1. DATA PREPARATION

January 23, 2026

0.0.1 Phishing URL Website detection using Machine Learning

Problem Statement The problem is to detect whether a URL is phishing or legitimate using Machine Learning. The phishing URLs looks like legitimate but redirects to a malicious website. So, our goal is to build a detection system that detects any type of URL using Machine Learning with low latency.

URLs are taken from different data sources and combined together. By carefully observing those URLs, the common patterns I found are - Some of the Phishing URLs have HTTP protocol. This may be due to the ease of setup and less secure. But most of the Phishing Websites have HTTPS protocol overcoming the security. - Rarely FTP protocol is used to download malicious content by clicking on the URL. - Many of the Phishing URLs use IP address as the domain name. This sometimes make the users harder to identify the site as fradulent. - Phishing URLs length is very larger. Many characters are present in Path, Domain and Query. - Phishing URLs embed hexadecimal characters to escape detection by security.

[1]:
```
import pandas as pd
import numpy as np
import warnings

warnings.filterwarnings('ignore')
```

[2]:
```
data = pd.read_csv('data/raw/raw_data.csv')

data.head()
```

[2]:

	url	label
0	https://www.visitcanada.com	legitimate
1	http://218.228.19.9/~yossi/9ssfpkz	phishing
2	https://www.msupress.msu.edu/series.php?series…	legitimate
3	https://docs.google.com/presentation/d/e/2PACX…	phishing
4	https://www.c250.columbia.edu/c250_celebrates/…	legitimate

[3]:
```
print(f'The dataset consists of {data.shape[0]} rows and {data.shape[1]}␣
↪columns')
```

```
The dataset consists of 253098 rows and 2 columns
```

[4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 253098 entries, 0 to 253097
Data columns (total 2 columns):
 #   Column  Non-Null Count   Dtype 
---  ------  --------------   ----- 
 0   url     253098 non-null  object
 1   label   253098 non-null  object
dtypes: object(2)
memory usage: 3.9+ MB
```

[5]:
```
data.describe()
```

[5]:

	url	label
count	253098	253098
unique	253098	2
top	https://www.visitcanada.com	legitimate
freq	1	129420

Cleaning the URLs

[6]:
```
import re

def clean_url(url):

    # Remove zero-width characters
    zero_width = r'[\u200B\u200C\u200D\u2060\uFEFF]'
    url = re.sub(zero_width, '', url)

    # Remove all ASCII control characters (0-31 and 127)
    url = re.sub(r'[\x00-\x1F\x7F]', '', url)

    # Remove non-printable Unicode control symbols
    unicode_controls = r'[\u202A-\u202E\u2066-\u2069]'
    url = re.sub(unicode_controls, '', url)

    # Remove soft hyphens & BOM issues
    url = url.replace('\u00AD', '').replace('\ufeff', '')

    # Normalize accidental triple slashes
    url = re.sub(r':\/\/\/+', '://', url)

    # Fix missing colon after https (caused by invisible chars earlier)
    url = re.sub(r'^(https?)(\/\/)', r'\1://', url)

    # Remove spaces around colon in scheme (malformed URLs)
    url = re.sub(r'^(https?)\s*:\s*\/\/', r'\1://', url)

    # Remove trailing slashes
```

```
    url = url.rstrip('/')

    return url

data['url'] = data['url'].apply(clean_url)
```

Checking null & duplicate values

[7]:
```
data.isnull().sum()
```

[7]:
```
url      0
label    0
dtype: int64
```

[8]:
```
data.duplicated().sum()
```

[8]:
```
np.int64(0)
```

Validating URLs

[9]:
```
url_pattern = r"^(([^:/?#]+):)?(//([^/?#]*))?([^?#]*)(\?([^#]*))?(#(.*))?"

print(sum(data['url'].str.match(url_pattern)))
```

253098

[10]:
```
data.to_csv('data/transformed/cleaned_raw_data.csv',index=False)
```

On carefully observing the data, I found that dataset contains, many shorten URLs. Some shorten URLs may work and some may not. The working shortened URLs are expanded by using a cache of shortening sub-domains and async functions.

[11]:
```
# dataset consists of unshorten URLs
df = pd.read_csv('data/raw/expanded_urls.csv')
df.head()
```

[11]:

	url	label	expanded_url
0	https://www.visitcanada.com	legitimate	https://www.visitcanada.com
1	http://218.228.19.9/~yossi/9ssfpkz	phishing	http://218.228.19.9/~yossi/9ssfpkz
2	https://www.msupress.msu.edu/series.php?series…	legitimate	https://www.msupress.msu.edu/series.php?series…
3	https://docs.google.com/presentation/d/e/2PACX…	phishing	https://docs.google.com/presentation/d/e/2PACX…
4	https://www.c250.columbia.edu/c250_celebrates/…	legitimate	https://www.c250.columbia.edu/c250_celebrates/…

[12]:
```
sum(df['url'] != df['expanded_url'])
```

[12]:
```
2
```

As many of the shortened URLs are not working properly, only some shortened URLs are expanded.

[13]:
```
def replace_shortened_url(url,expanded_url):
    if url == expanded_url:
        return url
    return expanded_url

data['url'] = df.apply(lambda row:␣
 ↪replace_shortened_url(row['url'],row['expanded_url']),axis=1)
```

[14]:
```
data.head()
```

[14]:

	url	label
0	https://www.visitcanada.com	legitimate
1	http://218.228.19.9/~yossi/9ssfpkz	phishing
2	https://www.msupress.msu.edu/series.php?series…	legitimate
3	https://docs.google.com/presentation/d/e/2PACX…	phishing
4	https://www.c250.columbia.edu/c250_celebrates/…	legitimate

[15]:
```
data.isnull().sum()
```

[15]:
```
url      0
label    0
dtype: int64
```

[16]:
```
data.duplicated(subset=['url']).sum()
```

[16]:
```
np.int64(0)
```

[17]:
```
data[data.duplicated(subset=['url'])]
```

[17]:
```
Empty DataFrame
Columns: [url, label]
Index: []
```

[18]:
```
data.drop_duplicates(subset=['url'],inplace=True,ignore_index=True)
```

[19]:
```
data.duplicated().sum()
```

[19]:
```
np.int64(0)
```

[20]:
```
data.shape
```

[20]:
```
(253098, 2)
```

Extracting URL Components : These elements represent the structural building blocks of a URL and are useful for detailed analysis & feature engineering. - Protocol/Scheme of the URL - Domain of the URL - Path of the URL - Query of the URL - TLD of the URL - SLD of the URL - Path of the URL - Query of the URL

Consider the below URL as an example: https://www.google.com/search/name=phishing_url

- Scheme/Protocol: `https`
- Sub-domain: `www`
- Domain name: `google.com`
- Second-Level Domain: `google`
- Top-Level Domain: `com`
- Path: `search`
- Query: `name=phishing_url`

[21]:
```
url_components_df = data[['url','label']]
```

[22]:
```
# Extracting protocol, domain, path, query, sld, tld & subdomain
from urllib.parse import urlsplit
import tldextract

def parse_url(url):
    try:
        result = urlsplit(url)
        protocol = result.scheme
        domain   = result.netloc
        path     = result.path
        query    = result.query
        ext = tldextract.extract(domain)
        subdomain = ext.subdomain
        sld = ext.domain
        tld = ext.suffix
    except Exception:   # Handles IPv6 Address
        protocol = ""
        domain = ""
        path = ""
        query = ""
        sld = ""
        tld = ""
        subdomain = ""

    return protocol, domain, subdomain, tld, sld, path, query

url_components_df[['protocol','domain','subdomain','tld','sld','path','query']]␣
↪= data['url'].apply(parse_url).apply(pd.Series)
```

[23]:
```
# Handling IP-based URLs
```

```
url_components_df[url_components_df['domain'].str.fullmatch(r'\d{1,3}(\.
↪\d{1,3}){3}')]
```

[23]:

```
                                                    url     label protocol  \
1                      http://218.228.19.9/~yossi/9ssfpkz  phishing     http
249                          http://72.230.82.80/ase5.png  phishing     http
304                               http://185.102.136.127  phishing     http
548                  http://185.75.46.138/information.cgi  phishing     http
588     http://93.171.202.34/module/272a5ad4a1b97a2ac8…  phishing     http
…                                                    …       …      …
252343  http://46.183.221.133/module/a104f2955999a2f1a…  phishing     http
252653  http://5.9.219.160/~rajkumar/webapps/72e63/websrc  phishing     http
252844                    http://78.157.227.34/weds12.pdf  phishing     http
252950                         http://185.66.10.57/upd/4  phishing     http
252969              http://61.221.169.31/images/kongj.jpg  phishing     http

                 domain subdomain tld              sld  \
1          218.228.19.9                    218.228.19.9
249        72.230.82.80                    72.230.82.80
304     185.102.136.127                 185.102.136.127
548       185.75.46.138                   185.75.46.138
588       93.171.202.34                   93.171.202.34
…                   …       …  ..                  …
252343   46.183.221.133                  46.183.221.133
252653      5.9.219.160                     5.9.219.160
252844    78.157.227.34                   78.157.227.34
252950     185.66.10.57                    185.66.10.57
252969    61.221.169.31                   61.221.169.31

