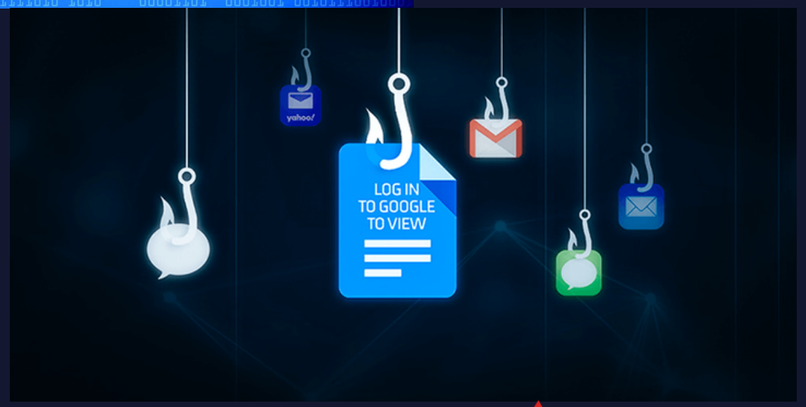
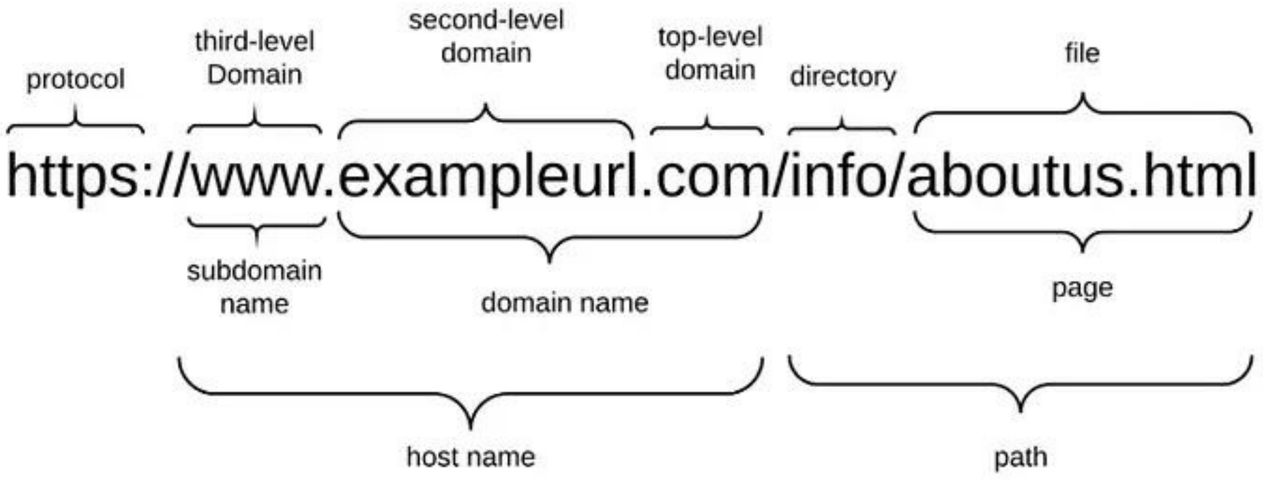


Predictive Modelling of Phishing Website Detection

Abhijit Das,
Sabrina Sanjida Rifa,
Abhirup Chanda

Phishing Websites and importance of its detection

- A phishing website is a popular social engineering technique that imitates reliable Addresses and online sites.



- Detecting phishing websites is important to protect personal and financial information, maintain reputation, and safeguard confidential business information.

Objectives of this project

- Gathering both phishing and benign URLs of websites.
- Extracting the key features from those URLs to form a dataset.
- Training machine learning models and deep neural nets on the dataset created to predict phishing websites.
- Measuring and comparing the performance level of each model.

Collecting Data - Phishing URLs

- An open-source service called PhishTank has been used. This service gives users access to a substantial database of phishing URLs that is updated hourly (csv, json).
  https://www.phishtank.com/developer_info.php, Company's official website.

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

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

- From this dataset, 5000 phishing URLs has been randomly selected for the purpose of this project.

Collecting Data - Legitimate URLs

- The free datasets offered by the University of New Brunswick - large collection of legitimate URLs.
- Contains a wide variety of URLs, including benign, spam, phishing, malware, and defacement URLs.

https://www.unb.ca/cic/datasets/url-2016.html

```
#Loading legitimate files
data1 = pd.read_csv("1.Benign_list_big_final.csv")
data1.columns = ['URLs']
data1.head()
```

	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...

- From this dataset, 5000 legitimate URLs has been randomly selected for the purpose of this project.

Feature Extraction

- Address Bar based Features
  - 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
- Domain based Features
  - DNS Record
  - Website Traffic
  - Age of Domain
  - End Period of Domain
- HTML and JavaScript based Features
  - IFrame Redirection
  - Status Bar Customization
  - Disabling Right Click
  - Website Forwarding

Reference:

Buber, Ebubekir. “Phishing URL Detection with ML.” Medium, Towards Data Science, 21 May 2018, towardsdatascience.com/phishing-domain-detection-with-ml-5be9c99293e5#:~:text=The%20purpose%20of%20Phishing%20Domain,provide%20useful%20information%20to%20us.

Mohammad, Rami, McCluskey, T.L. and Thabtah, Fadi (2012) An Assessment of Features Related to Phishing Websites using an Automated Technique. In: International Conferece For Internet Technology And Secured Transactions. ICITST 2012 . IEEE, London, UK, pp. 492-497. ISBN 978-1-4673-5325-0

Mohammad, Rami, Thabtah, Fadi Abdeljaber and McCluskey, T.L. (2014) Predicting phishing websites based on self-structuring neural network. Neural Computing and Applications, 25 (2). pp. 443-458. ISSN 0941-0643

Mohammad, Rami, McCluskey, T.L. and Thabtah, Fadi Abdeljaber (2014) Intelligent Rule based Phishing Websites Classification. IET Information Security, 8 (3). pp. 153-160. ISSN 1751-8709

Address Bar based Features

- ***Domain of the URL*** -
  extracting the domain present in the URL.

```python
# 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
```

- ***IP Address in the URL*** -
  - 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).

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

- ***"@" Symbol in 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).

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

Address Bar based Features

- ***Length of URL*** -
  - Phishers can use long URL to hide the doubtful part in the address bar.
  - If the length of URL >= 54 , the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

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

- ***Depth of URL -***
  - Calculates the number of sub pages in the given url based on the '/'.
  - The value of feature is a numerical based on the URL.

```
# 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
```

- ***"http/https" in Domain name -***
  - 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).

```
# 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
```

Address Bar based Features

- ***Redirection "//" in URL -***
  - The existence of “//” within the URL path means that the user will be redirected to another website.
  - 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).

```
# 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
```

- ***Prefix or Suffix "-" in Domain -***
  - 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).

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

Address Bar based Features

- ***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 else 0 (legitimate).

```
#listing shortening services
shortening_services = r"bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|" \
                      r"yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|" \
                      r"short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|" \
                      r"doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|db\.tt|" \
                      r"qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|q\.gs|is\.gd|" \
                      r"po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|x\.co|" \
                      r"prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.me|v\.gd|" \
                      r"tr\.im|link\.zip\.net"
```

```
# 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
```

Domain based Features

- ***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).

```python
# 11.DNS Record availability (DNS_Record)
import whois

def dnsrecord(url):
    try:
        domain = urlparse(url).netloc
        w = whois.whois(domain)
        if w.status:
            return 0   # legitimate
        else:
            return 1   # phishing
    except:
        return 1       # phishing
```

- ***Age of Domain*** -
  - This feature can be extracted from WHOIS database.
  - Most phishing websites live for a short period of time. The minimum age (different between creation and expiration time) of the legitimate domain is considered to be 12 months for this project.
  - If age of domain < 12 months, the vlaue of this feature is 1 (phishing) else 0 (legitimate).

```python
# 13.Survival time of domain: The difference between termination time and creation time (Domain_Age)
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) < 12):
      age = 1
    else:
      age = 0
  return age
```

Domain based Features

- ***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).

```
# 14.End time of domain: The difference between termination time and current time (Domain_End)
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
```

Domain based Features

- ***Web Traffic*** -
  - Measures the popularity of the website by determining the number of visitors and the number of pages they visit.
  - 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).
  - In worst scenarios, legitimate websites ranked among the top 100,000 on an average.
  - 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).

```python
# 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?cli=10&dat=s&url=" + url).read(), "xml").find(
        "REACH")['RANK']
    rank = int(rank)
  except TypeError:
        return 1
  if rank <100000:
    return 1
  else:
    return 0
```

HTML and JavaScript based Features

- ***IFrame Redirection*** -
  - IFrame is an HTML tag use to display an additional webpage into current page.
  - Phishers can take advantage of the “iframe” tag to make it invisible i.e. without frame borders. “frameBorder” attribute causes the browser to render a visual delineation.
  - If the iframe is empty or response is not found then, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

```
# 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
```

