PHISHING WEBSITE DETECTION

A PROJECT REPORT

Submitted by

SIGIRALA NAVEEN (RA2011042010086)
SEETHAMSETTI SRIKANTH(RA2011042010088)
BIJJAM VENKATA VISWANATH REDDY(RA2011042010092)
CHETHAN REDDY. V(RA2011042010093)
PATHIPATI RAGHAVENDRA LOHITH(RA2011042010139)

Under the guidance of

Dr. J JEBA SONIA

(Associate Professor, Department of Data Science and Business Systems)

in partial fulfillment for the award of the

degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND BUSINESS SYSTEMS
OF
FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Kancheepuram District

APRIL 2023

BONAFIDE CERTIFICATE

Certified that this project report titled

"Phishing Website Detection Feature Extraction"

is the bonafide work of "S. NAVEEN, SREEKANTH. S, VISWANATH REDDY. B, CHETHAN.V, LOHITH. P" who carried out the project work under my supervision as a batch. Certified further, that to the best of my knowledge the work reported herein does not form any other project report on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

Date : Project Supervisor Head of the Department

Submitted for University Examination held on ----------in the Department of Data Science and Business Systems, SRM Institute of Science and Technology, Kattankulathur.

Date: Internal Examiner External Examiner

ACKNOWLEDGEMENTS

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice Chancellor (I/C), SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to **Dr. Revathi Venkataraman**, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for his invaluable support.

We wish to thank **Dr. M. Lakshmi**, Professor & Head, Department of Data Science and Business Systems, SRM Institute of Science and Technology, for her valuable suggestions and encouragement throughout the period of the project work.

We are extremely grateful to our Academic Advisor **Dr. E. Sasikala**, Assistant Professor,
Department of Data Science and Business Systems, SRM Institute of Science and Technology, for her great support at all the stages of project work.

Our inexpressible respect and thanks to my guide, **Dr. J. Jeba Sonia,** Assistant Professor, Department of Data Science and Business Systems, SRM Institute of Science and Technology, for providing me an opportunity to pursue my project under her mentorship. She provided me the freedom and support to explore the research topics of my interest. Her passion for solving real problems and making a difference in the world has always been inspiring.

We sincerely thank staff and students of the Data Science and Business Systems Department, SRM Institute of Science and Technology, for their help during my research. Finally, we would like to thank my parents, our family members and our friends for their unconditional love, constant support and encouragement.

SIGIRALA NAVEEN (RA2011042010086)
SEETHAMSETTI SRIKANTH(RA2011042010088)
BIJJAM VENKATA VISWANATH REDDY(RA2011042010092)
CHETHAN REDDY. V(RA2011042010093)
PATHIPATI RAGHAVENDRA LOHITH(RA2011042010139)

Phishing Website Detection Feature Extraction

Final project of AI & Cybersecurity Course

1. Objective:

A phishing website is a common social engineering method that mimics trustful uniform resource locators (URLs) and webpages. The objective of this notebook is to collect data & extract the selctive features form the URLs.

This project is worked on Google Collaboratory.

2. Collecting the Data:

For this project, we need a bunch of urls of type legitimate (0) and phishing (1).

The collection of phishing urls is rather easy because of the opensource service called PhishTank. This service provide a set of phishing URLs in multiple formats like csv, json etc. that gets updated hourly. To download the data: https://www.phishtank.com/developer_info.php (https://www.phishtank.com/developer_info.php)

For the legitimate URLs, I found a source that has a collection of benign, spam, phishing, malware & defacement URLs. The source of the dataset is University of New Brunswick, https://www.unb.ca/cic/datasets/url-2016.html (https://www.unb.ca/cic/datasets/url-2016.html). The number of legitimate URLs in this collection are 35,300. The URL collection is downloaded & from that, *'Benign_list_big_final.csv'* is the file of our interest. This file is then uploaded to the Colab for the feature extraction.

2.1. Phishing URLs:

The phishing URLs are collected from the PhishTank from the link provided. The csv file of phishing URLs is obtained by using wget command. After downlaoding the dataset, it is loaded into a DataFrame.

In [0]:
```
#importing required packages for this module
import pandas as pd
```

In [0]:
```
#Downloading the phishing URLs file
!wget http://data.phishtank.com/data/online-valid.csv
```

```
--2020-05-10 07:33:37--  http://data.phishtank.com/data/online-valid.csv (htt
p://data.phishtank.com/data/online-valid.csv)
Resolving data.phishtank.com (data.phishtank.com)... 104.16.101.75, 104.17.17
7.85, 2606:4700::6810:654b, ...
Connecting to data.phishtank.com (data.phishtank.com)|104.16.101.75|:80... co
nnected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://data.phishtank.com/data/online-valid.csv (https://data.phis
htank.com/data/online-valid.csv) [following]
--2020-05-10 07:33:37--  https://data.phishtank.com/data/online-valid.csv (ht
tps://data.phishtank.com/data/online-valid.csv)
Connecting to data.phishtank.com (data.phishtank.com)|104.16.101.75|:443... c
onnected.
HTTP request sent, awaiting response... 302 Found
Location: https://d1750zhbc38ec0.cloudfront.net/datadumps/verified_online.cs
v?Expires=1589096027&Signature=NeznemrBS2h3ozoDsM8x9fZ73pTe1OhCjCyYEEtKcqjyJl
O62TdCD9eAh4tC0fvlytZAq4ihqhtRGtgkwaWfw6QJE8HhE-UfnzUlOxU6w-lnHJppNbsbWsIqCjY
eBoNbGvLTpa4CklK5Lo7PV6vd3bSl8wAq0PNjyct7f6qyO2nazZilc0NIdzHp2t-XwAozQj39S7cz
LORAzloGH98cqa1XBc3honvarNeV3S6d8QJCO8dHf3zk201KUSJFRIky6sFZP3--z5aDSL06fZj-y
AyIDE-Xn0SNaiqLFVuMQUx0tTo5eIdk98zC2D7R5XOvAkGdpo1fGHT45f77MzUv4Q__&Key-Pair-
Id=APKAILB45UG3RB4CSOJA (https://d1750zhbc38ec0.cloudfront.net/datadumps/veri
fied_online.csv?Expires=1589096027&Signature=NeznemrBS2h3ozoDsM8x9fZ73pTe1OhC
jCyYEEtKcqjyJlO62TdCD9eAh4tC0fvlytZAq4ihqhtRGtgkwaWfw6QJE8HhE-UfnzUlOxU6w-lnH
JppNbsbWsIqCjYeBoNbGvLTpa4CklK5Lo7PV6vd3bSl8wAq0PNjyct7f6qyO2nazZilc0NIdzHp2t
-XwAozQj39S7czLORAzloGH98cqa1XBc3honvarNeV3S6d8QJCO8dHf3zk201KUSJFRIky6sFZP3-
-z5aDSL06fZj-yAyIDE-Xn0SNaiqLFVuMQUx0tTo5eIdk98zC2D7R5XOvAkGdpo1fGHT45f77MzUv
4Q__&Key-Pair-Id=APKAILB45UG3RB4CSOJA) [following]
--2020-05-10 07:33:37--  https://d1750zhbc38ec0.cloudfront.net/datadumps/veri
fied_online.csv?Expires=1589096027&Signature=NeznemrBS2h3ozoDsM8x9fZ73pTe1OhC
jCyYEEtKcqjyJlO62TdCD9eAh4tC0fvlytZAq4ihqhtRGtgkwaWfw6QJE8HhE-UfnzUlOxU6w-lnH
JppNbsbWsIqCjYeBoNbGvLTpa4CklK5Lo7PV6vd3bSl8wAq0PNjyct7f6qyO2nazZilc0NIdzHp2t
-XwAozQj39S7czLORAzloGH98cqa1XBc3honvarNeV3S6d8QJCO8dHf3zk201KUSJFRIky6sFZP3-
-z5aDSL06fZj-yAyIDE-Xn0SNaiqLFVuMQUx0tTo5eIdk98zC2D7R5XOvAkGdpo1fGHT45f77MzUv
4Q__&Key-Pair-Id=APKAILB45UG3RB4CSOJA (https://d1750zhbc38ec0.cloudfront.net/
datadumps/verified_online.csv?Expires=1589096027&Signature=NeznemrBS2h3ozoDsM
8x9fZ73pTe1OhCjCyYEEtKcqjyJlO62TdCD9eAh4tC0fvlytZAq4ihqhtRGtgkwaWfw6QJE8HhE-U
fnzUlOxU6w-lnHJppNbsbWsIqCjYeBoNbGvLTpa4CklK5Lo7PV6vd3bSl8wAq0PNjyct7f6qyO2na
zZilc0NIdzHp2t-XwAozQj39S7czLORAzloGH98cqa1XBc3honvarNeV3S6d8QJCO8dHf3zk201KU
SJFRIky6sFZP3--z5aDSL06fZj-yAyIDE-Xn0SNaiqLFVuMQUx0tTo5eIdk98zC2D7R5XOvAkGdpo
1fGHT45f77MzUv4Q__&Key-Pair-Id=APKAILB45UG3RB4CSOJA)
Resolving d1750zhbc38ec0.cloudfront.net (d1750zhbc38ec0.cloudfront.net)... 14
3.204.101.142, 143.204.101.147, 143.204.101.48, ...
Connecting to d1750zhbc38ec0.cloudfront.net (d1750zhbc38ec0.cloudfront.net)|1
43.204.101.142|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3232768 (3.1M) [text/csv]
Saving to: ‘online-valid.csv’

online-valid.csv    100%[===================>]   3.08M  5.13MB/s    in 0.6s

2020-05-10 07:33:38 (5.13 MB/s) - ‘online-valid.csv’ saved [3232768/3232768]
```

