PHISHING LINK CLASSIFIER USING ML

SLOT : B2/TB2

Debalay Dasgupta 19BCE2423

Submitted to

Prof. Geraldine Bessie Amali, SCOPE

School of Computer Science and Engineering

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Table of Contents

Abstract

In this digital day and electronic world, the Internet plays a vital role in day-to-day activities like communication, business, transactions, personal needs, marketing, e-commerce etc. The Internet is a multifaceted facility which helps in completing many tasks readily and conveniently within a few seconds. Almost everything is presently accessible over the web in this period of progression of advances. Thus, increasing usage of the 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 a phishing web page. Phishing uses emails and websites, which are intended to look like from trusted organizations, to hoodwink clients into unveiling their own or money related data. The threatening party then uses 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 the user leads to several losses. 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.

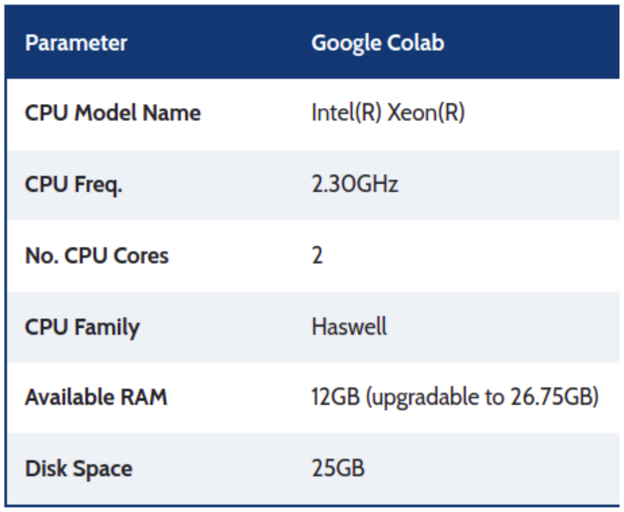
1. Introduction

Phishing is popular among attackers because it is easier to persuade someone to click on a malicious link that appears to be legitimate than it is to break through a computer's defenses. The malicious links in the message's body are made to look like they go to the faked organization by utilizing the spoofed organization's logos and other legitimate material. Every day and hour, a large number of users accidentally click phishing URLs. The attackers are going after both users and businesses. The annual global impact of phishing might be as high as $5 billion, according to the 3rd Microsoft Computing Safer Index Report, released in February 2014. The main cause is a lack of user knowledge. However, security defenders must take efforts to keep consumers away from these dangerous websites. Preventing these massive expenditures can begin with raising public awareness, as well as developing strong security systems that can detect and prevent phishing domains from reaching the user. In the academic literature and commercial applications, there are numerous algorithms and data formats for phishing detection. A phishing URL and its accompanying website have various characteristics that distinguish them from harmful URLs. Therefore we provide 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. Two large standard legitimate datasets with 73,575 URLs and 100,000 URLs were used. Two test modes (percentage split, K-Fold cross-validation) were utilized for conducting experiments and final predictions.

2. Hardware/Software Requirements

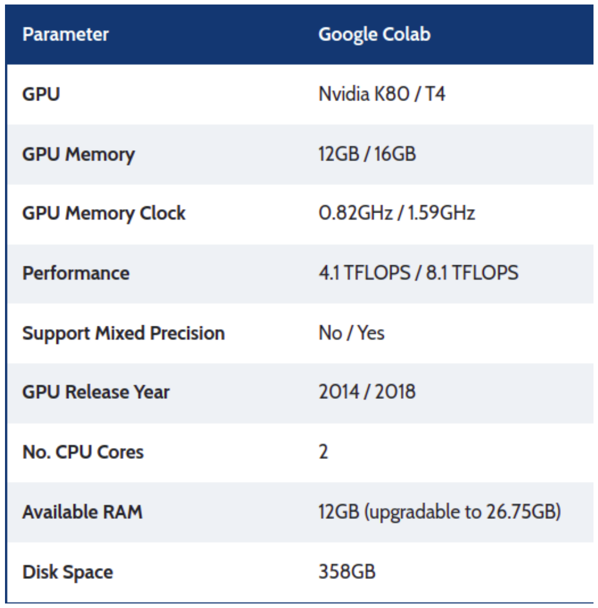
Used Google Collab for Running the python notebooks. The specifications of Google Collab free version are mentioned below:

Hardware Specs (CPU-only VMs)



Parameter	Google Colab
CPU Model Name	Intel(R) Xeon(R)
CPU Freq.	2.30GHz
No. CPU Cores	2
CPU Family	Haswell
Available RAM	12GB (upgradable to 26.75GB)
Disk Space	25GB

Software Specs (GPU VMs)



Parameter	Google Colab
GPU	Nvidia K80 / T4
GPU Memory	12GB / 16GB
GPU Memory Clock	0.82GHz / 1.59GHz
Performance	4.1 TFLOPS / 8.1 TFLOPS
Support Mixed Precision	No / Yes
GPU Release Year	2014 / 2018
No. CPU Cores	2
Available RAM	12GB (upgradable to 26.75GB)
Disk Space	358GB

3. Existing System method

3.1 Drawback of Existing System Approach

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 utilize 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 attacks.

4. Proposed Model

4.1 Design

List of Modules:

1. Phishing classifier model analysis between RF, Multinomial NB and Logistic Regression
2. VPN based chat application with phishing link detection ability to protect customers from potential scam
3. ERG-SVC model(Expandable Random Gradient Stacked Voting Classifier) based phishing classifier analysis on different dataset.

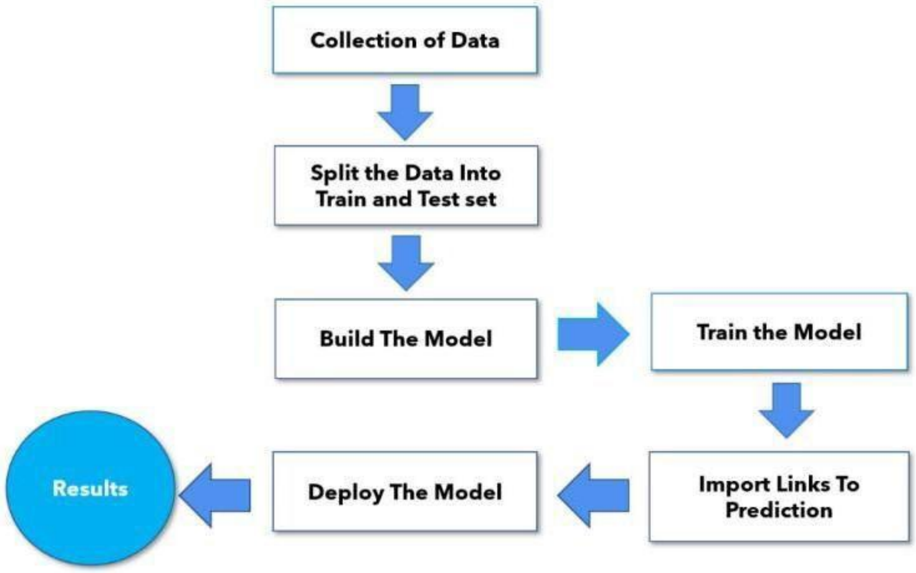
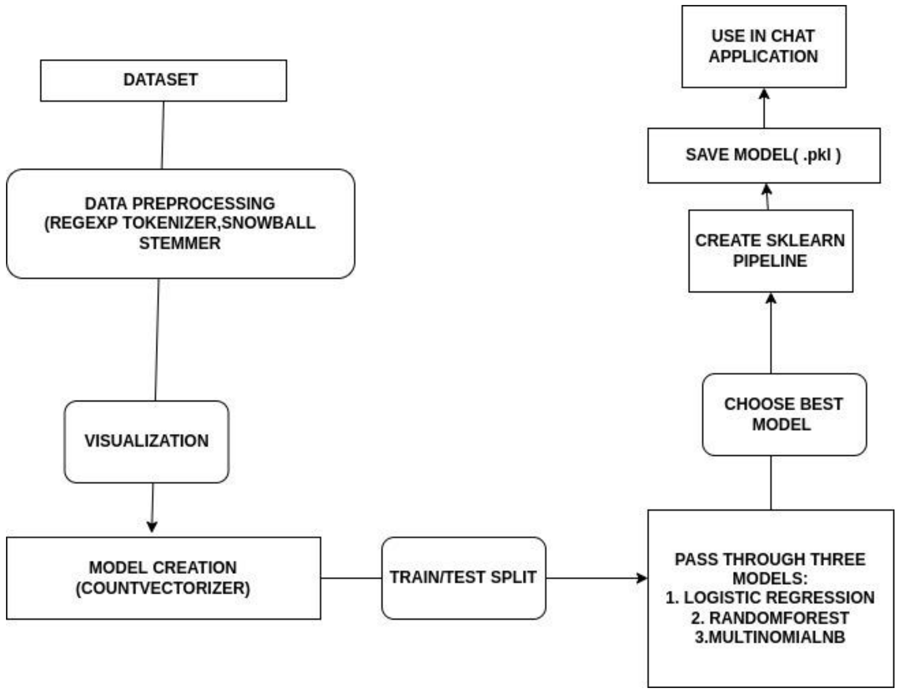


Fig : The ML pipeline followed in this project


DATASET
DATA PREPROCESSING (REGEXP TOKENIZER,SNOWBALL STEMMER
VISUALIZATION
MODEL CREATION (COUNTVECTORIZER)
TRAIN/TEST SPLIT
PASS THROUGH THREE MODELS: 1. LOGISTIC REGRESSION 2. RANDOMFOREST 3.MULTINOMIALNB
CHOOSE BEST MODEL
CREATE SKLEARN PIPELINE
SAVE MODEL(.pkl)
USE IN CHAT APPLICATION
Fig : Architecture for first two modules

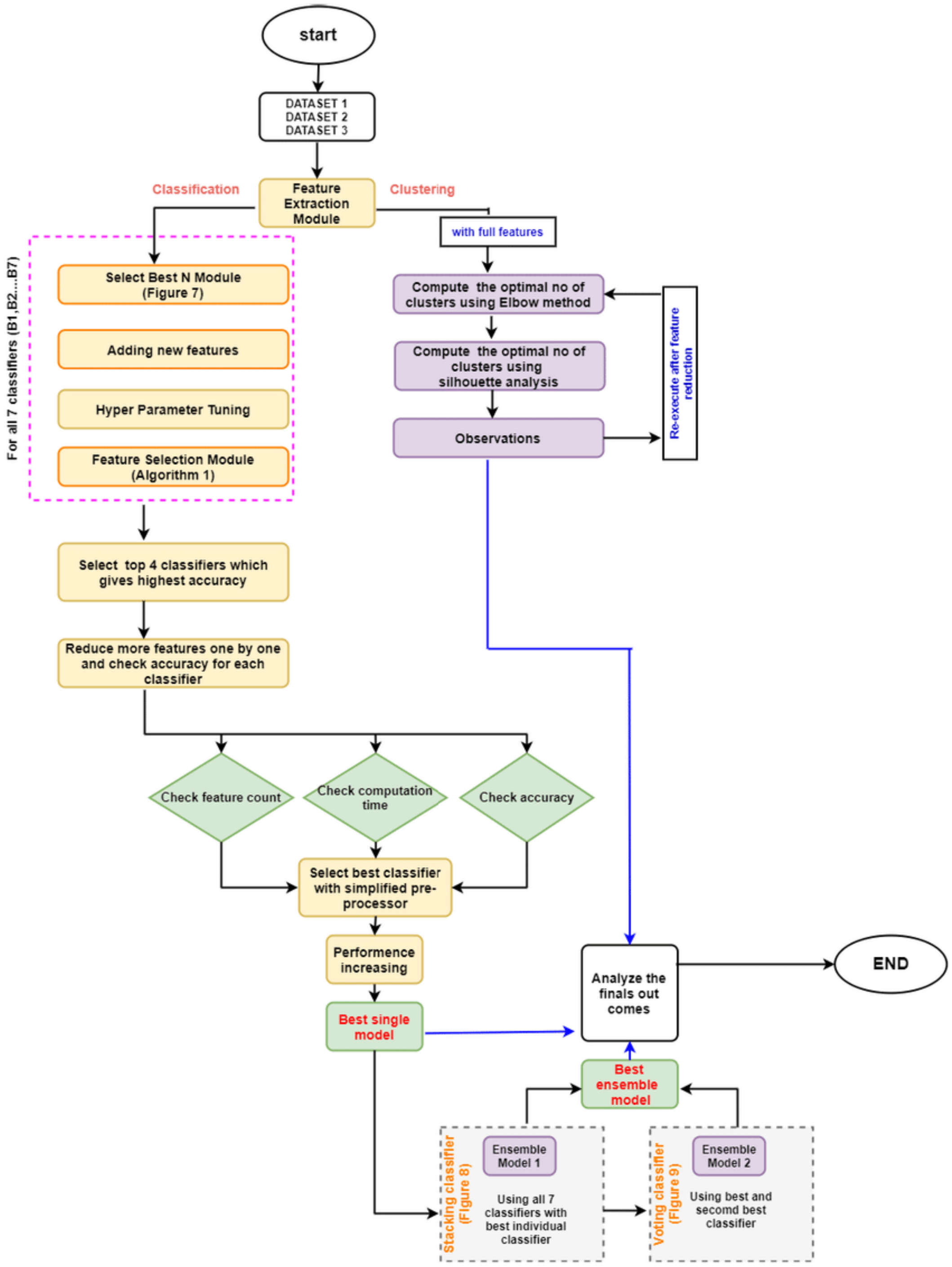

start
DATASET 1
DATASET 2
DATASET 3
Classification
Feature Extraction Module
Clustering
with full features
For all 7 classifiers (B1,B2...B7)
Select Best N Module (Figure 7)
Adding new features
Hyper Parameter Tuning
Feature Selection Module (Algorithm 1)
Compute the optimal no of clusters using Elbow method
Compute the optimal no of clusters using silhouette analysis
Observations
Re-execute after feature reduction
Select top 4 classifiers which gives highest accuracy
Reduce more features one by one and check accuracy for each classifier
Check feature count
Check computation time
Check accuracy
Select best classifier with simplified pre-processor
Performence increasing
Best single model
Analyze the finals out comes
END
Best ensemble model
Stacking classifier (Figure 8)
Ensemble Model 1
Using all 7 classifiers with best individual classifier
Voting classifier (Figure 9)
Ensemble Model 2
Using best and secomd best classifier

Fig: Ensemble Model methodology

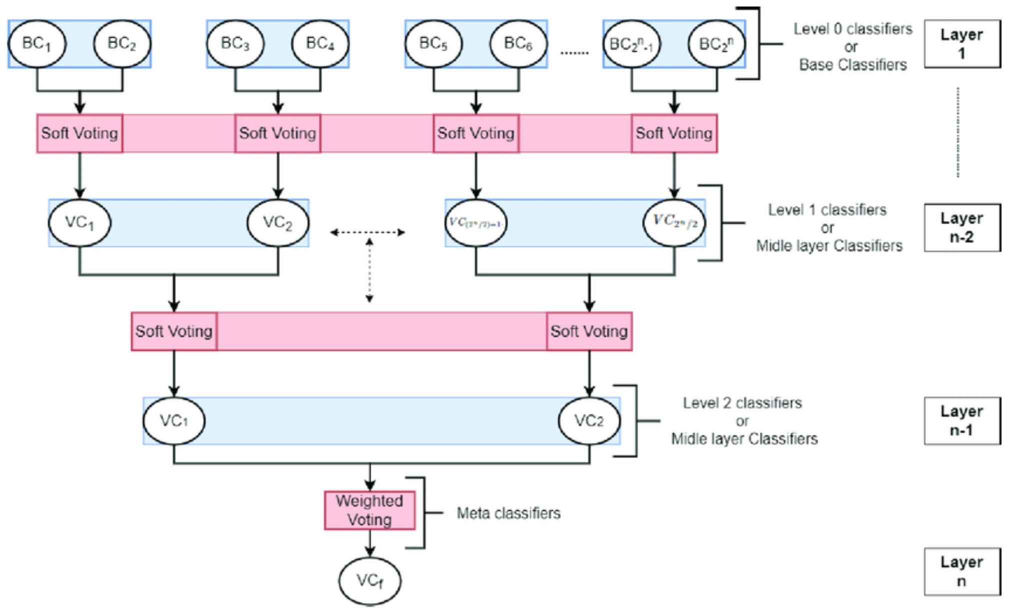

BC1
BC2
BC3
BC4
BC5
BC6
BC2n-1
BC2n
Level 0 classifiers or Base Classifiers
Layer 1
Soft Voting
VC1
VC2
Level 1 classifiers or Midle layer Classifiers
Layer n-2
Level 2 classifiers or Midle layer Classifiers
Layer n-1
Weighted Voting
Meta classifiers
VCf
Layer n