                                        path query
1                                /~yossi/9ssfpkz
249                                    /ase5.png
304
548                              /information.cgi
588     /module/272a5ad4a1b97a2ac874d6d3e5fff01d
…                                             …   …
252343  /module/a104f2955999a2f1a1c881e8930b82f6
252653           /~rajkumar/webapps/72e63/websrc
252844                                /weds12.pdf
252950                                    /upd/4
252969                           /images/kongj.jpg

[1629 rows x 9 columns]
```

[24]:

```
ip_idx = url_components_df[url_components_df['sld'].str.fullmatch(r'\d{1,3}(\.
↪\d{1,3}){3}')].index
```

```
url_components_df['is_ip'] = False
url_components_df.iloc[ip_idx,-1] = True
url_components_df.iloc[ip_idx,[3,6]] = ''

url_components_df.iloc[ip_idx]
```

[24]:
```
                                        url     label protocol domain  \
1          http://218.228.19.9/~yossi/9ssfpkz  phishing     http
38                   http://91.239.24.133:6892  phishing     http
249               http://72.230.82.80/ase5.png  phishing     http
304                    http://185.102.136.127  phishing     http
455     http://208.75.241.246:443/msearch.php  phishing     http
…                                         …       …      …    …
252844        http://78.157.227.34/weds12.pdf  phishing     http
252950              http://185.66.10.57/upd/4  phishing     http
252966    http://115.29.165.174:25663/s-3.rar  phishing     http
252969  http://61.221.169.31/images/kongj.jpg  phishing     http
253094              http://91.239.24.216:6892  phishing     http

       subdomain tld sld               path query  is_ip
1                            /~yossi/9ssfpkz         True
38                                           True
249                               /ase5.png         True
304                                          True
455                             /msearch.php         True
…          …  ..  ..                    …   …    …
252844                           /weds12.pdf         True
252950                                /upd/4         True
252966                               /s-3.rar         True
252969                     /images/kongj.jpg         True
253094                                       True

[2283 rows x 10 columns]
```

[25]:
```
url_components_df.head()
```

[25]:
```
                                                 url       label protocol  \
0                        https://www.visitcanada.com  legitimate    https
1                  http://218.228.19.9/~yossi/9ssfpkz    phishing     http
2  https://www.msupress.msu.edu/series.php?series…  legitimate    https
3  https://docs.google.com/presentation/d/e/2PACX…    phishing    https
4  https://www.c250.columbia.edu/c250_celebrates/…  legitimate    https

                  domain     subdomain  tld          sld  \
0    www.visitcanada.com           www  com  visitcanada
1
2   www.msupress.msu.edu  www.msupress  edu          msu
```

```
3        docs.google.com          docs  com       google
4  www.c250.columbia.edu      www.c250  edu     columbia

                                                  path  \
0
1                                        /~yossi/9ssfpkz
2                                           /series.php
3  /presentation/d/e/2PACX-1vRBjV4Bm4UxL3gJ8sCyQx…
4  /c250_celebrates/athletics/athletics_timeline…

                                           query  is_ip
0                                                 False
1                                                  True
2                                      seriesID=17  False
3  start=false&amp;loop=false&amp;delayms=3000  False
4                                                 False
```

URL Length Features : These metrics help capture patterns that may be indicative of URL behaviour or intent. - Length of the URL - Length of the Domain - Length of the Path - Length of the Query - URL Depth (no. of '/' segments in path) - No. of Subdomains

[26]:
```
url_len_features_df = data[['url','label']]
```

[27]:
```
# Length of the URL
url_len_features_df['url_len'] = data['url'].str.len()
```

[28]:
```
# Length of the Domain
url_len_features_df['domain_len'] = url_components_df['domain'].str.len()
```

[29]:
```
# Length of the Path
def path_length(path):
    if not isinstance(path,str) or not path.strip():
        return 0

    if path.startswith('/'):
        path = path[1:]

    segments = [seg for seg in path.split('/') if seg]

    return sum(len(seg) for seg in segments)

url_len_features_df['path_len'] = url_components_df['path'].apply(path_length)
```

[30]:
```
# Length of the Query
url_len_features_df['query_len'] = url_components_df['query'].str.len()
```

[31]:
```
# Depth of the URL
protocol_re = re.compile(r'^[a-zA-Z][a-zA-Z0-9+.-]*://)?')
```

```
def calculate_url_depth(url):
    url = protocol_re.sub('',url)
    parts = url.split('/',1)

    if len(parts) == 1:
        return 0
    path = parts[1]

    depth = sum(1 for seg in path.split('/') if seg)

    return depth

url_len_features_df['url_depth'] = data['url'].apply(calculate_url_depth)
```

[32]:
```
# Subdomain count
url_len_features_df['subdomain_count'] = url_components_df['subdomain'].str.
 ↪split('.').apply(lambda x: len(x))
url_len_features_df.loc[ip_idx,'subdomain_count'] = 0   # IP-based URLs
```

[33]:
```
url_len_features_df.head()
```

[33]:

	url	label	url_len	domain_len	path_len	query_len	url_depth	subdomain_count
0	https://www.visitcanada.com	legitimate	27	19	0	0	0	1
1	http://218.228.19.9/~yossi/9ssfpkz	phishing	34	0	13	0	2	0
2	https://www.msupress.msu.edu/series.php?series…	legitimate	51	20	10	11	1	2
3	https://docs.google.com/presentation/d/e/2PACX…	phishing	175	15	103	43	5	1
4	https://www.c250.columbia.edu/c250_celebrates/…	legitimate	79	21	47	0	3	2

Domain Structure Features : These features may provide insights into domain structure and potential indicators of unusual or suspicious patterns. - TLD Length - Is the domain a IP address? - Does the URL contains Port number?

[34]:
```
domain_features_df = data[['url','label']]
domain_features_df['tld'] = url_components_df[['tld']]
```

[35]:
```
# TLD Length
domain_features_df['tld_len'] = domain_features_df['tld'].str.len()
```

[36]:
```
# Is the domain a IPv4 address?
domain_features_df['url_has_ipv4'] = url_components_df['is_ip']
```

[37]:
```
# Does the URL contains Port number?
from urllib.parse import urlparse

def url_has_port(url):
    try:
        return urlparse(url).port is not None
    except Exception:
        return False

domain_features_df['url_has_port'] = url_components_df['url'].
 ↪apply(url_has_port)
```

[38]:
```
domain_features_df.head()
```

[38]:

	url	label	tld	\
0	https://www.visitcanada.com	legitimate	com	
1	http://218.228.19.9/~yossi/9ssfpkz	phishing		
2	https://www.msupress.msu.edu/series.php?series…	legitimate	edu	
3	https://docs.google.com/presentation/d/e/2PACX…	phishing	com	
4	https://www.c250.columbia.edu/c250_celebrates/…	legitimate	edu	

	tld_len	url_has_ipv4	url_has_port
0	3	False	False
1	0	True	False
2	3	False	False
3	3	False	False
4	3	False	False

SLD-based Features : These features carries meaningful information about the entity represented by the URL. - Length of SLD - SLD has digit? - SLD has hyphen? - SLD Token count

[39]:
```
sld_features_df = data[['url','label']]
sld_features_df['sld'] = url_components_df[['sld']]
```

[40]:
```
# Length of SLD
sld_features_df['sld_len'] = sld_features_df['sld'].str.len()
```

[41]:
```
# SLD has digit?
sld_features_df['sld_has_digit'] = sld_features_df['sld'].str.contains(r'\d')
```

[42]:
```
# SLD has hyphen?
sld_features_df['sld_has_hyphen'] = sld_features_df['sld'].str.contains('-')
```

[43]:
```
# No of tokens in SLD
sld_features_df['sld_token_count'] = (sld_features_df['sld'].str.split('-')).
 ↪str.len()
```

[44]:
```
sld_features_df.head()
```

[44]:
```
                                                 url       label          sld  \
0                        https://www.visitcanada.com  legitimate  visitcanada
1                  http://218.228.19.9/~yossi/9ssfpkz    phishing
2  https://www.msupress.msu.edu/series.php?series…  legitimate          msu
3  https://docs.google.com/presentation/d/e/2PACX…    phishing       google
4  https://www.c250.columbia.edu/c250_celebrates/…  legitimate     columbia

   sld_len  sld_has_digit  sld_has_hyphen  sld_token_count
0       11          False           False                1
1        0          False           False                1
2        3          False           False                1
3        6          False           False                1
4        8          False           False                1
```

Character Features : These features may help to reveal obfuscation techniques in phishing URLs. - Dot count in Domain - Hyphen count in domain + path - Underscore count in path + query - Slash count in URL - Digit count in URL - Alphabet count in URL

[45]:
```
char_features_df = data[['url','label']]
```

[46]:
```
# Dot count in Domain
char_features_df['dot_count_domain'] = url_components_df['domain'].str.
 ↪count(r'\.')
```

[47]:
```
# Hyphen count in domain + path
char_features_df['hyphen_count_domain_path'] = (url_components_df['domain'] +␣
 ↪url_components_df['path']).str.count('-')
```

[48]:
```
# Underscore count in path + query
char_features_df['underscore_count_path_query'] = (url_components_df['path'] +␣
 ↪url_components_df['query']).str.count('\_')
```

[49]:
```
# Slash count in URL
char_features_df['slash_count'] = data['url'].str.count('/')
```

[50]:
```
# Digit count in URL
char_features_df['digit_count'] = data['url'].str.count(r'\d')
```

[51]:
```
# Alphabet count in URL
char_features_df['alphabet_count'] = data['url'].str.count(r'[a-zA-Z]')
```

[52]:
```
# Special characters count in URL
char_features_df['spl_char_count'] = data['url'].str.count(r'[^a-zA-Z0-9]')
```

[53]:
```
char_features_df.head()
```

[53]:
```
                                                 url       label  \
0                        https://www.visitcanada.com  legitimate
```

```
1                http://218.228.19.9/~yossi/9ssfpkz    phishing
2  https://www.msupress.msu.edu/series.php?series…  legitimate
3  https://docs.google.com/presentation/d/e/2PACX…    phishing
4  https://www.c250.columbia.edu/c250_celebrates/…  legitimate

   dot_count_domain  hyphen_count_domain_path  underscore_count_path_query  \
0                 2                         0                            0
1                 0                         0                            0
2                 3                         0                            0
3                 2                         2                            1
4                 3                         0                            2

   slash_count  digit_count  alphabet_count  spl_char_count
0            2            0              22               5
1            4           10              15               9
2            3            2              39              10
3            7           19             135              21
4            5            6              61              12
```

Entropy features : Higher entropy often indicates obfuscation or automatically generated strings, which may be strong signals in detecting phishing URLs. - Entropy of URL - Entropy of Domain - Entropy of SLD - Entropy of Path

[54]:
```
# Entropy calculation
import math
from collections import Counter

def shannon_entropy(str_):
    counts = Counter(str_)
    length = len(str_)

    return -sum((count / length) * math.log2(count / length) for count in␣
 ↪counts.values())
```

[55]:
```
entropy_feature_df = data[['url','label']]
```

[56]:
```
# Entropy of URL
entropy_feature_df['url_entropy'] = data['url'].apply(shannon_entropy)
```

[57]:
```
# Entropy of Domain
entropy_feature_df['domain_entropy'] = url_components_df['domain'].
 ↪apply(shannon_entropy)
```

[58]:
```
# Entropy of SLD
entropy_feature_df['sld_entropy'] = url_components_df['sld'].
 ↪apply(shannon_entropy)
```

[59]:
```
# Entropy of Path
entropy_feature_df['path_entropy'] = url_components_df['path'].
 ↪apply(shannon_entropy)
```

[60]:
```
entropy_feature_df.head()
```

[60]:

	url	label	url_entropy	\
0	https://www.visitcanada.com	legitimate	3.856196	
1	http://218.228.19.9/~yossi/9ssfpkz	phishing	3.962032	
2	https://www.msupress.msu.edu/series.php?series…	legitimate	3.965393	
3	https://docs.google.com/presentation/d/e/2PACX…	phishing	5.569700	
4	https://www.c250.columbia.edu/c250_celebrates/…	legitimate	4.274946	

	domain_entropy	sld_entropy	path_entropy
0	3.431624	2.845351	0.000000
1	0.000000	0.000000	3.240224
2	3.008695	1.584963	2.913977
3	2.973557	1.918296	5.540696
4	3.748995	3.000000	3.845213

Token based Features : Tokenization helps revealing structural patterns and semantic clues within the URL. - Token count in domain - Token count in path - Token Count in domain + path - Avg Token Length

[61]:
```
token_features_df = data[['url','label']]
```

[62]:
```
# Calculating Token count in path + domain
def count_tokens(domain,path):
    try:
        domain_tokens = domain.replace('.',' ').replace('-',' ').replace('_',' 
 ↪').split()
        path_tokens = path.replace('.',' ').replace('-',' ').replace('_',' ').
 ↪split()

        total_tokens = len(domain_tokens) + len(path_tokens)

        return len(domain_tokens), len(path_tokens), total_tokens

    except Exception:
        return 0,0,0

token_features_df[['domain_token_count','path_token_count','total_tokens']] = 
 ↪url_components_df.apply(lambda row:
 ↪count_tokens(row['domain'],row['path']),axis=1).apply(pd.Series)
```

[63]:
```
# Average token length of the URL
protocol_re = re.compile(r'^[a-zA-Z][a-zA-Z0-9+.-]*://',re.IGNORECASE)
split_re = re.compile(r'[./?=\-_&:%]+')
```

```
def avg_token_length(url):
    url = protocol_re.sub('',url)
    tokens = split_re.split(url)
    tokens = [t for t in tokens if t]

    if not tokens:
        return 0

    return sum(len(t) for t in tokens) / len(tokens)

token_features_df['avg_token_length'] = data['url'].apply(avg_token_length)
```

[64]:
```
token_features_df.head()
```

[64]:

	url	label	domain_token_count	path_token_count	total_tokens	avg_token_length
0	https://www.visitcanada.com	legitimate	3	0	3	5.666667
1	http://218.228.19.9/~yossi/9ssfpkz	phishing	0	1	1	3.666667
2	https://www.msupress.msu.edu/series.php?series…	legitimate	4	2	6	4.500000
3	https://docs.google.com/presentation/d/e/2PACX…	phishing	3	4	7	8.882353
4	https://www.c250.columbia.edu/c250_celebrates/…	legitimate	4	4	8	6.200000

Hexadecimal based Features - URL has Hexadecimal Characters - No. of Hexadecimal Characters - Hexadecimal Ratio

[65]:
```
hex_feature_df = data[['url','label']]
```

[66]:
```
# URL has Hexadecimal Characters?
hex_feature_df['has_hex'] = data['url'].str.contains(r'%[0-9A-Fa-f]{2}')
```

[67]:
```
# No. of hexadecimal characters

def count_hex_chars(url):
    count = 0
    i = 0
    while i < len(url):
        if url[i] == '%' and i+2 < len(url):
            c1, c2 = url[i+1], url[i+2]

            if (c1.isdigit() or c1.lower() in 'abcdef') and (c2.isdigit() or c2.
 ↪lower() in 'abcdef'):
```

```
                count += 2
                i += 3
                continue
        i += 1

    return count

hex_feature_df['hex_char_count'] = data['url'].apply(count_hex_chars)
```

[68]:
```
# Hexadecimal Ratio
hex_feature_df['hex_ratio'] = hex_feature_df['hex_char_count'] /␣
 ↪url_len_features_df['url_len']
```

[69]:
```
hex_feature_df.head()
```

[69]:
```
                                                 url       label  has_hex  \
0                       https://www.visitcanada.com  legitimate    False
1                  http://218.228.19.9/~yossi/9ssfpkz    phishing    False
2  https://www.msupress.msu.edu/series.php?series…  legitimate    False
3  https://docs.google.com/presentation/d/e/2PACX…    phishing    False
4  https://www.c250.columbia.edu/c250_celebrates/…  legitimate    False

   hex_char_count  hex_ratio
0               0        0.0
1               0        0.0
2               0        0.0
3               0        0.0
4               0        0.0
```

[70]:
```
url_components_df.drop(columns=['is_ip'],inplace=True)
```

[71]:
```
# Saving all the extracted features in the corresponding CSV files.

data.to_csv('data/transformed/final_raw_data.csv',index=False)
url_components_df.to_csv('data/transformed/1.url_components_data.
 ↪csv',index=False)
url_len_features_df.to_csv('data/transformed/2.component_len_features_data.
 ↪csv',index=False)
domain_features_df.to_csv('data/transformed/3.domain_features_data.
 ↪csv',index=False)
sld_features_df.to_csv('data/transformed/4.sld_features_data.csv',index=False)
char_features_df.to_csv('data/transformed/5.char_features_data.csv',index=False)
entropy_feature_df.to_csv('data/transformed/6.entropy_features_data.
 ↪csv',index=False)
token_features_df.to_csv("data/transformed/7.token_features_data.
 ↪csv",index=False,encoding="utf-8",float_format="%.6f")
hex_feature_df.to_csv('data/transformed/8.hex_features_data.csv',index=False)
```