



School of Computer Science and Engineering

Phishing Website Detection using Machine Learning Algorithms

J component Report

CSE3013 Artificial Intelligence

Slot: C1+TC1

Divyanshu Gupta(19BCE2093)

B. Tech Computer Science and Engineering

Under Guidance of:
Arun Kumar G
Associate Professor, SCOPE
VIT

Abstract

Aim

To detect phishing URLs as well as narrow down to best machine learning algorithm by comparing accuracy rate, false positive and false negative rate of each algorithm.

Objective

- To overcome the drawbacks of blacklist and heuristics-based method
- Focus on machine learning techniques.

Motivation

- Phishing becomes a main area of concern for security researchers because it is not difficult to create the fake website which looks so close to legitimate website. Main aim of the attacker is to steal banks account credentials.
- Phishing attacks are becoming successful because lack of user awareness.
- The general method to detect phishing websites by updating blacklisted URLs, Internet Protocol (IP) to the antivirus database which is also known as "blacklist" method.
- To overcome the drawbacks of blacklist and heuristics-based method, many security researchers now focused on machine learning techniques.
- Using this technique, algorithm will analyze various blacklisted and legitimate URLs and their features to accurately detect the phishing websites including zero- hour phishing websites.

Introduction

In this digital day and electronic world, Internet plays a vital role in day-to-day activities like communication, business, transactions, personal needs, marketing, e-commerce etc. Internet is a multifaceted facility which helps in completing many tasks readily and conveniently within few seconds. Almost everything is presently accessible over web in this period of progression of advances. Thus, increasing usage of internet leads to cybercrime and other malware activities. The information divulged in online leaves digital imprint and if it happens to drop into the wrong hands, it will result in data theft, identity theft and monetary loss. Cybercrime includes many kinds of security issues over the internet and one of the most threatening problems is Phishing. Phishing is a fraudulent technique achieved by phishing web page. Phishing uses e-mails and websites, which are intended to look like from trusted organization, to hoodwink clients into unveiling their own or money related data. The threatening party then use these data for criminal purposes, such as, identity or data theft and extortion. Clients are deceived into revealing their data either by giving touchy data through a web shape or downloading and introducing unfriendly codes, which seek clients' PCs or checking clients' online actions to get data. Luring Internet users by making them click on rogue links that seem trustworthy is an easy task because of widespread credulity and unawareness. It is important to prevent user's confidential data from unauthorized access. The procedure for the most part includes sending messages that then cause the beneficiary to either visit a deceitful site and enter their data or to visit an authentic site through a phishing intermediary attack or using spoofed website, which then gathers the details of user leads to several loss. The Phishing problem needs to be mitigated by anti-Phishing approaches. This research provides a solution that helps in detecting and preventing Phishing attacks using the features of phishing URLs and an automated real-time detection of phishing websites by machine learning approach.

Advantages and Disadvantages of the previous used methods

Blacklists: Blacklists hold URLs (or parts thereof) that refer to sites that are considered malicious. Whenever a browser loads a page, it queries the blacklist to determine whether the currently visited URL is on this list. If so, appropriate countermeasures can be taken. Otherwise, the page is considered legitimate. The blacklist can be stored locally at the client or hosted at a central server.

Advantages:

Obviously, an important factor for the effectiveness of a blacklist is its coverage. The coverage indicates how many phishing pages on the Internet are included in the list. Another factor is the quality of the list. The quality indicates how many non-phishing sites are incorrectly included into the list. For each incorrect entry, the user experiences a false warning when she visits a legitimate site, undermining her trust in the usefulness and correctness of the solution.

Finally, the last factor that determines the effectiveness of a blacklist-based solution is the time it takes until a phishing site is included. This is because many phishing pages are short-lived and most of the damage is done in the time span between going online and vanishing. Even when a blacklist contains many entries, it is not effective when it takes too long until new information is included or reaches the clients.

Disadvantages:

The overall technique to identify phishing sites by refreshing boycotted URLs, Internet Protocol (IP) to the antivirus information base which is otherwise called "boycott" strategy. To dodge boycotts, assailants utilizes innovative procedures to trick clients by adjusting the URL to seem real through muddling and numerous other basic methods including: quick motion, in which intermediaries are naturally produced to have the page; algorithmic age of new URLs; and so forth. Significant disadvantage of this strategy is that it can't recognize Zero-hour phishing attack.

Test Data and Statistics

For our study, a large number of phishing pages were necessary. We concatenated three databases from Kaggle and merged it into one. The information collected by this method is freely available and the amount of reported phishing sites is very large.

Table: (a) Domains that host phishing sites. (b) Popular phishing targets.

(a)		(b)	
No domain (numerical)	3,864	paypal	1,301
.com	1,286	53.com	940
.biz	1,164	ebay	807
.net	469	bankofamerica	581
.info	432	barclays	514
.ws	309	volksbank	471
.jp	307	sparkasse	273
.bz	256	openplan	182
.nz	228	Total	5,069
.org	156		
.de	111		
.ru	106		
.us	105		

LITERATURE SURVEY

In **Phishing E-mail Detection Based on Structural Properties** [1], the proposed approach explains to find phishing through appropriate identification and usage of structural properties of email. The experiment is done by SVM and classification technique to classify phishing e-mails. The technique used in this classification method is not large enough and it uses only one approach to identify phishing e-mails, which is low in efficiency and scalability. This is purely based on structural properties of e-mail and it has to extend more structural or content properties to reduce error results.

In **Discovering Phishing Target Based on Semantic Link Network** [2], the paper proposes a novel approach to discover phishing website by calculating association relation among webpages that include malicious webpages and its associated webpages to measure the combination of link

relation, search relation, and text relation. The semantic link network proposes a strategy based on four convergent situations to identify the suspicious webpage as phishing. The demerits in this approach are more kind of association has to be done, similarities between visual, layout and domain have to be related. This method is considered as a time-consuming approach and also various sub-relations in the combined association relations be studied.

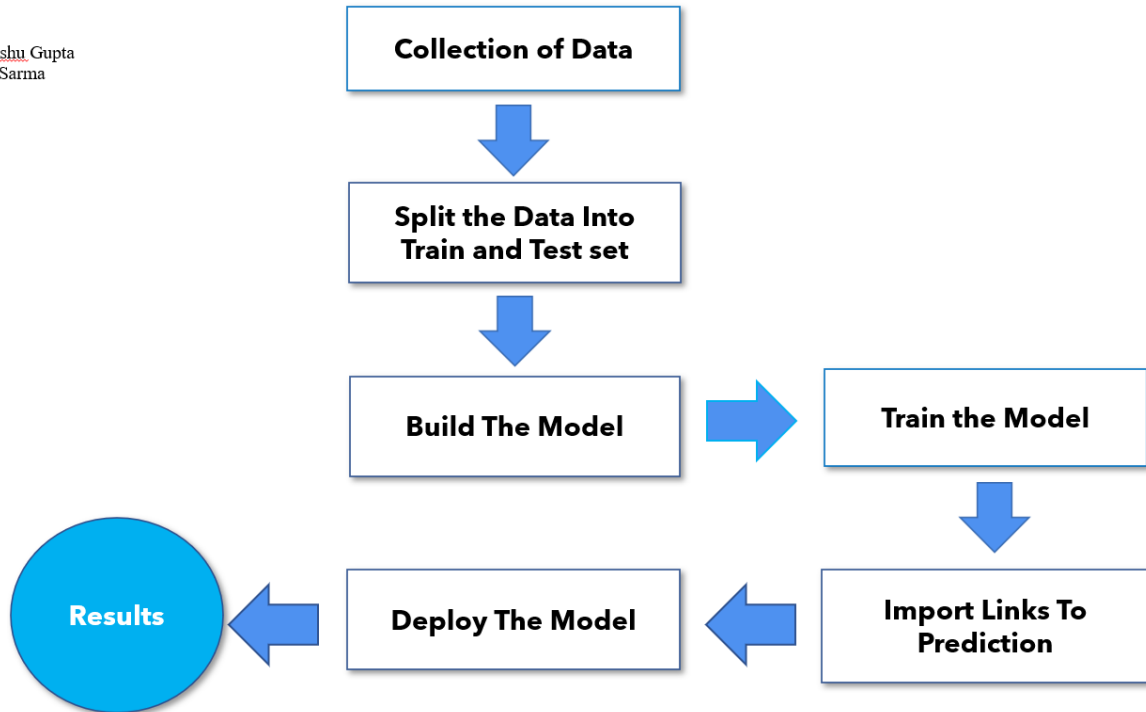
In **Evolving Fuzzy Neural Network for Phishing Emails Detection** [3], deals with zero-day phishing email. It differentiates phishing email and ham email in online mode. It is adopted on feature fetching, rank fetching and grouping similar features of email. The technique is based on binary value 0 or 1 to produce the result for all features used in this method, where 1 denotes a phishing feature and 0 for non-phishing. This technique does not have more dynamic system, so it is less in performance to produce accurate results.

In **Intelligent Phishing Website Detection and Prevention System by Using Link Guard Algorithm** [4], proposed a system using link guard algorithm which works for hyperlinks. The algorithm performs certain tests like comparison of the DNS of actual and visual links, checks dotted decimal of IP address, checks encoded links and pattern matching. The drawbacks of this system is, it produce the false positive results if any genuine site has IP address instead of domain name, and it considers some phishing site as normal one if the user does not visit the original site. This results in false negative conclusions.

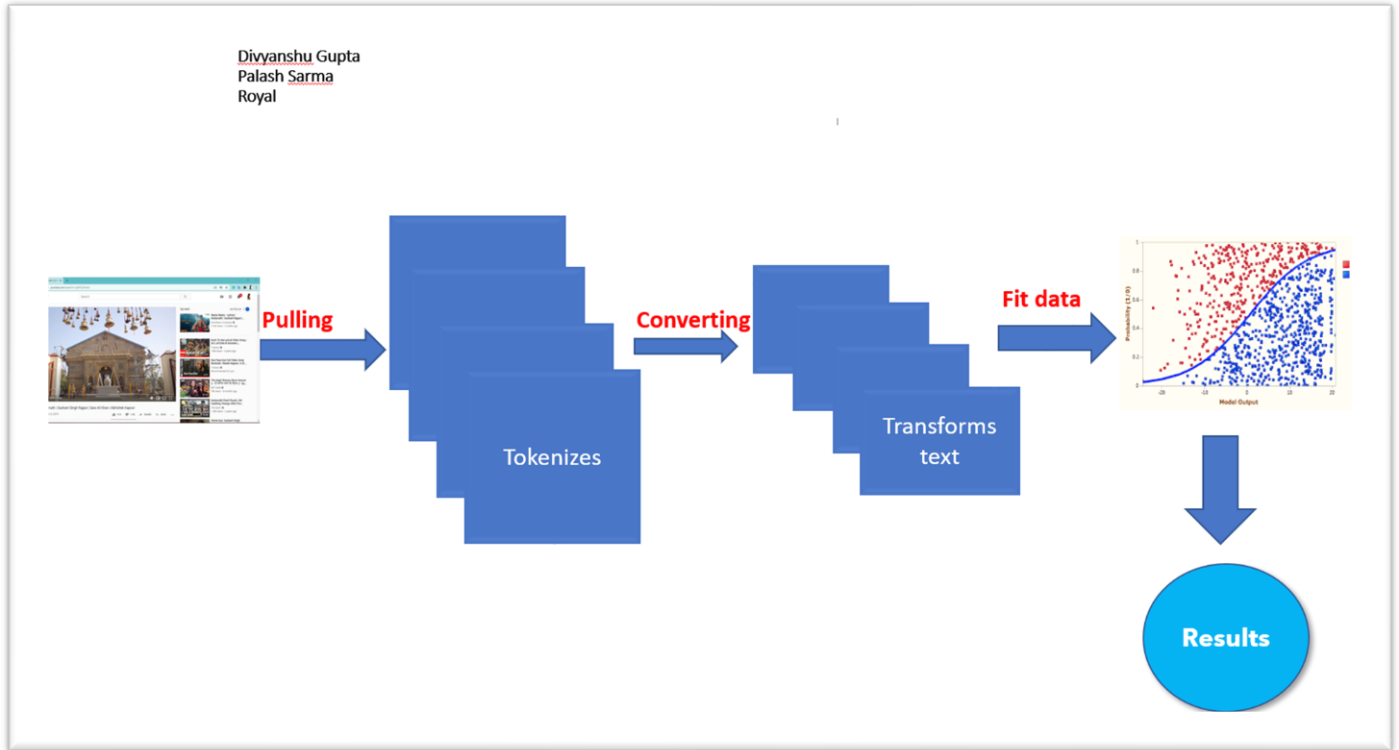
In **Said Afroz, Rachel Greenstadt – Phishzoo Approach** [5], the algorithm detects current phishing sites by matching their content with genuine site. This will match images, contents and the structure of website with trusted one in order to avoid phishing. Drawbacks of this algorithm is, it requires matching image site, and it is less robust for detecting phishing attacks.

Proposed Methods

Divyanshu Gupta
Palash Sarma
Royal



Methodology



Detailed Description of Methodology

MACHINE LEARNING ALGORITHMS

- Logistic regression

Logistic regression is a statistical analysis method used to predict a data value based on prior observations of a data set. A logistic regression model predicts a dependent data variable by analyzing the relationship between one or more existing independent variables. It is used in statistical software to understand the relationship between the dependent variable and one or more independent variables by estimating probabilities using a logistic regression equation.

- Random Forest Algorithm

Random forest algorithm is one of the most powerful algorithms in machine learning technology and it is based on concept of decision tree algorithm. Random forest algorithm creates the forest with number of decision trees. High number of trees gives high detection accuracy. Creation of trees are based on bootstrap method. In bootstrap method features and samples of dataset are randomly selected with replacement to construct single tree. Among randomly selected features, random forest algorithm will choose best splitter for the classification and like decision tree algorithm; Random forest algorithm also uses gini index and information gain methods to find the best splitter. This process will get continue until random forest creates n number of trees. Each tree in forest predicts the target value and then algorithm will calculate the votes for each predicted target. Finally, random forest algorithm considers high voted predicted target as a final prediction.

- Multinomial Naive Bayes

Multinomial Naive Bayes algorithm is a probabilistic learning method that is mostly used in Natural Language Processing (NLP). The algorithm is based on the Bayes theorem and predicts the tag of a text such as a piece of email or newspaper article. It calculates the probability of each tag for a given sample and then gives the tag with the highest probability as output. The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts.

Modules

DATASET

We combined various datasets from Kaggle and concatenate them into one. The final file name after concatenation is phishing_classifier_url.csv. The dataset furthermore contains 2 attributes which are:

- URL: this contains a list of many URLs which are either phishing sites or are not phishing sites.
- LABEL: this comprises of a binary entry i.e. bad or good.

Data collection

Collecting data for training the ML model is the basic step in the machine learning pipeline. The predictions made by ML systems can only be as good as the data on which they have been trained. Following are some of the problems that can arise in data collection:

Inaccurate data. The collected data could be unrelated to the problem statement.

Missing data. Sub-data could be missing. That could take the form of empty values in columns or missing images for some class of prediction.

Data imbalance. Some classes or categories in the data may have a disproportionately high or low number of corresponding samples. As a result, they risk being under-represented in the model.

Data bias. Depending on how the data, subjects and labels themselves are chosen, the model could propagate inherent biases on gender, politics, age or region, for example. Data bias is difficult to detect and remove

Train-Test Split Evaluation

The train-test split is a technique for evaluating the performance of a machine learning algorithm.

It can be used for classification or regression problems and can be used for any supervised learning algorithm.

The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset.

Train Dataset: Used to fit the machine learning model.

Test Dataset: Used to evaluate the fit machine learning model.

The objective is to estimate the performance of the machine learning model on new data: data not used to train the model. This is how we expect to use the model in practice. Namely, to fit it on available data with known inputs and outputs, then make predictions on new examples in the future where we do not have the expected output or target values.

Model Building

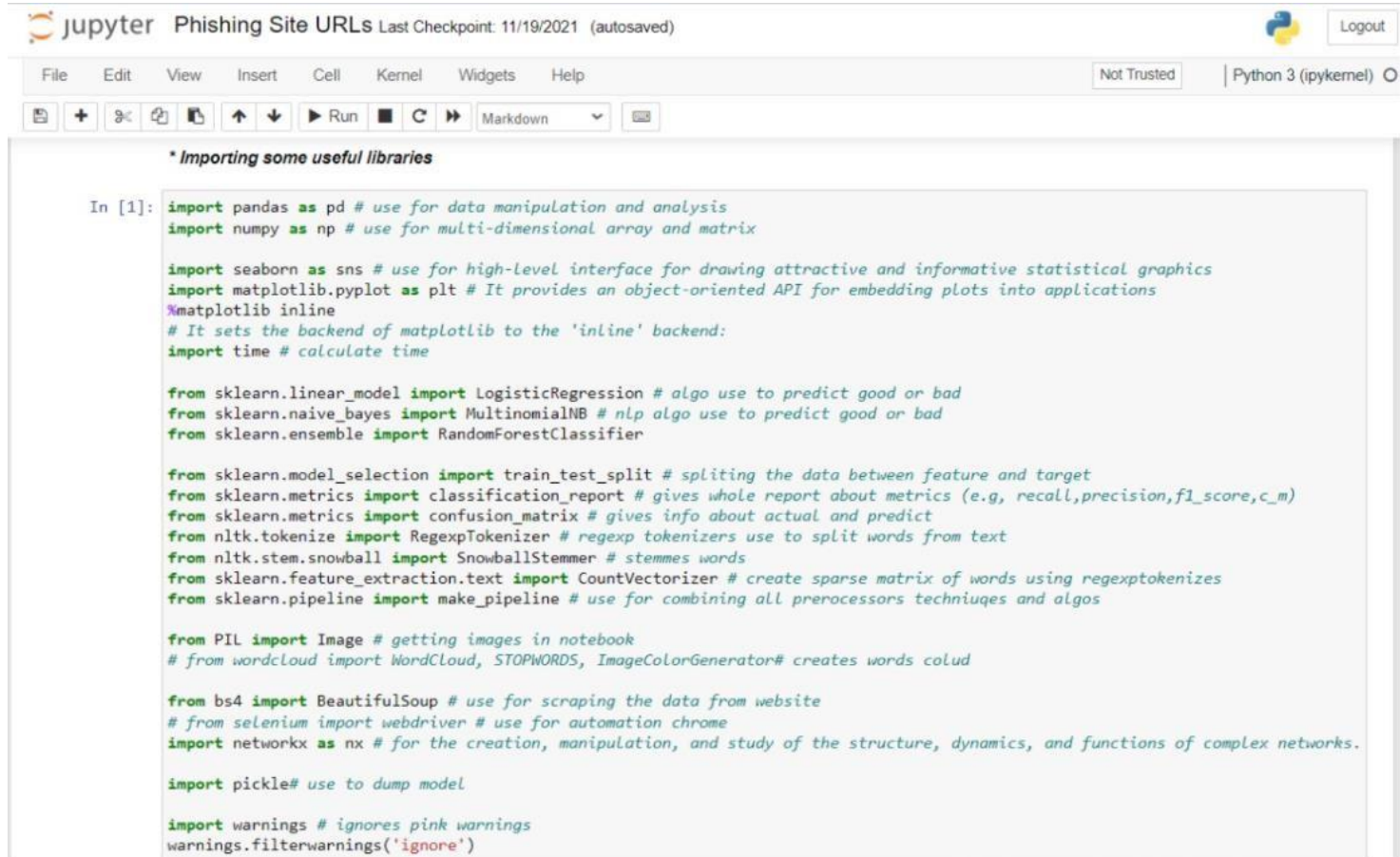
We finally now use the prepared data for model building. Depending on the data type (qualitative or quantitative) of the target variable (commonly referred to as the Y variable) we are either going to be building a classification model. We will be using logistic regression, MultinomialNB and Random Forest.

Model training

It consists of the sample output data and the corresponding sets of input data that have an influence on the output. The training model is used to run the input data through the algorithm to correlate the processed output against the sample output. The result from this correlation is used to modify the model.

Deployment is the method by which you integrate a machine learning model into an existing production environment to make practical business decisions based on data. It is one of the last stages in the machine learning life cycle and can be one of the most cumbersome.

Code



The image shows a Jupyter Notebook interface. At the top, the title bar reads "jupyter Phishing Site URLs Last Checkpoint: 11/19/2021 (autosaved)". On the right, there is a "Logout" button and a Python 3 (ipykernel) logo. Below the title bar is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Widgets, Help. To the right of the menu bar is a "Not Trusted" warning and a "Python 3 (ipykernel)" label. Below the menu bar is a toolbar with icons for saving, adding, undo, redo, up, down, run, interrupt, and a dropdown menu currently set to "Markdown".

The main area of the notebook contains a code cell with the following text:

```
* Importing some useful libraries

In [1]: import pandas as pd # use for data manipulation and analysis
import numpy as np # use for multi-dimensional array and matrix

import seaborn as sns # use for high-level interface for drawing attractive and informative statistical graphics
import matplotlib.pyplot as plt # It provides an object-oriented API for embedding plots into applications
%matplotlib inline
# It sets the backend of matplotlib to the 'inline' backend:
import time # calculate time

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.ensemble import RandomForestClassifier

from sklearn.model_selection import train_test_split # splitting the data between feature and target
from sklearn.metrics import classification_report # gives whole report about metrics (e.g, recall, precision, f1_score, c_m)
from sklearn.metrics import confusion_matrix # gives info about actual and predict
from nltk.tokenize import RegexpTokenizer # regexp tokenizers use to split words from text
from nltk.stem.snowball import SnowballStemmer # stemmes words
from sklearn.feature_extraction.text import CountVectorizer # create sparse matrix of words using regextokenizes
from sklearn.pipeline import make_pipeline # use for combining all prerocessors techniues and algos

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

from bs4 import BeautifulSoup # use for scraping the data from website
# from selenium import webdriver # use for automation chrome
import networkx as nx # for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.

import pickle# use to dump model

import warnings # ignores pink warnings
warnings.filterwarnings('ignore')
```

- | URL | Label |
|-----|-------|
|-----|-------|

▶ Run

In [21]: `phish_data.info()`

```
RangeIndex: 549346 entries, 0 to 549345
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  -
 0   URL     549346 non-null  object
 1   Label   549346 non-null  object
memory usage: 8.4+ MB
```

- Date is c onlaing 5.4g.3do unique enTrws.
- There are o columns
- Lehel colHmn in gredictiuo nlyyhirh nes 2 categories A which means the url\$ is roi+:nnTai ing mRlir.inus sl:f# and lhl aha ienot a Phishing Bite. B. Bad - which means lne ur's contd.'ns malicious skulls and la ad IM Rhichlrrg Site.
- there is no milking vairie in lthe daiasel

ir. [22]: `phish_data.isnull().sum()` # there is no missing values

Out[22]: URL 0

dtype: into

```
7a6el roucts : pd.DataFrame(phish_data.lated.value ,ounts())
```

```
sns.set_style('darkgrid')
sns.baupJot { Labe E_count s . i r+dex , labe T_c or nl s.L abe l }
```

Out[24]: <AxesSubplot:ylabel='Label'>

- A token*et thai spits e suing using s <eguie exotesson, rich mulches eiihet ihe id<ens or me separatos between td<ens.

```
I» f25,: totmnizmr « SegonpToConizor(r'{fi..u.: '* ')
```

```
phish_data.at('URL')[n]
```

```
[20]: 'nobell.it/70ffb52d079109dca5664cce6f3173782/login.SkyPe.com/en/cgi-bin/verification/login/70ffb52d079109dca5664cce6f317373/index.php?cmd=_profile-ach&outdated_page_tmpl=p/gen/failed-to-load&nav=0.5.1&login_access=1322408526'
```

```
tokenizer.tokenize(phish_data.URL[0]) # using first row
```

```
['oehell',
```

```
'ffb',
```

```
'cce',
```

```
'SkyPe',
```

```
'e ilicztien',
```

```
'd.P
```

```
'dca',
```

```
'php'
```



at

Getting words stemmed ...
 Time taken 2.983d26d9S9999095 sec

In [29]: phi_hdmcc.mmaplo(C)

Out[29]:

67551	tools.ietf.org/html/rfc1679	good	[tools, ietf, org, html, rfc]
508306			

- Snowball is a small string processing language, gives root words

In [30]: stemmer = SnowballStemmer('english')

```
In [31]: print('Getting words stemmed ...')
t0 = time.perf_counter()
phish_data['text_stemmed'] = phish_data['text_tokenized'].map(lambda l: [stemmer.stem(word) for word in l])
t1 = time.perf_counter() - t0
print('Time taken', t1, 'sec')

Getting words stemmed ...
Time taken 39.97157340000001 sec
```

In [32]: phish_data.sample(5)


```
In [32]: phishing_data.sample(5)
```

Out[32]:

	URL	Label	text_tokenized	text_summary
260229				
499860	hdxsp1kk.top/?xH2AcreZKxnNCIE=i3SKfPrfJxzFGMSU...	bad	[hdxsp, kk, top, xH, AcreZKxnNCIE, i, SKfPrfJx...	[hdxsp, kk, top, xh, acrezkxnnci, i, skfprfjz...
298898				

```
In [33]: print('Getting joining words ...')
         t0 = time.perf_counter()
         phish_data['text_sent'] = phish_data['text_stemmed'].map(lambda x: ' '.join(1))
```

$p \bullet i \text{ in } I('I' w t + k: e \bullet i'', t \forall, 'se r')$

i ice t amen 0 . 201764499999979 3 soc

```
l< !""_ : phs fl dat». *aeple(5)
```

103042	slideanddivide.co.uk/js/microsoft secure.ency...	bad	[slideanddivide.co.uk/js/microsoft secure...	[slideanddivid.co.uk/js secur...	secure encrypt...
210811	manhattan.about.com/od/glbtscene/a/lavender-ill...	good	[msWea...com.o0.gdscene.a Bve...	[man«ana ,ao>x.cmm oo glbtscene, a,lave...	com c< glbtscene a,laveid igh...
544583					

Creating Model

Co+nfusctodzer

- CountVectorizer is used to transform a corpora of text to a vector of term token counts.

```
cv = CountVectorizer()
```

```
[45]: cv.fit_transform(phish_data.text_sent)
```

```
In [46]: feature = cv.fit_transform(phish_data.text_sent) #transform all text which we tokenize and stemmed
```

```
In [47]: feature[:5].toarray() # convert sparse matrix into array to print transformed features
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

```
. '[ch] *raint, tes*M, *rainy, tesCy - train tes* split(fea*uze, phi*N da*a.Label)
```

```
In [54]: rf = RandomForestClassifier(n_estimators=100)
```

```
rf.fit(trainX, trainY)
```

```
In [ ]: rf.score(testX, testY)
```

Logistic Regression

- Logistic Regression is a machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that takes data coded as 1 (yes, success, etc) or 0 (no, failure, etc). In other words this logistic regression model predicts $P(Y=1)$ as a function of X .

```
lr = LogisticRegression()
```

```
In [43]: lr.fit(trainX,trainY)
```

```
[44]: lr.score(testX,testY)
```

```
Out[44]: 0.9636514559077306
```

" Logistic Regression is giving ~96% accuracy, Now we will store scores in dict to see which model performs best

```
In [45]: Scores_ml = {}
Scores_ml['Logistic Regression'] = np.round(lr.score(testX,testY),2)
```

```
In [46]: # We will store the scores of all models in a dictionary
# We will use the pd.Series to store the scores
# We will use the pd.DataFrame to store the scores
```

```
print('\nCLASSIFICATION REPORT\n')
print(classification_report(lr.predict(testX), testY,
```

```
print(classification_report(lr.predict(testX), testY,
                             labels=[0,1],
                             pos_label=1,
                             drop_in_training=False,
                             drop_in_validation=False,
                             drop_in_testing=False,
                             show_digits=4))
```

```
print ( c l e s s i f i c e t i o n r e p e a t ( i e . p r e d i c t ( t e s t D ) , t e s t Y ,
      t a r g e t _ n a m e s - { ' B a r ' , ' T o o a ' } ) )
```

```
print('\nCONFUSION MATRIX')
plt.figure(figsize= (6,4))
sns.heatmap(con_mat, annot = True,fmt='d',cmap="YlGnBu")
```

Training Accuracy: 0.97824884 79795345

Testing Accuracy: 0.96365 J'4 559e77306

CLASSIFICATION @ CIE T

	precision	recall	f1-score	support
Bad	0.95	0.87	0.93	36597
Good	0.99	0.96	0.97	100740
accuracy			0.96	137587
macro avg	0.95	0.96	0.95	137337
sighted avg	0.97	0.96	0.96	137337

CONFUSION MATRIX

Out[46]: <matplotlib.figure.Figure>



MultinomialNB

- Applying Multinomial Naive Bayes to NLP Problems. Naive Bayes Classifier Algorithm is a family of probabilistic algorithms based on applying Bayes' theorem across the training set of conditional independence of every pair of features.

```
mnb = MultinomialNB()
```

```
[48]: lb.fit(train_M, train_no)
      multinomialNB()
```

```
[49]: lb.score(test_X, test_Y)
```

```
Out[49]: 0.9574550194048217
```

*** MultinomialNB gives us 95% accuracy**

```
In [50]: scores_T["Hacker's Lair"] = np.round(lb.score(test_X, test_Y), 2)
```

```
[51]: print('Training Accuracy: ', lb.score(train_X, train_Y))
      print('Testing Accuracy: ', lb.score(test_X, test_Y))
      con_mat = pd.DataFrame(confusion_matrix(mnb.predict(test_X), test_Y),
                             columns=['Predicted:Bad', 'Predicted:Good'],
```

```
print('\nCLASSIFICATION REPORT\n')
print(classification_report(mnb.predict(test_X), test_Y,
                             target_names=['Bad', 'Good']))
```

```
print('\nCONFUSION MATRIX')
```

```
so % .head(top_categorical, axis=1, sort=True, columns=['Bad', 'Good'])
```

Training Accuracy : 0.9741437687840817

Testing Accuracy : 0.9574550194048217



File View Insert Settings Help | Zsh +!

Not Trusted

Python 3 (ipykernel) C

```
#####
```

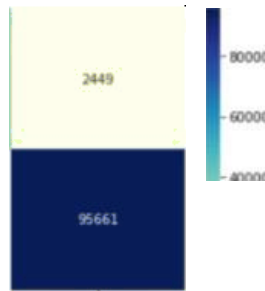
Training Accuracy : 0.9741437670
Testing Accuracy : 0.957455194882

Classification Report

	precision	recall	f1-score	support
Bad	0.91	0.94	0.92	18282
Good	0.98	0.97	0.98	137337
accuracy			0.96	137337
macro avg	0.94	0.95	0.95	137337
weighted avg	0.96	0.96	0.96	137337

```
t4FuSlQn MmrRl*
```

```
Out[51]: <matplotlib.axes._subplots.AxesSubplot at 0x1ec84f9c908>
```



```
In [52]: acc = pd.DataFrame(rate, index=rate.index, columns=['Accuracy'])
sns.set_style('flat')
sns.barplot(acc.index, acc['Accuracy'])
```

```
Out[52]: <matplotlib.axes._subplots.AxesSubplot at 0x1ec84f9c908>
```

```
to |sz]: acc = pd.DataFrame.from_dict(Scores, orient='index', columns=('n', 'x', 'conf', 's'))
sns.set_style('darkgrid')
sns.barplot(ccc.index, acc.accuracy)
```

Out[52]: <matplotlib.axes._subplots.AxesSubplot at 0x1ec84f71348>



" So, Logistic Regression is the best fit model. Now we make sklearn pipeline using Logistic Regression

```
In [53]: pipeline_ls = make_pipeline(CountVecorizer(tokenizer = RegexpTokenizer(r'[A-Za-z]+').tokenize, stop_words='english'), LogisticRe
```

```
l)[0:4]: traino, testX, traino, testY, train test split this data. URL, eh is b_d z t a. ka bet }
```

```
In [55]: pipeline_ls.fit(trainX, trainY)
```

```
..: Pipeline with steps:
  CountVecorizer(steps=[('tokenizer', RegexpTokenizer(r'[A-Za-z]+').tokenize, stop_words='english'),
  ('vectorizer', CountVecorizer(tokenizer='tokenizer', stop_words='english'))])
```

```
In [56]: pipeline_ls.score(trainX, trainY)
```

```
Out[56]: 0.9674AS043943BB*6
```

```
In [57]: print('Training Accuracy :', pipeline_ls.score(trainX, trainY))
print('Testing Accuracy :', pipeline_ls.score(testX, testY))
con_mat = pd.DataFrame(confusion_matrix(pipeline_ls.predict(testX), testY),
                           columns = ['Predicted:Bad', 'Predicted:Good'],
                           index = ['Actual:Bad', 'Actual:Good'])
```

```
print('\nCLASSIFICATION REPORT\n')
print('Classification Report of the model on the test set')
```

```
print('\nConfusion Matrix\n')
plt.figure(figsize=(6,4))
sns.heatmap(con_mat, annot=True, fmt='d', cmap="YlGnBu")
```

```
Training Accuracy : 0.980891*9#*JA't003
Testing Accuracy : 0.9674ASBA39t38B36
```

CLASSIFICATION REPORT

	precision	recall	F1-score	support
Bad	0.97	0.97	0.94	36841
Good	0.99	0.97	0.98	100496
accuracy			0.97	137337
macro avg	0.95	0.97	0.96	137337
weighted avg	0.97	0.97	0.97	137337

CONFUSION MATRIX

```
Out[57]: <matplotlib.figure.Figure at 0x1ec83fc a58B>
```





File Edit View Insert Cell Kernel Widgets Help

NotT/usleD

Orton 2 (ipykernel) I

```
In [5B]: pickle.dump(pipeline_ls, open('phs>l »b.pcl', 'wL'))
```

```
In [5S]: loaded_model = pickle.load(open('phi:hing.pk1', 'rb'))
result = loaded_model.score(testM, testW)
print(result)
```