The above command downlaods the file of phishing URLs, *online-valid.csv* and stores in the */content/* folder.

In [0]:
```
#loading the phishing URLs data to dataframe
data0 = pd.read_csv("online-valid.csv")
data0.head()
```

Out[3]:

	phish_id	url	phish_detail_url
0	6557033	http://u1047531.cp.regruhosting.ru/acces-inges...	http://www.phishtank.com/phish_detail.php?phis...
1	6557032	http://hoysalacreations.com/wp-content/plugins...	http://www.phishtank.com/phish_detail.php?phis...
2	6557011	http://www.accsystemprblemhelp.site/checkpoint...	http://www.phishtank.com/phish_detail.php?phis...
3	6557010	http://www.accsystemprblemhelp.site/login_atte...	http://www.phishtank.com/phish_detail.php?phis...
4	6557009	https://firebasestorage.googleapis.com/v0/b/so...	http://www.phishtank.com/phish_detail.php?phis...

In [0]:
```
data0.shape
```

Out[4]:
```
(14858, 8)
```

So, the data has thousands of phishing URLs. But the problem here is, this data gets updated hourly. Without getting into the risk of data imbalance, I am considering a margin value of 10,000 phishing URLs & 5000 legitimate URLs.

Thereby, picking up 5000 samples from the above dataframe randomly.

In [0]:
```
#Collecting 5,000 Phishing URLs randomly
phishurl = data0.sample(n = 5000, random_state = 12).copy()
phishurl = phishurl.reset_index(drop=True)
phishurl.head()
```

Out[5]:

	phish_id	url	phish_detail_url	sub
0	6485787	https://eevee.tv/Bootstrap/assets/css/acces	http://www.phishtank.com/phish_detail.php?phis...	04T0
1	6422543	https://appleid.apple.com-sa.pm/appleid/?	http://www.phishtank.com/phish_detail.php?phis...	27T1
2	6543602	https://grandcup.xyz/	http://www.phishtank.com/phish_detail.php?phis...	02T2
3	6528783	https://villa-azzurro.com/onedrive/	http://www.phishtank.com/phish_detail.php?phis...	25T2
4	6498136	http://mygpstrip.net/ii/u.php	http://www.phishtank.com/phish_detail.php?phis...	10T1

In [0]:
```
phishurl.shape
```

Out[6]:
```
(5000, 8)
```

As of now we collected 5000 phishing URLs. Now, we need to collect the legitimate URLs.

2.2. Legitimate URLs:

From the uploaded *Benign_list_big_final.csv* file, the URLs are loaded into a dataframe.

In [0]:
```
#Loading legitimate files
data1 = pd.read_csv("Benign_list_big_final.csv")
data1.columns = ['URLs']
data1.head()
```

Out[7]:

	URLs
0	http://1337x.to/torrent/1110018/Blackhat-2015-...
1	http://1337x.to/torrent/1122940/Blackhat-2015-...
2	http://1337x.to/torrent/1124395/Fast-and-Furio...
3	http://1337x.to/torrent/1145504/Avengers-Age-o...
4	http://1337x.to/torrent/1160078/Avengers-age-o...

As stated above, 5000 legitimate URLs are randomaly picked from the above dataframe.

In [0]:
```
#Collecting 5,000 Legitimate URLs randomly
legiurl = data1.sample(n = 5000, random_state = 12).copy()
legiurl = legiurl.reset_index(drop=True)
legiurl.head()
```

Out[8]:

	URLs
0	http://graphicriver.net/search?date=this-month...
1	http://ecnavi.jp/redirect/?url=http://www.cros...
2	https://hubpages.com/signin?explain=follow+Hub...
3	http://extratorrent.cc/torrent/4190536/AOMEI+B...
4	http://icicibank.com/Personal-Banking/offers/o...

In [0]:
```
legiurl.shape
```

Out[9]:
```
(5000, 1)
```

3. Feature Extraction:

In this step, features are extracted from the URLs dataset.

The extracted features are categorized into

1. Address Bar based Features
2. Domain based Features
3. HTML & Javascript based Features

3.1. Address Bar Based Features:

Many features can be extracted that can be consided as address bar base features. Out of them, below mentioned were considered for this project.

- Domain of URL
- IP Address in URL
- "@" Symbol in URL
- Length of URL
- Depth of URL
- Redirection "//" in URL
- "http/https" in Domain name
- Using URL Shortening Services “TinyURL”
- Prefix or Suffix "-" in Domain

Each of these features are explained and the coded below:

In [0]:
```
# importing required packages for this section
from urllib.parse import urlparse,urlencode
import ipaddress
import re
```

3.1.1. Domain of the URL

Here, we are just extracting the domain present in the URL. This feature doesn't have much significance in the training. May even be dropped while training the model.

In [0]:
```
# 1.Domain of the URL (Domain)
def getDomain(url):
  domain = urlparse(url).netloc
  if re.match(r"^www.",domain):
            domain = domain.replace("www.","")
  return domain
```

3.1.2. IP Address in the URL

Checks for the presence of IP address in the URL. URLs may have IP address instead of domain name. If an IP address is used as an alternative of the domain name in the URL, we can be sure that someone is trying to steal personal information with this URL.

If the domain part of URL has IP address, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

In [0]:
```
# 2.Checks for IP address in URL (Have_IP)
def havingIP(url):
  try:
    ipaddress.ip_address(url)
    ip = 1
  except:
    ip = 0
  return ip
```

3.1.3. "@" Symbol in URL

Checks for the presence of '@' symbol in the URL. Using “@” symbol in the URL leads the browser to ignore everything preceding the “@” symbol and the real address often follows the “@” symbol.

If the URL has '@' symbol, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

In [0]:
```
# 3.Checks the presence of @ in URL (Have_At)
def haveAtSign(url):
  if "@" in url:
    at = 1
  else:
    at = 0
  return at
```

3.1.4. Length of URL

Computes the length of the URL. Phishers can use long URL to hide the doubtful part in the address bar. In this project, if the length of the URL is greater than or equal 54 characters then the URL classified as phishing otherwise legitimate.