Fig : Voting classifier Architecture for choosing Ensemble Architecture

4.2 Module Wise Description

1. Phishing link classifier

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

2. **Chat application with phishing link detection ability**

Server instance is created and cipher is negotiated then the methodology in below diagram is followed.

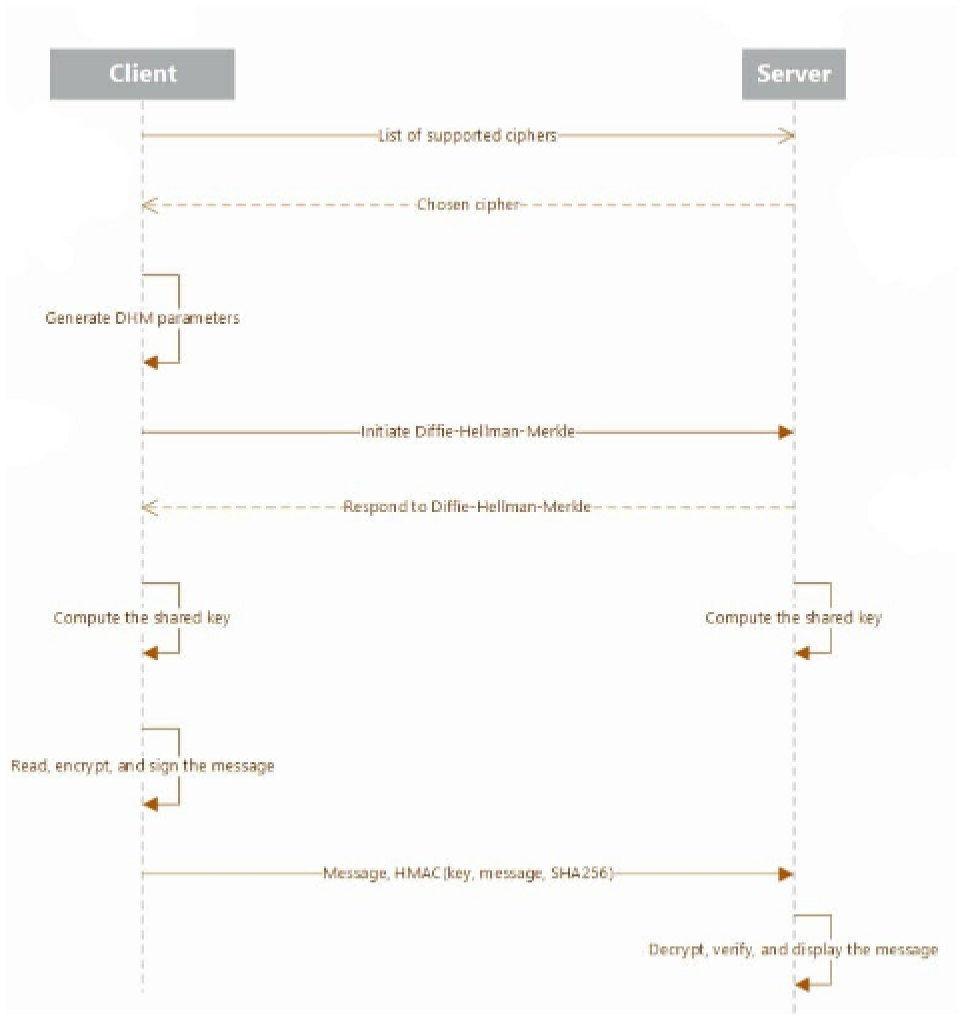

Client
Server
List of supported ciphers
Chosen cipher
Generate DHM parameters
Initiate Diffie-Hellman-Merkle
Respond to Diffie-Hellman-Merkle
Compute the shared key
Compute the shared key
Read, encrypt, and sign the message
Message, HMAC(key, message, SHA256)
Decrypt, verify, and display the message

Fig : How the VPN chat will work

3. **Expandable Random Gradient Stacked Voting Classifier**

Preprocessing

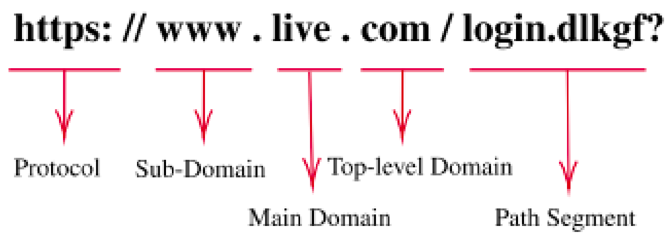
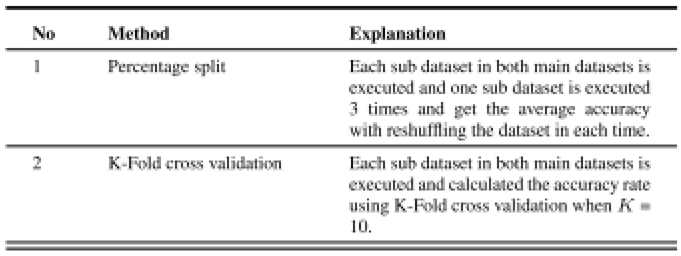


Fig : URL structure to be used for extracting features

Best N module



No	Method	Explanation
1	Percentage split	Each sub dataset in both main datasets is executed and one sub dataset is executed 3 times and get the average accuracy with reshuffling the dataset in each time.
2	K-Fold cross validation	Each sub dataset in both main datasets is executed and calculated the accuracy rate using K-Fold cross validation when K = 10.

Fig : 2 best modes used for best N module

PAIR PLOT VISUALIZATION

Pair plot visualization included in the Python Seaborn library is used to gain an interpretation of the nature of a dataset. A pair plot calculates by a variant combination between every feature combination and plots the results in a 2D diagram

Classifier Choice

As The Database Contains Additional Fragmentation, It Was Decided To Bypass A List Of Indirect Classification Algorithms That Are Useful For Building Different Ml Models. Decision Trees (Dts), Random Forests (Rfs), Closest Neighbors (Knn), And Other Indirect Segregation Algorithms Can Solve This Problem Quickly.

Hyperparameter Tuning

The Parameters That Show The Structure Of The Model Are Called Hyperparameters And The Search For The Model Of Structures Is Called Hyperparameter Tuning.

Feature Selection Module

Excessive And Repetitive Features Increase Algorithm Training Time And Reduce Its Efficiency. Therefore, In Order To Reduce Features And Size, Six Different Feature Selection Techniques Were Used To Select The Best Features In The First Feature Set.

Select ML models

Features Are Reduced In Sequence Using Six Feature Engineer Techniques. The Predictive Accuracy Of Each Ml Model And Calculation Time Is Calculated At The End Of Each Method.

Building An Ensemble Model

Many Categories Are Grouped Together To Formulate Combinations For Better Performance. Ensemble Strategies Benefit From The Achievement Of Improved Outcomes By Two Or More Division Dividers. It Can Also Be Argued That Ensemble Models Integrate Multiple Single

Models And Form A Large Volume System With High Variability Associated With Single Models.

4.3 Implementation

1. Phishing link classifier

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.

Deploy Model

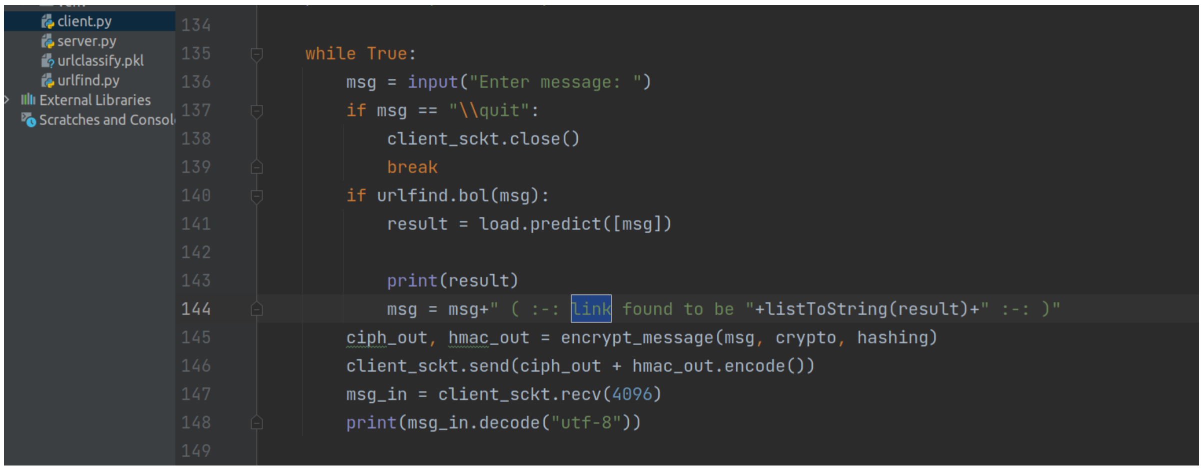
Deployment is the method by which you integrate a machine learning model into an existing production environment to make practical business decisions based on data. It is one of the last stages in the machine learning life cycle and can be one of the most cumbersome.

2. Chat application with phishing link detection ability

There are mainly two scripts running in this module server.py and client.py. A socket is created wherein automatically after running server script an instance is created .Then host and port are chosen automatically. When the client side is called then connection is made.The ciphers are negotiated out of supported ciphers. Next the key is negotiated for communication established.The cryptosystem is initialized and then exchange of messages takes place.

Along with socket, Crypto.Cipher and the pkl that was dumped and saved is slo used using model.load() method.

If any url is detected in the chat using urlfind.py script(uses regex) then urlclassifier pkl model is loaded to verify legitimacy of the link shared.



client.py
server.py
urlclassify.pkl
urlfind.py
External Libraries
Scratches and Consol

```python

    while True:
        msg = input("Enter message: ")
        if msg == "\\quit":
            client_sckt.close()
            break
        if urlfind.bol(msg):
            result = load.predict([msg])

            print(result)
            msg = msg+" ( :-: link found to be "+listToString(result)+" :-: )"
        ciph_out, hmac_out = encrypt_message(msg, crypto, hashing)
        client_sckt.send(ciph_out + hmac_out.encode())
        msg_in = client_sckt.recv(4096)
        print(msg_in.decode("utf-8"))

```

Fig : if link detected then load url classifier model

3. Expandable Random Gradient Stacked Voting Classifier

Preprocessing is kept lightweight.Followed by its classifier selection takes place,after that Best N module.Finally Hyperparameter tuning ,feature selection takes place after that top 2 models which are too stacked are finalized.

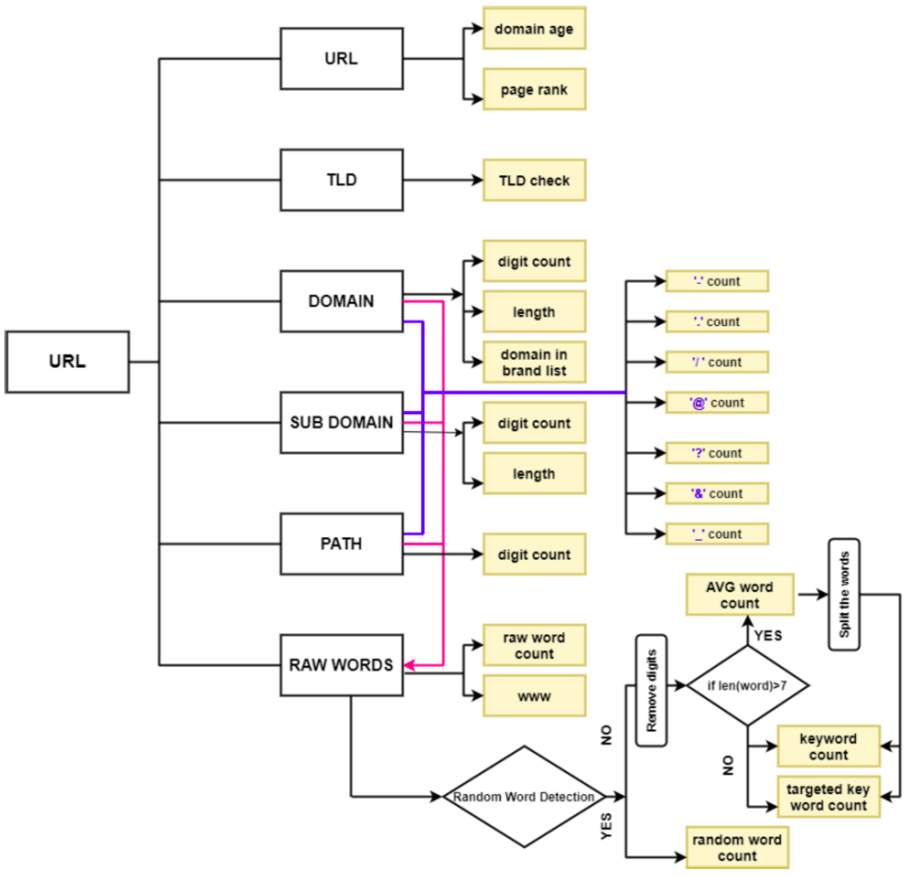


Fig : proposed Pre-processor model

The rate of identifying malicious URLs for the most part was measured using a variety of classification, ensemble, and clustering techniques, which is quite significant. Two test modes, six classification algorithm performance evaluation approaches, and two clustering and model calculation time evaluation metrics basically were employed to kind of compose three separate datasets for the assessment procedure, which particularly is quite significant.

After the top two individual classifiers(RF and GB) stacking is performed.Then voting is performed layer by layer.Finally clustering and then analysis is performed.

5. Results and Description

1. Phishing link classifier

Used the same legitimate URL "google.com" on each algorithm and found the following accuracy:

Random Forest Classifier:
Accuracy: 70%

Logistic regression classifier:

```
CONFUSION MATRIX

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

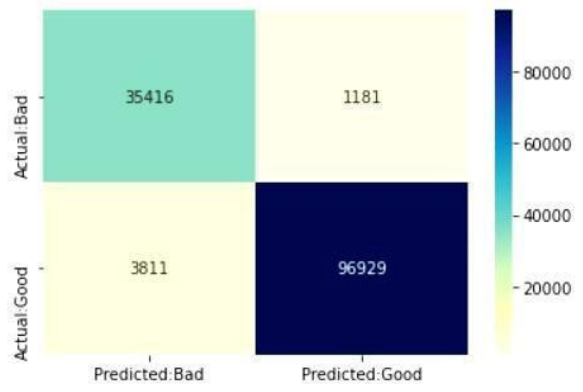


Fig: Confusion Matrix

MultinomialNB classifier:

```
         CONFUSION MATRIX

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

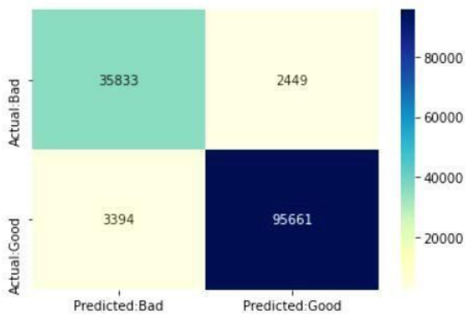


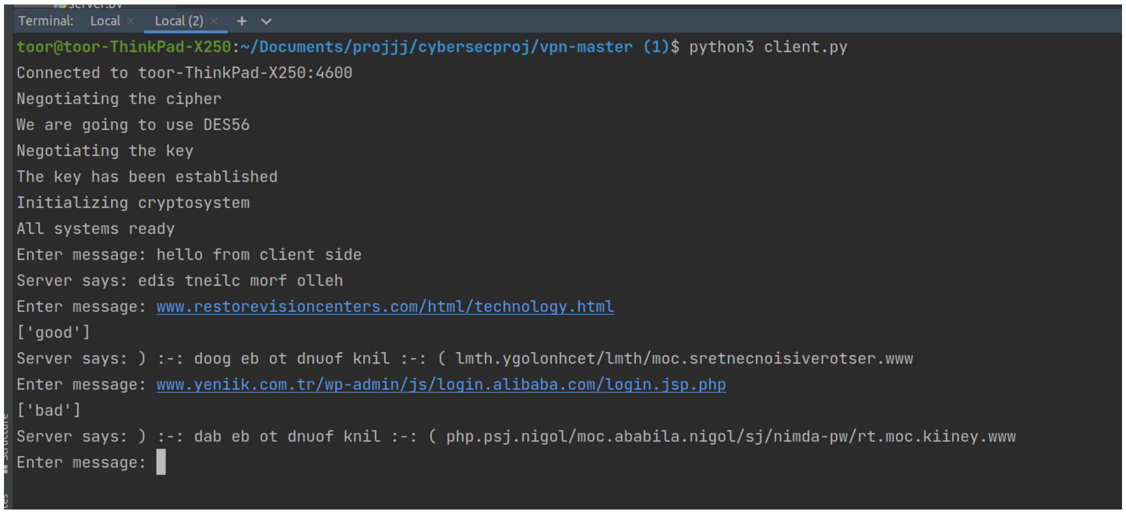
Fig: Confusion Matrix

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 produces the best result.

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

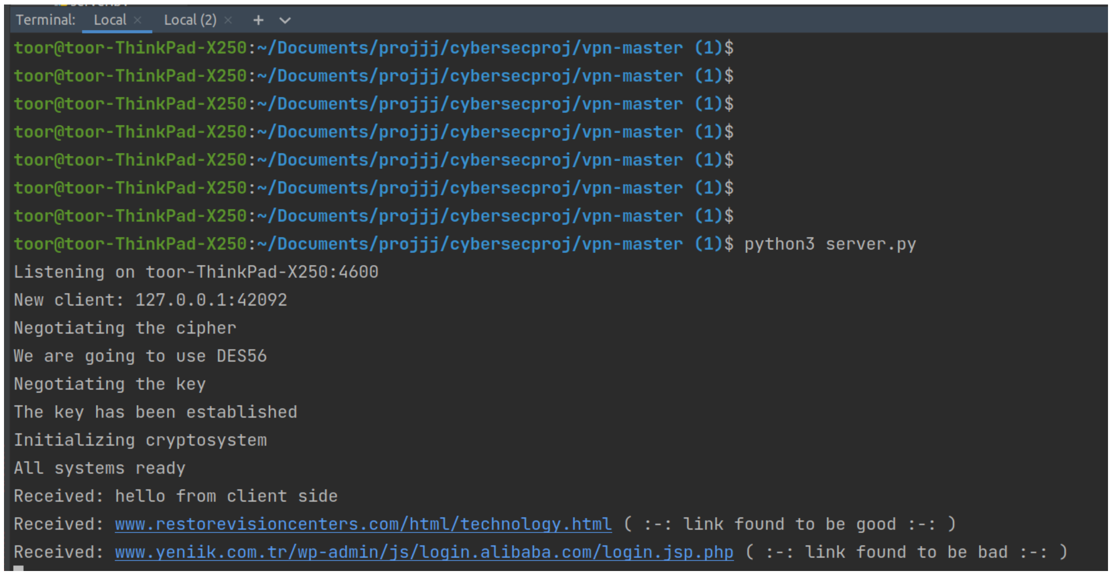
2. Chat application with phishing link detection ability

The phishing pkl file after model dumping is used in a vpn chat application to detect phishing links when sent to protect the customer from scams.



```
toor@toor-ThinkPad-X250:~/Documents/projjj/cybersecproj/vpn-master (1)$ python3 client.py
Connected to toor-ThinkPad-X250:4600
Negotiating the cipher
We are going to use DES56
Negotiating the key
The key has been established
Initializing cryptosystem
All systems ready
Enter message: hello from client side
Server says: edis tneilc morf olleh
Enter message: www.restorevisioncenters.com/html/technology.html
['good']
Server says: ) :-: doog eb ot dnuof knil :-: ( lmth.ygolonhcet/lmth/moc.sretnecnoisiverotser.www
Enter message: www.yeniik.com.tr/wp-admin/js/login.alibaba.com/login.jsp.php
['bad']
Server says: ) :-: dab eb ot dnuof knil :-: ( php.psj.nigol/moc.ababila.nigol/sj/nimda-pw/rt.moc.kiiney.www
Enter message: 
```

Fig : Client side(have sent one good link one bad link)

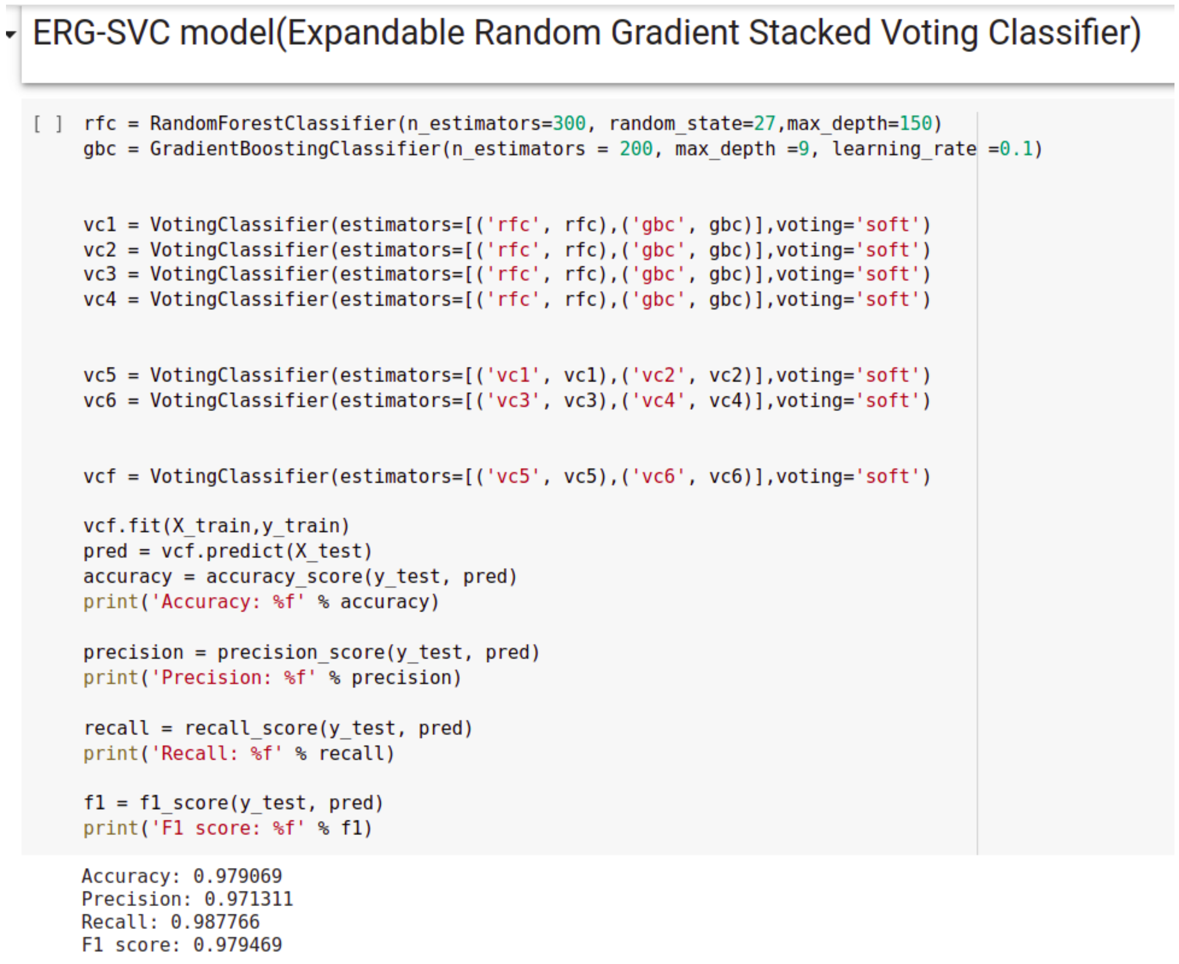


```
toor@toor-ThinkPad-X250:~/Documents/projjj/cybersecproj/vpn-master (1)$
toor@toor-ThinkPad-X250:~/Documents/projjj/cybersecproj/vpn-master (1)$
toor@toor-ThinkPad-X250:~/Documents/projjj/cybersecproj/vpn-master (1)$
toor@toor-ThinkPad-X250:~/Documents/projjj/cybersecproj/vpn-master (1)$
toor@toor-ThinkPad-X250:~/Documents/projjj/cybersecproj/vpn-master (1)$
toor@toor-ThinkPad-X250:~/Documents/projjj/cybersecproj/vpn-master (1)$
toor@toor-ThinkPad-X250:~/Documents/projjj/cybersecproj/vpn-master (1)$
toor@toor-ThinkPad-X250:~/Documents/projjj/cybersecproj/vpn-master (1)$ python3 server.py
Listening on toor-ThinkPad-X250:4600
New client: 127.0.0.1:42092
Negotiating the cipher
We are going to use DES56
Negotiating the key
The key has been established
Initializing cryptosystem
All systems ready
Received: hello from client side
Received: www.restorevisioncenters.com/html/technology.html ( :-: link found to be good :-: )
Received: www.yeniik.com.tr/wp-admin/js/login.alibaba.com/login.jsp.php ( :-: link found to be bad :-: )
```

Fig : Server side (one good link one bad link)

3. Expandable Random Gradient Stacked Voting Classifier

When compared with several models


ERG-SVC model(Expandable Random Gradient Stacked Voting Classifier)

```
[ ] rfc = RandomForestClassifier(n_estimators=300, random_state=27,max_depth=150)
    gbc = GradientBoostingClassifier(n_estimators = 200, max_depth =9, learning_rate =0.1)


    vc1 = VotingClassifier(estimators=[('rfc', rfc),('gbc', gbc)],voting='soft')
    vc2 = VotingClassifier(estimators=[('rfc', rfc),('gbc', gbc)],voting='soft')
    vc3 = VotingClassifier(estimators=[('rfc', rfc),('gbc', gbc)],voting='soft')
    vc4 = VotingClassifier(estimators=[('rfc', rfc),('gbc', gbc)],voting='soft')


    vc5 = VotingClassifier(estimators=[('vc1', vc1),('vc2', vc2)],voting='soft')
    vc6 = VotingClassifier(estimators=[('vc3', vc3),('vc4', vc4)],voting='soft')


    vcf = VotingClassifier(estimators=[('vc5', vc5),('vc6', vc6)],voting='soft')

    vcf.fit(X_train,y_train)
    pred = vcf.predict(X_test)
    accuracy = accuracy_score(y_test, pred)
    print('Accuracy: %f' % accuracy)

    precision = precision_score(y_test, pred)
    print('Precision: %f' % precision)

    recall = recall_score(y_test, pred)
    print('Recall: %f' % recall)

    f1 = f1_score(y_test, pred)
    print('F1 score: %f' % f1)
```

```
Accuracy: 0.979069
Precision: 0.971311
Recall: 0.987766
F1 score: 0.979469
```

6. Conclusion

Out of RF, MultinomialNB and Logistic Regression, Logistic Regression gave the best results hence a sklearn pipeline was created based on LR and model was dumped in pkl form and used in an VPN application. Logistic regression can be modified or combined with other ML models for better results.
In nature, ensemble models take significantly longer to execute than simple models. If the model needs to be put on a cloud platform for model retraining circumstances, more fees will be incurred.The results demonstrate that the GB classifier outperformed with the fewest NLP-based features, obtaining a prediction accuracy of 98.118 percent. Furthermore, the stacking ensemble model and suggested voting ensemble model (ERG-SVC) surpassed other evaluated approaches in detecting malicious URLs, with rates of 98.23% and 98.27%, respectively, of reliable prediction accuracy.

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 targets based on a semantic link network." Future Generation Computer Systems 26.3 (2010): 381- 38

[3] Almomani, Ammar, et al. "Evolving fuzzy neural networks 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.

[6] Purvi Pujara, MB Chaudhari. "Phishing website detection using machine learning." International Journal of Scientific Research in Computer Science, Engineering and Information Technology 3 (7), 395-399, 2018

[7] Waleed Ali."Phishing website detection based on supervised machine learning with wrapper features selection". International Journal of Advanced Computer Science and Applications 8 (9), 72-78, 2017

[8] Abdulhamit Subasi, Esraa Molah, Fatin Almkallawi, Touseef J Chaudhery. "Intelligent phishing website detection using random forest classifier". 2017 International conference on electrical and computing technologies and applications (ICECTA), 1-5, 2017

[9] Ekta Gandotra, Deepak Gupta. "Improving spoofed website detection using machine learning". Cybernetics and Systems 52 (2), 169-190, 2021

[10] Ammara Zamir, Hikmat Ullah Khan, Tassawar Iqbal, Nazish Yousaf, Farah Aslam, Almas Anjum, Maryam Hamdani. "Phishing website detection using diverse machine learning algorithms". The Electronic Library, 2020

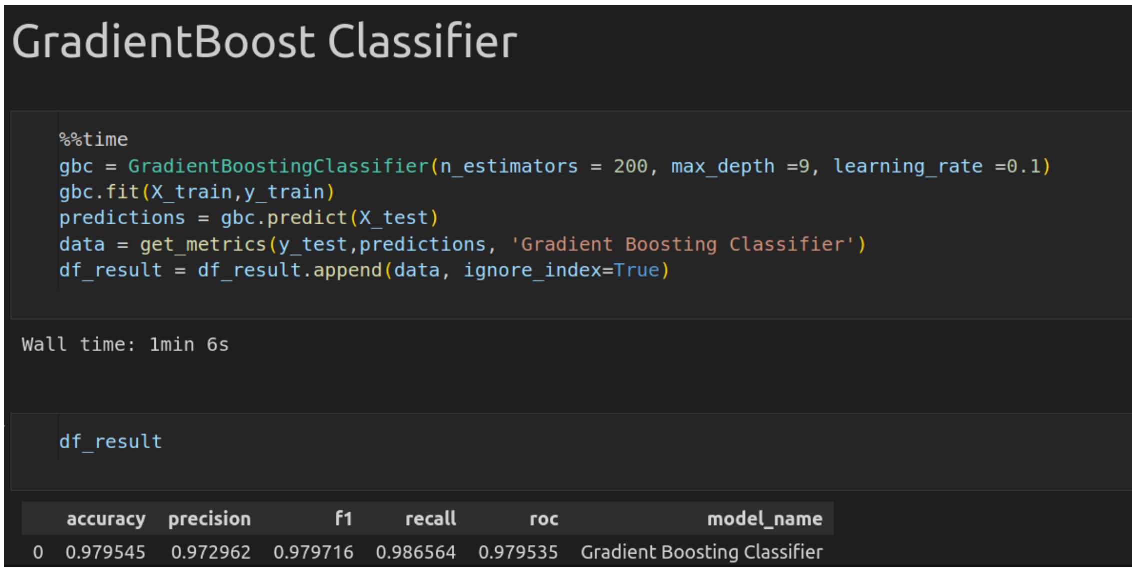
[11] Kulkarni, Arun D., and Leonard L. Brown III. "Phishing website detection using machine learning." (2019).