```
6.967445043943B0 #6
```

```
In [ ]: * Bad links => this are phishing sites
        yeniik.com.tr/wp-admin/js/login.alibaba.com/login.jsp.pNp
        +tub vT ez.exe
        svis1on-online.dey fi/ada%tnistnator/cmponeotsyccra babakup/classes/fx29Tdl.txt
```

```
* Good links => this are not phishing sites
```

```
youtube . cw/ua4•chtvsqI0TQ3I3vdU
```

```
restorevisioncenters.com/htmlYtechnology.html
```

```
In [60]: predict_bad = ['yeniik.com.tr/wp-admin/js/login.alibaba.com/login.jsp.php', 'fazan-pacir.rs/temp/libraries/ipad', 'tubemoviez.exe',
predict_good = ['youtube.com/', 'youtube.com/watch?v=qI0TQ3I3vdU', 'retailhellunderground.com/', 'restorevisioncenters.com/html/tec
loaded_model = pickle.load(open('phishing.pk1', 'rb'))
#predict_bad = vectorizers.transform(predict_bad)
# predict_good = vectorizer.transform(predict_good)
result = loaded_model.predict(predict_bad)
result2 = loaded_model.predict(predict_good)
print(result)
```

```
print(result2)
```

```
['bad' 'bad' 'bad' 'bad']
```

Implementation

Not a phishing site

127.0.0.1:8000/docs#/default/predict_predict_feature_get

GET /predict/{feature} Predict

Parameters

Cancel

Name	Description
features required (query)	<input type="text" value="https://www.youtube.com/watch?v=uyONiEVI"/>

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/predict/{feature}?features=https%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3DuyONiEVEudM%26ab_channel%3DTaarakMehtaKaOoltahChashmah' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/predict/{feature}?features=https%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3DuyONiEVEudM%26ab_channel%3DTaarakMehtaKaOoltahChashmah
```

Server response

Code	Details
200	<p>Response body</p> <pre>["https://www.youtube.com/watch?v=uyONiEVEudM%26ab_channel=TarakMehtaKaOoltahChashmah", "This is not a Phishing Site"]</pre> <p>Download</p>

Phishing site

GET /predict/{feature} Predict

Parameters

Cancel

Name	Description
features required (query)	<input type="text" value="www.paypay/123/df.html"/>

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/predict/{feature}?features=www.paypay%2F123%2Fdf.html' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/predict/{feature}?features=www.paypay%2F123%2Fdf.html
```

Server response

Code	Details
200	<p>Response body</p> <pre>["www.paypay/123/df.html", "This is a Phishing Site"]</pre> <p>Download</p>

RESULT AND ANALYSIS OF PROPOSED METHOD

We used the same legitimate URL “google.com” on each algorithm and found the following accuracy:

1. Random Forest Classifier:

Accuracy: 70%

2. Logistic regression classifier:

CONFUSION MATRIX

: <matplotlib.axes._subplots.AxesSubplot at 0x1ec84c387c8>



3. MultinomialNB classifier:

CONFUSION MATRIX

Out[51]: <matplotlib.axes._subplots.AxesSubplot at 0x1ec84f9c908>



Table below shows the results of classifiers used for the classification process in python. From the table it is shown that the classifier **Logistic Regression** produce the best result.

Classifier	Accuracy
Random Forest	70%
Logistic regression	96.36%
MultinomialNB	95.78%

CONCLUSION

This project intends to upgrade recognition technique to recognize phishing sites utilizing machine learning innovation. From our experiment we found logistic regression has the highest accuracy rate as 96.3% as compared to random forest and logistic regression classifier. Likewise result shows that classifiers give better execution when we utilized more information as preparing information. Hence the users can get rid of phishing sites and be safe by avoiding them using this technique. The final confusion matrix after optimizing the logistic regression and completing the pipelining work is given below.



Future Scope

Incorporating large datasets

Designing web interface, by using html and css

Increasing model accuracy by using bagging and boosting techniques

REFERENCES

- [1] Chandrasekaran, Madhusudhanan, Krishnan Narayanan, and Shambhu Upadhyaya. "Phishing email detection based on structural properties." NYS Cyber Security Conference. 2006.
- [2] Wenyin, Liu, et al. "Discovering phishing target based on semantic link network." Future Generation Computer Systems 26.3 (2010): 381- 388.
- [3] Almomani, Ammar, et al. "Evolving fuzzy neural network for phishing emails detection." Journal of Computer Science 8.7 (2012): 1099.
- [4] Madhuri, M., K. Yeseswini, and U. Vidya Sagar. "Intelligent phishing website detection and prevention system by using link guard algorithm." Int. J. Commun. Netw. Secur 2 (2013): 9-15.
- [5] --Afroz, Sadia, and Rachel Greenstadt. "Phishzoo: Detecting phishing websites by looking at them." Semantic Computing (ICSC), 2011 Fifth IEEE International Conference on. IEEE, 2011.