further to query the similar URL.

```
def create_dictionary(clean_list):
    word_count = {}

    for word in clean_list:
        if word in word_count:
            word_count[word] += 1
        else:
            word_count[word] = 1
    #print(word_count)
    c = Counter(word_count)

    top = c.most_common(5)
    return top
```

- **Query Result** function is used to search the similar URL from the signature formed with help of most frequent words and the domain name.

```
def queryResult(top,url):
    li = []
    for i in top:
        li.append(i[0])
    #adding root domain name to query list
    res=re.findall(r'(?<=\.)([^.]+)(?:\.(?:co\.uk|ac\.us|[^·.]+(?:$|\n)))',url)
    li.append(res[0])
    ##li.append(urlparse("https://spotify.update-billing.sctrlgin.com").netloc.split('.')[0])
    query = listToString(li)
    print("Keywords to be searched on Google : ")
    print()
    print(li)
    print("------------------------")
    print("Top google query results : ")
    print()
    global urlList
    for j in search(query, tld="co.in", num=5, stop=5, pause=2):
       urlList += [j]
```

- **Check Similarity** is used to check the similarity between the text extracted from Provided URL and the top 5 Queried URL. And on the basis of that it provide output of similarity in range of 0 to 1 (both inclusive).

The cosine similarity is the cosine of the angle between two vectors. In text analysis, each vector can represent a document. The greater the value of θ, the less the value of cos θ, thus the less the similarity between two documents. Mathematical Representation of Cosine Similarity:

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \cdot \|\vec{b}\|}$$

```
def check_similarity(documents):
    for str in documents:
        if str == '':
            return -1

    stemmer = nltk.stem.porter.PorterStemmer()
    def StemTokens(tokens):
        return [stemmer.stem(token) for token in tokens]
    remove_punct_dict = dict((ord(punct), None) for punct in string.punctuation)

    def StemNormalize(text):
        return StemTokens(nltk.word_tokenize(text.lower().translate(remove_punct_dict)))

    lemmer = nltk.stem.WordNetLemmatizer()

    def LemTokens(tokens):
        return [lemmer.lemmatize(token) for token in tokens]
    remove_punct_dict = dict((ord(punct), None) for punct in string.punctuation)

    def LemNormalize(text):
        return LemTokens(nltk.word_tokenize(text.lower().translate(remove_punct_dict)))

    LemVectorizer = CountVectorizer(tokenizer=LemNormalize, stop_words='english')
    LemVectorizer.fit_transform(documents)

    tf_matrix = LemVectorizer.transform(documents).toarray()

    tfidfTran = TfidfTransformer(norm="l2")
    tfidfTran.fit(tf_matrix)

    def idf(n,df):
        result = math.log((n+1.0)/(df+1.0)) + 1
        return result

    tfidf_matrix = tfidfTran.transform(tf_matrix)

    cos_similarity_matrix = (tfidf_matrix * tfidf_matrix.T).toarray()
    semi = cos_similarity_matrix[0]
    semi.sort()
    return(semi[-2])
```

5.3 Integration Module

- **runHTTPSServer** is used to run HTTPS server on port 4443 with custom generated certificate.

```
def runHTTPSServer():
    server_address = (listen_host, listen_port)
    #httpd = http.server.HTTPServer(server_address, http.server.SimpleHTTPRequestHandler)
    httpd = http.server.HTTPServer(server_address, MyCustomHandler)
    httpd.socket = ssl.wrap_socket(httpd.socket,
                                   server_side=True,
                                   certfile='mycertificate_combined.pem',
                                   ssl_version=ssl.PROTOCOL_TLS)
    httpd.serve_forever()
    # Generated certificate via commands:
    #openssl req -newkey rsa:2048 -new -nodes -x509 -days 3650 -keyout key.pem -out cert.pem
    #cat cert.pem key.pem > mycertificate_combined.pem

    # Parsing Attribute/Endpoint:
```

- **MyCustomHandler** is used to provide endpoint */phish_api*. It takes *phish_url* as GET parameter with value of encoded URL which is needed to check whether phishing or not and return JSON response containing *phish_status* as true/false and *phish_score*.

```
class MyCustomHandler(http.server.BaseHTTPRequestHandler):
    def do_GET(self):
        if self.path.__contains__("/phish_api?"):
            #print(self.path)
            # Parsing the parameter from the URL:
            # Courtesy: https://stackoverflow.com/questions/8928730/processing-http-get-input-parameter-on-server-side-in-python/8930440
            query = urlparse(self.path).query
            query_components = dict(qc.split("=") for qc in query.split("&"))
            if query_components.get("phish_url"):
                # Sending response header:
                self.send_response(200)
                self.send_header('Content-Type', 'application/json')
                self.send_header('Access-Control-Allow-Origin','*')
                self.end_headers()

                url = unquote(query_components.get("phish_url"))        #url-decode
                print(url)

                # Calling function for firing Different Phishing detection modules:
                result = runPhishDetectionModule(url)
                result_json = json.dumps(result)

                self.wfile.write(bytes(result_json,'UTF-8'))


            else:
                self.send_response(200)
                self.send_header("Content-type", "text/html")
                self.end_headers()
                #self.wfile.write(self._html("My BODY"))
                self.wfile.write(b"Supply phish_url as parameter to query")

        else:
                self.send_response(404)
                self.send_header("Content-type", "text/html")
                self.end_headers()
```

- **lookup_cache** is used to lookup from the cache file and directly return the phishing status & phishing score if present in cache.

```
output_file = "output_phishing_detection.csv"

# cache lookup
def lookup_cache(url):
    result={}

    # if file not already exist
    if not os.path.isfile(output_file):
        result={"phish_status": "#","phish_score": "#"}
        return result

    file1 = open(output_file, "r")
    for line in file1 :
        line=line.replace('\n','')
        if url in line:
            lst=line.split(",")
            result['phish_score']=float(lst[1])
            #result['phish_status']=bool(lst[2])
            if lst[2] == "False":
                result['phish_status']=False
            else:
                result['phish_status']=True

            print("======> Value found from cache")
            print("phish_status = {} phish_score = {}".format(result['phish_status'],result['phish_score']))
            return result

    # if entry not found already in cache table
    result={"phish_status": "#","phish_score": "#"}
    return result
```

- **store_result** is used to used to store result generated from the module into cache file for faster lookup.

```
# stores output in output_file which will be used as cache.
def store_result(url,result):
    if url == "" :
        return
    if result["phish_status"] == "" :
        return
    if result["phish_score"] == "" :
        return
    file1=open(output_file,"a")
    file1.write(url+","+str(result["phish_score"])+","+str(result["phish_status"])+"\n")
```

- **runPhishDetectionModule** is used to fire Module-1 and Module-2 (if required) and returns a dictionary containing keys *phish_status* and *phish_score*.

```
def runPhishDetectionModule(url):
    # result dictionary will store my results which will me send via json response.
    result={"phish_status": "#","phish_score": "#"}

    # Importing different modules:
    url_parse = SourceFileLoader("Url_parse", "../Module-1/parser_final.py").load_module()
    ml_predict = SourceFileLoader("prediction", "../Module-1/predict.py").load_module()
    #ml_predict = SourceFileLoader("prediction", "../Module-1/ML_predict.py").load_module()
    module2 = SourceFileLoader("driver_func", "../Module-2/WebResults.py").load_module()
    #cosine = SourceFileLoader("check_similarity", "../Module-2/Cosine.py").load_module()

    # Lookup in cache table
    result=lookup_cache(url)
    if result["phish_status"] != "#" and result["phish_score"] != "#":
        return result

    # Firing parse attribute script:
    print("======> Parsing Attributes")
    parser  = url_parse.ParseURl()
    url_attr_list=parser.parseUrl(url)

    print(url_attr_list)
    #url_attr_list is going to be feeded to Module-1 Phishing detection system. It will return phishing_status & phishing_score.


    # Firing Module-1 Machine Learning script:
    print("======> Firing MODULE-1")
    phish_score,phish_status = ml_predict.predict(url_attr_list)
    print("MODULE 1 VALUES:",phish_status,phish_score)

    # Chaning output to boolean form
    if phish_status == "NON-PHISHING":
        phish_status=False
    else:
        phish_status=True
```

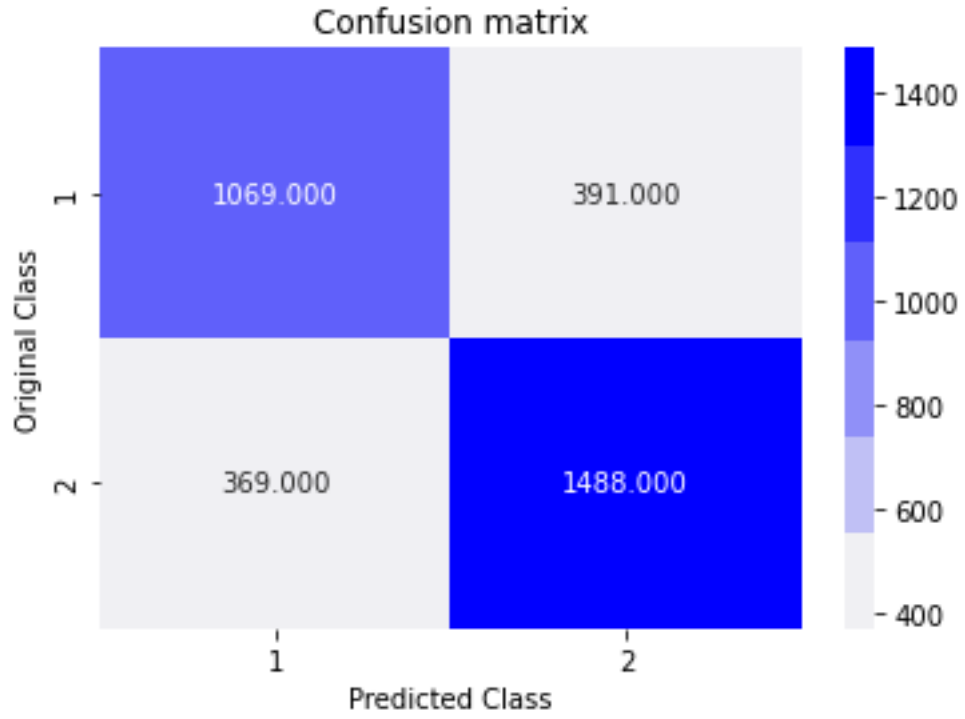
```
    # Firing Module-2 Structure Analysis script:
    if phish_score > 0.65 and phish_score < 0.81:
        print("======> Firing MODULE-2")
        phish_status,phish_score=module2.driver_func(url)           # phish_status return will be in boolean already
        print("MODULE 2 VALUES:",phish_status,phish_score)

        # Chaning output to boolean form
        if phish_status == "False":
            phish_status=False
        else:
            phish_status=True

    ##result["phish_status"]=False
    ##result["phish_score"]="90"

    phish_score=phish_score*100

    result["phish_status"]=phish_status
    result["phish_score"]=phish_score

    store_result(url,result)
    return result
```

Chapter 6

Testing and Result

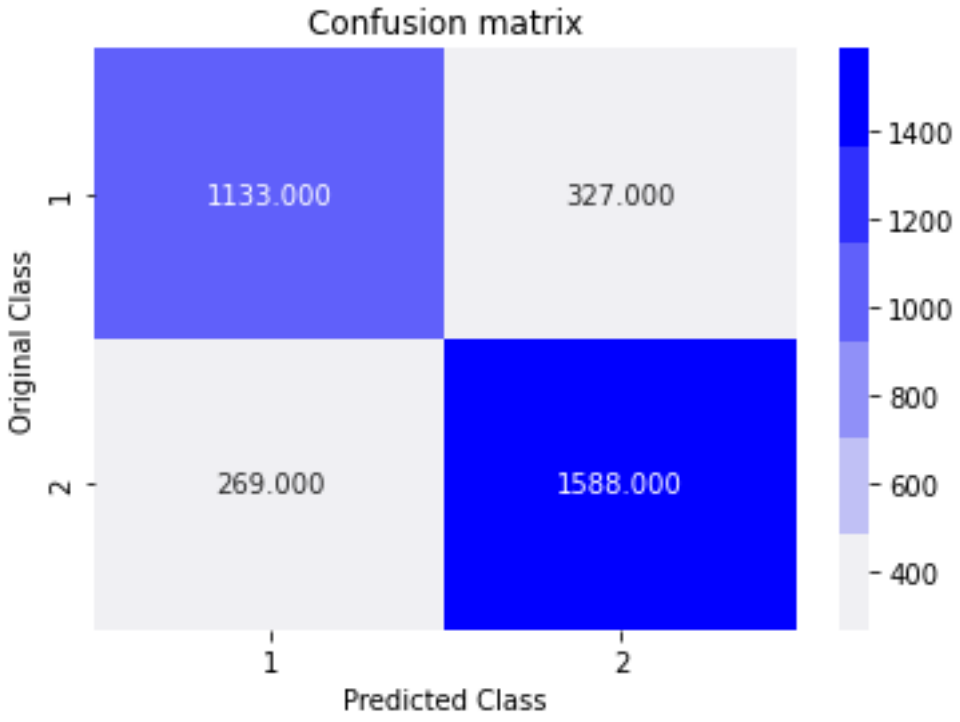
6.1 Module 1 Testing:

Confusion Matrix after Applying Logistic Regression:



Efficiency of Logistic Regression : 0.7708772987639433

Confusion Matrix after Applying Support Vector Machine:



Efficiency of SVM : 0.8203195658727767

6.2 Final Testing:

(After Integration of Both the Modules)

Module 1 : testing

```
loaded_model = joblib.load('model_svm.sav')
prediction = loaded_model.predict(test_X)
print('accuracy for module 1 : ',accuracy_score(prediction,test_Y))
```

```
accuracy for module 1 :  0.8203195658727767
```

Module1 + Module2

```
import requests
prediction = []
for url in data_url:
    base_url = 'https://phish.local:4443/phish_api?phish_url='

    #hitting our api (Module1 + Module2)
    response = requests.get(base_url+url,verify=False)
    prediction.append(response.json()['phish_score']/100)
print('accuracy for module1+module2 : ',accuracy_score(prediction,test_Y))
```

```
accuracy for module1+module2 :  0.9175257731958762
```

Chapter 7

Conclusion

Thus to summarize, we have seen how phishing is a huge threat to the security and safety of the web and how phishing detection is an important problem domain. We have reviewed some of the traditional approaches to phishing detection namely the Machine learning approach and CANTINA approach, We then selected the combination of both the approaches based on their performance and built a Chrome extension for detecting phishing web pages. The extension allows easy deployment of our phishing detection model to end-users. The reported **accuracy score** on the data-set on module 1 is **82.032%**. And overall Efficiency (**accuracy score**) of our project (i.e. after integrating module 1 and module 2) is **91.75%**.
For future enhancements, we intend to build the phishing detection system as a scale-able web service that 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.

Chapter 8

References

1. Phishing Websites Dataset1 [Online]:
   https://archive.ics.uci.edu/ml/datasets/phishing+websites
2. Phishing Websites Dataset2 [Online]:
   https://www.phishtank.com/
3. Existing Phishing detection techniques & domain survey [Research Paper]:
   https://ieeexplore.ieee.org/document/8862697
4. URL based Phishing detection using Machine Learning [Research Paper]:
   https://ieeexplore.ieee.org/abstract/document/8858325
5. Detecting phishing websites using machine learning [Online]:
   https://medium.com/intel-software-innovators/detecting-phishing-websites-using-machine-learning-de723bf2f946
6. Content-Based Approach to Detecting Phishing Web Sites (CANTINA) [Research Paper]:
   https://www.cs.cmu.edu/~jasonh/publications/www2007-cantina-final.pdf
7. A Novel Phishing Page Detection Mechanism Using HTML Source CodeComparison and Cosine Similarity [Research Paper]:
   https://ieeexplore.ieee.org/abstract/document/6906016

Appendix A

Screenshots

Module 1:

```
parser = ParseURl()
parser.parseUrl('https://www.google.com')
```

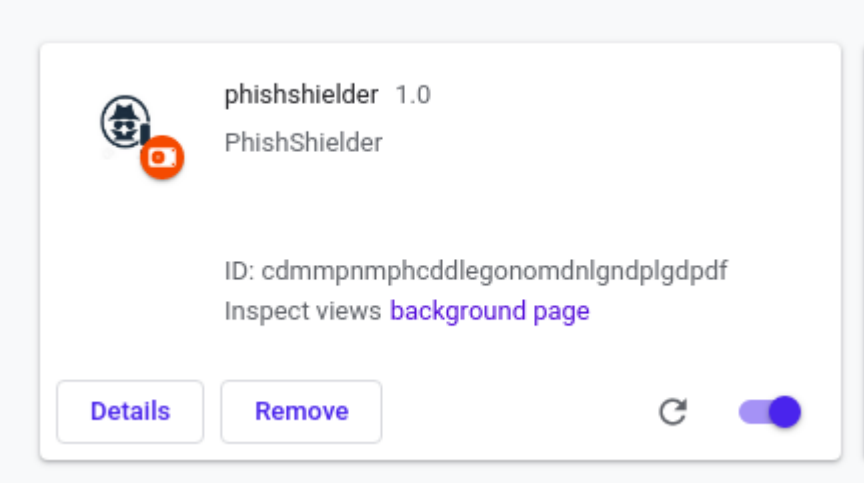
```
contains_ip_address : 1
long_url : 1
url_shortner_used : 1
has_at : 1
has_double_redirection : 1
has_multidomain : 1
new_domain : 1
has_https : 1
findRequestURL : 1
findSFH : 1
redirects : 1
about_to_expire : 1
dns_record : 1
detectWebTraffic : 1
pageRank : 1
google_indexed : 1
```

Module 2:

```
116  def driver_func(url):
117
118      temporary = fetch_wordlist(url)
119      clean_list = clean_wordlist(temporary)
120      top = create_dictionary(clean_list)
121      queryResult(top,url)
122      contentList = []
123      #original website
124      contentList.append(' '.join([str(elem) for elem in clean_list]))
125      #top 5 results
126      fiveUrlList = urlList[:5]
127      print(fiveUrlList)
128      for urls in fiveUrlList:
129          temp_word_list = fetch_wordlist(urls)
130          temp_clean_list = clean_wordlist(temp_word_list)
131          contentList.append(' '.join([str(elem) for elem in temp_clean_list]))
132
133      #print(contentList)
```

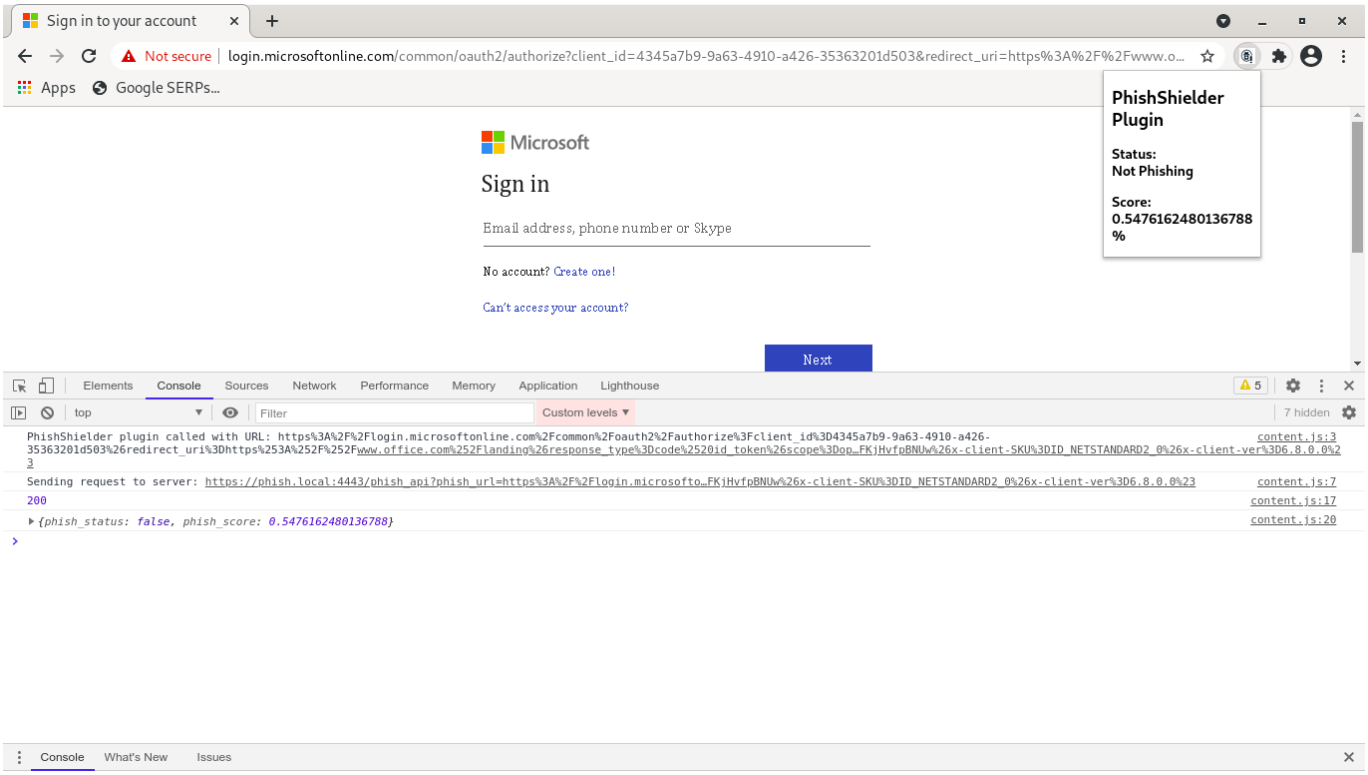
```
OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS 2

n-2021.4.765268190\pythonFiles\lib\python\debugpy\launcher' '52457' '--' 'c:\Users\pasta\OneDrive\Desktop\print.py'
https://spotify.update-billing.sctrlgin.com
Keywords to be searched on Google :

['continue', 'log', 'spotify', 'facebook', 'username']

-----------------------

Top google query results :

['https://support.spotify.com/us/article/facebook-login-help/',
'https://community.spotify.com/t5/Accounts/How-do-i-find-my-username-when-using-Facebook-login/td-p/859795',
'https://accounts.spotify.com/en/login/',
'https://community.spotify.com/t5/Accounts/Removing-the-Facebook-login-and-changing-it-for-a-standard/td-p/169994',
'https://community.spotify.com/t5/Accounts/Disconnect-facebook-and-create-a-username-for-spotify/td-p/4809236']

Phish_status: True
Cosine Similarity Score: 0.63780673480547724
PS C:\Users\pasta\OneDrive\Desktop>
```

Phish-shielder (Chrome Plugin):