[12] Indrasiri, P. L., Halgamuge, M. N., & Mohammad, A. (2021). Robust Ensemble Machine Learning Model for Filtering Phishing URLs: Expandable Random Gradient Stacked Voting Classifier (ERG-SVC). IEEE Access, 9, 150142-150161.

Appendix

Top 2 classifier models out of many came out to be Gradient Boost Classifier and random forest ML model hence can be used for ensemble model.

1. Gradient Boost Classifier



GradientBoost Classifier

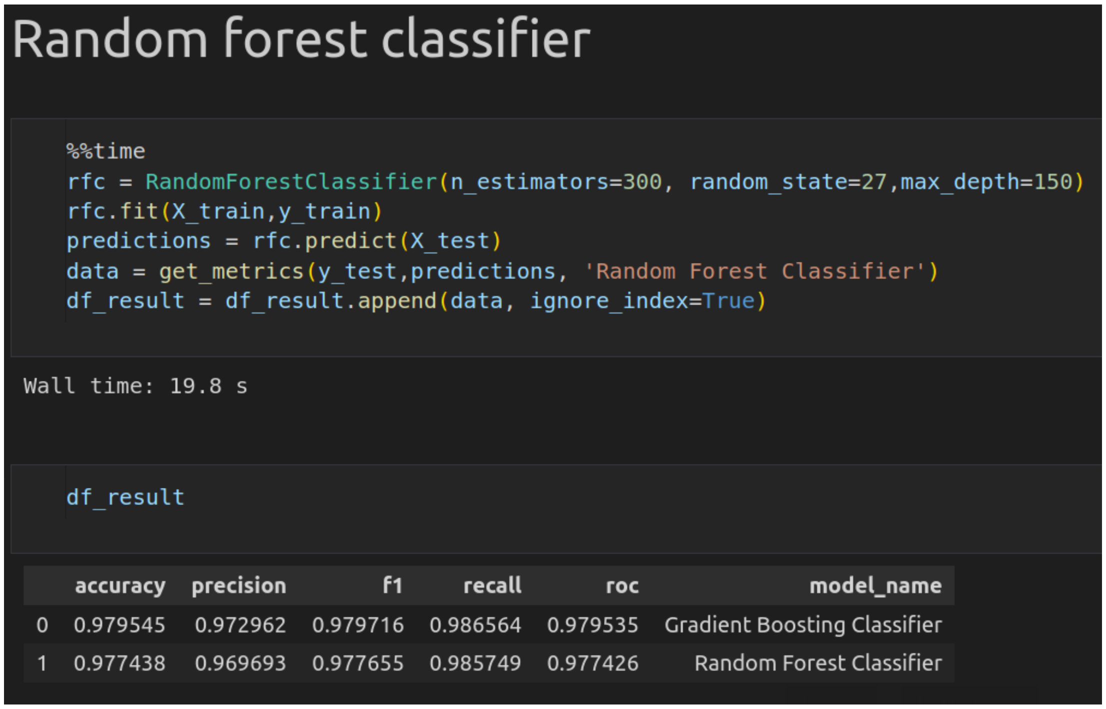
```
%%time
gbc = GradientBoostingClassifier(n_estimators = 200, max_depth =9, learning_rate =0.1)
gbc.fit(X_train,y_train)
predictions = gbc.predict(X_test)
data = get_metrics(y_test,predictions, 'Gradient Boosting Classifier')
df_result = df_result.append(data, ignore_index=True)
```

```
Wall time: 1min 6s
```

```
df_result
```

	accuracy	precision	f1	recall	roc	model_name
0	0.979545	0.972962	0.979716	0.986564	0.979535	Gradient Boosting Classifier

2. Random Forest Classifier


Random forest classifier

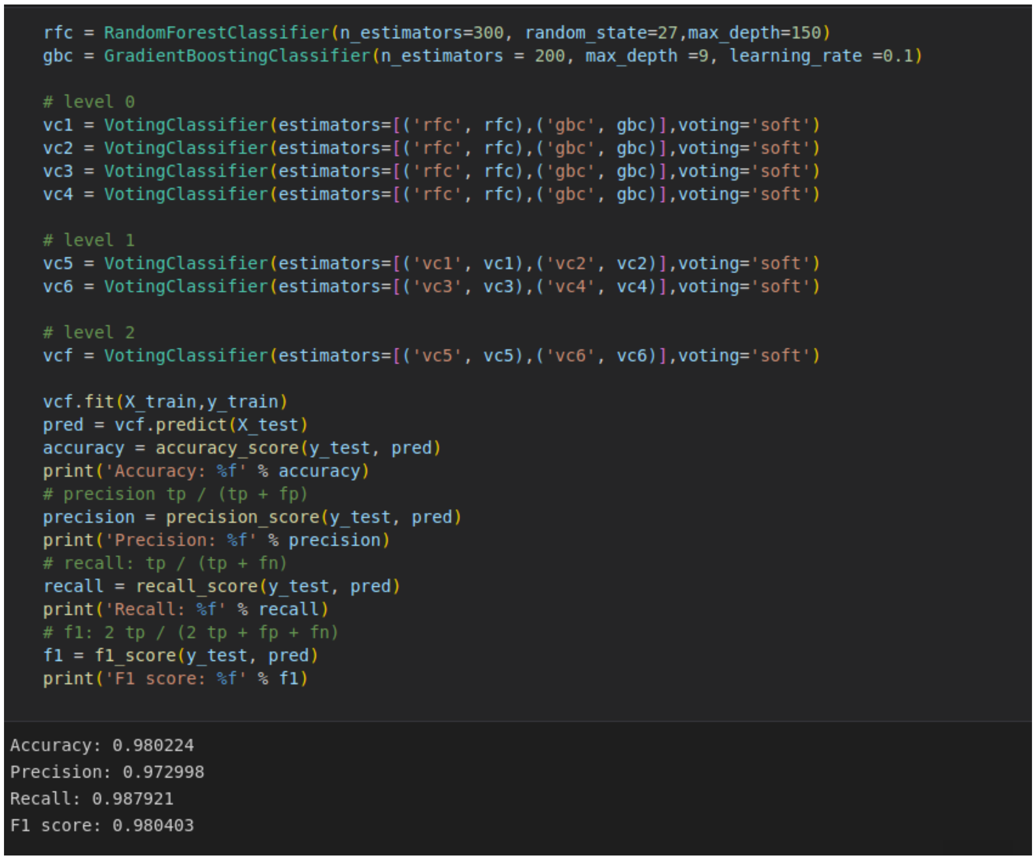
```
%%time
rfc = RandomForestClassifier(n_estimators=300, random_state=27,max_depth=150)
rfc.fit(X_train,y_train)
predictions = rfc.predict(X_test)
data = get_metrics(y_test,predictions, 'Random Forest Classifier')
df_result = df_result.append(data, ignore_index=True)
```

```
Wall time: 19.8 s
```

```
df_result
```

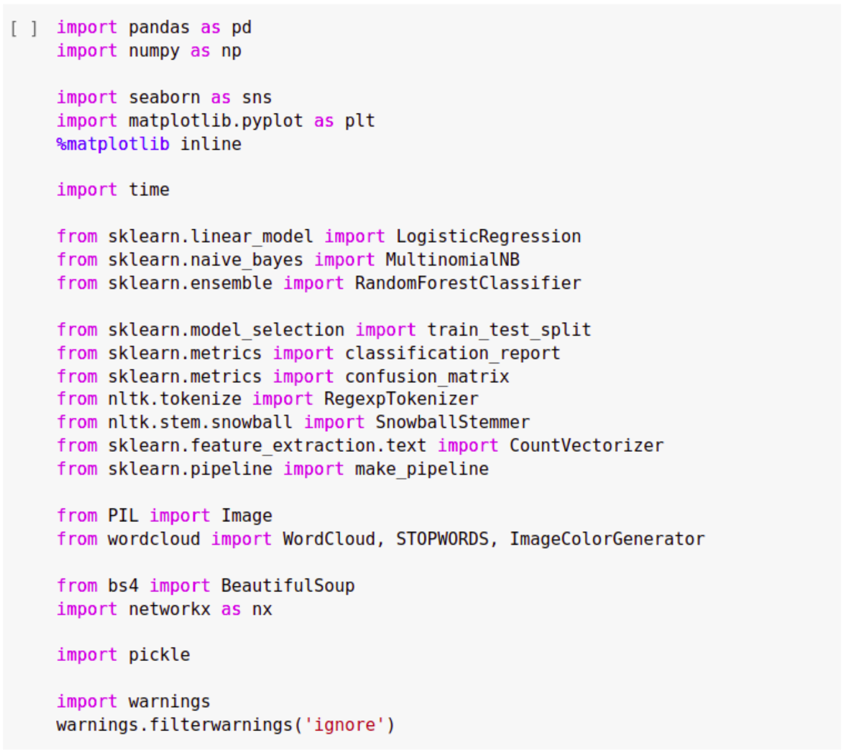
	accuracy	precision	f1	recall	roc	model_name
0	0.979545	0.972962	0.979716	0.986564	0.979535	Gradient Boosting Classifier
1	0.977438	0.969693	0.977655	0.985749	0.977426	Random Forest Classifier

3. Proposed Ensemble model


```
rfc = RandomForestClassifier(n_estimators=300, random_state=27,max_depth=150)
gbc = GradientBoostingClassifier(n_estimators = 200, max_depth =9, learning_rate =0.1)

# level 0
vc1 = VotingClassifier(estimators=[('rfc', rfc),('gbc', gbc)],voting='soft')
vc2 = VotingClassifier(estimators=[('rfc', rfc),('gbc', gbc)],voting='soft')
vc3 = VotingClassifier(estimators=[('rfc', rfc),('gbc', gbc)],voting='soft')
vc4 = VotingClassifier(estimators=[('rfc', rfc),('gbc', gbc)],voting='soft')

# level 1
vc5 = VotingClassifier(estimators=[('vc1', vc1),('vc2', vc2)],voting='soft')
vc6 = VotingClassifier(estimators=[('vc3', vc3),('vc4', vc4)],voting='soft')

# level 2
vcf = VotingClassifier(estimators=[('vc5', vc5),('vc6', vc6)],voting='soft')

vcf.fit(X_train,y_train)
pred = vcf.predict(X_test)
accuracy = accuracy_score(y_test, pred)
print('Accuracy: %f' % accuracy)
# precision tp / (tp + fp)
precision = precision_score(y_test, pred)
print('Precision: %f' % precision)
# recall: tp / (tp + fn)
recall = recall_score(y_test, pred)
print('Recall: %f' % recall)
# f1: 2 tp / (2 tp + fp + fn)
f1 = f1_score(y_test, pred)
print('F1 score: %f' % f1)
```

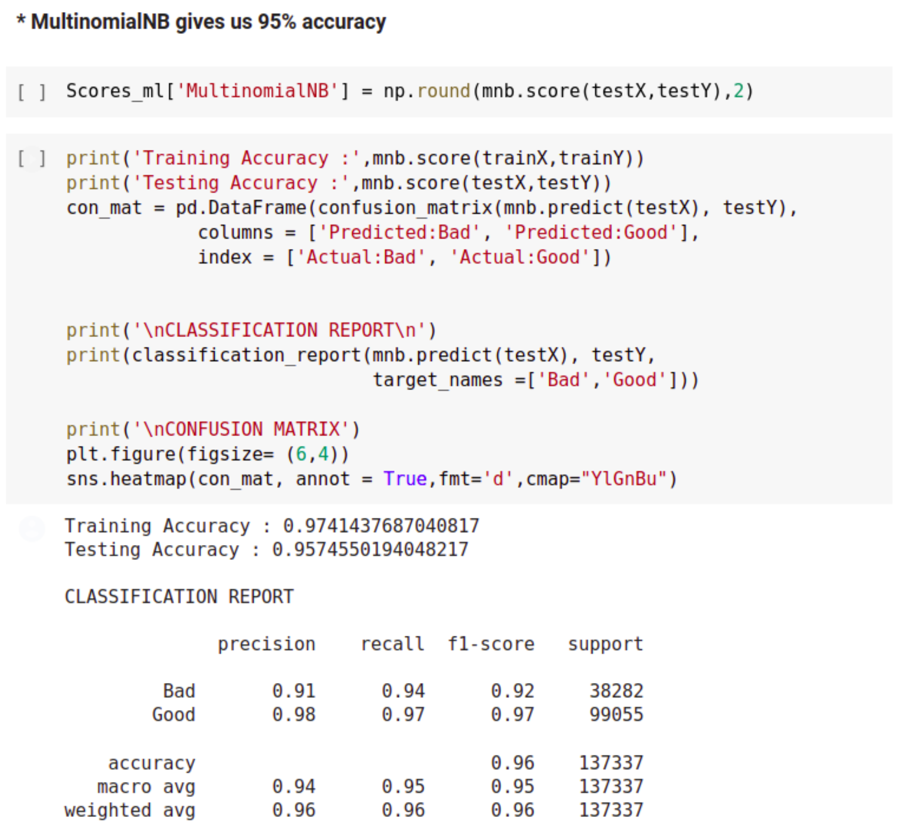
```
Accuracy: 0.980224
Precision: 0.972998
Recall: 0.987921
F1 score: 0.980403
```

4. Important imports



```python
import pandas as pd
import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import time

from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from nltk.tokenize import RegexpTokenizer
from nltk.stem.snowball import SnowballStemmer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.pipeline import make_pipeline

from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator

from bs4 import BeautifulSoup
import networkx as nx

import pickle

import warnings
warnings.filterwarnings('ignore')
```

5. Multinomial NB



*** MultinomialNB gives us 95% accuracy**

```
[ ] Scores_ml['MultinomialNB'] = np.round(mnb.score(testX,testY),2)
```

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


    print('\nCLASSIFICATION REPORT\n')
    print(classification_report(mnb.predict(testX), testY,
                                target_names =['Bad','Good']))

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

```
Training Accuracy : 0.9741437687040817
Testing Accuracy : 0.9574550194048217

CLASSIFICATION REPORT

              precision    recall  f1-score   support

         Bad       0.91      0.94      0.92     38282
        Good       0.98      0.97      0.97     99055

    accuracy                           0.96    137337
   macro avg       0.94      0.95      0.95    137337
weighted avg       0.96      0.96      0.96    137337
```

7. Logistic Regression

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

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


print('\nCLASSIFICATION REPORT\n')
print(classification_report(lr.predict(testX), testY,
                            target_names =['Bad','Good']))

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

```
Training Accuracy : 0.9782480479795345
Testing Accuracy : 0.9636514559077306

CLASSIFICATION REPORT

              precision    recall  f1-score   support

         Bad       0.90      0.97      0.93     36597
        Good       0.99      0.96      0.97    100740

    accuracy                           0.96    137337
   macro avg       0.95      0.96      0.95    137337
weighted avg       0.97      0.96      0.96    137337
```

8. Dump sklearn pipeline

```
[ ]  pickle.dump(pipeline_ls,open('phishing.pkl','wb'))
```

```
[ ]  loaded_model = pickle.load(open('phishing.pkl', 'rb'))
     result = loaded_model.score(testX,testY)
     print(result)
```

```
0.9674450439430016
```