Shri Govindram Seksaria Institute of Technology and Science

CSE Major Project 2021

"PHISHING WEBSITES DETECTION SYSTEM"

Guide:
Mr. Rajesh Dhakad

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

Table of Content

Problem Description:

- 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 and noticed more in Tier 2 and Tier 3 cities.

What is Phishing??

- **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**.

Literature/Domain Survey:

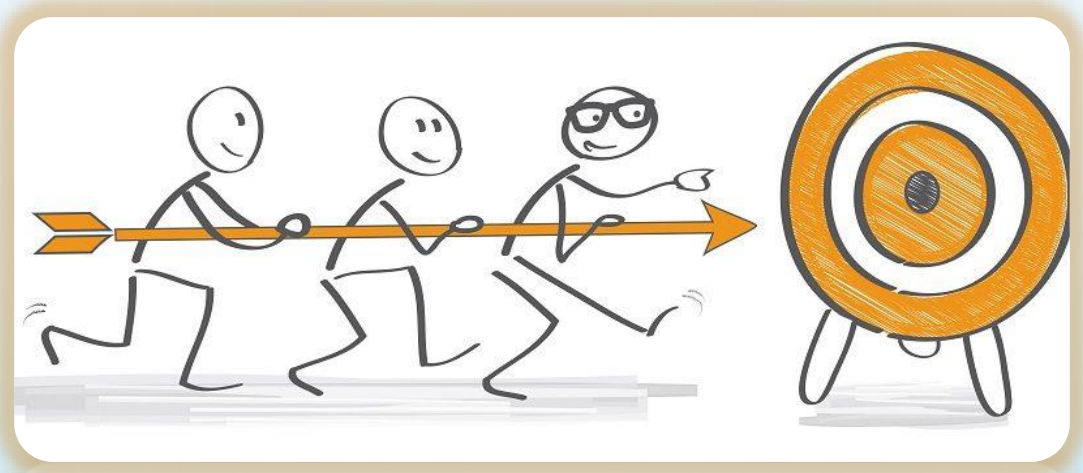
Best software's available in the market for phishing detection:

1. 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.

2. 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.

3. 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.

Proposed Solution:



- 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.

Tools & Technologies Used

Module – 1:

Python Libraries used:

- python-whois
- Ipaddress
- Requests
- Numpy
- Scikit learn
- Pandas

Module – 2:

Python Libraries used:

- Gensim
- Google
- bs4

Module – 3:

chrome plugin development using

- html
- css
- javascript

Requirements:

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 subdomains, very long URL's, URL's using URL shortener etc. Phishing URLs often hide the real URL-destination. Subdomains 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.

Non-Functional Requirements:

- **Minimum Time should be taken** to detect a URL is phishing or not, so in order to achieve that, we have implemented **caching mechanism**.
- **Higher accuracy** in detecting whether a website is phishing or not.
- 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.

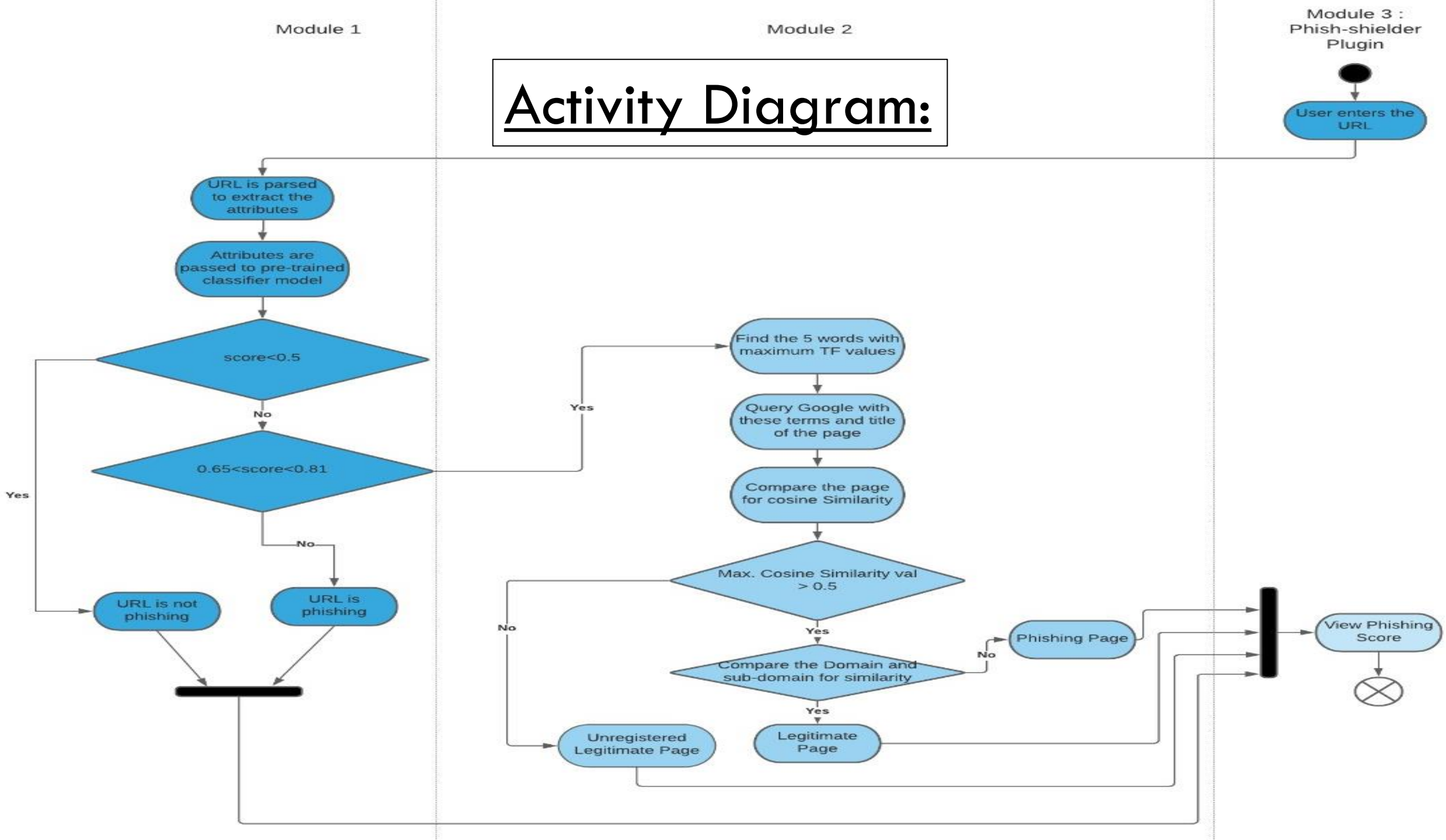
Analysis:

The project contains two major modules:

- **Module 1 - URL Detector:** Detecting URL is phishing or not via machine learning
- **Module 2 - Structure Analyzer:** Verify the URL is phishing or not by analysing HTML Structure.

The URL is passed to Module 1 & generate Phishing Score & if the score is around the threshold score then the URL is passed to Module2 for further analysis.

Activity Diagram:



Demo Time

Module: 1

Data Source:

Phishing data set1 (Primary): https://archive.ics.uci.edu/ml/datasets/phishing+websites
Phishing data set2 : https://www.phishtank.com/
Reputed site's rank data: https://www.alexa.com

Raw Dataset (Dataset 2):

	A	B	C	D	E	F	G	H
1	phish_id	url	phish_detail_url	submission_time	verified	verification_time	online	target
2	6817789	s://pileofpencils.com/pplsecure/CLEAN-PrinceDuscam/PrinceDuscam/app/s	http://www.phishtank.com/phish_detail.php?phish_id=6817789	2020-10-22T06:14:52+00:00	yes	2020-10-22T06:20:59+00:00	yes	PayPal
3	6817775	http://srv90174.ht-test.ru/crypt/crypt/	http://www.phishtank.com/phish_detail.php?phish_id=6817775	2020-10-22T06:09:39+00:00	yes	2020-10-22T06:15:08+00:00	yes	Other
4	6817771	http://srv90174.ht-test.ru/crypt/crypt/index.html?email=USER@DOMAIN.ch	http://www.phishtank.com/phish_detail.php?phish_id=6817771	2020-10-22T05:52:07+00:00	yes	2020-10-22T05:53:52+00:00	yes	Other
5	6817768	https://amazon-c.top/amazon/login/login.php	http://www.phishtank.com/phish_detail.php?phish_id=6817768	2020-10-22T05:46:51+00:00	yes	2020-10-22T05:47:59+00:00	yes	Amazon.com
6	6817767	https://amazon-c.top	http://www.phishtank.com/phish_detail.php?phish_id=6817767	2020-10-22T05:46:36+00:00	yes	2020-10-22T05:47:59+00:00	yes	Amazon.com
7	6817761	https://bittrax.me/log/id/	http://www.phishtank.com/phish_detail.php?phish_id=6817761	2020-10-22T05:24:49+00:00	yes	2020-10-22T05:26:12+00:00	yes	Other
8	6817759	https://amsuwhfa.co.hgdrh.xyz	http://www.phishtank.com/phish_detail.php?phish_id=6817759	2020-10-22T05:12:13+00:00	yes	2020-10-22T06:23:56+00:00	yes	Amazon.com
9	6817752	https://paypalservicesharesupport.com/?sign	http://www.phishtank.com/phish_detail.php?phish_id=6817752	2020-10-22T04:49:52+00:00	yes	2020-10-22T04:53:14+00:00	yes	PayPal
10	6817751	https://www.amazon.111f7k.top/	http://www.phishtank.com/phish_detail.php?phish_id=6817751	2020-10-22T04:48:44+00:00	yes	2020-10-22T04:53:14+00:00	yes	Amazon.com
11	6817750	https://www.amazzoncojp.top/	http://www.phishtank.com/phish_detail.php?phish_id=6817750	2020-10-22T04:45:42+00:00	yes	2020-10-22T04:48:10+00:00	yes	Amazon.com
12	6817743	https://reaccessontoonlinevhost122805.lowhost.ru/fibank/account/	http://www.phishtank.com/phish_detail.php?phish_id=6817743	2020-10-22T04:42:24+00:00	yes	2020-10-22T04:47:26+00:00	yes	Fifth Third Bank
13	6817730	https://interntbakiingdbssg.com/	http://www.phishtank.com/phish_detail.php?phish_id=6817730	2020-10-22T04:13:54+00:00	yes	2020-10-22T04:24:13+00:00	yes	Development Bank of Singapore
14	6817726	http://itau-app.net/inicio.php	http://www.phishtank.com/phish_detail.php?phish_id=6817726	2020-10-22T04:00:27+00:00	yes	2020-10-22T04:03:39+00:00	yes	Other
15	6817719	://secure.runescape.com-as.ru/m=weblogin/loginform441,482,146,71297412	http://www.phishtank.com/phish_detail.php?phish_id=6817719	2020-10-22T03:44:55+00:00	yes	2020-10-22T03:47:37+00:00	yes	Other
16	6817711	http://post-i.top/in.html	http://www.phishtank.com/phish_detail.php?phish_id=6817711	2020-10-22T03:30:12+00:00	yes	2020-10-22T05:26:12+00:00	yes	Other
17	6817710	http://post-i.top/mmi.html	http://www.phishtank.com/phish_detail.php?phish_id=6817710	2020-10-22T03:29:27+00:00	yes	2020-10-22T05:23:07+00:00	yes	Other
18	6817703	://secure.runescape.com-as.ru/m=weblogin/loginform621,363,768,94152249	http://www.phishtank.com/phish_detail.php?phish_id=6817703	2020-10-22T03:16:03+00:00	yes	2020-10-22T03:26:20+00:00	yes	Other
19	6817693	https://yahooatt1.weebly.com/	http://www.phishtank.com/phish_detail.php?phish_id=6817693	2020-10-22T03:01:50+00:00	yes	2020-10-22T03:05:05+00:00	yes	Other
20	6817691	/www.avoidsuspension-nowupdate.world/banks/online.lloydsbank.co.uk/n	http://www.phishtank.com/phish_detail.php?phish_id=6817691	2020-10-22T03:01:00+00:00	yes	2020-10-22T03:05:05+00:00	yes	Other
21	6817690	http://vbse.co.in/mircosoftonline	http://www.phishtank.com/phish_detail.php?phish_id=6817690	2020-10-22T03:00:56+00:00	yes	2020-10-22T03:05:05+00:00	yes	Other

Pre-processed Dataset (Attributes):

The data set contains 31 attributes which are calculated from the URL Attributes are:

having_IP_Address { -1,1 }
having_At_Symbol { 1,-1 }
having_Sub_Domain { -1,0,1 }
Favicon { 1,-1 }
Request_URL { 1,-1 }
SFH { -1,1,0 }
Redirect { 0,1 }
popUpWidnow { 1,-1 }
DNSRecord { -1,1 }
Google_Index { 1,-1 }
Result { -1,1 }
URL_Length { 1,0,-1 }
double_slash_redirecting { -1,1 }
SSLfinal_State { -1,1,0 }
port { 1,-1 }
URL_of_Anchor { -1,0,1 }
Submitting_to_email { -1,1 }
on_mouseover { 1,-1 }
Iframe { 1,-1 }
web_traffic { -1,0,1 }
Links_pointing_to_page { 1,0,-1 }
Shortining_Service { 1,-1 }
Prefix_Suffix { -1,1 }
Domain_registeration_length { -1,1 }
HTTPS_token { -1,1 }
Links_in_tags { 1,-1,0 }
Abnormal_URL { -1,1 }
RightClick { 1,-1 }
age_of_domain { -1,1 }
Page_Rank { -1,1 }
Statistical_report { -1,1 }

-1 indicates: Phishing Attribute
0 indicates: Suspicious Attribute
1 indicates: Non - Phishing Attribute

Pre-processed Dataset

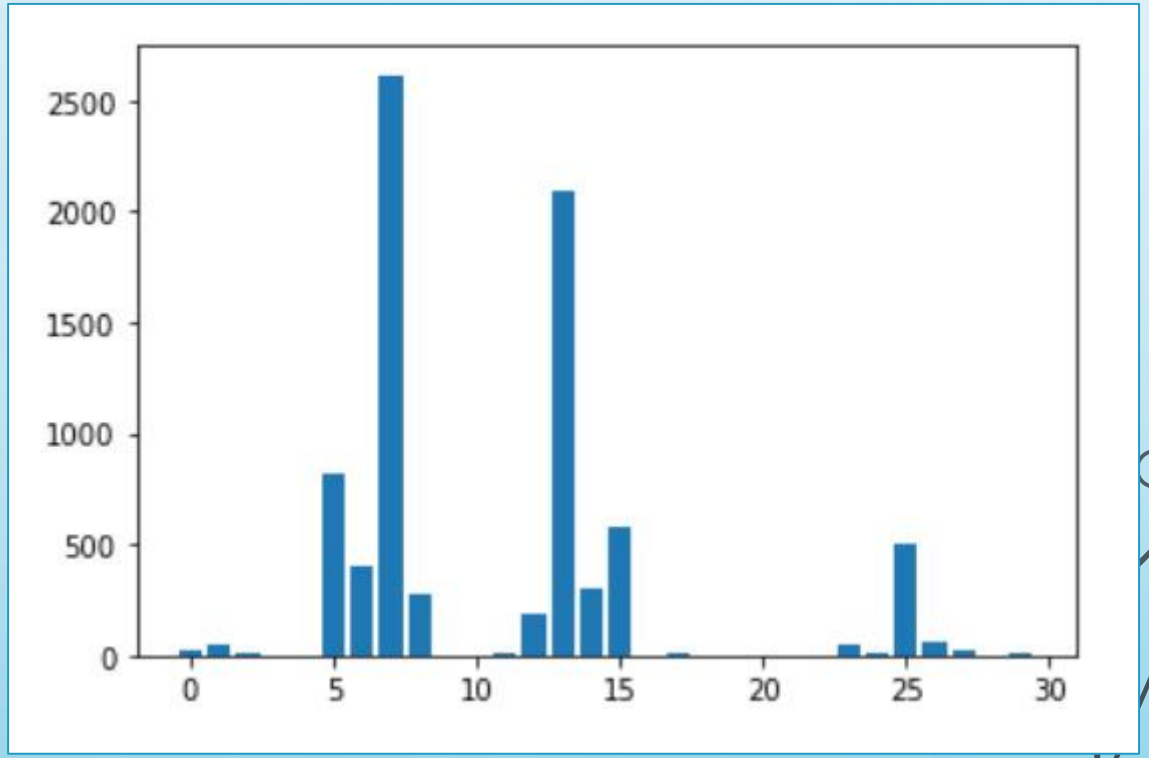
	having_IP_Address	URL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain	SSLfinal_State	Domai
0	-1	1	1	1	-1	-1	-1	-1	
1	1	1	1	1	1	-1	0	1	
2	1	0	1	1	1	-1	-1	-1	
3	1	0	1	1	1	-1	-1	-1	
4	1	0	-1	1	1	-1	1	1	
5	-1	0	-1	1	-1	-1	1	1	
6	1	0	-1	1	1	-1	-1	-1	
7	1	0	1	1	1	-1	-1	-1	
8	1	0	-1	1	1	-1	1	1	
9	1	1	-1	1	1	-1	-1	1	
10	1	1	1	1	1	-1	0	1	
11	1	1	-1	1	1	-1	1	-1	
12	-1	1	-1	1	-1	-1	0	0	
13	1	1	-1	1	1	-1	0	-1	
14	1	1	-1	1	1	1	-1	1	
15	1	-1	-1	-1	1	-1	0	0	
16	1	-1	-1	1	1	-1	1	1	
17	1	-1	1	1	1	-1	-1	0	
18	1	1	1	1	1	-1	-1	1	
19	1	1	1	1	1	-1	-1	1	

20 rows × 31 columns

Feature/Attribute 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 is used for categorical features in a dataset.

```
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
```



Attribute Parsing

Finally 16 attributes that we will consider for training our model after applying feature selection:

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

```
1 parser = ParseURl()
2 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
```

Model Selection

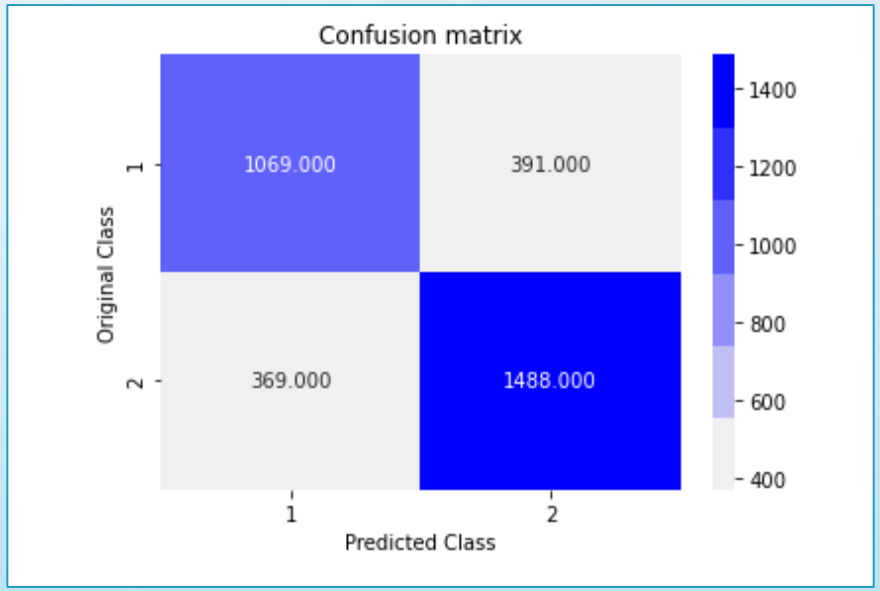
This module intakes URL as an input and after applying machine learning algorithms generate a phishing score. According to the generated score, the website will be tagged as phishing or normal.

Machine Learning Models that can be used are:

- Logistic Regression
- Support Vector Machine

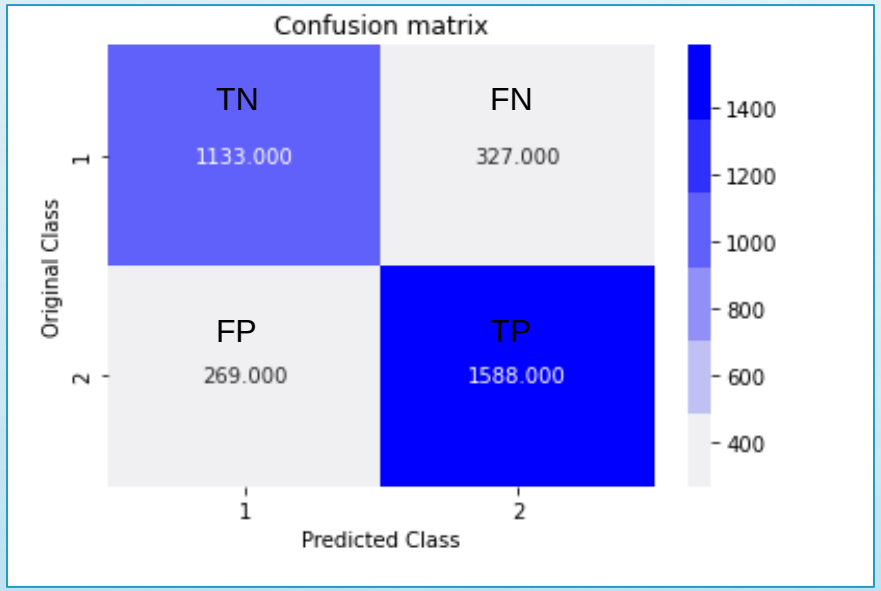
Model Selection

Logistic Regression



Accuracy of Logistic Regression :
0.7708772987639433

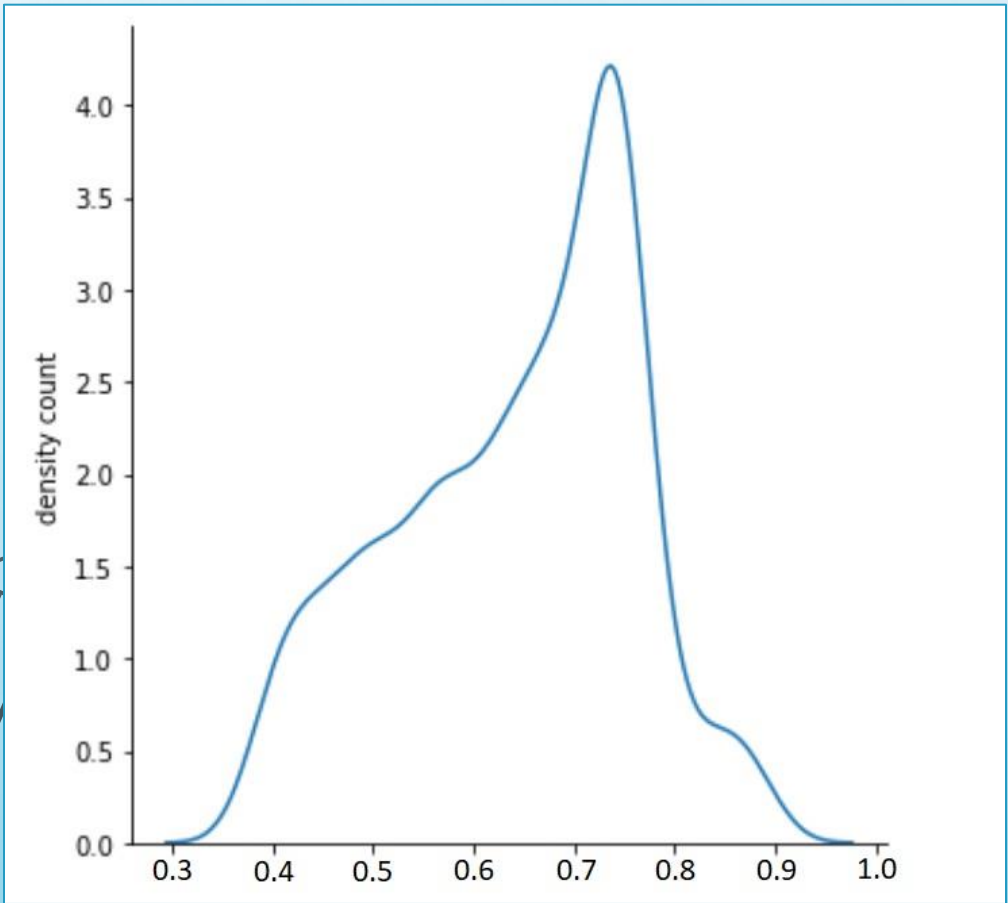
Support Vector Machine



Accuracy of Support Vector Machine :
0.8203195658727767

Range Selection

- Support Vector Machine (SVM) is used as the machine learning model for module 1.
- False Negative values obtained in the range are passed to module 2.



False Negative Output in the range **[0.6507983 , 0.81118389]** consists of **65%** of total false negative.

Hence the range of above density graph is passed to module 2.

Module: 2

Module 2
(Structure Analyzer)

This module examines the content of a web page to determine whether it is legitimate or not, in contrast to other approaches that look at surface characteristics of a web page, for example, the Web page content and its domain name.

To identify the duplicate or near-duplicate webpages we will use the technique:

- **URL Mining :**

  In this method, a signature is made using the title of the web page and 5 words of it with maximum TF (term frequency) values. Put that signature for Google search and get the first n web pages. Next step is to compare the web pages each other by HTML code matching and also by checking the textual part of these web pages for cosine similarity.

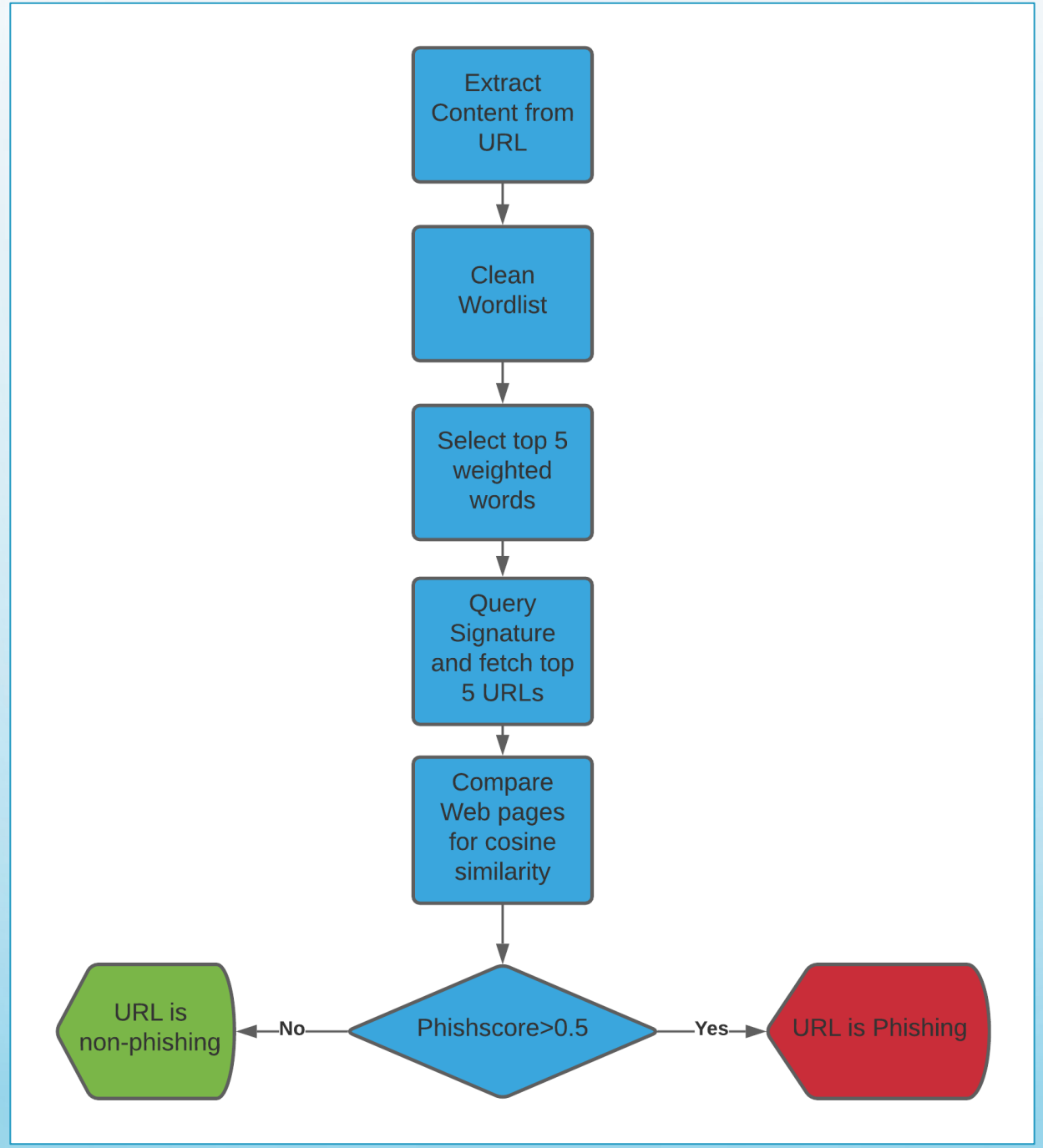
Cosine Similarity

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.

How to Convert Web page into vector:

- Raw texts are preprocessed with the most common words and punctuation removed, and tokenization is done.
- A dictionary of unique terms found in the whole corpus is created. Texts are quantified first by calculating the term frequency (tf) for each document. The numbers are used to create a vector for each document where each component in the vector stands for the term frequency in that document.

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


Extract Content from URL
Clean Wordlist
Select top 5 weighted words
Query Signature and fetch top 5 URLs
Compare Web pages for cosine similarity
Phishscore>0.5
No
URL is non-phishing
Yes
URL is Phishing

Implementation

```python
def driver_func(url):

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

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>
```

Module Integration

- **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.*

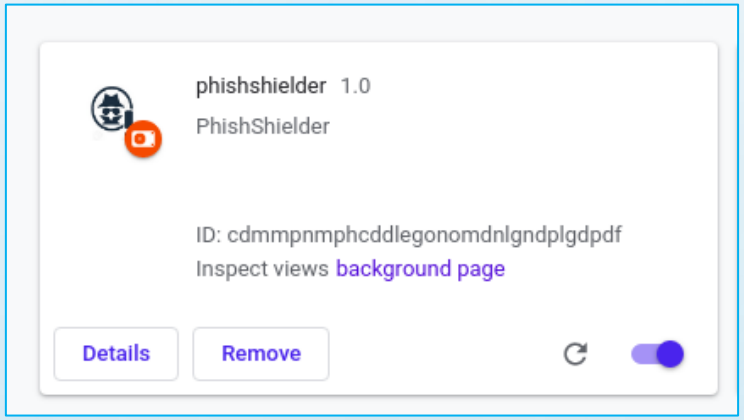
```
class MyCustomHandler(http.server.BaseHTTPRequestHandler):
    def do_GET(self):
        if self.path.__contains__("/phish_api?"):
            #print(self.path)
            # Parsing the parameter from the URL:
            # Courtesy: https://stackoverflow.com/questions/8928730/processing-http-get-input-parameter-on-server-side-in-python/8930440
            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()
```

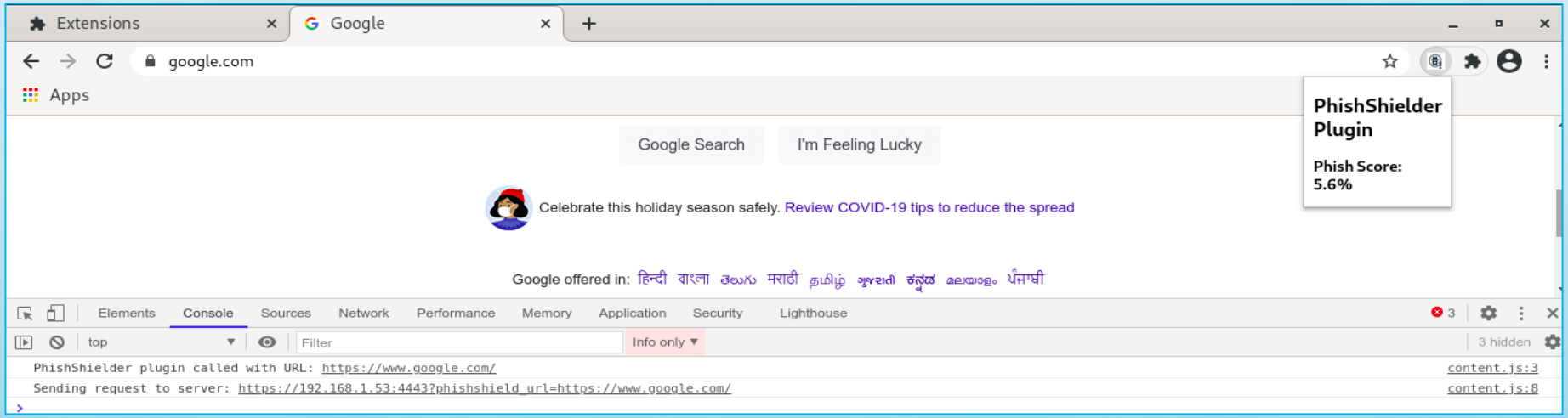
- **runPhishDetectionModule** is used to fire Module-1 and Module-2 (if required) and returns a dictionary containing keys *phish_status* and *phish_score*.

```
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
```

Implementation





Testing and Result

Accuracy of Module 1: **82.03%**
Accuracy After Integration of both the modules: **91.75%**

Module 1 : testing

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

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

Module1 + Module2

```
1 import requests
2 prediction = []
3 for url in data_url:
4     base_url = 'https://phish.local:4443/phish_api?phish_url='
5
6     #hitting our api (Module1 + Module2)
7     response = requests.get(base_url+url,verify=False)
8     prediction.append(response.json()['phish_score']/100)
9 print('accuracy for module1+module2 : ',accuracy_score(prediction,test_Y))
```

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

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%**.
- Module 2 helps us to increase the Efficiency by around **9.72%**.

Code

Implementation:
https://github.com/mansoorrangwala/PhishingDetectionSystem

Reference:

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 Code Comparison and Cosine Similarity [Research Paper]:** https://ieeexplore.ieee.org/abstract/document/6906016

Thank You!!!