- ***Status Bar Customization*** -
  - Using JavaScript to show a fake URL in the status bar. 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).

```
# 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
```

HTML and JavaScript based Features

- ***Disabling Right Click*** -
  - Phishers may use JavaScript to disable the right-click function, so users cannot view and save the webpage source code.
  - This feature is treated exactly as “Using onMouseOver”. 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).

```python
# 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
```

- ***Website Forwarding*** -
  - The fine line that distinguishes phishing websites from legitimate ones is how many times a website has been redirected.
  - If the forwarding response is more than 2 times, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

```python
# 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
```

Computing URL Features and Creating a Dataset

- Creating a list and a function that calls the other functions and stores all the features of the URL in the list and extracting the features of each URL and append to this list.

- Forming two dataframes of legitimate & phishing URL features using the list.

- Combining them to a single dataframe and export the data to csv file for training the Machine Learning Model

```
#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))
```

```
#converting the list to dataframe
feature_names = ['Domain', 'Have_IP', 'Have_At', 'URL_Length', 'URL_Depth','Redirection',
                 'https_Domain', 'TinyURL', 'Prefix/Suffix', 'DNS_Record', 'Web_Traffic',
                 'Domain_Age', 'Domain_End', 'iFrame', 'Mouse_Over','Right_Click', 'Web_Forwards', 'Lab

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

```
#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))
```

```
#converting the list to dataframe
feature_names = ['Domain', 'Have_IP', 'Have_At', 'URL_Length', 'URL_Depth','Redirection',
                 'https_Domain', 'TinyURL', 'Prefix/Suffix', 'DNS_Record', 'Web_Traffic',
                 'Domain_Age', 'Domain_End', 'iFrame', 'Mouse_Over','Right_Click', 'Web_Forwards', 'Label']

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

```
#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)

  # 11.DNS Record availability (DNS_Record)
  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
```

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

Final Dataset

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

	Domain	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	TinyURL	Prefix/Suffix	DNS_Record	Web_Traffic	Domain_Age	Domain_End	iFrame	Mouse_Over	Right_Click	Web_Forwards	Label
0	graphicriver.net	0	0	1	1	0	0	0	0	0	1	1	1	0	0	1	0	0
1	ecnavi.jp	0	0	1	1	1	0	0	0	0	1	1	1	0	0	1	0	0
2	hubpages.com	0	0	1	1	0	0	0	0	0	1	0	1	0	0	1	0	0
3	extratorrent.cc	0	0	1	3	0	0	0	0	0	1	0	1	0	0	1	0	0
4	icicibank.com	0	0	1	3	0	0	0	0	0	1	0	1	0	0	1	0	0

```
urldata.tail()
```

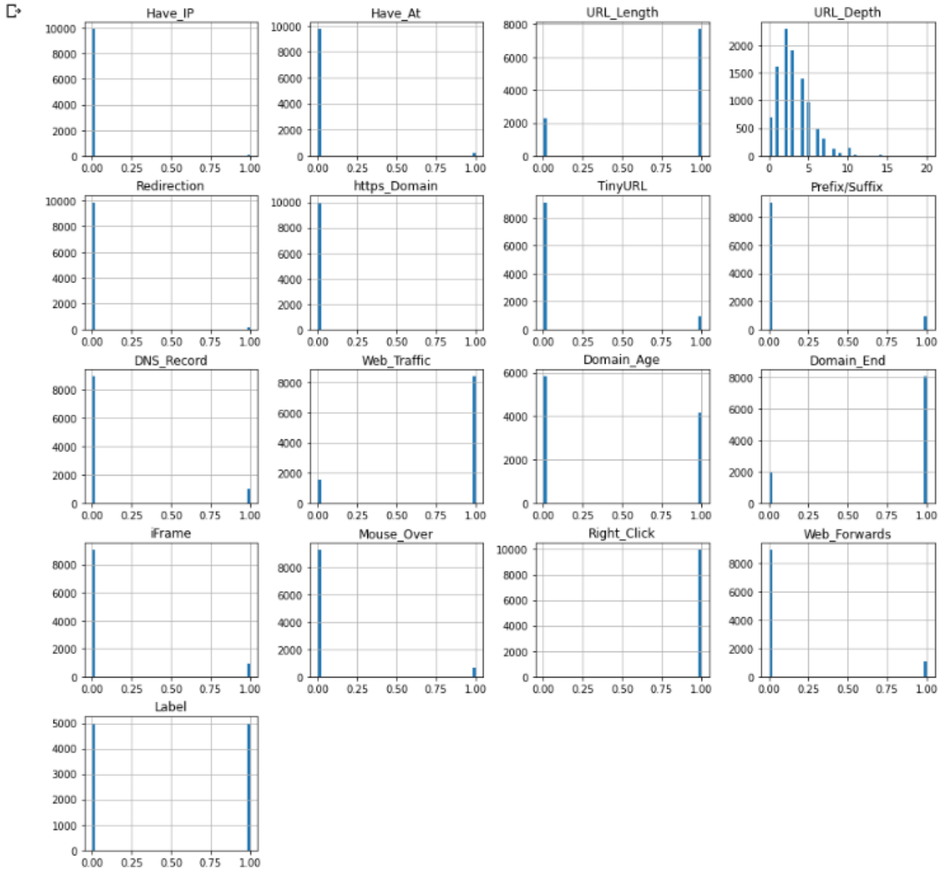
	Domain	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	TinyURL	Prefix/Suffix	DNS_Record	Web_Traffic	Domain_Age	Domain_End	iFrame	Mouse_Over	Right_Click	Web_Forwards	Label
9995	wvk12-my.sharepoint.com	0	0	1	5	0	0	1	1	0	1	1	1	0	0	1	0	1
9996	adplife.com	0	0	1	4	0	0	0	0	0	1	0	1	0	0	1	0	1
9997	kurortnoye.com.ua	0	1	1	3	0	0	1	0	0	0	1	1	1	0	1	0	1
9998	norcaltc-my.sharepoint.com	0	0	1	5	0	0	1	1	0	1	1	1	0	0	1	0	1
9999	sieck-kuehlsysteme.de	0	1	1	4	0	0	1	1	0	1	1	1	0	0	1	0	1

```
[ ] urldata.shape
```

```
(10000, 18)
```

Feature Distribution

```
#Plotting the data distribution
data0.hist(bins = 50,figsize = (15,15))
plt.show()
```



The obtained result shows that the most of the data is made of 0's & 1's except 'Domain' & 'URL_Depth' columns. The Domain column doesnt have any significance to the machine learning model training. So dropped the 'Domain' column from the dataset. As, the 'URL_Depth' maximum value is 20, there is no necessity to change this column.

```
[ ]
data0.describe()
```

	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	TinyURL	Prefix/Suffix	DNS_Record	Web_Traffic	Domain_Age	Domain_End	iFrame	Mouse_Over	Right_Click	Web_Forwards	Label
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.0000	10000.000000	10000.00000	10000.00000	10000.000000	10000.000000
mean	0.005500	0.022600	0.773400	3.072000	0.013500	0.000200	0.090300	0.093200	0.100800	0.845700	0.413700	0.8099	0.090900	0.06660	0.99930	0.105300	0.500000
std	0.073961	0.148632	0.418653	2.128631	0.115408	0.014141	0.286625	0.290727	0.301079	0.361254	0.492521	0.3924	0.287481	0.24934	0.02645	0.306955	0.500025
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000	0.000000	0.00000	0.00000	0.000000	0.000000
25%	0.000000	0.000000	1.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	1.0000	0.000000	0.00000	1.00000	0.000000	0.000000
50%	0.000000	0.000000	1.000000	3.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	1.0000	0.000000	0.00000	1.00000	0.000000	0.500000
75%	0.000000	0.000000	1.000000	4.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	1.000000	1.0000	0.000000	0.00000	1.00000	0.000000	1.000000
max	1.000000	1.000000	1.000000	20.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.0000	1.000000	1.00000	1.00000	1.000000	1.000000

Machine Learning Models

As the input URL is classified as phishing (1) or legitimate (0), this data set comes under classification problem.

Used machine learning models (classification) to train the dataset:

- Decision Tree
- Random Forest
- Multilayer Perceptrons
- XGBoost
- Autoencoder Neural Network
- Support Vector Machine

Models Performance

- The models are evaluated, and the considered metric is accuracy.

```
#creating dataframe
results = pd.DataFrame({ 'ML Model': ML_Model,
    'Train Accuracy': acc_train,
    'Test Accuracy': acc_test})
results
```

	ML Model	Train Accuracy	Test Accuracy
0	Decision Tree	0.839	0.834
1	Random Forest	0.842	0.832
2	Multilayer Perceptrons	0.867	0.850
3	XGBoost	0.869	0.852
4	AutoEncoder	0.002	0.001
5	SVM	0.801	0.806

- XGBoost model gives better performance

Working on this project is very knowledgeable and worth the effort. Through this project, one learned a lot about the phishing websites and how they are differentiated from legitimate ones.

Questions?