Project: Phishing Prevention for Twitter: Machine Learning Approach

In [24]:
```
import os                       # OS system library
from os.path import splitext  # OS library package to split URL into parts
import numpy as np              # numpy library for storing multi dimensional ar
rays
import matplotlib               # libary to plot graphs and visualize
import pandas as pd             # data set manipulation library
import ipaddress as ip          # library to detect IP address in URL
import tldextract               # library to separate TLD from domains in URL
try:                            # URL library to parse into string
    from urllib.parse import urlparse
except ImportError:
     from urlparse import urlparse
```

In [25]:
```
dataset = pd.read_csv("dataset1.csv")   # read the dataset file
dataset = dataset.sample(frac=1).reset_index(drop=True) # 10 percent random sa
mpling from dataset, with no index in columns
# dataset.head(n=10) # display first N rows
```

In [26]:
```
Suspicious_TLD=['zip','cricket','link','work','party','gq','kim','country','sc
ience','tk']
Suspicious_Domain=['luckytime.co.kr','mattfoll.eu.interia.pl','trafficholder.c
om','dl.baixaki.com.br','bembed.redtube.comr','tags.expo9.exponential.com','de
epspacer.com','funad.co.kr','trafficconverter.biz']
```

In [27]:
```
# Method to count number of delimeters
def countdelim(url):
    count = 0
    delim=[';','_','?','=','&']
    for each in url:
        if each in delim:
            count = count + 1
    return count
# Method to count number of dots
def countdots(url):
    return url.count('.')
def isip(uri):
    try:
        if ip.ip_address(uri):
            return 1
    except:
        return 0
#method to check the presence of hyphens
def isPresentHyphen(url):
    return url.count('-')

#method to check the presence of @

def isPresentAt(url):
    return url.count('@')

def isPresentDSlash(url):
    return url.count('//')

def countSubDir(url):
    return url.count('/')

def get_ext(url):
    """Return the filename extension from url, or ''."""

    root, ext = splitext(url)
    return ext

def countSubDomain(subdomain):
    if not subdomain:
        return 0
    else:
        return len(subdomain.split('.'))

def countQueries(query):
    if not query:
        return 0
    else:
        return len(query.split('&'))
```

In [28]:
```
featureSet = pd.DataFrame(columns=('url','dots count','hyphen present','url le
ngth','at present',\
'double slash present','subdir count','subdomain count','domain length','queri
es count','IP present','suspicious TLD',\
'suspicious domain','label'))
```

In [29]:
```
def getFeatures(url, label):
    result = []
    url = str(url)

    #add the url to feature set
    result.append(url)

    #parse the URL and extract the domain information
    path = urlparse(url)
    ext = tldextract.extract(url)

    #counting number of dots in subdomain
    result.append(countdots(ext.subdomain))

    #checking hyphen in domain
    result.append(isPresentHyphen(path.netloc))

    #length of URL
    result.append(len(url))

    #checking @ in the url
    result.append(isPresentAt(path.netloc))

    #checking presence of double slash
    result.append(isPresentDSlash(path.path))

    #Count number of subdir
    result.append(countSubDir(path.path))

    #number of sub domain
    result.append(countSubDomain(ext.subdomain))

    #length of domain name
    result.append(len(path.netloc))

    #count number of queries
    result.append(len(path.query))

    #Adding domain information

    #if IP address is being used as a URL
    result.append(isip(ext.domain))

    #presence of Suspicious_TLD
    result.append(1 if ext.suffix in Suspicious_TLD else 0)

    #presence of suspicious domain
    result.append(1 if '.'.join(ext[1:]) in Suspicious_Domain else 0 )

    result.append(str(label))

    return result
```

In [30]:
```
for i in range(len(dataset)):
    features = getFeatures(dataset["URL"].loc[i], dataset["Lable"].loc[i])
    featureSet.loc[i] = features
```

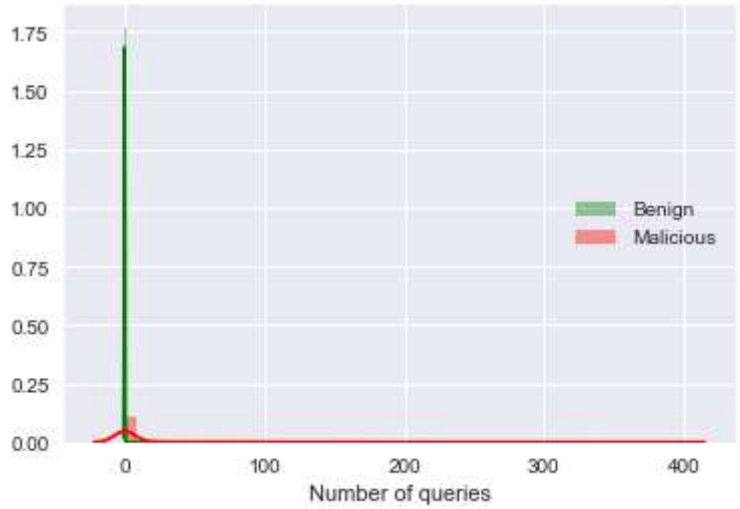
In [31]:
```
featureSet.head(n=5)
```

Out[31]:

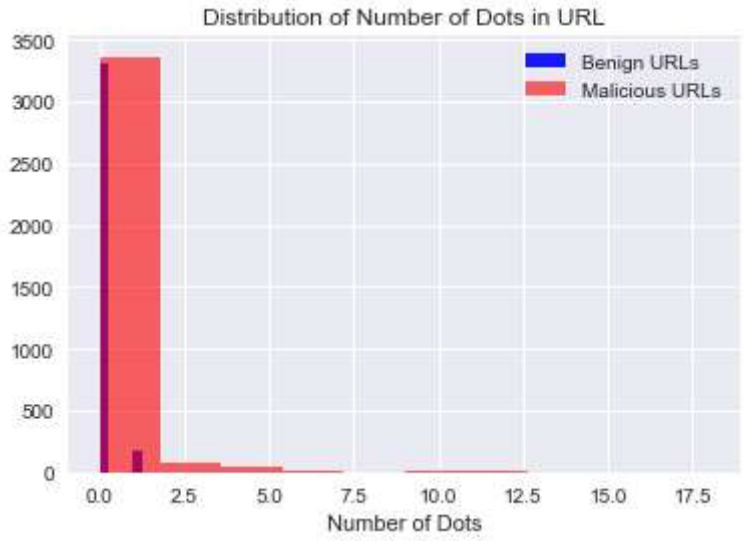
	url	dots count	hyphen present	url length	at present	double slash present	subdir count
0	http://tomxyz.han-solo.net/error_docs/forbidde...	0.0	1.0	51.0	0.0	0.0	2.0
1	http://www.beautyatbelladonna.com.au/irs/confi...	0.0	0.0	59.0	0.0	0.0	3.0
2	http://www.etaeng.com.br/js/SistemaAutorizado/...	0.0	0.0	70.0	0.0	0.0	4.0
3	http://www.mgnet.org/	0.0	0.0	21.0	0.0	0.0	1.0
4	http://livingstonefellowship.co.za/update-payp...	0.0	0.0	72.0	0.0	0.0	4.0

In [32]:
```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import pickle as pkl
from __future__ import division
```

In [33]:
```
sns.set(style="darkgrid")
sns.distplot(featureSet[featureSet['label']=='0']['queries count'],color='green',label='Benign')
sns.distplot(featureSet[featureSet['label']=='1']['queries count'],color='red',label='Malicious')
plt.legend(loc='center right')
plt.xlabel('Number of queries')
plt.show()
```



In [34]:
```
x=featureSet[featureSet['label']=='0']['dots count']
y=featureSet[featureSet['label']=='1']['dots count']
plt.hist(x,bins=8, alpha=0.9, label='Benign URLs',color='blue')
#sns.distplot(x,bins=8,color='blue',label='Benign URLs')
plt.hist(y,bins=10, alpha=0.6, label='Malicious URLs',color='red')
#sns.distplot(y,bins=8,color='red',label='Malicious URLs')
plt.legend(loc='upper right')
plt.xlabel('Number of Dots')
plt.title('Distribution of Number of Dots in URL')
plt.show()
```



In [35]:
```
import sklearn.ensemble as ek
from sklearn import cross_validation, tree, linear_model
from sklearn.feature_selection import SelectFromModel
from sklearn.externals import joblib
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
from sklearn.pipeline import make_pipeline
from sklearn import preprocessing
from sklearn import svm
from sklearn.linear_model import LogisticRegression
```

In [36]:
```
featureSet.groupby(featureSet['label']).size()
```

Out[36]:
```
label
0    3494
1    3536
dtype: int64
```

In [37]:
```
X = featureSet.drop(['url','label'],axis=1).values
y = featureSet['label'].values
```

In [38]:
```
model = { "DecisionTree":tree.DecisionTreeClassifier(max_depth=10),
          "Adaboost":ek.AdaBoostClassifier(n_estimators=50),
          "LogisticRegression":LogisticRegression()
}
```

In [39]:
```
X_train, X_test, y_train, y_test = cross_validation.train_test_split(X, y ,tes
t_size=0.2)
```

In [40]:
```
results = {}
for algo in model:
    clf = model[algo]
    clf.fit(X_train,y_train)
    score = clf.score(X_test,y_test)
    print ("%s : %s " %(algo, score))
    results[algo] = score
```

```
DecisionTree : 0.894736842105
LogisticRegression : 0.842816500711
Adaboost : 0.871977240398
```

In [41]:
```
winner = max(results, key=results.get)
print(winner)
# winner = "DecisionTree"
```

```
DecisionTree
```

In [42]:
```
clf = model[winner]
res = clf.predict(X)
mt = confusion_matrix(y, res)
print("False positive rate : %f %%" % ((mt[0][1] / float(sum(mt[0])))*100))
print('False negative rate : %f %%' % ( (mt[1][0] / float(sum(mt[1])))*100))
```

```
False positive rate : 6.611334 %
False negative rate : 7.494344 %
```

Test

In [43]:
```
result = pd.DataFrame(columns=('url','no of dots','presence of hyphen','len of
url','presence of at',\
'presence of double slash','no of subdir','no of subdomain','len of domain','n
o of queries','is IP','presence of Suspicious_TLD',\
'presence of suspicious domain','label'))

results = getFeatures('http://tdngrdgdc.blogspot.ru', '0')
# results = getFeatures('http://www.google.com/', '1')
result.loc[0] = results
result = result.drop(['url','label'],axis=1).values

print(clf.predict(result))
```

```
['1']
```

In [44]:
```
import json
import codecs
import os
import shutil
import requests
from collections import Counter


global depth
global dir_id
global counter
global counter_blacklist
global url_link
global url_profile
global total_list
global google_status
global phishtank_status
global urlblacklist_status

url_profile=""
url_link=""
total_list=[]
depth=0
counter=0
counter_blacklist=0
dir_id=""
google_status = False
phishtank_status = False
urlblacklist_status = False

def init_auto():
    global dir_id
    global depth
    global counter
    global counter_blacklist
    global url_profile
    global google_status
    global phishtank_status
    global urlblacklist_status

    while depth<3:
        dir_depth = 'E:/Twitter/Depth-%d' % depth
        list_dir = os.listdir(dir_depth)

        for doc in list_dir:
            ## Initialize/Reset the condition flags.
            google_status = False
            phishtank_status = False
            urlblacklist_status = False
            dir_count()
            counter +=1
            print "User id    : ",str(doc)

            dir_id = '%s/%s' % (dir_depth,str(doc))
            os.system('cls' if os.name == 'nt' else 'clear')
```

```
            xtract_url()

            urlList = url_link.split('\n')

            for subURL in urlList:
                result = pd.DataFrame(columns=('url','no of dots','presence of
hyphen','len of url','presence of at',\
                'presence of double slash','no of subdir','no of subdomain','l
en of domain','no of queries','is IP','presence of Suspicious_TLD',\
                'presence of suspicious domain','label'))

                results = getFeatures(str(subURL), str(1))
                print "URL: ", subURL
                result.loc[0] = results
                result = result.drop(['url','label'],axis=1).values
#                 print(clf.predict(result))
                out = int(str(clf.predict(result))[2])
                if out:
                    print "*** MALICIOUS ***"
                else: print "*** BENIGN ***"

                print "----------------------------------------------------------
---------"

        depth +=1

def xtract_url():
    global url_link
    global total_list
    total_list = []
    url_link=""
    try:
        url_count=0
        c=0
        dir_tweet = '%s/tweets.dump' % dir_id
        jfile2 = codecs.open(dir_tweet,'rb','utf-8')
        print("Extracting URL from tweets.... ")
        print("Resolving URL.....")
        for jdoc in jfile2:
            jobj = json.loads(jdoc)
            list_url = jobj['entities']['urls']
            for url in list_url:

                url = str(url['expanded_url'])
                total_list.append(url)
                url_link = url_link + url+"\n"
                url_count+=1

            if url_count> 490:
                break

        #url_link = "%d\n%s" %(url_count,url_link)
        jfile2.close()
        #print("URL Count :",url_count)
        #print("")
    except FileNotFoundError as e:
        print(e, " in extract tweets")
```

```
def dir_count():
    global counter_blacklist

    counter_blacklist = len(os.listdir("E:/Classify/Blacklist/"))

def decision(user):
    if (google_status is True) or (phishtank_status is True) or (urlblacklist_
status is True):
        list_l = os.listdir(dir_id)
        dst_dir = "E:/Classify/Blacklist/%s" % user
        os.makedirs(dst_dir)
        for file in list_l:
            src_f = "%s/%s" %(dir_id,file)
            dst_f = "%s/%s" %(dst_dir,file)
            shutil.move(src_f, dst_f)
        os.rmdir(dir_id)
        return True

init_auto()
```

```
User id    :  86953262
Extracting URL from tweets....
Resolving URL......
URL:  http://bing.com
*** BENIGN ***
------------------------------------------------------------------
URL:  http://rutgers.com
*** MALICIOUS ***
------------------------------------------------------------------
URL:  http://facebook.com
*** BENIGN ***
------------------------------------------------------------------
URL:  http://twitter.com
*** MALICIOUS ***
------------------------------------------------------------------
URL:  http://tdngrdgdc.blogspot.ru
*** MALICIOUS ***
------------------------------------------------------------------
URL:  http://linkedin.com
*** BENIGN ***
------------------------------------------------------------------
URL:  http://wikipedia.com
*** BENIGN ***
------------------------------------------------------------------
URL:  http://google.com/wikipedia
*** MALICIOUS ***
------------------------------------------------------------------
URL:  http://davaoblog.com/id/onlin/nn.nz/index.htm
*** MALICIOUS ***
------------------------------------------------------------------
URL:
*** MALICIOUS ***
------------------------------------------------------------------
```

In [ ]: