Report check

by Muddasar Razzaq

Submission date: 28-Apr-2022 06:32AM (UTC+0100)
Submission ID: 177586301
File name: Final_20Report_20v5.docx (688.68K)
Word count: 3841
Character count: 19635

Phishing URLs detection using ensemble machine learning

With web application

***Abstract*— The transparency of the Web uncovered open doors for lawbreakers to transfer malignant substances. As a matter of fact, notwithstanding broad exploration, email-based spam separating strategies can't safeguard other web administrations. Hence, a countermeasure should be taken that sums up across web administrations to safeguard the client from phishing host URLs. This project portrays a methodology that characterizes URLs naturally founded on their lexical and have based highlights. Bunching is performed on the whole dataset and a group ID (or name) is determined for every URL, which thusly is utilized as a prescient element by the arrangement framework. Online URL notoriety administrations are utilized to order URLs and the classes returned are utilized as a supplemental wellspring of data that would empower the framework to rank URLs. The ensemble classifier accomplishes 96% precision by recognizing countless phishing has, while keeping a humble bogus positive rate. URL bunching, URL characterization, and URL order instruments work related to giving URLs a position, along with machine learning, we have designed a web-based application that will be helpful to detect the URL as spam or not graphically, on the machine learning side, we have trained three different ensemble models, the list included three major models like Gradient Boosting, XGBoost and Random Forest, we use the ensemble learning way and gathered these models into one base voting classifier.**

Keywords: URL, machine learning, web application, phishing

I. Introduction

Throughout recent years, the Web has seen a huge development in the number and sorts of web benefits that incorporate social organizing, discussions, online journals, and video sharing locales [1]. As these web administrations drive new open doors for individuals to cooperate, they likewise set out new open doors for lawbreakers. The reality that Google identifies around 300,000 noxious sites every month is an obvious sign and verification that these amazing open doors are widely utilized by lawbreakers [2]. Phishing sites are utilized to take individual data, for example, Mastercard also, passwords, and to carry out drive-by downloads. Hoodlums likewise utilize phishing to draw clients into visiting their phony sites [3]. The vector that is utilized to trap clients is a Uniform Resource Finder (URL). The client should perform second looks for good measures previously clicking a URL, like giving close consideration to the spelling of the site's location and assessing the related gamble that may be experienced by visiting the URL [4]. Late work has shown that security professionals have created strategies, for example, boycotting, to shield clients from phishing sites.

Their procedure of utilizing honey-profiles to recognize single spam bots as well as largescale crusades doesn't give total assurance to clients. Interpersonal organizations comprise constant collaboration, yet at the same, the strategy for identifying fake records can bring about delays because of the requirement for building a profile of the improper movement. The ongoing work proposes a framework equipped for bunching, arranging, sorting, and positioning URLs progressively while adjusting to new and advancing patterns in URL qualities.

An endeavor to work on existing strategies for distinguishing phishing URLs is made by adding novel elements (for example indicator factors), and making the learning system more productive by utilizing Mahout, an Apache project containing versatile AI calculations [5]. Online calculations adjust to changes rapidly and are a superior decision since the qualities for indicator factors of URLs can change after some time [6].

One significant commitment of the ongoing paper is the execution of a crossbreed approach consolidating both grouping and order. Grouping is performed before order, what's more, group IDs from it are added as a bunch step indicator variable to grow each element vector present in the dataset. The most common way of performing grouping and utilizing bunch IDs (or bunch names) helps in acquiring a hyperplane with a bigger edge which thusly brings about higher classifier productivity because of better speculation. Another major commitment of the ongoing paper is URL arrangement by Microsoft Reputation Services (MRS) [7]. The classifications acquired from MRS are utilized to perform positioning of URLs by

utilizing a clever URL Ranking methodology, which comprises of utilizing the order result and an inward scale (Red/Yellow/Green) of seriousness determined for every URL.

Problem Statement

There are several methods for enhancing the level of security. Thus, the database administrator's primary objective is to detect harmful activity associated with the websites. People from all over the world have been the victims of fake websites and spam emails, there are various techniques used by different internet services providers, web hosting, etc. To avoid such harmful activities, we are going to make a predictive model based on historical data to predict the URL as spam or not spam. Many works have been done on this, but the way we are predicting the URLs as spam or not is by using the ensemble learning technique.

Aim

The main aim of this project is to use machine learning ensemble models to predict that either the URL is spam or not spam. We will also design a web-based application that will be integrated with the model we trained; the web application will predict the URL when a user enters any. The overall project is useful and can be implemented in the industry.

Objective

- To perform basic data analysis on the historical data
- To implement a Machine learning-ensemble model which will be able to predict the URL is spam or not
- To make a web-based application that can be used in real-time for predictions

II. Literature Review

Presently, various diaries, gatherings have unique studies and exploration for identification of phishing sites and one of the methodologies that had been proposed was a multi-level arrangement for phishing URL sifting. In this, the creators set forward an inventive technique for separating phishing URL highlights because of the weightage of message content [8]. A numerous arrangement calculation is utilized which incorporates SVM, AdaBoost, and Naive Bayes. These calculations are partitioned into three levels utilizing 21 fixed at this point various elements [8]. Then a two-venture methodology happens with the assistance of Procedures of the second International Conference on Inventive Communication and Computational Technologies (ICICCT 2018) IEEE Xplore Compliant - Part Number: CFP18BAC-ART; ISBN:978-1-5386-1974-2 another characterization calculation however the issue here is the time consumed and the intricacy in question, overheads involved and the presentation issues and subsequently this wasn't an ideal technique.

One more technique that was taken on by the creators of one of the IEEE 2017 papers was the example acknowledgment for example unique highlights are separated from messages to acquire a model that makes a difference to segregate phishing messages from non-phishing ones [9]. One of the essential techniques utilized in this setting is to perceive the assaults and afterward utilize include extraction and afterward grouping. The principal impediment of this proposition is that there are an excessive number of elements that are assessed without taking into account whether they truly are fundamental to distinguish phishing. Subsequently, it could prompt superfluous computational expense.

According to the International Journal of Advance research and innovative ideas in education (IJARIIE) journal paper, the Multi-Label Classifier based Associative Classification (MCAC) data mining approach is also one of the methods that are used for detecting phishing websites. The associative classification algorithm detects phishing websites with mediocre accuracy. [10] MCAC consists of three main steps which are Rule discovery, classifier building, and class assignment. In the first step of this algorithm, rules are found and extracted by iterating over the training data set (historical websites features or data collected from various sources).

[10] In this step, the merging of any of the resulting rules that have the same antecedent (left-hand side) takes place and is linked with different classes to produce the multi-label rules. Along with this, redundant rules are eliminated. The outcome of the second step is the classifier which contains single and multilabel rules. The last step involves testing the classifier on the testing data set to measure its performance. In the prediction process, the rule in the classifier group which matches the test data features is often fired to guess its type (class). The MCAC algorithm generates rules further those rules are sorted by using the sorting algorithm. [10]. The main problem that MCAC faced was difficulty in determining minimum confidence and minimum support when there is a large amount of data later on there came more sophisticated algorithms to replace this which were more accurate and had lesser time complexity.

2. Methodology

Our proposition for the previously mentioned subject is to work on productivity in the discovery of phishing sites. A ton of examination work and overview was done to think about different characterization calculations to best accommodate our model. We have the data of different website URLs in CSV; the goal is to first perform the basic analysis of the data to get the deep-down insights from the data. The next step is to clean the data and prepare it for machine learning models. On the machine learning side, we are using the ensemble learning technique, we will train three ensemble models on the data, then we will use a base voting classifier based on the three trained models to perform the predictions.

We are training the following three ensemble models.

- Gradient Boosting
- Random Forest
- Xgboost

Before training any machine learning model, we have to perform some data pre-processing to model the data for machine learning. What we have done in data pre-processing is given below.

Data Preprocessing

data preprocessing can allude to the control or dropping of information before it is utilized to guarantee or improve execution, and is a significant stage in the information mining process. The expression "trash in, trash out" is especially relevant to information mining and AI projects. We are following the following steps.

- Loading the data
- Familiarizing with data & EDA
- Visualizing the data
- Splitting the data
- Training the data
- Comparison of Model
- Conclusion

Loading the data

Data loading is the process to add the data from your computer into a Python environment to perform operations on the data. We have data in CSV format, we are using Python's Pandas library to import and load the data into a Pandas data frame. We will apply all the required data pre-processing after uploading the data into Jupyter Notebook.

Familiarizing with data & EDA

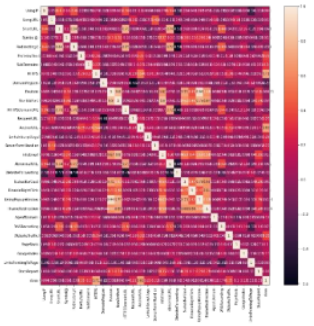
This is the main part of our project, before going to analyze the data, the first thing to perform is to familiarize yourself with the data. After taking a look at the data, we found the following information.

- There are 11054 instances and 31 features in the dataset.
- Out of which 30 are independent features whereas 1 is the dependent feature.
- Each feature is in int data type, so there is no need to use Label Encoder.
- There is no outlier present in the dataset.
- There is no missing value in the dataset.

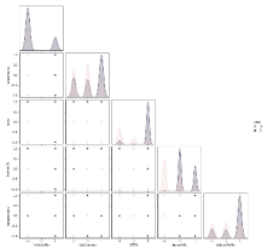
Visualizing the data

To dive deep into the data, we did some basic visualization to gain more knowledge about the dataset we have. Few plots and graphs are displayed to find how the data is distributed and how features are related to each other.

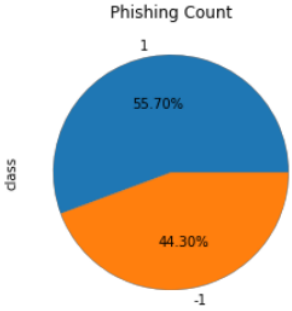
We have drawn the correlation plot to see the correlation of features in our data with each other.



In the correlation plot, we saw some features are highly correlated. We are now plotting the pair plot of those features to see their distribution in the data very well.



In our data, we have a feature which is the class labeled feature, we will predict this feature based on all the other independent features. This feature has values as 1 and -1, here 1 shows that the URL is spam and -1 shows that the URL is not spam, let's see the total distribution of 1 and -1 in our dataset.



We can see that there are 55% of URLs in our dataset are spam and 44% are not spam. After getting some insights from our data, we will move ahead.

Splitting the data

After performing some basic analysis on our data, the next step we took is to split the data into test and train sets. The train set will be used to train our models and we will use the test set to test our models and perform some predictions on our model using the testing set. We are splitting the dataset into an 80:20 ratio which means that 80% of the data will be used as training data and the remaining 20% will be used for testing purposes.

Training the data

After performing data analysis, visualization, and splitting the data into test and train sets, the next step was to import the models we are going to train and then train them using the training data and test them and perform predictions on the testing data. We are training 10 models to see the best model for our project.

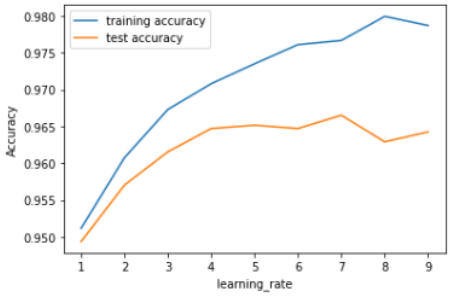
1. Gradient Boosting Classifier

The first ensemble model we trained is the Gradient Boosting model. Gradient boosting classifiers are a gathering of AI and machine learning algorithms that join numerous frail learning models together to make a solid prescient model. Choice trees are typically utilized while doing slope helping.

We trained the Gradient Boosting classifier bypassing some of its parameters as max_depth=4, learning_rate=0.7, we got the accuracy same as Random Forest gave which is 97%.

	precision	recall	f1-score	support
-1	0.99	0.96	0.97	976
1	0.97	0.99	0.98	1235
accuracy			0.97	2211
macro avg	0.98	0.97	0.97	2211
weighted avg	0.97	0.97	0.97	2211

The training and testing accuracy graph is given.



We tried GridSearchCV to perform the parameters tuning of the model, the purpose of doing so is to see if we can increase the accuracy to some extent?

```
from scipy.stats import uniform as sp_randFloat
from scipy.stats import randint as sp_randInt
gbr = GradientBoostingClassifier()
parameters = {"learning_rate": sp_randFloat(),
              "subsample"    : sp_randFloat(),
              "n_estimators" : sp_randInt(100, 1000),
              "max_depth"    : sp_randInt(4, 10)
             }
```

We performed the parameters tuning and trained the model on the best parameters, the accuracy we got is 95%.

2. *XGBoost Classifier*

XGBoost, which represents Extreme Gradient Boosting, is an adaptable, circulated angle helped choice tree (GBDT) AI library. It gives equal tree support and is the main AI library for relapse, grouping, and positioning issues.

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance that is dominative competitive machine learning. In this post, you will discover how you can install and create your first XGBoost model in Python

The accuracy the XGBoost classifier gave on testing data was also 96%.

```
XGBoost Classifier : Accuracy on training Data: 0.987
XGBoost Classifier : Accuracy on test Data: 0.969

XGBoost Classifier : f1_score on training Data: 0.988
XGBoost Classifier : f1_score on test Data: 0.973
```

We got an accuracy of 98% on the training set and 96% on the testing dataset.

We also performed the GridSearchSV parameter tuning to see if we can increase the accuracy.

```
from xgboost import XGBClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
import time

# A parameter grid for XGBoost
params = {
    'n_estimators':[500],
    'min_child_weight':[4,5],
    'gamma':[i/10.0 for i in range(3,6)],
    'subsample':[i/10.0 for i in range(6,11)],
    'colsample_bytree':[i/10.0 for i in range(6,11)],
    'max_depth': [2,3,4,6,7],
    'objective': ['reg:squarederror', 'reg:tweedie'],
    'booster': ['gbtree', 'gblinear'],
    'eval_metric': ['rmse'],
    'eta': [i/10.0 for i in range(3,6)],
}

xgb_tuned = XGBClassifier(nthreads=-1)

# run randomized search
n_iter_search = 100
random_search = RandomizedSearchCV(xgb_tuned, param_distributions=params,
                                   n_iter=n_iter_search, cv=5, scoring='neg_mean_squared_error')

start = time.time()
random_search.fit(X_train, y_train)
print("RandomizedSearchCV took %.2f seconds for %d candidates"
      " parameter settings." % ((time.time() - start), n_iter_search))
```

We trained the XGBoost again on the suitable parameters using GridSearchCV, we got the accuracy of 96% on testing data which is the same as we got without parameter tuning.

3. *Random Forest Classifier*

Random forest or irregular choice trees is a group learning strategy for arrangement, relapse, and different errands that works by building a huge number of choice trees at preparation time. For arrangement errands, the result of the arbitrary timberland is the class chosen by most trees.

Random Forest is a well-known AI and machine learning algorithm that has a place with the managed learning method. It very well may be utilized for both Classification and Regression issues in ML. It depends on the idea of troupe realizing, which is a course of joining various classifiers to take care of a complicated issue and to work on the presentation of the model.

We trained the Random Forest classifier, and tested it on the testing data, the model gave us an accuracy of 96%.

We performed GridSearchCV on random forest too to train it on the suitable parameters, we got the accuracy of 96% after parameters tuning.

Ensembling the models

We trained three different machine learning ensemble models, the goal of doing so was to check that which model is performing well on our data.

The goal of this project is to predict that either a URL is spam or not. We trained these three ensemble models and we got the accuracy above 90%. Now we will put all the trained models into one base voting model and train it again, this is the ensemble learning way.

An ensemble is a machine learning model that combines the predictions from two or more models. The models that contribute to the ensemble, referred to as ensemble members, may be the same type or different types and may or may not be trained on the same training data.

We will combine all the three models into one base voting classifier. A Voting Classifier is a machine learning model that trains on an ensemble of numerous models and predicts an output (class) based on their highest probability of chosen class as the output.

```
from sklearn.ensemble import VotingClassifier
votingC = VotingClassifier(estimators=[('rfc', rand_tuned),
('xgb', xgb_tuned),('gbc',gbr_tuned)], voting='soft', n_jobs=4)

votingC = votingC.fit(X_train, y_train)
```

```
votingC.score(X_test,y_test)
```

```
0.9696969696969697
```

We trained a voting classifier as a base model and passed all the trained model into it. Upon testing it on the test data, the voting ensemble model gave the accuracy of 96%.

So, we converted the voting ensemble model into a pickle file, we will now use this model for our

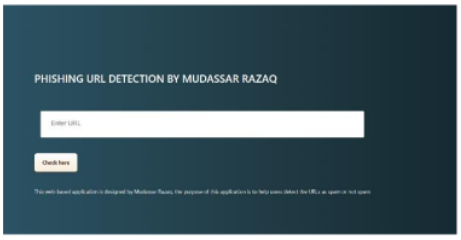
predictions, and we will integrate it into a web-based application through Flask APIs.

Web-based Application

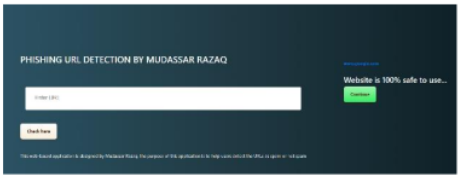
The goal of the project was to train three different ensemble models on the data we have, after training all the model, we will use a base voting model and combine all the ensemble trained models into it, we will train the base model again by passing the three models, then we will integrate the model into a web-based application which will then predict a URL we enter as spam or not.

The User Interface of the web-based application is designed in HTML and CSS. We used the Flask APIs to integrate the model we have selected into the web-based application.

The Flask APIs is the bridge interface between the application and the model we have trained on the historical data. When a user enters ULR in the application, the Flask APIs will take the URL we entered and feed it into the model, the model will predict whether the URL is spam or not and give back the output on the screen.



The above is the user interface of the web application, there is a textbox that is used to enter the URL, when we enter the URL in the textbox and press the button, the button will communicate with the model through Flask APIs, the model will take the URL as an input, based on the historical data that the model is trained on, the model will predict that either the URL is spam or not. The demo can be seen in the following pictures.



We have entered Google's URL, the model predicted that the website is 100% safe, which means that the model predicted very well. Let's check for some spammed websites.

By entering the spammed URL, the model predicted that the URL is 100% unsafe.

How it works?

A URL consists of different parts, for example, the scheme, subdomain, top-level domain, second-level domain, and subdirectory, etc. when a user enter a URL in the input field, the URL is taken to the model by Flask APIs, once the model receives the URL, it split the URL into its components, as the machine learning models don't understand string data so it'll first the URL into different parts and assign different numerical numbers to it. For example, if the URL is having HTTPS, then the model will assign a value 1 to it which indicates that the URL is secured but if the entered URL is having HTTP only, it'll assign a negative one (-1) to it as unsecured URL. Every necessary component of a URL which is important in predicting the URL spam or not spam will be converted into numerical values. Based on the numerical data, the model will further perform the predictions and send back the information through Flask APIs and displayed it on the screen.

Discussion

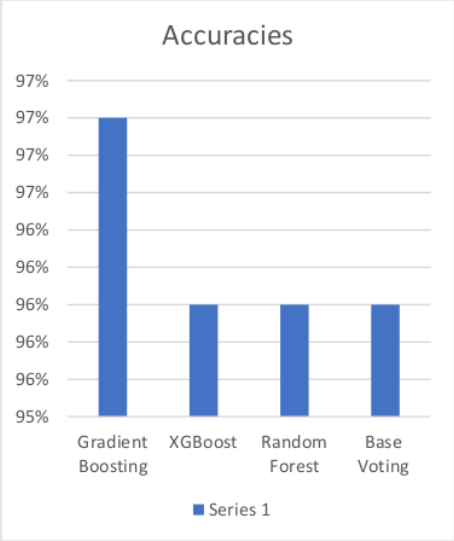
The goal of the project was to train an ensemble base machine learning model on the historical data we have to predict that either the URL is spam or not? Our dataset has the data of different URLs with all their information and each URL is labeled as spammed or not spammed. The ratio of spam URLs is 56% and 44% are not spammed URLs. We trained three different ensemble machine learning models, and the purpose of doing so was ensemble these three models into one base model, this way we can have the strength of all the models in one place.

We got an accuracy of 96% with Gradient Boosting classifier. We moved on to train other models, Random Forest gave us the accuracy of 96% and XGBoost also gave the same accuracy.

We performed the GridSearchCV to find the most suitable parameters for our models, after performing the parameters tuning, we got the same accuracies.

We gathered all the ensemble models into one base voting model, after the training the voting classifier, we got the accuracy of 96%.

We pickled the base voting classifier model to integrate it into a web-based application using Flask APIs, the application will be used to detect the URLs as spammed or not spammed in real-time.



Conclusion

In this project, different methodologies have been used to detect a URL as phishing or not, three different machine learning models have been trained and then we used the base ensemble model called voting classifier, we passed all the trained models into it and trained the voting classifier again and deployment as a pickled model. The performance metrics along with our literature survey also proved the accuracy level of the base voting model Classifier to be the highest around 96.4%.

A web-based application has been designed which is integrated with the model we have selected for predictions; the application will be helpful in the industry to help users know whether the website they are surfing is safe or not.

Future work will intend to create a more intelligent framework that can advance without anyone else about new sorts of phishing assaults by adding a more upgraded element to the identification process.

Bibliography

[1] K. G. C. M. J. P. V. a. S. D. Thomas, "Design," *IEEE Symposium on Security and*, 2011.

[2] E. Mills, "Google finds 9,500 new malicious Web sites a day," CNET News, 2012. [Online]. Available: http://news.cnet.com/8301-1009_3-57455614-.

[3] R. Siciliano, "What is Typosquatting?," McAfee Blog Central, 2013. [Online]. Available: http://blogs.mcafee.com/consumer/what-is-typosquatting.

[4] J. S. L. S. S. a. V. G. Ma, ""Identifying," *International Conference*, 2009.

[5] S. A. R. D. T. a. F. E. Owen, "Mahout In," *Manning Publications*, 2010.

[6] J. S. L. S. S. a. V. G. Ma, "Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs," 2009.

[7] M. R. Services, "Feedback and Error Reporting," http://www.microsoft.com/security/portal/mrs/. [Online].

[8] "A Review of Various Techniques for Detection and Prevention for Phishing attack," *International Journal of Advanced Computer Technology (IJACT).*

[9] "Feature selection for machine learning based detection of phishing websites," *IEEE 2017*, 2017.

[10] B. N. ,. G. A. ,. Prof.T.BhaskarAher Sonali, "Detection of Website Phishing Using MCAC Technique implementation".

[11] G. K. C. a. V. G. Stringhini, “detecting spammers on social media,” *ACSAC’10,* 2010.

[12] G. a. M. K. Orr, “ Neural Networks: Tricks of the Trade,” *Springer,* 1998.

Report check

ORIGINALITY REPORT

25%	5%	19%	12%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Shraddha Parekh, Dhwanil Parikh, Srushti Kotak, Smita Sankhe. "A New Method for Detection of Phishing Websites: URL Detection", 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), 2018 Publication	10%
2	Feroz, Mohammed Nazim, and Susan Mengel. "Phishing URL Detection Using URL Ranking", 2015 IEEE International Congress on Big Data, 2015. Publication	7%
3	machinelearningmastery.com Internet Source	2%
4	Submitted to Coventry University Student Paper	1%
5	mc.ai Internet Source	1%
6	Submitted to University of Central England in Birmingham Student Paper	1%

7	Submitted to University of Ulster Student Paper	1 %
8	R. Kanmani, S. Muthulakshmi, R. Radhapoorani, N. Suganya, K. Sri subitcha, M. Sriranjani. "Prediction of Covid Disease using Advanced IoT and Support Vector Machine", 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS), 2021 Publication	1 %
9	www.coursehero.com Internet Source	<1 %
10	"Information and Communications Security", Springer Science and Business Media LLC, 2020 Publication	<1 %
11	www.ijtsrd.com Internet Source	<1 %

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		