If the length of URL >= 54 , the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

In [0]:
```
# 4.Finding the length of URL and categorizing (URL_Length)
def getLength(url):
  if len(url) < 54:
    length = 0
  else:
    length = 1
  return length
```

3.1.5. Depth of URL

Computes the depth of the URL. This feature calculates the number of sub pages in the given url based on the '/'.

In [0]:
```
# 5.Gives number of '/' in URL (URL_Depth)
def getDepth(url):
  s = urlparse(url).path.split('/')
  depth = 0
  for j in range(len(s)):
    if len(s[j]) != 0:
      depth = depth+1
  return depth
```

3.1.6. Redirection "//" in URL

Checks the presence of "//" in the URL. The existence of “//” within the URL path means that the user will be redirected to another website. The location of the “//” in URL is computed. We find that if the URL starts with “HTTP”, that means the “//” should appear in the sixth position. However, if the URL employs “HTTPS” then the “//” should appear in seventh position.

If the "//" is anywhere in the URL apart from after the protocal, thee value assigned to this feature is 1 (phishing) or else 0 (legitimate).

In [0]:
```
# 6.Checking for redirection '//' in the url (Redirection)
def redirection(url):
  pos = url.rfind('//')
  if pos > 6:
    if pos > 7:
      return 1
    else:
      return 0
  else:
    return 0
```

3.1.7. "http/https" in Domain name

Checks for the presence of "http/https" in the domain part of the URL. The phishers may add the “HTTPS” token to the domain part of a URL in order to trick users.

If the URL has "http/https" in the domain part, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

In [0]:
```
# 7.Existence of “HTTPS” Token in the Domain Part of the URL (https_Domain)
def httpDomain(url):
  domain = urlparse(url).netloc
  if 'https' in domain:
    return 1
  else:
    return 0
```

3.1.8. Using URL Shortening Services “TinyURL”

URL shortening is a method on the “World Wide Web” in which a URL may be made considerably smaller in length and still lead to the required webpage. This is accomplished by means of an “HTTP Redirect” on a domain name that is short, which links to the webpage that has a long URL.

If the URL is using Shortening Services, the value assigned to this feature is 1 (phishing) or

In [0]:
```
#Listing shortening services
shortening_services = r"bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.c
                      r"yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac
                      r"short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite
                      r"doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly
                      r"qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com
                      r"po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cu
                      r"prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.co
                      r"tr\.im|link\.zip\.net"
```

In [0]:
```
# 8. Checking for Shortening Services in URL (Tiny_URL)
def tinyURL(url):
    match=re.search(shortening_services,url)
    if match:
        return 1
    else:
        return 0
```

3.1.9. Prefix or Suffix "-" in Domain

Checking the presence of '-' in the domain part of URL. The dash symbol is rarely used in legitimate URLs. Phishers tend to add prefixes or suffixes separated by (-) to the domain name so that users feel that they are dealing with a legitimate webpage.

If the URL has '-' symbol in the domain part of the URL, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

In [0]:
```
# 9.Checking for Prefix or Suffix Separated by (-) in the Domain (Prefix/Suffi
def prefixSuffix(url):
    if '-' in urlparse(url).netloc:
        return 1            # phishing
    else:
        return 0            # legitimate
```

3.2. Domain Based Features:

Many features can be extracted that come under this category. Out of them, below mentioned were considered for this project.

- DNS Record
- Website Traffic
- Age of Domain
- End Period of Domain

Each of these features are explained and the coded below:

In [0]:
```
!pip install python-whois
```

```
Collecting python-whois
  Downloading https://files.pythonhosted.org/packages/f0/ab/11c2d01db2554bbaa
bb2c32b06b6a73f7277372533484c320c78a304dfd7/python-whois-0.7.2.tar.gz (http
s://files.pythonhosted.org/packages/f0/ab/11c2d01db2554bbaabb2c32b06b6a73f727
7372533484c320c78a304dfd7/python-whois-0.7.2.tar.gz) (90kB)
     |████████████████████████████████| 92kB 5.5MB/s eta 0:00:011
Requirement already satisfied: future in /usr/local/lib/python3.6/dist-packag
es (from python-whois) (0.16.0)
Building wheels for collected packages: python-whois
  Building wheel for python-whois (setup.py) ... done
  Created wheel for python-whois: filename=python_whois-0.7.2-cp36-none-any.w
hl size=85245 sha256=900afbc18f144913762a57978778098dda65b687b3b5a1f14f7998e9
631564e8
  Stored in directory: /root/.cache/pip/wheels/69/e6/62/1e6a746ca8e690f472611
511b6948c325b232aaf693245ce46
Successfully built python-whois
Installing collected packages: python-whois
Successfully installed python-whois-0.7.2
```

In [0]:
```
# importing required packages for this section
import re
from bs4 import BeautifulSoup
import whois
import urllib
import urllib.request
from datetime import datetime
```

3.2.1. DNS Record

For phishing websites, either the claimed identity is not recognized by the WHOIS database or no records founded for the hostname. If the DNS record is empty or not found then, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

In [0]:
```
# 11.DNS Record availability (DNS_Record)
# obtained in the featureExtraction function itself
```

3.2.2. Web Traffic

This feature measures the popularity of the website by determining the number of visitors and the number of pages they visit. However, since phishing websites live for a short period of time, they may not be recognized by the Alexa database (Alexa the Web Information Company., 1996). By reviewing our dataset, we find that in worst scenarios, legitimate websites ranked among the top 100,000. Furthermore, if the domain has no traffic or is not recognized by the Alexa database, it is classified as “Phishing”.

If the rank of the domain < 100000, the vlaue of this feature is 1 (phishing) else 0 (legitimate).

In [0]:
```
# 12.Web traffic (Web_Traffic)
def web_traffic(url):
  try:
    #Filling the whitespaces in the URL if any
    url = urllib.parse.quote(url)
    rank = BeautifulSoup(urllib.request.urlopen("http://data.alexa.com/data?cl
        "REACH")['RANK']
    rank = int(rank)
  except TypeError:
        return 1
  if rank <100000:
    return 1
  else:
    return 0
```

3.2.3. Age of Domain

This feature can be extracted from WHOIS database. Most phishing websites live for a short period of time. The minimum age of the legitimate domain is considered to be 12 months for this project. Age here is nothing but different between creation and expiration time.

If age of domain > 12 months, the vlaue of this feature is 1 (phishing) else 0 (legitimate).

In [0]:
```
# 13.Survival time of domain: The difference between termination time and crea
def domainAge(domain_name):
  creation_date = domain_name.creation_date
  expiration_date = domain_name.expiration_date
  if (isinstance(creation_date,str) or isinstance(expiration_date,str)):
    try:
      creation_date = datetime.strptime(creation_date,'%Y-%m-%d')
      expiration_date = datetime.strptime(expiration_date,"%Y-%m-%d")
    except:
      return 1
  if ((expiration_date is None) or (creation_date is None)):
      return 1
  elif ((type(expiration_date) is list) or (type(creation_date) is list)):
      return 1
  else:
    ageofdomain = abs((expiration_date - creation_date).days)
    if ((ageofdomain/30) < 6):
      age = 1
    else:
      age = 0
  return age
```

3.2.4. End Period of Domain

This feature can be extracted from WHOIS database. For this feature, the remaining domain time is calculated by finding the different between expiration time & current time. The end period considered for the legitimate domain is 6 months or less for this project.

If end period of domain > 6 months, the vlaue of this feature is 1 (phishing) else 0 (legitimate).

In [0]:
```
# 14.End time of domain: The difference between termination time and current t
def domainEnd(domain_name):
  expiration_date = domain_name.expiration_date
  if isinstance(expiration_date,str):
    try:
      expiration_date = datetime.strptime(expiration_date,"%Y-%m-%d")
    except:
      return 1
  if (expiration_date is None):
      return 1
  elif (type(expiration_date) is list):
      return 1
  else:
    today = datetime.now()
    end = abs((expiration_date - today).days)
    if ((end/30) < 6):
      end = 0
    else:
      end = 1
  return end
```

3.3. HTML and JavaScript based Features

Many features can be extracted that come under this category. Out of them, below mentioned were considered for this project.

- IFrame Redirection
- Status Bar Customization
- Disabling Right Click
- Website Forwarding

Each of these features are explained and the coded below:

In [0]:
```
# importing required packages for this section
import requests
```

3.3.1. IFrame Redirection

IFrame is an HTML tag used to display an additional webpage into one that is currently shown. Phishers can make use of the “iframe” tag and make it invisible i.e. without frame borders. In this regard, phishers make use of the “frameBorder” attribute which causes the browser to render a visual delineation.

If the iframe is empty or repsonse is not found then, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

In [0]:
```
# 15. IFrame Redirection (iFrame)
def iframe(response):
  if response == "":
      return 1
  else:
      if re.findall(r"[<iframe>|<frameBorder>]", response.text):
          return 0
      else:
          return 1
```

3.3.2. Status Bar Customization

Phishers may use JavaScript to show a fake URL in the status bar to users. To extract this feature, we must dig-out the webpage source code, particularly the “onMouseOver” event, and check if it makes any changes on the status bar

If the response is empty or onmouseover is found then, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

In [0]:
```
# 16.Checks the effect of mouse over on status bar (Mouse_Over)
def mouseOver(response):
  if response == "" :
    return 1
  else:
    if re.findall("<script>.+onmouseover.+</script>", response.text):
      return 1
    else:
      return 0
```

3.3.3. Disabling Right Click

Phishers use JavaScript to disable the right-click function, so that users cannot view and save the webpage source code. This feature is treated exactly as “Using onMouseOver to hide the Link”. Nonetheless, for this feature, we will search for event “event.button==2” in the webpage source code and check if the right click is disabled.

If the response is empty or onmouseover is not found then, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

In [0]:
```
# 17.Checks the status of the right click attribute (Right_Click)
def rightClick(response):
  if response == "":
    return 1
  else:
    if re.findall(r"event.button ?== ?2", response.text):
      return 0
    else:
      return 1
```

3.3.4. Website Forwarding

The fine line that distinguishes phishing websites from legitimate ones is how many times a website has been redirected. In our dataset, we find that legitimate websites have been redirected one time max. On the other hand, phishing websites containing this feature have been redirected at least 4 times.

In [0]:
```
# 18.Checks the number of forwardings (Web_Forwards)
def forwarding(response):
  if response == "":
    return 1
  else:
    if len(response.history) <= 2:
      return 0
    else:
      return 1
```

4. Computing URL Features

Create a list and a function that calls the other functions and stores all the features of the URL in the list. We will extract the features of each URL and append to this list.

In [0]:
```
#Function to extract features
def featureExtraction(url,label):

  features = []
  #Address bar based features (10)
  features.append(getDomain(url))
  features.append(havingIP(url))
  features.append(haveAtSign(url))
  features.append(getLength(url))
  features.append(getDepth(url))
  features.append(redirection(url))
  features.append(httpDomain(url))
  features.append(tinyURL(url))
  features.append(prefixSuffix(url))

  #Domain based features (4)
  dns = 0
  try:
    domain_name = whois.whois(urlparse(url).netloc)
  except:
    dns = 1

  features.append(dns)
  features.append(web_traffic(url))
  features.append(1 if dns == 1 else domainAge(domain_name))
  features.append(1 if dns == 1 else domainEnd(domain_name))

  # HTML & Javascript based features (4)
  try:
    response = requests.get(url)
  except:
    response = ""
  features.append(iframe(response))
  features.append(mouseOver(response))
  features.append(rightClick(response))
  features.append(forwarding(response))
  features.append(label)

  return features
```

4.1. Legitimate URLs:

Now, feature extraction is done on legitimate URLs.

In [0]:
```
legiurl.shape
```

Out[33]:
```
(5000, 1)
```

In [0]:
```
#Extracting the feautres & storing them in a list
legi_features = []
label = 0

for i in range(0, 5000):
  url = legiurl['URLs'][i]
  legi_features.append(featureExtraction(url,label))
```

In [0]:
```
#converting the list to dataframe
feature_names = ['Domain', 'Have_IP', 'Have_At', 'URL_Length', 'URL_Depth','Re
                      'https_Domain', 'TinyURL', 'Prefix/Suffix', 'DNS_Record'
                      'Domain_Age', 'Domain_End', 'iFrame', 'Mouse_Over','Righ

legitimate = pd.DataFrame(legi_features, columns= feature_names)
legitimate.head()
```

Out[35]:

	Domain	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	TinyU
0	graphicriver.net	0	0	1	1	0	0	
1	ecnavi.jp	0	0	1	1	1	0	
2	hubpages.com	0	0	1	1	0	0	
3	extratorrent.cc	0	0	1	3	0	0	
4	icicibank.com	0	0	1	3	0	0	

In [0]:
```
# Storing the extracted legitimate URLs fatures to csv file
legitimate.to_csv('legitimate.csv', index= False)
```

4.2. Phishing URLs:

Now, feature extraction is performed on phishing URLs.

In [0]:
```
phishurl.shape
```

Out[37]:
```
(5000, 8)
```

In [0]:
```
#Extracting the feautres & storing them in a list
phish_features = []
label = 1
for i in range(0, 5000):
  url = phishurl['url'][i]
  phish_features.append(featureExtraction(url,label))
```

In [0]:
```
#converting the list to dataframe
feature_names = ['Domain', 'Have_IP', 'Have_At', 'URL_Length', 'URL_Depth','Re
                 'https_Domain', 'TinyURL', 'Prefix/Suffix', 'DNS_Record'
                 'Domain_Age', 'Domain_End', 'iFrame', 'Mouse_Over','Righ

phishing = pd.DataFrame(phish_features, columns= feature_names)
phishing.head()
```

Out[40]:

	Domain	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	Ti
0	eevee.tv	0	0	0	4	0	0	
1	appleid.apple.com-sa.pm	0	0	0	1	0	0	
2	grandcup.xyz	0	0	0	0	0	0	
3	villa-azzurro.com	0	0	0	1	0	0	
4	mygpstrip.net	0	0	0	2	0	0	

In [0]:
```
# Storing the extracted legitimate URLs fatures to csv file
phishing.to_csv('phishing.csv', index= False)
```

5. Final Dataset

In the above section we formed two dataframes of legitimate & phishing URL features. Now, we will combine them to a single dataframe and export the data to csv file for the Machine Learning training done in other notebook.

In [0]:
```
#Concatenating the dataframes into one
urldata = pd.concat([legitimate, phishing]).reset_index(drop=True)
urldata.head()
```

Out[45]:

	Domain	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	TinyU
0	graphicriver.net	0	0	1	1	0	0	
1	ecnavi.jp	0	0	1	1	1	0	
2	hubpages.com	0	0	1	1	0	0	
3	extratorrent.cc	0	0	1	3	0	0	
4	icicibank.com	0	0	1	3	0	0	

In [0]:
```
urldata.tail()
```

Out[46]:

	Domain	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain
9995	wvk12-my.sharepoint.com	0	0	1	5	0	0
9996	adplife.com	0	0	1	4	0	0
9997	kurortnoye.com.ua	0	1	1	3	0	0
9998	norcaltc-my.sharepoint.com	0	0	1	5	0	0
9999	sieck-kuehlsysteme.de	0	1	1	4	0	0

In [0]:
```
urldata.shape
```

Out[47]:
```
(10000, 18)
```

In [0]:
```
# Storing the data in CSV file
urldata.to_csv('urldata.csv', index=False)
```

6. Conclusion

With this the objective of this notebook is achieved. We finally extracted 18 features for 10,000 URL which has 5000 phishing & 5000 legitimate URLs.

7. References

- https://archive.ics.uci.edu/ml/datasets/Phishing+Websites (https://archive.ics.uci.edu/ml/datasets/Phishing+Websites)