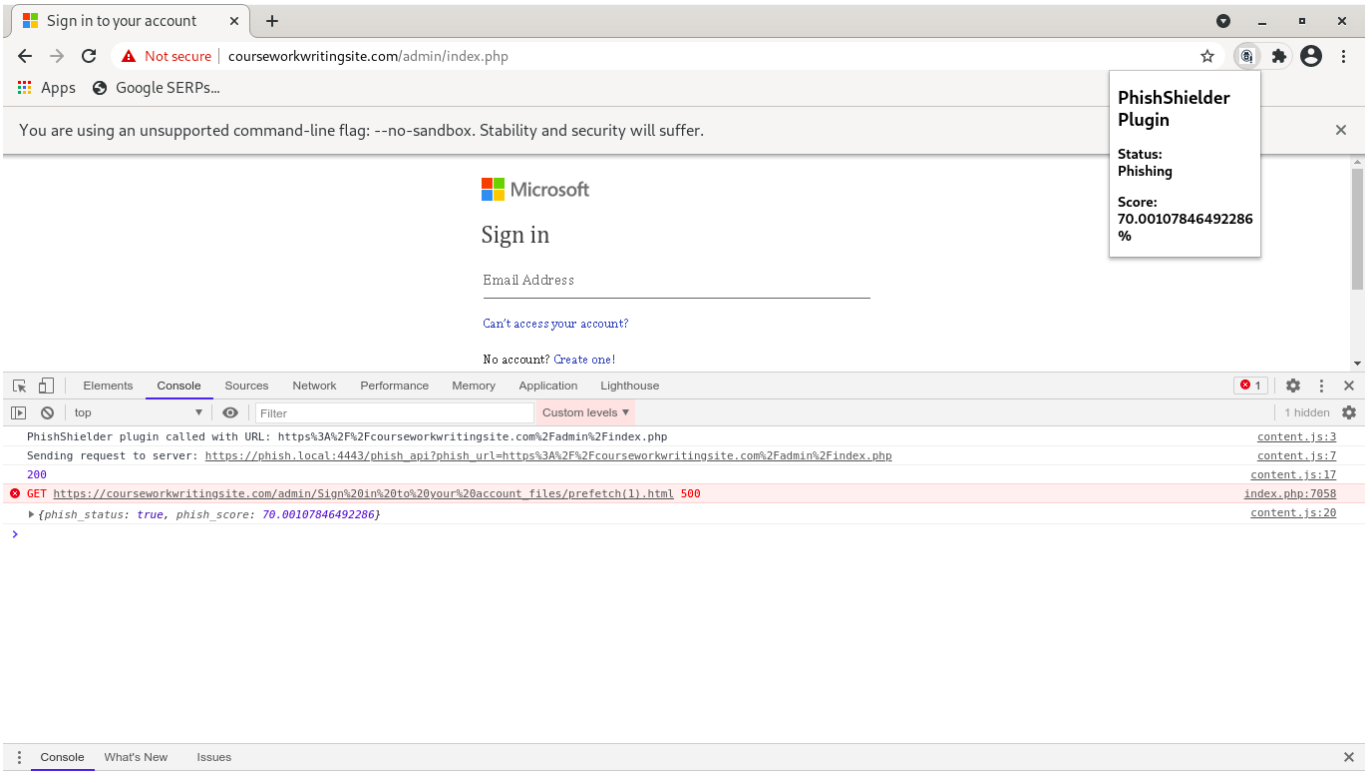
Legitimate Microsoft online login page:

```
127.0.0.1 - - [26/Mar/2021 19:06:51] "GET /phish_api?phish_url=https%3A%2F%2Flogin.microsoftonline.com%2Fcommon%2Foauth2%2Fauthorize%3Fclient_id%3D4345a7b9-9a63-4910-a426-35363201d503%26redirect_uri%3Dhttps%253A%252F%252Fwww.office.com%252Flanding%26response_type%3Dcode%2520id_token%26scope%3Dopenid%2520profile%26response_mode%3Dform_post%26nonce%3D637523626079616982.ZDgzODJlODEtMDU5Zi00ZjgyLTljM2QtODQyZmE2YWRhNzhkNThlM2Y3YjQtMDBjZC00ZWE3LTlhMzktNjE2YTQyNmJhZjEx%26ui_locales%3Den-GB%26mkt%3Den-GB%26client-request-id%3Dde88ec3a-a218-475e-8d35-a090bad118fd%26state%3DR7TzrCfoH5uTxFZ7nAtnldFSEvxUDghtUIAs7jCVGU77Zy5eH6zgI7AzghD707wlKs4vI43gAMvOpTlbs0mpdf-XUzLpYCmudOfpk-ZEb-MzzHd01IUKxy5RlS0gq1fEizjslCVgf3LdOIhZ3PHozkRWp1w9Dxwu8Fc-COA7GzS0iYlOtLb0jcGfF5ejSn4Ah_oRbUr80guQ7u-YLnQ-iAW-UvCvR2-pf-zVUXeRL6FtEw9OdC1TVLifcPmlzJ23D0msZD-NoM6FKjHvfpBNUw%26x-client-SKU%3DID_NETSTANDARD2_0%26x-client-ver%3D6.8.0.0%23 HTTP/1.1" 200 -
https://login.microsoftonline.com/common/oauth2/authorize?client_id=4345a7b9-9a63-4910-a426-35363201d503&redirect_uri=https%3A%2F%2Fwww.office.com%2Flanding&response_type=code%20id_token&scope=openid%20profile&response_mode=form_post&nonce=637523626079616982.ZDgzODJlODEtMDU5Zi00ZjgyLTljM2QtODQyZmE2YWRhNzhkNThlM2Y3YjQtMDBjZC00ZWE3LTlhMzktNjE2YTQyNmJhZjEx&ui_locales=en-GB&mkt=en-GB&client-request-id=de88ec3a-a218-475e-8d35-a090bad118fd&state=R7TzrCfoH5uTxFZ7nAtnldFSEvxUDghtUIAs7jCVGU77Zy5eH6zgI7AzghD707wlKs4vI43gAMvOpTlbs0mpdf-XUzLpYCmudOfpk-ZEb-MzzHd01IUKxy5RlS0gq1fEizjslCVgf3LdOIhZ3PHozkRWp1w9Dxwu8Fc-COA7GzS0iYlOtLb0jcGfF5ejSn4Ah_oRbUr80guQ7u-YLnQ-iAW-UvCvR2-pf-zVUXeRL6FtEw9OdC1TVLifcPmlzJ23D0msZD-NoM6FKjHvfpBNUw&x-client-SKU=ID_NETSTANDARD2_0&x-client-ver=6.8.0.0#
contains_ip_address : 1
long_url : -1
url_shortner_used : -1
has_at : 1
has_double_redirection : 1
has_multidomain : 1
new_domain : -1
has_https : 1
findRequestURL : -1
findSFH : 1
redirects : 1
about_to_expire : -1
dns_record : -1
detectWebTraffic : 1
pageRank : -1
google_indexed : 1
[1, -1, -1, 1, 1, 1, -1, 1, -1, 1, 1, -1, -1, 1, -1, 1]
NON-PHISHING 0.00547616248013678 8
```



Phishing page of Microsoft online login:

```
127.0.0.1 - - [26/Mar/2021 19:04:40] "GET /phish_api?phish_url=https%3A%2F%2Fcourseworkwritingsite.com%2Fadmin%2Findex.php HTTP/1.1" 200 -
https://courseworkwritingsite.com/admin/index.php
contains_ip_address : 1
long_url : 1
url_shortner_used : 1
has_at : 1
has_double_redirection : 1
has_multidomain : 1
new_domain : -1
has_https : 1
findRequestURL : 1
findSFH : 1
redirects : 1
about_to_expire : -1
dns_record : -1
detectWebTraffic : -1
pageRank : -1
google_indexed : -1
[1, 1, 1, 1, 1, 1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1]
PHISHING 0.7000107846492285
```


Sign in to your account
Not secure | courseworkwritingsite.com/admin/index.php
Apps
Google SERPs...
You are using an unsupported command-line flag: --no-sandbox. Stability and security will suffer.
PhishShielder Plugin
Status: Phishing
Score: 70.00107846492286 %
Microsoft
Sign in
Email Address
Can't access your account?
No account? Create one!
Elements
Console
Sources
Network
Performance
Memory
Application
Lighthouse
top
Filter
Custom levels
1 hidden
PhishShielder plugin called with URL: https%3A%2F%2Fcourseworkwritingsite.com%2Fadmin%2Findex.php
content.js:3
Sending request to server: https://phish.local:4443/phish_api?phish_url=https%3A%2F%2Fcourseworkwritingsite.com%2Fadmin%2Findex.php
content.js:7
200
content.js:17
GET https://courseworkwritingsite.com/admin/Sign%20in%20to%20your%20account_files/prefetch(1).html 500
index.php:7058
{phish_status: true, phish_score: 70.00107846492286}
content.js:20
Console
What's New
Issues

Appendix B

Code

https://github.com/mansoorrangwala/PhishingDetectionSystem