

```
In [1]: import pandas as pd
import numpy as np # use for multi-dimensional array and matrix
from sklearn.linear_model import LogisticRegression # algo use to predict good or bad
from sklearn.naive_bayes import MultinomialNB # nlp algo use to predict good or bad

from sklearn.model_selection import train_test_split # splitting the data between fe
from sklearn.metrics import classification_report # gives whole report about metric
from sklearn.metrics import confusion_matrix # gives info about actual and predict
from nltk.tokenize import RegexpTokenizer # regexp tokenizers use to split words fr
from nltk.stem.snowball import SnowballStemmer # stemmes words
from sklearn.feature_extraction.text import CountVectorizer # create sparse matrix
from sklearn.pipeline import make_pipeline # use for combining all prerocessors tec

from PIL import Image # getting images in notebook
# from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator# creates words co

from bs4 import BeautifulSoup # use for scraping the data from website
# use for automation chrome
import networkx as nx # for the creation, manipulation, and study of the structure,

import pickle# use to dump model

import warnings # ignores pink warnings
df = pd.read_csv("malicious_phish.csv")
df.columns = ['URL', 'Label']
df
```

```
Out[1]:
```

	URL	Label
0	br-icloud.com.br	phishing
1	mp3raid.com/music/krizz_kaliko.html	benign
2	bopsecrets.org/rexroth/cr/1.htm	benign
3	http://www.garage-pirenne.be/index.php?option=...	defacement
4	http://adventure-nicaragua.net/index.php?optio...	defacement
...
651186	xbox360.ign.com/objects/850/850402.html	phishing
651187	games.teamxbox.com/xbox-360/1860/Dead-Space/	phishing
651188	www.gamespot.com/xbox360/action/deadspace/	phishing
651189	en.wikipedia.org/wiki/Dead_Space_(video_game)	phishing
651190	www.angelfire.com/goth/devilmaycrytonite/	phishing

651191 rows × 2 columns

```
In [2]: df.describe()
```



```

def abnormal_url(URL):
    hostname = urlparse(URL).hostname
    hostname = str(hostname)
    match = re.search(hostname, URL)
    if match:
        print(match.group())
        return 1
    else:
        print('No matching pattern found')
        return 0

df['abnormal_url'] = df['URL'].apply(lambda i: abnormal_url(i))

from googlesearch import search

def google_index(URL):
    site = search(URL, 5)
    return 1 if site else 0
df['google_index'] = df['URL'].apply(lambda i: google_index(i))

def count_dot(URL):
    count_dot = URL.count('.')
    return count_dot

df['count.'] = df['URL'].apply(lambda i: count_dot(i))

def count_www(URL):
    URL.count('www')
    return URL.count('www')

df['count-www'] = df['URL'].apply(lambda i: count_www(i))

def count_atrate(URL):

    return URL.count('@')

df['count@'] = df['URL'].apply(lambda i: count_atrate(i))

def no_of_dir(URL):
    urldir = urlparse(URL).path
    return urldir.count('/')

df['count_dir'] = df['URL'].apply(lambda i: no_of_dir(i))

def no_of_embed(URL):
    urldir = urlparse(URL).path
    return urldir.count('///')

df['count_embed_domian'] = df['URL'].apply(lambda i: no_of_embed(i))

def shortening_service(URL):
    match = re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tiny
        'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.p
        'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|
        'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.l
        'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.cc
        'q\.gs|is\.gd|po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\
        'x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.cc

```

```

        'tr\\.im|link\\.zip\\.net',
        URL)

    if match:
        return 1
    else:
        return 0

df['short_url'] = df['URL'].apply(lambda i: shortening_service(i))

def count_https(URL):
    return URL.count('https')

df['count-https'] = df['URL'].apply(lambda i : count_https(i))

def count_http(URL):
    return URL.count('http')

df['count-http'] = df['URL'].apply(lambda i : count_http(i))

def count_per(URL):
    return URL.count('%')

df['count%'] = df['URL'].apply(lambda i : count_per(i))

def count_ques(URL):
    return URL.count('?')

df['count?'] = df['URL'].apply(lambda i: count_ques(i))

def count_hyphen(URL):
    return URL.count('-')

df['count-'] = df['URL'].apply(lambda i: count_hyphen(i))

def count_equal(URL):
    return URL.count('=')

df['count='] = df['URL'].apply(lambda i: count_equal(i))

def url_length(URL):
    return len(str(URL))

#Length of URL
df['url_length'] = df['URL'].apply(lambda i: url_length(i))
#Hostname Length

def hostname_length(URL):
    return len(urllib.parse(URL).netloc)

df['hostname_length'] = df['URL'].apply(lambda i: hostname_length(i))

df.head()

def suspicious_words(URL):
    match = re.search('PayPal|login|signin|bank|account|update|free|lucky|service|t
        URL)

    if match:
        return 1
    else:
        return 0
df['sus_url'] = df['URL'].apply(lambda i: suspicious_words(i))

```

```
def digit_count(URL):
    digits = 0
    for i in URL:
        if i.isnumeric():
            digits = digits + 1
    return digits

df['count-digits'] = df['URL'].apply(lambda i: digit_count(i))

def letter_count(URL):
    letters = 0
    for i in URL:
        if i.isalpha():
            letters = letters + 1
    return letters

df['count-letters'] = df['URL'].apply(lambda i: letter_count(i))

# pip install tld

from urllib.parse import urlparse
from tld import get_tld
import os.path

#First Directory Length
def fd_length(URL):
    urlpath = urlparse(URL).path
    try:
        return len(urlpath.split('/')[1])
    except:
        return 0

df['fd_length'] = df['URL'].apply(lambda i: fd_length(i))

#Length of Top Level Domain
df['tld'] = df['URL'].apply(lambda i: get_tld(i, fail_silently=True))

def tld_length(tld):
    try:
        return len(tld)
    except:
        return -1

df['tld_length'] = df['tld'].apply(lambda i: tld_length(i))
```

6/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

[illegible]

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

12/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

16/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

18/8401

19/8401

20/8401

8200/8401

8201/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8203/8401

8204/8401

[illegible]

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8208/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8210/8401

8211/8401

8212/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8216/8401

8217/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8219/8401

8220/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8224/8401

8225/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8227/8401

8228/8401

8229/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8231/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

[illegible]

8236/8401

8237/8401

8238/8401

8239/8401

8240/8401

8241/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

[illegible]

8244/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8246/8401

8247/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8251/8401

8252/8401

8253/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8255/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8257/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8260/8401

8261/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8264/8401

8265/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8267/8401

8268/8401

[illegible]

8270/8401

8271/8401

8272/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8274/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8276/8401

8277/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8279/8401

8280/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8282/8401

8283/8401

8284/8401

8285/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8287/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8289/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8291/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8295/8401

8296/8401

8297/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8299/8401

8300/8401

8301/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8303/8401

8304/8401

8305/8401

8306/8401

8307/8401

8308/8401

8309/8401

8310/8401

8311/8401

8312/8401

8313/8401

8314/8401

8315/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8317/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8320/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8324/8401

8325/8401

8326/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8328/8401

8329/8401

8330/8401

[illegible]

8332/8401

8333/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8335/8401

8336/8401

8337/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8339/8401

8340/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8342/8401

[illegible]

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8348/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8352/8401

8353/8401

8354/8401

8355/8401

8356/8401

8357/8401

8358/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8360/8401

8361/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8363/8401

8364/8401

8365/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8373/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8376/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8379/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8381/8401

8382/8401

8383/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8385/8401

8386/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8389/8401

8390/8401

8391/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8393/8401

8394/8401

localhost:8888/nbconvert/html/pranav ml.ipynb?download=false

8396/8401

8397/8401

8398/8401

No matching pattern found

No matching pattern found

IOPub data rate exceeded.

The notebook server will temporarily stop sending output to the client in order to avoid crashing it.

To change this limit, set the config variable

`--NotebookApp.iopub_data_rate_limit`.

Current values:

NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)

NotebookApp.rate_limit_window=3.0 (secs)

In [7]: `print(df.columns)`

```
Index(['URL', 'Label', 'use_of_ip', 'abnormal_url', 'google_index', 'count.',
      'count-www', 'count@', 'count_dir', 'count_embed_domian', 'short_url',
      'count-https', 'count-http', 'count%', 'count?', 'count-', 'count=',
      'url_length', 'hostname_length', 'sus_url', 'count-digits',
      'count-letters', 'fd_length', 'tld', 'tld_length'],
      dtype='object')
```

In [8]: `df.head()`

Out[8]:

	URL	Label	use_of_ip	abnormal_url	google_index	count.
0	br-icloud.com.br	phishing	0	0	1	2
1	mp3raid.com/music/krizz_kaliko.html	benign	0	0	1	2
2	bopsecrets.org/rexroth/cr/1.htm	benign	0	0	1	2
3	http://www.garage-pirenne.be/index.php?option=...	defacement	0	1	1	3
4	http://adventure-nicaragua.net/index.php?optio...	defacement	0	1	1	2

5 rows × 7 columns

In [9]: `from sklearn.preprocessing import LabelEncoder`

```
lb_make = LabelEncoder()
df["type_code"] = lb_make.fit_transform(df["Label"])
```

In [11]: `#Predictor Variables`
`# filtering out google_index as it has only 1 value`
`X = df[['use_of_ip', 'abnormal_url', 'count.', 'count-www', 'count@',`
 `'count_dir', 'count_embed_domian', 'short_url', 'count-https',`
 `'count-http', 'count%', 'count?', 'count-', 'count=', 'url_length',`
 `'hostname_length', 'sus_url', 'fd_length', 'tld_length', 'count-digits',`
 `'count-letters']]`
`#Target Variable`
`y = df['type_code']`

In [12]: `X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0.2)`

In [13]: `# Random Forest Model`
`from sklearn.ensemble import RandomForestClassifier`

```

from sklearn import metrics
rf = RandomForestClassifier(n_estimators=100,max_features='sqrt')
rf.fit(X_train,y_train)
y_pred_rf = rf.predict(X_test)
print(classification_report(y_test,y_pred_rf,target_names=['benign', 'defacement', 'phishing', 'malware'])

score = metrics.accuracy_score(y_test, y_pred_rf)
print("accuracy:  %0.3f" % score)

```

	precision	recall	f1-score	support
benign	0.97	0.98	0.98	85621
defacement	0.98	0.99	0.99	19292
phishing	0.99	0.95	0.97	6504
malware	0.91	0.86	0.88	18822
accuracy			0.97	130239
macro avg	0.96	0.95	0.95	130239
weighted avg	0.97	0.97	0.97	130239

accuracy: 0.966

```

In [14]: import pickle
pickle_out = open("rf.pkl","wb")
pickle.dump(rf, pickle_out)
pickle_out.close()

```

```

In [18]: def main(URL):

    status = []

    status.append(having_ip_address(URL))
    status.append(abnormal_url(URL))
    status.append(count_dot(URL))
    status.append(count_www(URL))
    status.append(count_atrate(URL))
    status.append(no_of_dir(URL))
    status.append(no_of_embed(URL))

    status.append(shortening_service(URL))
    status.append(count_https(URL))
    status.append(count_http(URL))

    status.append(count_per(URL))
    status.append(count_ques(URL))
    status.append(count_hyphen(URL))
    status.append(count_equal(URL))

    status.append(url_length(URL))
    status.append(hostname_length(URL))
    status.append(suspicious_words(URL))
    status.append(digit_count(URL))
    status.append(letter_count(URL))
    status.append(fd_length(URL))
    tld = get_tld(URL,fail_silently=True)

    status.append(tld_length(tld))

    return status

# predict function
def get_prediction_from_url(test_url):
    features_test = main(test_url)
    # Due to updates to scikit-learn, we now need a 2D array as a parameter to the

```

```

features_test = np.array(features_test).reshape((1, -1))
pred = rf.predict(features_test)
if int(pred[0]) == 0:

    res="SAFE"
    return res
elif int(pred[0]) == 1.0:

    res="DEFAACEMENT"
    return res
elif int(pred[0]) == 2.0:
    res="PHISHING"
    return res

elif int(pred[0]) == 3.0:

    res="MALWARE"
    return res

```

predicting sample raw URLs

```
urls = ['kekma.net', 'www.bmsit.ac.in']
```

```

for URL in urls:
    print(get_prediction_from_url(URL))

```

```

No matching pattern found
No matching pattern found
MALWARE
No matching pattern found
No matching pattern found
SAFE

```

C:\Users\npran\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names

```
warnings.warn(
```

C:\Users\npran\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names

```
warnings.warn(
```

In []:

In []:

In []:

In []:

In []:

In []:

In []:

In []:

In []: