**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 supervisions. Hence, a countermeasure should be taken that sums up across web administrations to safeguard the client from phishing swarm 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

# 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

# Literature Review

As of now, different journals, get-togethers have extraordinary

studies and investigation for distinguishing proof of phishing destinations and one of the approaches that had been proposed was a staggered plan for phishing URL filtering. In this, the makers put forth an imaginative method for isolating phishing URL features as a result of the weightage of message content [8]. A various game plan computation is used which integrates SVM, AdaBoost, and Naive Bayes. These estimations are divided into three levels using 21 fixed now different components [8]. Then a two-adventure system occurs with the help 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 one more portrayal estimation anyway the issue here is the time consumed and the multifaceted design being referred to, overheads included and the show issues and along these lines this was certainly not an optimal method.

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.

As indicated by the International Journal of Advance exploration and inventive thoughts in training (IJARIIE) diary paper, the Multi-Label Classifier based Associative Classification (MCAC) information mining approach is additionally one of the techniques that are utilized for distinguishing phishing sites. The cooperative grouping calculation identifies phishing sites with fair precision. [10] MCAC comprises of three primary advances which are Rule disclosure, classifier building, and class task. In the initial step of this calculation, rules are found and separated by repeating over the preparation informational index (authentic sites elements or information gathered from different sources).

[10] In this progression, the converging of any of the subsequent standards that have a similar precursor (left-hand side) happens and is connected with various classes to deliver the multi-mark rules. Alongside this, repetitive standards are dispensed with. The result of the subsequent advance is the classifier which contains single and multilabel rules. The last advance includes testing the classifier on the testing informational collection to gauge its exhibition. In the forecast interaction, the standard in the classifier bunch which matches the test information highlights is frequently terminated to figure its sort (class). The MCAC calculation creates leads further those rules are arranged by utilizing the arranging calculation. [10]. The fundamental issue that MCAC confronted was trouble in deciding least certainty and least help when there is a lot of information later on there came more refined calculations to supplant this which were more precise and had lesser time intricacy.

# 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 refers to the regulat or dropping of data before it is utilized to guarantee or improve execution, and is an important 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

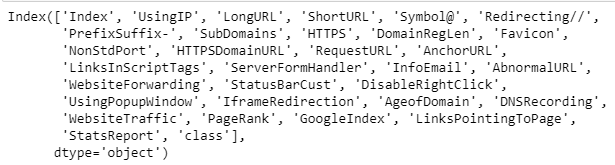
## Loading the data

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.

## About the data

We have the data in CSV format, the dataset is the collection of over 11000+ websites URLs, in the data, we have over 32 different parameters for each website URL and we have a class label as 1 and -1 which indicates that the website is phishing or not?

The features in the data is given below.



## Familiarizing with data & EDA

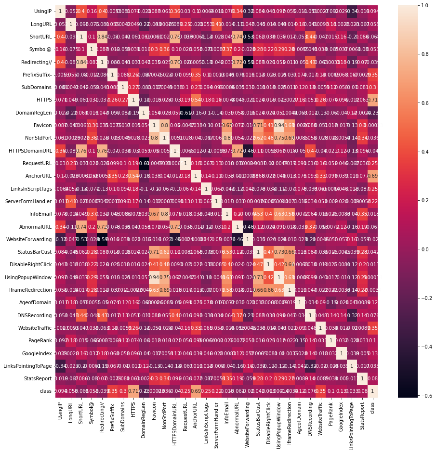
This is the main part of our project, before going to analyze the data, the first thing to perform is to familiarize yourself with the data. After taking a look at the data, we found the following information.

* There are 11054 instances and 32 features in the dataset.
* Out of which 31 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.

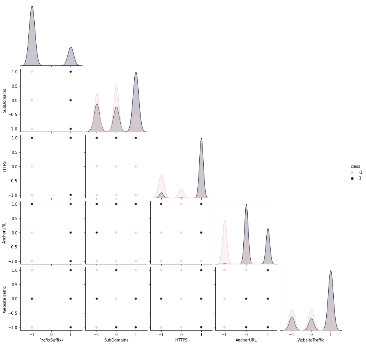
## Visualizing the data

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.

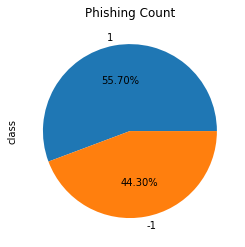
We have drawn the correlation plot to see the correlation of features in our data with each other.



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.

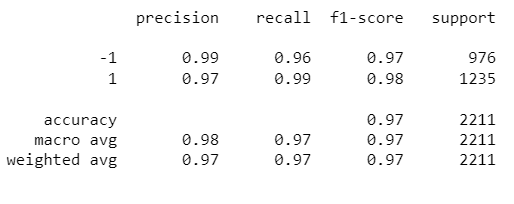
## Training the data

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.

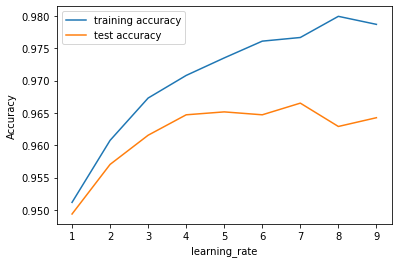
## Gradient Boosting Classifier

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.

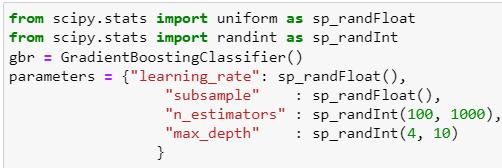
We trained the Gradient Boosting classifier bypassing some of its parameters as max\_depth=4, learning\_rate=0.7, we got the accuracy same as Random Forest gave which is 97%.



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?



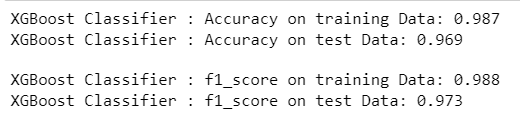
We performed the parameters tuning and trained the model on the best parameters, the accuracy we got is 95%.

## 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 execution of slope helped decision trees intended for speed and execution that is dominative cutthroat AI. Here, you will find how you can introduce and make your most memorable XGBoost model in Python

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



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.



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.

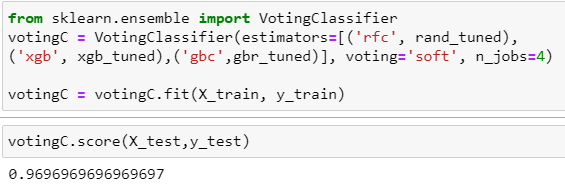
## Ensembling the models

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 an machine learning model that joins the predictions from at least two models. The models that add to the group, alluded to as outfit individuals, might be similar sort or various sorts and could possibly be prepared on similar preparation information.

We will consolidate every one of the three models into one base voting classifier. A Voting Classifier is an AI model that trains on a gathering of various models and predicts a result (class) in light of their most elevated likelihood of picked class as the result.



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%.

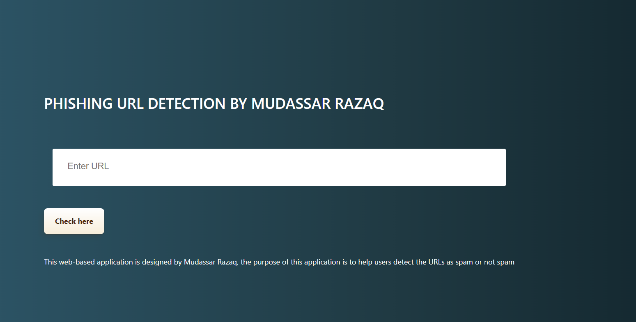
So, we converted the voting ensemble model into a pickle file, we will now use this model for our predictions, and we will integrate it into a web-based application through Flask APIs.

## Web-based Application

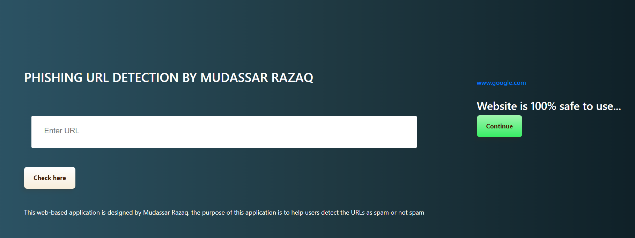
The goal of the project was to train three different ensemble models on the data we have, after training all the model, we will use a base voting model and combine all the ensemble trained models into it, we will train the base model again by passing the three models, then we will integrate the model 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.

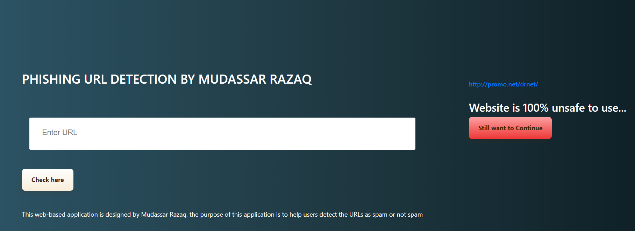
The Flask APIs is the bridge interface between the application and the model we have trained on the historical data. When a user enters ULR in the application, the Flask APIs will take the URL we entered and feed it into the model, the model will predict whether the URL is spam or not and give back the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