Table of Contents:

Problem Statement:

i. **Idea**

Malicious attackers have considered COVID-19 as an opportunity to launch attacks for financial gains and to promote their evil intents. Healthcare systems are being attacked with ransomware and resources such as patient's records confidentiality, and integrity is being compromised. People are falling prey to phishing attacks through COVID-19 related content. We have tried to identify different aspects of cyber security that have been threatened during this time.

In this project we are focusing on detecting the phishing websites

ii. **Project Scope**

Nowadays people generally use random forest classifiers or the decision tree classifier but after this the XGBoost classifier will come to rise. In future people use the XGboost classifier and can classify whether the website is phishing or not with more accuracy not only in this it can be used in any ml project.

iii. **Objective:**

The main objective of this particular paper is a generalised study to review the capabilities of Machine Learning methods for detection of phishing websites and the different techniques that can be used for this specific purpose. They also aimed at comparing different machine learning techniques and came up with the best and most effective possible model for malicious URL detection.

In this project, we are doing a comparative analysis by doing three different iterations and also develop an ensembled ML model to detect Phishing website.

We have used the following algorithms to build a model to detect phishing websites.

Decision Tree Model

Support Vector Machine

XGBoost Model

Bagging Tree

Ada Boost

Grid Search

iv. Process: We will collect the dataset from open-source platforms, extract the required features from the URL database, analyse and pre-process the data. Then we will be using machine learning algorithms with deep learning techniques like SVM, Random Forest Classifier and Autoencoder on the dataset. We will finally compare the different approaches and find out which one is the better alternative for this purpose.

v. Novelty

The main objective of this project is to find out if a dataset containing phishing websites is legitimate or not. Even though many journals were working on the phishing website detection, very few journals were reported about the comparative analysis on different datasets having different attributes to improve efficiency. There are many methods on detecting phishing websites but the most efficient one will always be found by doing comparative study so we have mainly focused on comparing the various algorithms in order to determine the legitimacy of the website. And then we are using three different datasets which have different attributes to find the best dataset which gives a maximum efficiency so that we'll consider the best dataset and the attributes present in this dataset in order to find the efficient algorithm.

We have added new attributes in the dataset for increasing the efficiency for detection of phishing websites.

vi. Comparative statement (Related Works)

We have gone through five research papers and concluded the results along with few advantages and disadvantages from each paper as follows:

1. In the first paper, they calculated the consequences of various classifiers like SVM, Random Forest, Naive Bayes etc. After proper comparison of all the different algorithms, they came to a conclusion that Random Forest classifier had the most accuracy and they considered it best for the purpose of detecting phishing websites and to check the legitimacy of a website.

   Advantages: They have created their own dataset so the features extracted would be 100% accurate. The comparison of different algorithms guarantees to some extent an improvement in work. They have proposed a fresh new phishing detection technique that uses URL based features and furthermore, they also created new classifiers with their own calculations.

   Disadvantages and Suggestions: They have only considered the features they have extracted from the URLs. There are a lot of other types of features that help in detecting whether a website is legitimate or not. Other than the basic URL features, there are finer features of the page, hosting domain features, security features etc which haven't been taken into account in this study.

2. In the second paper, From the comparison and visualisation of all the results it becomes clear that the Random Forest algorithm has a higher R-Squared Value and better Root Mean Square Error, Mean Absolute Error, Mean Squared Error. Also, the Random Forest classifier has better phishing detection accuracy of 91.4% compared with other machine learning models they used in this study.

   Advantages: They have divided the entire procedure into different modules which helps in flexibility and scalability of the code. It is also much easier to understand and much more comprehensible. It makes the code cleaner and easier to judge. They have also used a lot of measures to compare the results of all the models instead of just accuracy and recall as most studies do so that's a really good plus point of this study.

   Disadvantages and Suggestions: They have considered URL features and domain based features. They could've added a few more types of features like security features, site popularity features, network features and comparison with similar surveys. Network layer data like the number of packets sent and received to gain a connection and the count of ports opened on the web server also play a role in detecting phishing websites. They haven't taken these types of features into account.

3. In the third paper, after exploring all the algorithms and classifier models, they compare the accuracies, exactness of all the different models by using a few measures namely, accuracy, true positive rate, false positive rate and positive predictive value. But they mainly used TPR in this study to determine the efficiency of the algorithm. After those experiments are conducted in a standalone mode. As the number of layers increase in the feature extraction process, the accuracy decreases hence they used only two layers to avoid overfitting. They explored two types of features in the paper: original features and interactive features. Then they used DBN, trained the model and later they used the big dataset to test the model and according to TPR the accuracy came up to be 90%.

   Advantages: This is a self-learning model. Instead of normal classifiers they explored deep learning techniques so that the system doesn't get obsolete. It will keep learning and the accuracy will come up to a good measure.

   Disadvantages and Suggestions: They have only considered two kinds of features: original features and interactive features. Other features like domain-based hosting features and security features haven't been taken into account. Also, the measures used to judge accuracy are very limited. They could've taken a few more measures like the root mean square errors, mean square error, recall etc.

4. In the fourth paper, the writers have reviewed articles including synopsis of the explored works after a detailed and thorough study of Phishing Website Detection frameworks. They planned to assist analysts and designers and researchers with the knowledge of the advancement accomplished in the past years. Regardless of the colossal advancement in the field of network safety, phishing site discovery actually represents a testing issue with the always developing innovation and methods.

Advantages: They have helped to collect all past discoveries in the phishing website detection field. For researchers this could be a one stop journal to get an overview of all the approaches that have been used and explored till now regarding malicious website detection and classifying any website as legitimate or malicious/phishing. It has a lot of information and has mostly gone through all aspects or work in this field giving good overview knowledge about the subject.

Disadvantages and Suggestions: They haven't gone through many deep learning mechanisms of detecting phishing websites. They have also not gone through any studies which go through physical features of the website to detect whether it is a phishing website or not. They have limited feature analysis in some studies while some are extensive. They haven't come to any conclusion on the most effective approach and they also haven't compared the different approaches to this problem. They have just stated them.

5. In the fifth paper, The three models the authors of this journal implemented on the given dataset for phishing website detection concludes that XG Boost model is more efficient than other models. The accuracy of XG boost is quite high as compared to other machine learning models and algorithms.

Algorithm	Accuracy
Logistic Regression	92.29
Neutral Network	54.81
XG Boost	94.7

Advantages: The comparison of different algorithms guarantees to some extent an improvement in work. Using three very diverse algorithms for this classification purpose has given them a broad idea of which type of model to explore more. Also, they achieved a pretty high accuracy of 94.7 which is quite difficult to achieve in this field.

Disadvantages and Suggestions: They have only considered the features they have extracted from the URLs. Out of the 30 available traits, they only picked 8 of them which is not sufficient to tell whether a website is phishing or not. They have also missed out on some very basic but really helpful machine learning models in their study like Random Forest Algorithm which is quite a good model in phishing website detection according to the other studies I have gone through. They could've also explores SSL based features, network-based features and a lot more to make their project more scalable and versatile.

From the above research carried out,

Every day, more than 1000 million phishing emails need to be blocked. There are millions of new phishing websites and emails being created and sent especially now, during this pandemic related to false information regarding medical facilities and healthcare to lure people into

signing off their personal details. These are all just an addition to more than 250 million Covid related spam messages. They are using the worsening conditions of the pandemic to give false hope to people and taking advantage of their personal information they provided in hopes of receiving help or information.

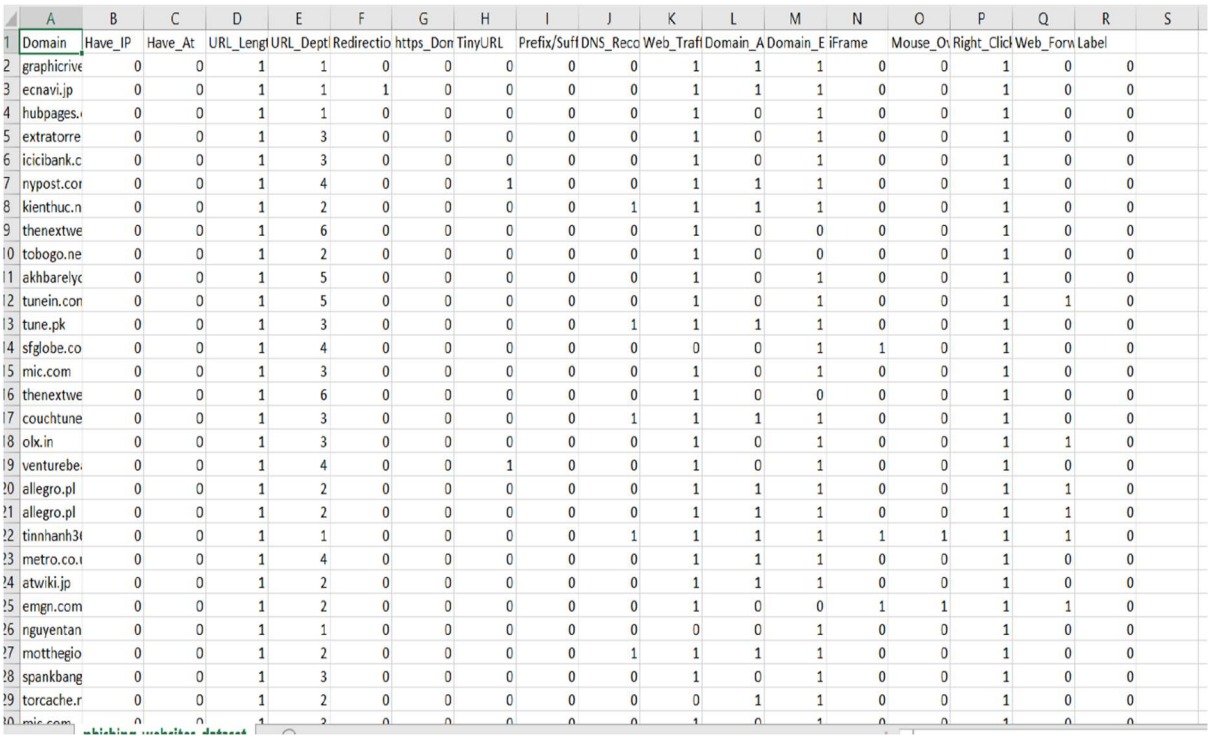
The main reason why this is so prominent in today's world is the lack of awareness of these kinds of practices. The best solutions for this would be for every person to have a piece of software of application in their devices to prominently check whether a website is a phishing website or not given the url of the website you wish to investigate.

After going through the five research papers, this is our proposed system:

We are taking some important attributes from the datasets used by these journals to make a dataset which are the best suitable in detecting the phishing websites in an efficient way.

vii. Dataset

Iteration 1 Dataset:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Domain	Have_IP	Have_At	URL_Lengt	URL_Dept	Redirectio	https_Don	TinyURL	Prefix/Suff	DNS_Reco	Web_Traff	Domain_A	Domain_E	iFrame	Mouse_Ov	Right_Clic	Web_Forw	Label	
2	graphicrive	0	0	1	1	0	0	0	0	0	1	1	1	0	0	1	0	0	
3	ecnavi.jp	0	0	1	1	1	0	0	0	0	1	1	1	0	0	1	0	0	
4	hubpages.(	0	0	1	1	0	0	0	0	0	1	0	1	0	0	1	0	0	
5	extratorre	0	0	1	3	0	0	0	0	0	1	0	1	0	0	1	0	0	
6	icicibank.c	0	0	1	3	0	0	0	0	0	1	0	1	0	0	1	0	0	
7	nypost.cor	0	0	1	4	0	0	1	0	0	1	1	1	0	0	1	0	0	
8	kienthuc.n	0	0	1	2	0	0	0	0	1	1	1	1	0	0	1	0	0	
9	thenextwe	0	0	1	6	0	0	0	0	0	1	0	0	0	0	1	0	0	
10	tobogo.ne	0	0	1	2	0	0	0	0	0	1	0	0	0	0	1	0	0	
11	akhbarelyc	0	0	1	5	0	0	0	0	0	1	0	1	0	0	1	0	0	
12	tunein.con	0	0	1	5	0	0	0	0	0	1	0	1	0	0	1	1	0	
13	tune.pk	0	0	1	3	0	0	0	0	1	1	1	1	0	0	1	0	0	
14	sfglobe.co	0	0	1	4	0	0	0	0	0	0	0	1	1	0	1	0	0	
15	mic.com	0	0	1	3	0	0	0	0	0	1	0	1	0	0	1	0	0	
16	thenextwe	0	0	1	6	0	0	0	0	0	1	0	0	0	0	1	0	0	
17	couchtune	0	0	1	3	0	0	0	0	1	1	1	1	0	0	1	0	0	
18	olx.in	0	0	1	3	0	0	0	0	0	1	0	1	0	0	1	1	0	
19	venturebe	0	0	1	4	0	0	1	0	0	1	0	1	0	0	1	0	0	
20	allegro.pl	0	0	1	2	0	0	0	0	0	1	1	1	0	0	1	1	0	
21	allegro.pl	0	0	1	2	0	0	0	0	0	1	1	1	0	0	1	1	0	
22	tinnhanh3(	0	0	1	1	0	0	0	0	1	1	1	1	1	1	1	1	0	
23	metro.co.(	0	0	1	4	0	0	0	0	0	1	1	1	0	0	1	0	0	
24	atwiki.jp	0	0	1	2	0	0	0	0	0	1	1	1	0	0	1	0	0	
25	emgn.com	0	0	1	2	0	0	0	0	0	1	0	0	1	1	1	1	0	
26	nguyentan	0	0	1	1	0	0	0	0	0	0	0	1	0	0	1	0	0	
27	motthegio	0	0	1	2	0	0	0	0	1	1	1	1	0	0	1	0	0	
28	spankbang	0	0	1	3	0	0	0	0	0	1	0	1	0	0	1	0	0	
29	torcache.r	0	0	1	2	0	0	0	0	0	0	1	1	0	0	1	0	0	
30	mic.com	0	0	1	3	0	0	0	0	0	1	0	1	0	0	1	0	0	

Iteration-2:

Legitimate websites Dataset:

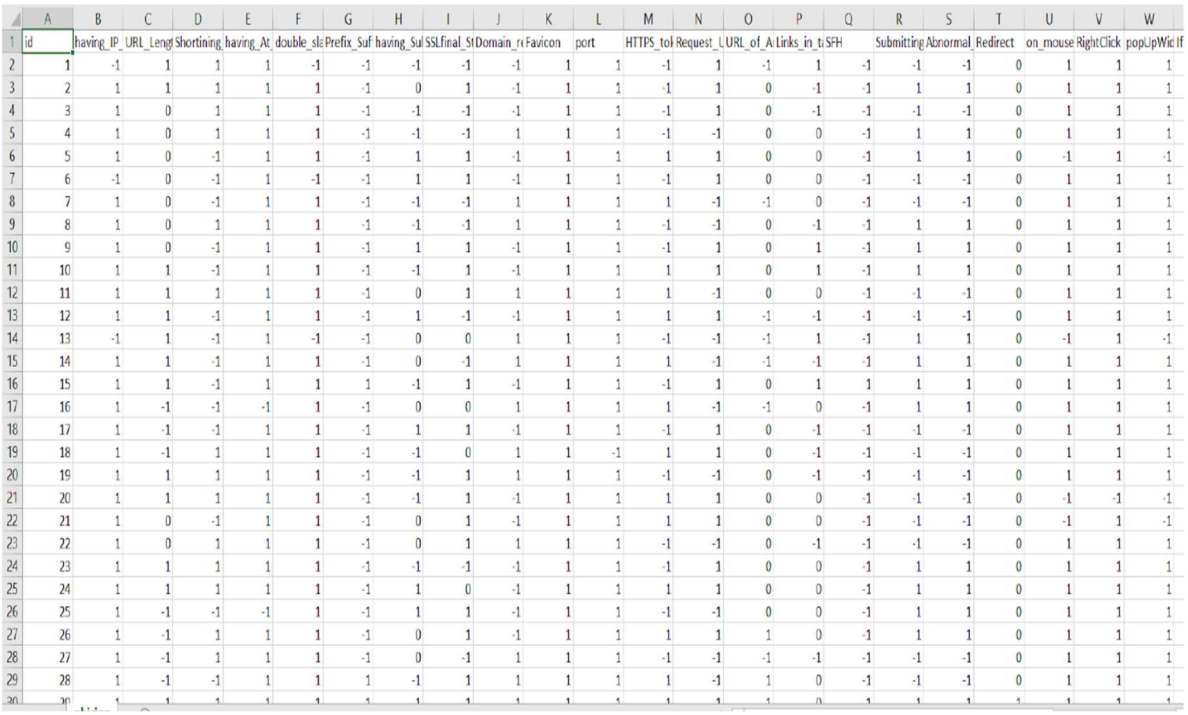
```
websites
http://www.emuck.com:3000/archive/egan.html
http://danoday.com/summit.shtml
http://groups.yahoo.com/group/voice_actor_appreciation/lir
http://voice-international.com/
http://www.livinglegendsltd.com/
http://voicechasers.com/forum/viewforum.php?f=8
http://hollywoodcollectorshow.com/
http://www.geocities.com/hollywood/hills/8944/
http://asifa.proboards61.com/index.cgi?action=calendarviev
http://groups.yahoo.com/group/voice_actor_appreciation/cal
http://us.imdb.com/name/nm0267724/
http://www.pamelynferdin.com/
http://www.pamferdin.com/
http://www.monkeydog.com/
http://teamknightrider.com/cast/plato/plato.html
http://us.imdb.com/name/nm0281486/
```

Phishing websites Dataset:

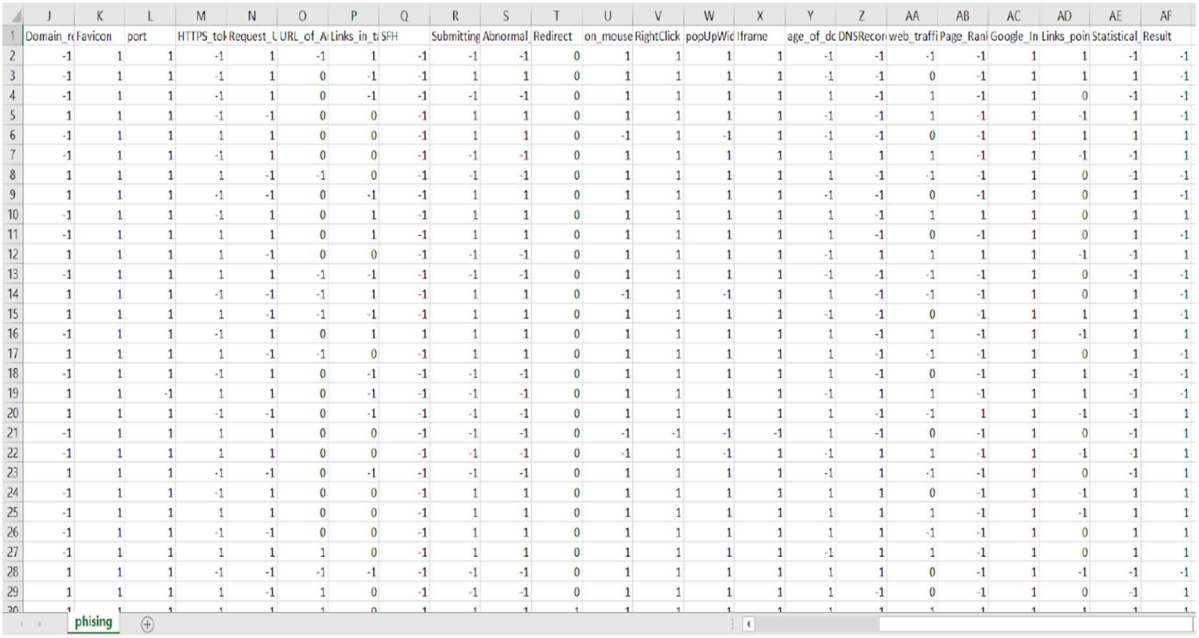
```
http://asesoresvelfit.com/media/datacredito.co/
http://caixa.com.br.fgtsagendesaqueconta.com/consulta85232
http://hissoulreason.com/js/homepage/home/
http://unauthorizd.newebpage.com/webapps/66fbf/
http://133.130.103.10/23/
http://dj00.co.vu/css/?bsoul=Qg@xIHW%//yh/en/?i=34453&amp;
http://133.130.103.10/21/logar/
http://httpssicredi.esy.es/servico/sicredi/validarclientes
http://gamesaty.ga/wp-content///yh/en/?i=31416&amp;i=31416
http://luxuryupgradepro.com/ymailNew/ymailNew/
http://133.130.103.10/1/
http://133.130.103.10/24/sicredi/psmlId/31/paneid/index.ht
http://smscaixaacesso.hol.es
http://133.130.103.10/7/SIIBC/siwinCtrl.php
http://tinyurl.com/kjmmw57
http://wrightlandscapes.org/no/T/Y1.html
http://mautic.eto-cms.ru/themes/goldstar/mtbonline/newmanc
```

Iteration-3 Dataset:

Phishing Dataset:



id	having_IP	URL_Leng	Shortining	having_At	double_sla	Prefix_Suf	having_Sul	SSLfinal_St	Domain_re	Favicon	port	HTTPS_tok	Request_U	URL_of_Ai	Links_in_ti	SFH	Submitting	Abnormal_	Redirect	on_mouse	RightClick	popUpWid	If
1	-1	1	1	1	-1	-1	-1	-1	-1	1	1	-1	1	-1	1	-1	-1	-1	0	1	1	1	
2	1	1	1	1	1	-1	0	1	-1	1	1	-1	1	0	-1	-1	1	1	0	1	1	1	
3	1	0	1	1	1	-1	-1	-1	-1	1	1	-1	1	0	-1	-1	-1	-1	0	1	1	1	
4	1	0	1	1	1	-1	-1	-1	1	1	1	-1	-1	0	0	-1	1	1	0	1	1	1	
5	1	0	-1	1	1	-1	1	1	-1	1	1	1	1	0	0	-1	1	1	0	-1	1	-1	
6	-1	0	-1	1	-1	-1	1	1	-1	1	1	-1	1	0	0	-1	-1	-1	0	1	1	1	
7	1	0	-1	1	1	-1	-1	-1	1	1	1	1	-1	-1	0	-1	-1	-1	0	1	1	1	
8	1	0	1	1	1	-1	-1	-1	1	1	1	-1	-1	0	-1	-1	1	1	0	1	1	1	
9	1	0	-1	1	1	-1	1	1	-1	1	1	-1	1	0	1	-1	1	1	0	1	1	1	
10	1	1	-1	1	1	-1	-1	1	-1	1	1	1	1	0	1	-1	1	1	0	1	1	1	
11	1	1	1	1	1	-1	0	1	1	1	1	1	-1	0	0	-1	-1	-1	0	1	1	1	
12	1	1	-1	1	1	-1	1	-1	-1	1	1	1	1	-1	-1	-1	-1	-1	0	1	1	1	
13	-1	1	-1	1	-1	-1	0	0	1	1	1	-1	-1	-1	1	-1	1	1	0	-1	1	-1	
14	1	1	-1	1	1	-1	0	-1	1	1	1	1	-1	-1	-1	-1	1	1	0	1	1	1	
15	1	1	-1	1	1	1	-1	1	-1	1	1	-1	1	0	1	1	1	1	0	1	1	1	
16	1	-1	-1	-1	1	-1	0	0	1	1	1	1	-1	-1	0	-1	1	1	0	1	1	1	
17	1	-1	-1	1	1	-1	1	1	-1	1	1	-1	1	0	-1	-1	-1	-1	0	1	1	1	
18	1	-1	1	1	1	-1	-1	0	1	1	-1	1	1	0	-1	-1	-1	-1	0	1	1	1	
19	1	1	1	1	1	-1	-1	1	1	1	1	-1	-1	0	-1	-1	-1	-1	0	1	1	1	
20	1	1	1	1	1	-1	-1	1	-1	1	1	1	1	0	0	-1	-1	-1	0	-1	-1	-1	
21	1	0	-1	1	1	-1	0	1	-1	1	1	1	1	0	0	-1	-1	-1	0	-1	1	-1	
22	1	0	1	1	1	-1	0	1	1	1	1	-1	-1	0	-1	-1	-1	-1	0	1	1	1	
23	1	1	1	1	1	-1	-1	-1	-1	1	1	-1	1	0	0	-1	1	1	0	1	1	1	
24	1	1	1	1	1	-1	1	0	-1	1	1	1	1	0	0	-1	1	1	0	1	1	1	
25	1	-1	-1	-1	1	-1	1	1	-1	1	1	-1	-1	0	0	-1	1	1	0	1	1	1	
26	1	-1	1	1	1	-1	0	1	-1	1	1	1	1	1	0	-1	1	1	0	1	1	1	
27	1	-1	1	1	1	-1	0	-1	1	1	1	-1	-1	-1	-1	-1	-1	-1	0	1	1	1	
28	1	-1	-1	1	1	1	-1	1	1	1	1	1	-1	1	0	-1	-1	-1	0	1	1	1	



Domain_re	Favicon	port	HTTPS_tok	Request_U	URL_of_Ai	Links_in_ti	SFH	Submitting	Abnormal_	Redirect	on_mouse	RightClick	popUpWid	Iframe	age_of_dc	DNSRecor	web_traffi	Page_Rani	Google_In	Links_poin	Statistical	Result
-1	1	1	-1	1	-1	1	-1	-1	-1	0	1	1	1	1	-1	-1	-1	-1	1	1	-1	-1
-1	1	1	-1	1	0	-1	-1	1	1	0	1	1	1	1	-1	-1	0	-1	1	1	1	-1
-1	1	1	-1	1	0	-1	-1	-1	-1	0	1	1	1	1	1	-1	1	-1	1	0	-1	-1
1	1	1	-1	-1	0	0	-1	1	1	0	1	1	1	1	-1	-1	1	-1	1	-1	1	-1
-1	1	1	1	1	0	0	-1	1	1	0	-1	1	-1	1	-1	-1	0	-1	1	1	1	1
-1	1	1	-1	1	0	0	-1	-1	-1	0	1	1	1	1	1	1	1	-1	1	-1	-1	1
1	1	1	1	-1	-1	0	-1	-1	-1	0	1	1	1	1	1	-1	-1	-1	1	0	-1	-1
1	1	1	-1	-1	0	-1	-1	1	1	0	1	1	1	1	-1	-1	0	-1	1	0	1	-1
-1	1	1	-1	1	0	1	-1	1	1	0	1	1	1	1	1	-1	1	1	1	0	1	1
-1	1	1	1	1	0	1	-1	1	1	0	1	1	1	1	1	-1	0	-1	1	0	1	-1
1	1	1	1	-1	0	0	-1	-1	-1	0	1	1	1	1	-1	1	1	1	1	-1	-1	1
-1	1	1	1	1	-1	-1	-1	-1	-1	0	1	1	1	1	-1	-1	-1	-1	1	0	-1	-1
1	1	1	-1	-1	-1	1	-1	1	1	0	-1	1	-1	1	1	-1	-1	-1	1	0	1	-1
1	1	1	1	1	-1	-1	-1	1	1	0	1	1	1	1	-1	-1	0	-1	1	1	1	-1
-1	1	1	-1	1	0	1	1	1	1	0	1	1	1	1	1	-1	1	-1	1	-1	1	1
1	1	1	1	-1	-1	0	-1	1	1	0	1	1	1	1	1	-1	-1	-1	1	0	1	-1
-1	1	1	-1	1	0	-1	-1	-1	-1	0	1	1	1	1	1	-1	0	-1	1	1	-1	-1
1	1	-1	1	1	0	-1	-1	-1	-1	0	1	1	1	1	-1	1	1	-1	1	1	-1	-1
1	1	1	-1	-1	0	-1	-1	-1	-1	0	1	1	1	1	1	-1	-1	1	1	-1	-1	1
-1	1	1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	1	-1	0	-1	1	1	0	1
-1	1	1	1	1	0	0	-1	-1	-1	0	-1	1	-1	1	-1	1	1	-1	1	-1	-1	1
1	1	1	-1	-1	0	-1	-1	-1	-1	0	1	1	1	1	-1	1	-1	-1	1	0	-1	1
-1	1	1	-1	1	0	0	-1	1	1	0	1	1	1	1	1	1	0	-1	1	-1	1	1
-1	1	1	1	1	0	0	-1	1	1	0	1	1	1	1	1	1	1	-1	1	-1	1	1
-1	1	1	-1	-1	0	0	-1	1	1	0	1	1	1	1	1	1	-1	-1	1	0	1	1
-1	1	1	1	1	1	0	-1	1	1	0	1	1	1	1	-1	1	1	-1	1	0	1	1
1	1	1	-1	-1	-1	-1	-1	-1	-1	0	1	1	1	1	1	1	0	-1	1	-1	-1	1
1	1	1	1	-1	1	0	-1	-1	-1	0	1	1	1	1	1	-1	0	-1	1	0	-1	1

phising

vii. Test bed:

Tool used: Google Colab

We have used some important libraries like

• Numpy

• Pandas

• Matplotlib

• Tensorflow

• Keras

• Sklearn

• Tf learn

• Xg boost

• Svm

viii. Expected result:

The dataset extracted by us (which contains new attributes) should give better accuracy than existing datasets with all the machine learning algorithms.

Architecture:

HIGH LEVEL DESIGN (BLACK BOX DESIGN)

A black box system is a design simple system which

Broadly describes the input we are going to our project or system and the output we are expecting it to deliver.

There are no details regarding the internal working of the algorithms. It doesn't take into account the exact performance measures of the efficiency of the internal algorithms at work to give the expected output.

It just gives us an idea about the input and the output of our system or project. In neural networking or heuristic algorithms (laptop phrases usually used to explain 'learning' computer systems or 'AI simulations'), a black field is used to explain the continuously converting segment of this system's surroundings which cannot without problems be examined through the programmers. This is likewise known as a white field withinside the context that this system code may be seen, however the code is so complicated that it's far functionally equal to a black field.

In our project, we are aiming to detect whether a website is a phishing website or if it is a legitimate one using various machine learning algorithms. So irrespective of the algorithm we use in various iterations of our project, our high level design pretty much remains the same. But there are two variations.

PHASE 1: In this phase of the project, we took a prepared dataset which contained most of the important features we needed to check and compare to find out if the website is legitimate or a fraud.

INPUT: Dataset containing the following features:

• Domain

• Have_IP

• Have_at

• URL length

• URL depth

• Redirection

• Https domain

• Tiny URL

• Prefix

• Suffix

• DNS record

• Web traffic

• Domain Age

• Domain end

• Iframe

• Mouse_over

• Right_clicks

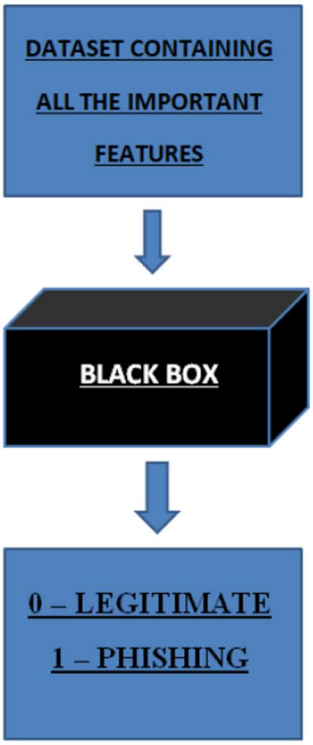
• Web_forwards

OUTPUT:

• 0-Legitimate

• 1-Phishing

HIGH LEVEL DESIGN



PHASE 2: In this phase of the project, instead of taking a pre-created dataset, we decided to work on creating our own dataset and then following up with phishing website detection to gain more control over the amount and shape of data as well as more accurate results.

INPUT:

We initially picked up a few datasets containing phishing website URLS as well as a few datasets of just plain legitimate website URLS and combined them to make one hue dataset.

After getting that dataset, we had to apply feature extraction and extract the kinds of attributes which we carefully researched were more important for phishing website detection. We had to apply this procedure to all the URLS in the dataset we had formed in the initial step.

We extracted the following features for this phase of the project:

• Protocol

• Domain name

• Address

• Long_URL

• Having @ symbol

• Redirection // symbol

• Prefix - Suffix deparation

• Sub-domains

• Having IP address

• Shortening service

• Https token

• Web traffic

• Domain Registration length

• Age of domain

• DNS record

After all of these steps, we have created our very own dataset. Then we applied the phishing website detection algorithms to classify the URLS according to the features we had extracted into the dataset.

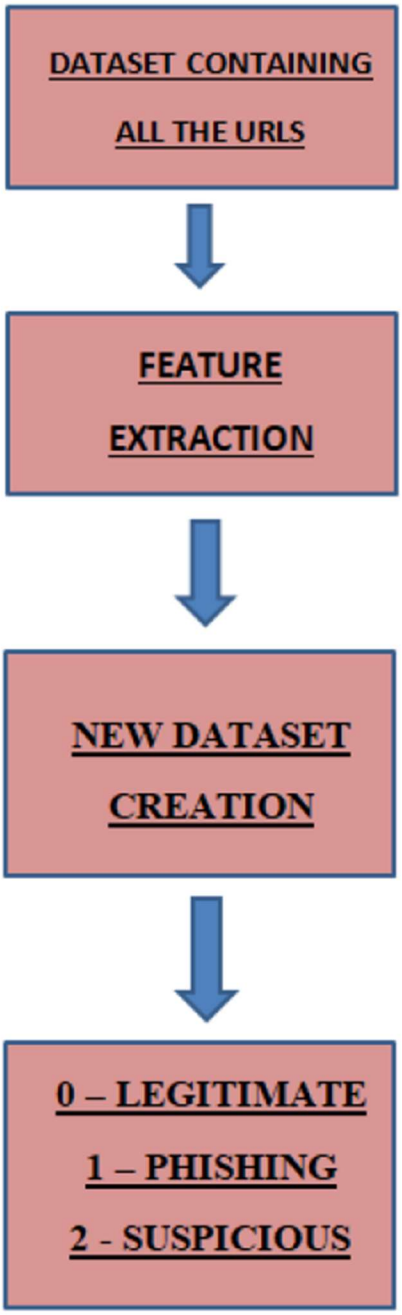
OUTPUT:

0 - Legitimate

1 - Phishing

2 - Suspicious

HIGH LEVEL DESIGN



LOW LEVEL DESIGN: DETAILED ARCHITECTURE

We have implemented python program to extract capabilities from URL. Below are the features or attributes that we've got extracted for detection of phishing URLs.

1) Presence of IP cope with in URL: If IP cope with found in URL then the characteristic is ready to one else set to 0. Most of the benign webweb sites do now no longer use IP cope with as an URL to download a webpage. Use of IP cope with in URL indicates that attacker is trying to steal touchy statistics.

2) Presence of @ image in URL: If @ image found in URL then the characteristic is ready to one else set to 0. Phishers upload unique image @ withinside the URL leads the browser to disregard the whole lot previous the "@" image and the actual cope with frequently follows the "@" image [4].

3) Number of dots in Hostname: Phishing URLs have many dots in URL. For instance http://shop.fun.amazon.phishing.com, in this URL phishing.com is an real area call, while use of "amazon" phrase is to trick customers to click on on it. Average wide variety of dots in benign URLs is three. If the wide variety of dots in URLs is extra than three then the characteristic is ready to one else to 0.

4) Prefix or Suffix separated with the aid of using (-) to area: If area call separated with the aid of using sprint (-) image then characteristic is ready to one else to 0. The sprint image is hardly ever utilized in legitimate URLs. Phishers upload sprint image (-) to the area call in order that customers experience that they may be coping with a legitimate webpage. For instance Actual site is http://www.onlineamazon.com but phisher can create any other faux internet site like http://www.online-amazon.com to confuse the harmless customers.

5) URL redirection: If "//" gift in URL course then characteristic is set to one else to 0. The existence of "//" withinside the URL course method that the consumer may be redirected to any other internet site [4].

6) HTTPS token in URL: If HTTPS token gift in International Journal of Computer Applications (0975 – 8887) Volume 181 – No. 23, October 2018 forty six URL then the characteristic is set to one else to 0. Phishers may also upload the "HTTPS" token to the area a part of a URL in order to trick customers. For instance, http://https-www-paypal-it-mpp-home.soft-hair.com [4].

7) Information submission to Email: Phisher would possibly use "mail()" or "mailto:" capabilities to redirect the consumer's statistics to his private email[4]. If such capabilities are gift withinside the URL then characteristic is ready to one else to 0.

8) URL Shortening Services "TinyURL": TinyURL service allows phisher to hide long phishing URL with the aid of using making it short. The aim is to redirect consumer to phishing web sites. If the URL is crafted the usage of shortening services (like bit.ly) then characteristic is ready to one else 0

9) Length of Host call: Average period of the benign URLs is determined to be a 25, If URL's period is more than 25 then the characteristic is ready to one else to 0

10) Presence of touchy phrases in URL: Phishing web sites use touchy phrases in its URL in order that customers experience that they may be dealing with a legitimate webpage. Below are the phrases that determined in many phishing URLs :- 'confirm', 'account', 'banking', 'secure', 'ebyisapi', 'webscr', 'signin', 'mail', 'install', 'toolbar', 'backup', 'paypal', 'password', 'username', etc;

11) Number of .'s in URL: The wide variety of slashes in benign URLs is determined to be a five; if wide variety of slashes in URL is more than five then the characteristic is ready to one else to 0.

12) Presence of Unicode in URL: Phishers could make a use of Unicode characters in URL to trick customers to click on on it. For instance the area "xn--80ak6aa92e.com" is equivalent to "apple.com". Visible URL to consumer is "apple.com" but after clicking on this URL, consumer will go to to "xn--80ak6aa92e.com" that is a phishing site.

13) IFRAME: We have extracted this selection with the aid of using crawling the source code of the URL. This tag is used to upload any other internet web page into existing principal webpage. Phishers can employ the "iframe" tag and make it invisible i.e. without frame borders [4]. Since border of inserted webpage is invisible, consumer appears that the inserted internet web page is likewise the a part of the principle internet web page and may input touchy statistics.

14) Website Rank: We extracted the rank of web sites and evaluate it with the primary One hundred thousand web sites of Alexa database. If rank of the internet site is more than 10,0000 then characteristic is ready to one else to 0.

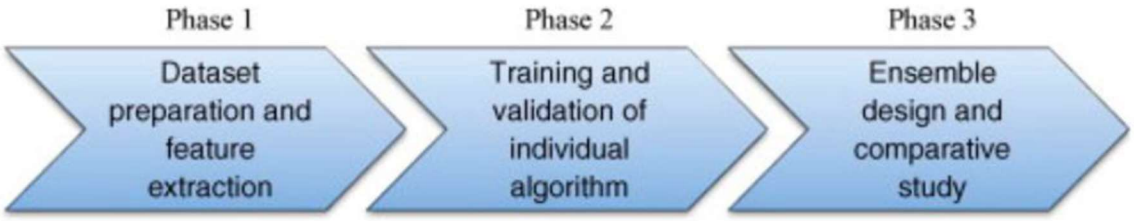
After extracting all the features and making our own dataset, we then fit the dataset into various machine learning models and then with each of them we train and test the data and calculate performance measures and finally we have compared the different algorithms and their efficiency when it comes to this purpose.

The different algorithms we have used in our project are:

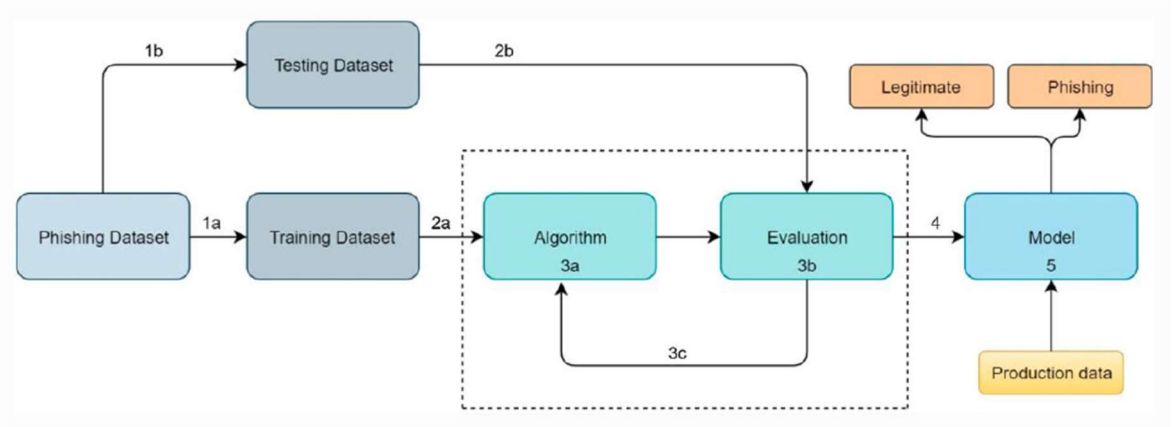
1. Decision Tree Algorithm
2. XgBoost Algorithm
3. Support Vector Machine
4. Bagging Tree
5. Grid Search
6. Ada Boost

General High-Level Design:

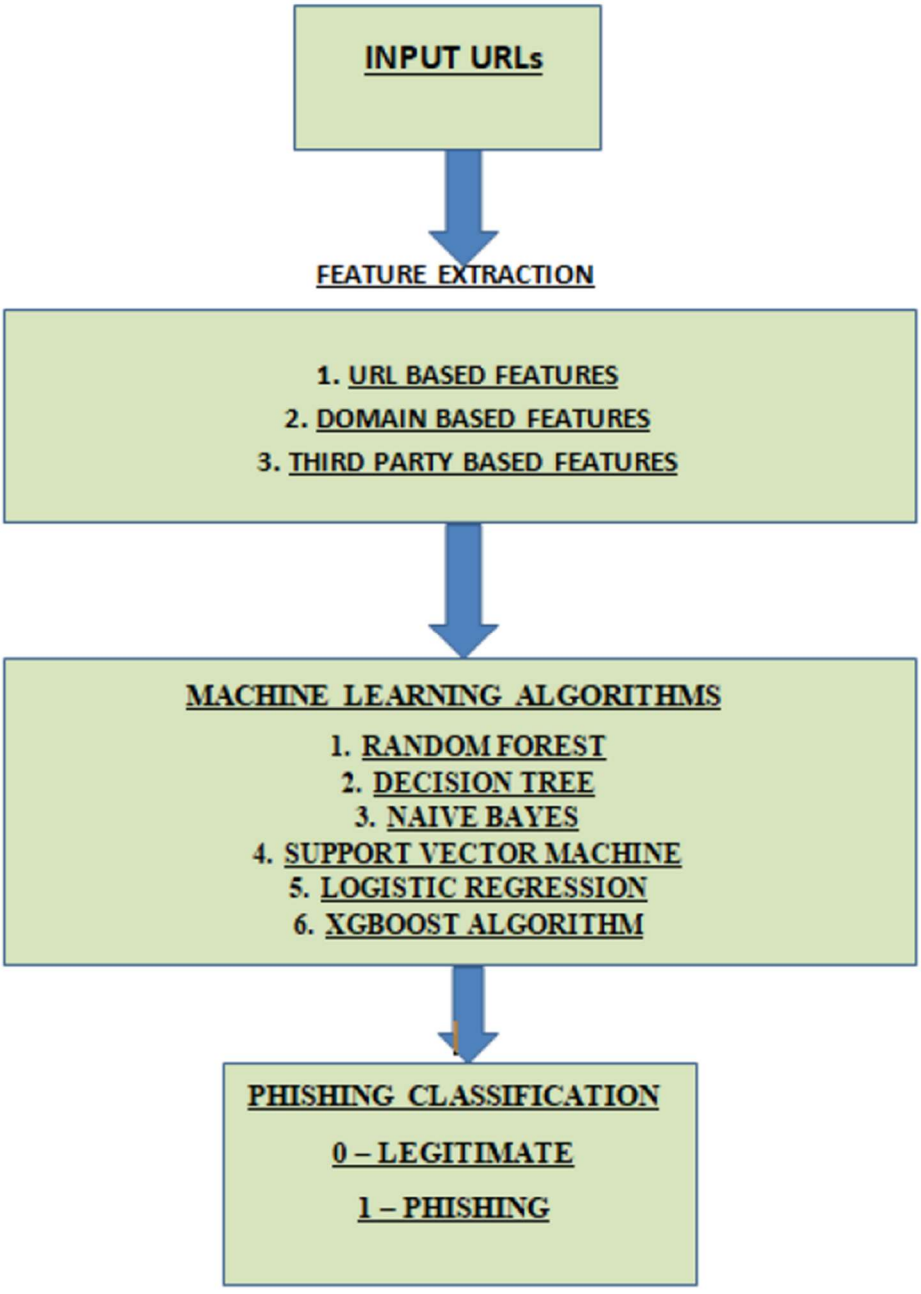
The steps can be divided into 3 broad phases



One the dataset is created this is the general flow of program to detect if the website is legitimate or not



According to each algorithm, the detailed internal working differs but the general architecture of the program is as shown below:



IMPLEMENTATION

In the span of three different iterations of our project, we have used the following algorithms to build a model to detect phishing websites. We have used:

1. Decision Tree Algorithm
2. XgBoost Algorithm
3. Support Vector Machine
4. Bagging Tree
5. Grid Search
6. Ada Boost

ALGORITHMS USED and MATHEMATICAL MODELS:

1. DECISION TREE

Decision Trees are very effective classifiers with no parameters. As the name suggests, a decision tree is a tree structure in which each non-terminal node designates a test for an attribute, each branch represents a test result, and leaf nodes designate classes.

The basic algorithm for the induction of decision trees is a greedy algorithm that creates the decision tree in a recursive manner by breaking it down and conquering it .

At each non-terminal node, one of the features is selected for the division. The attribute that gives the maximum information gain is A well-known algorithm for decision trees is the C4.5 algorithm, in which the entropy is used as the criterion for calculating the information gain.

The information gain is defined as the difference between the entropy before the division and the entropy after the division.

To calculate the information obtained below:

$$H(T) = -\sum_{j} p_j \log_2 \left(p_j \right)$$

$$Hs(T) = -\sum_{i} p_i Hs(T_i)$$

$$Gain(s) = H(T) - Hs(T)$$

Where H(T) is the entropy before division, Hs(T) is the entropy after division, and pj is the probability of class j. One of the main problems with the decision tree classifier is that it leads to overfitting of the data.

2. SUPPORT VECTOR MACHINE

SVM is a popular supervised machine learning algorithm which can be utilized for both classification and regression problems. It uses a specific technique called the famous kernel trick to minutely transform the data and then on the basis of these transformations it aims to find an optimal boundary between all the possible results or outputs.

This particular classifier uses non-linear mapping to transform the original training data into a higher dimension and finds hyperplanes that subdivide the data samples into the higher-dimensional feature space.

$$Wx + b=0$$

where W is a weight matrix and b is a constant. SV algorithms find the weight matrix in such a way that the distance between the hyperplanes separating two classes is maximized. The tuples that fall within the hyperplanes are called support vectors.

3. XGBOOST ALGORITHM:

XGBOOST (Extreme Gradient Boosted Tree) is an optimized implementation of gradient-controlled trees. It is mainly used in classification tasks where it is used as a classifier to map input patterns in a particular class. Learning algorithm that implements a process called Boost to improve the performance of trees with increased gradient.

XGBOOST has many strengths compared to traditional gradient boost implementations. Its strengths include better regularization ability, which helps reduce overfitting, high speed and performance due to the parallelism of the trees, flexibility due to the goals and evaluation criteria for costume optimization, and the built-in routines for handling missing values.

These and many other benefits of XGBOOST have made it a great tool of choice for many data science researchers. and machine learning. Some of the researchers used these techniques.

The prediction model ($\hat{y}$) can be written as the summation of all the prediction scores for each tree for a sample (x). As an example, consider the i-th sample,

$$\hat{y}i = \sum_{k}^{K} f_k(x), f_k \in F$$

Where K is the total number of trees, f is the function in the space $\mathcal{F}$ and $\mathscr{F}$ is the total of all the possible set of trees having the prediction score in all leaf nodes.

Boosted trees are trained through a strategy known as additive training. A new tree is added with each iteration of the phishing detection process. The final predictive score of the model is obtained by adding the predictive score of the individual tree. written as:

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)$$

The newest tree is created to compensate for the instances of the previous poor learning predicted websites. We need to optimize a certain objective function in order to choose the best model for the training data.

Here we recommend that a model have good predictive power. than simple nature (deals with fewer functions). we know that minimizing the loss function ((Θ)) promotes prediction models, as well as optimizing the regularization (Ω (Θ)) promotes a simpler model with less variance in future predictions, which makes the prediction stable. The closed shape of the target is given below:

$$obj(\Theta) = \sum_i^n l(y_i, \hat{y}_i) + \sum_{k=1}^K \Omega(f_k)$$

4. BAGGING TREE

Bagging (Bootstrap Aggregation) is used when our goal is to reduce the variance of a decision tree. Here idea is to create several subsets of data from training sample chosen randomly with replacement. Now, each collection of subset data is used to train their decision trees. As a result, we end up with an ensemble of different models. Average of all the predictions from different trees are used which is more robust than a single decision tree

5. GRID SEARCH

Grid-searching is the process of scanning the data to configure optimal parameters for a given model. Depending on the type of model utilized, certain parameters are necessary. Grid-searching does NOT only apply to one model type. Grid-searching can be applied across machine learning to calculate the best parameters to use for any given model. It is important to note that Grid-searching can be extremely computationally expensive and may take your machine quite a long time to run. Grid-Search will build a model on each parameter combination possible. It iterates through every parameter combination and stores a model for each combination.

6. ADA BOOST

AdaBoost algorithm, short for Adaptive Boosting, is a Boosting technique used as an Ensemble Method in Machine Learning. It is called Adaptive Boosting as the weights are re-assigned to each instance, with higher weights assigned to incorrectly classified instances. Boosting is used to reduce bias as well as variance for supervised learning. It works on the principle of learners growing sequentially. Except for the first, each subsequent learner is grown from previously grown learners. In simple words, weak learners are converted into strong ones. The AdaBoost algorithm works on the same principle as boosting with a slight difference.

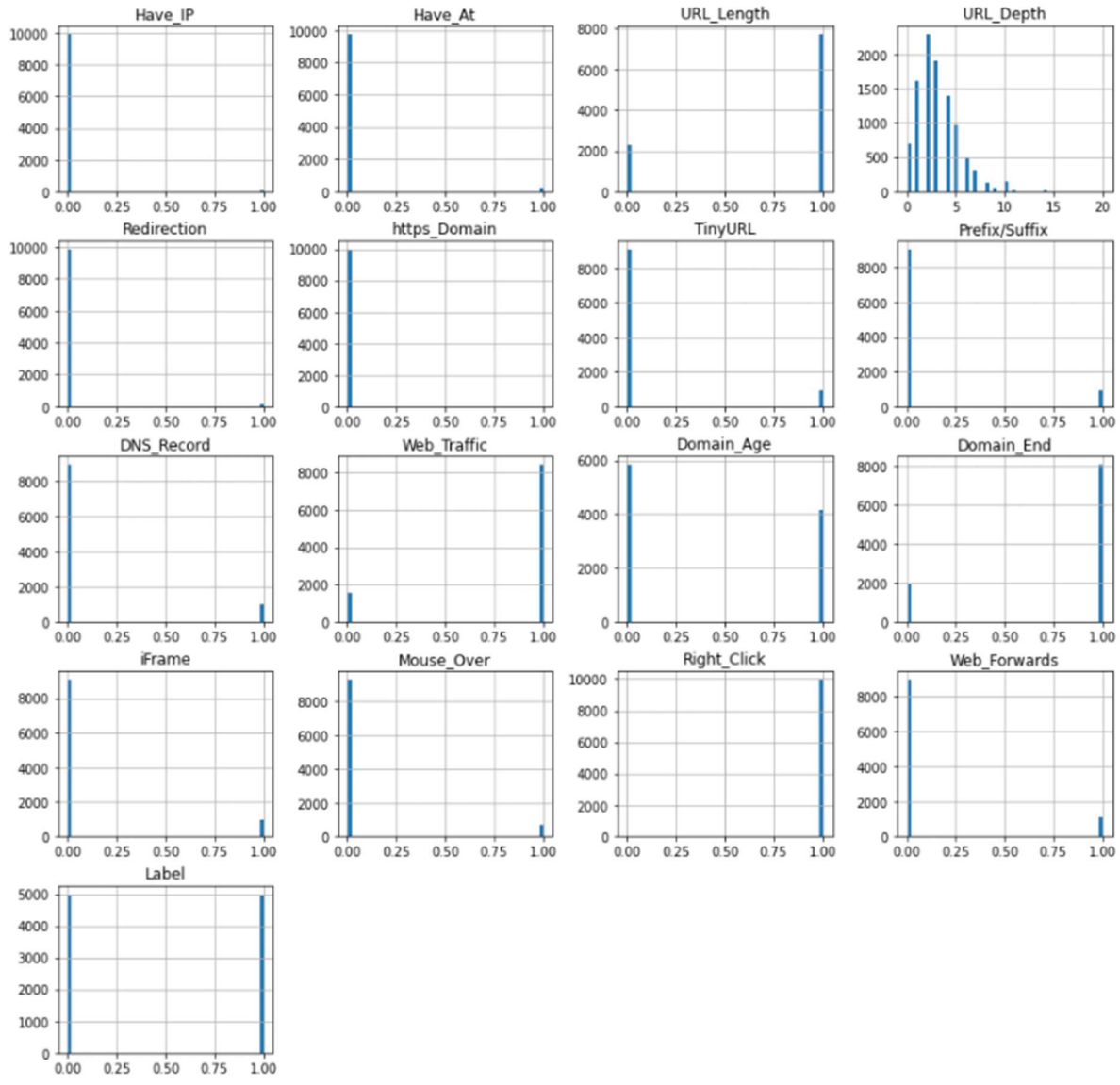
Results and discussion

i. Implementation with coding

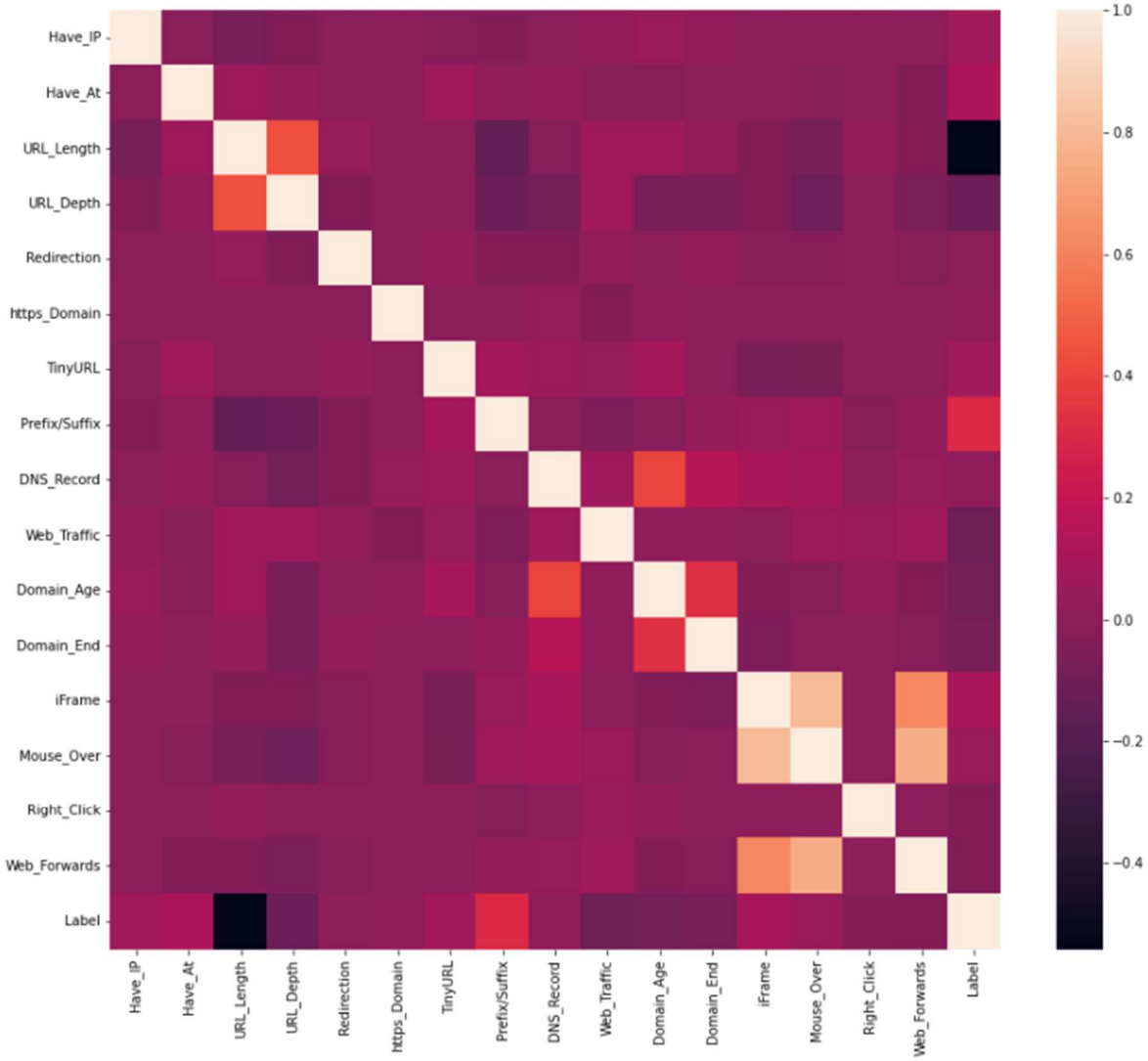
ii. Results in table/Graph/Data

data0.hist(bins = 50,figsize = (15,15))

plt.show()

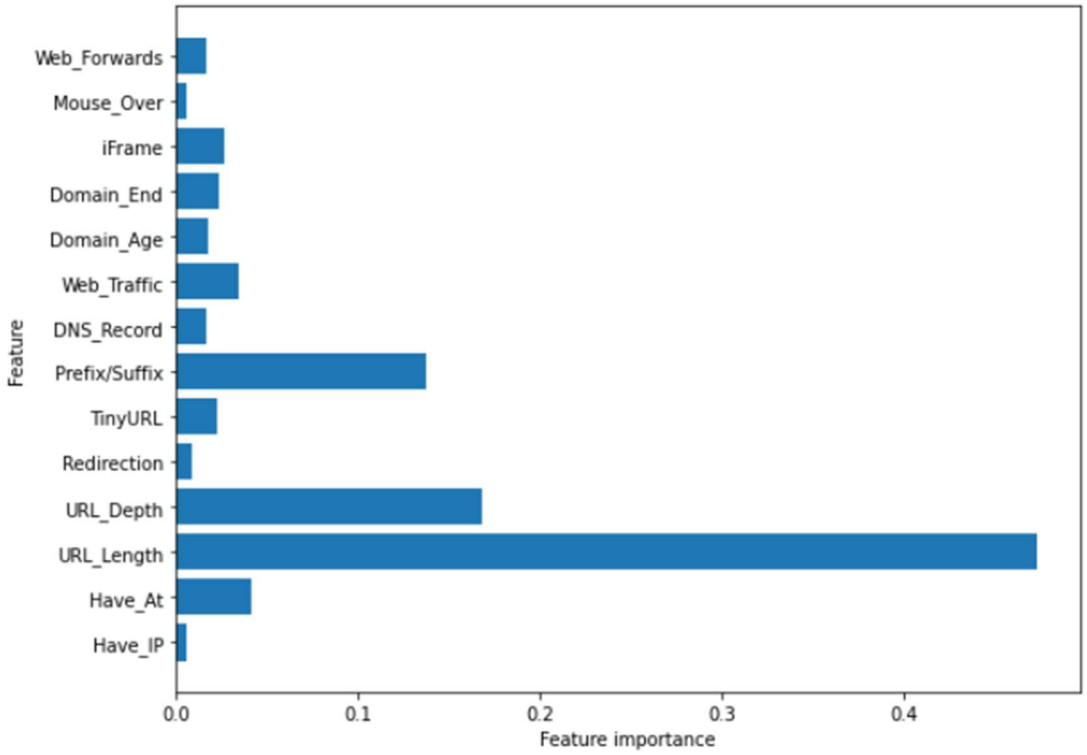


```
plt.figure(figsize=(15,13))
sns.heatmap(data0.corr())
plt.show()
```




```
# Creating holders to store the model performance results
ML_Model = []
acc_train = []
acc_test = []

#function to call for storing the results
def storeResults(model, a,b):
  ML_Model.append(model)
```

```
  acc_train.append(round(a, 3))
  acc_test.append(round(b, 3))

from sklearn.tree import DecisionTreeClassifier

# instantiate the model
tree = DecisionTreeClassifier(max_depth = 30)

# fit the model
tree.fit(X_train, y_train)

#predicting the target value from the model for the samples
y_test_tree = tree.predict(X_test)
y_train_tree = tree.predict(X_train)

#computing the accuracy of the model performance
acc_train_tree = accuracy_score(y_train,y_train_tree)
acc_test_tree = accuracy_score(y_test,y_test_tree)
print("Decision Tree: Accuracy on training Data: {:.3f}".format(acc_train_tree))
print("Decision Tree: Accuracy on test Data: {:.3f}".format(acc_test_tree))



plt.figure(figsize=(9,7))
n_features = X_train.shape[1]
plt.barh(range(n_features), tree.feature_importances_, align='center')
plt.yticks(np.arange(n_features), X_train.columns)
plt.xlabel("Feature importance")
plt.ylabel("Feature")
plt.show()
```




```
from xgboost import XGBClassifier

# instantiate the model
xgb = XGBClassifier(learning_rate=0.3,max_depth=30,n_jobs=-1,n_estimators=500)

#fit the model
xgb.fit(X_train, y_train)

#predicting the target value from the model for the samples
y_test_xgb = xgb.predict(X_test)
y_train_xgb = xgb.predict(X_train)

#computing the accuracy of the model performance
acc_train_xgb = accuracy_score(y_train,y_train_xgb)
acc_test_xgb = accuracy_score(y_test,y_test_xgb)
```

```
print("XGBoost: Accuracy on training Data: {:.3f}".format(acc_train_xgb))
print("XGBoost : Accuracy on test Data: {:.3f}".format(acc_test_xgb))

#Support vector machine model
from sklearn.svm import SVC

# instantiate the model
svm = SVC(kernel='linear', C=1.0, random_state=15)

#fit the model
svm.fit(X_train, y_train)

#predicting the target value from the model for the samples
y_test_svm = svm.predict(X_test)
y_train_svm = svm.predict(X_train)

#computing the accuracy of the model performance
acc_train_svm = accuracy_score(y_train,y_train_svm)
acc_test_svm = accuracy_score(y_test,y_test_svm)
print("SVM: Accuracy on training Data: {:.3f}".format(acc_train_svm))
print("SVM : Accuracy on test Data: {:.3f}".format(acc_test_svm))

# comparing the above models
results = pd.DataFrame({ 'ML Model': ML_Model,
   'Train Accuracy': acc_train,
   'Test Accuracy': acc_test})
results
```

	ML Model	Train Accuracy	Test Accuracy
0	Decision Tree	0.869	0.854
1	XGBoost	0.869	0.858
2	SVM	0.804	0.792

```
results.sort_values(by=['Test Accuracy', 'Train Accuracy'], ascending=False)
```

	ML Model	Train Accuracy	Test Accuracy
1	XGBoost	0.869	0.858
0	Decision Tree	0.869	0.854
2	SVM	0.804	0.792

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier
clf = DecisionTreeClassifier(random_state=35)
bag_clf = BaggingClassifier(base_estimator=clf, n_estimators=1000,
                            bootstrap=True,n_jobs=-1,
                            random_state=35)
bag_clf.fit(X_train, y_train)
#computing the accuracy of the model performance
acc_train_bag = accuracy_score(y_train,y_train_bag)
acc_test_bag = accuracy_score(y_test,y_test_bag)
print("bagging tree: Accuracy on training Data: {:.3f}".format(acc_train_bag))
print("bagging tree: Accuracy on test Data: {:.3f}".format(acc_test_bag))

from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
rf_clf = RandomForestClassifier(random_state=50)
params_grid = {"max_depth": [25, None],
               "min_samples_split": [2, 3, 10],
               "min_samples_leaf": [1, 3, 10],
               "bootstrap": [True, False],
               "criterion": ['gini', 'entropy']}
grid_search = GridSearchCV(rf_clf, params_grid,
                           n_jobs=-1, cv=5,
                           verbose=1, scoring='accuracy')
```

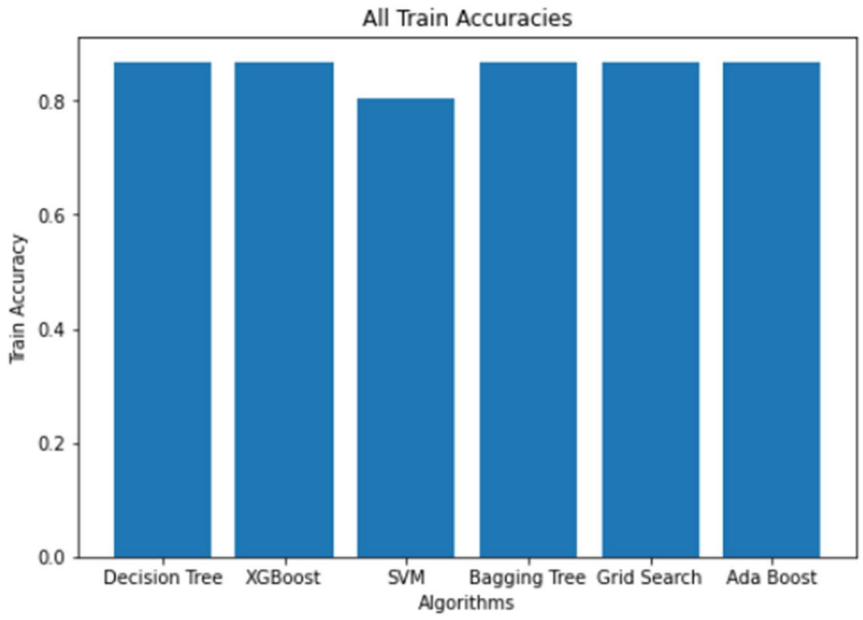
```
grid_search.fit(X_train, y_train)
y_test_gs = grid_search.predict(X_test)
y_train_gs = grid_search.predict(X_train)
acc_train_gs = accuracy_score(y_train,y_train_gs)
acc_test_gs = accuracy_score(y_test,y_test_gs)
print("grid Search : Accuracy on training Data: {:.3f}".format(acc_train_gs))
print("grid Search: Accuracy on test Data: {:.3f}".format(acc_test_gs))

from sklearn.ensemble import AdaBoostClassifier
ada_clf = AdaBoostClassifier(RandomForestClassifier(),learning_rate=0.3,n_estimators=100)
ada_clf.fit(X_train, y_train)
y_test_ar = grid_search.predict(X_test)
y_train_ar = grid_search.predict(X_train)
acc_train_ar = accuracy_score(y_train,y_train_ar)
acc_test_ar = accuracy_score(y_test,y_test_ar)
print("Ada Boost : Accuracy on training Data: {:.3f}".format(acc_train_ar))
print("Ada Boost: Accuracy on test Data: {:.3f}".format(acc_test_ar))

results = pd.DataFrame({ 'ML Model': ML_Model,
   'Train Accuracy': acc_train,
   'Test Accuracy': acc_test})
results
```

	ML Model	Train Accuracy	Test Accuracy
0	Decision Tree	0.869	0.854
1	XGBoost	0.869	0.858
2	SVM	0.804	0.792
3	Bagging Tree	0.869	0.857
4	Grid Search	0.869	0.856
5	Ada Boost	0.869	0.856

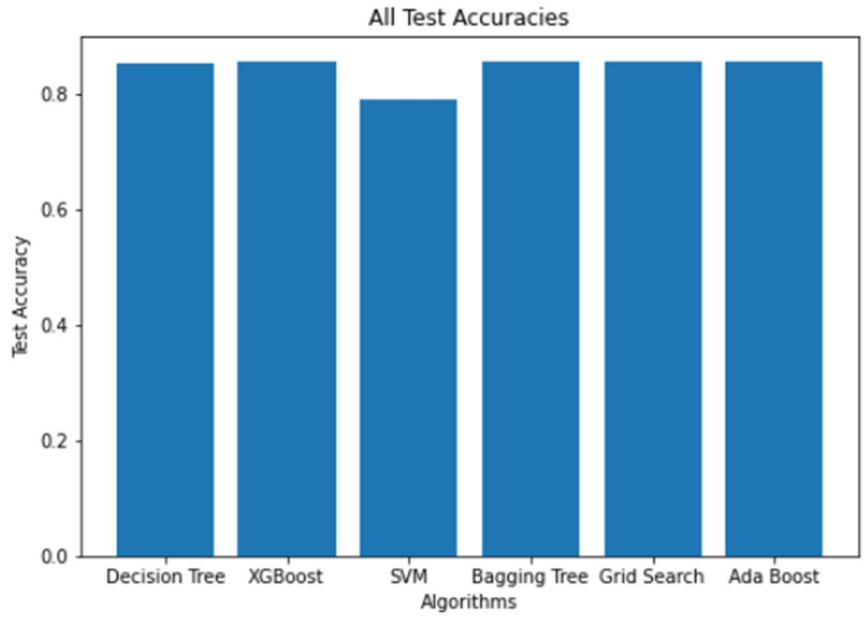
results.sort_values(by=['Test Accuracy', 'Train Accuracy'], ascending=False)

	ML Model	Train Accuracy	Test Accuracy
1	XGBoost	0.869	0.858
3	Bagging Tree	0.869	0.857
4	Grid Search	0.869	0.856
5	Ada Boost	0.869	0.856
0	Decision Tree	0.869	0.854
2	SVM	0.804	0.792

```
fig=plt.figure()
ax =fig.add_axes([0,0,1,1])
langs =results['ML Model']
students =results['Train Accuracy']
ax.bar(langs,students)
plt.xlabel('Algorithms')
plt.ylabel('Train Accuracy')
plt.title('All Train Accuracies')
plt.show()
```



```
fig=plt.figure()
ax =fig.add_axes([0,0,1,1])
langs =results['ML Model']
students =results['Test Accuracy']
ax.bar(langs,students)
plt.xlabel('Algorithms')
plt.ylabel('Test Accuracy')
plt.title('All Test Accuracies')
plt.show()
```



Testing the URL

```
final_features = []
url = "http://vvvvvv.paypal.com.xn-----vldfgifeq7dua2hckj.net/webscr.php?cmd=_login-run&dispatch=5885d80a13c0db1f1ff80d546411d7f8a8350c132bc41e0934cfc023d4e8f9e5de18f1c65e75492761cbf96927fedbefde18f1c65e75492761cbf96927fedbef"
final_features.append(featureExtraction(url))

import pandas as pd
feature_names = ['Domain', 'Have_IP', 'Have_At', 'URL_Length', 'URL_Depth','Redirection',
                 'https_Domain', 'TinyURL', 'Prefix/Suffix', 'DNS_Record', 'Web_Traffic',
```

'Domain_Age', 'Domain_End', 'iFrame', 'Mouse_Over','Right_Click', 'Web_Forwards']

```
final = pd.DataFrame(final_features, columns= feature_names)
final
```

	Domain	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	TinyURL	Prefix/Suffix	DNS_Record	Web_Traffic	Domain_Age	Domain_End	iFrame	Mouse_Over	Right_Click	Web_Forwards
0	vvvvvv.paypal.com.xn-----vldfgifeq7dua2hckj.net	0	0	1	1	0	0	0	1	1	1	1	1	1	1	1	1

```
final1 = final.drop(['Domain','Right_Click','https_Domain'], axis = 1).copy()
final1
```

	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	TinyURL	Prefix/Suffix	DNS_Record	Web_Traffic	Domain_Age	Domain_End	iFrame	Mouse_Over	Web_Forwards
0	0	0	1	1	0	0	1	1	1	1	1	1	1	1

```
test=xgb.predict(final1)
tmp=final['Domain']
if(test[0]==0):
  print('Legitimate')
else:
  print('Phishing')
print(tmp)
```

```
Phishing
0    vvvvvv.paypal.com.xn-----vldfgifeq7dua2hckj.net
Name: Domain, dtype: object
```

iii. Mapping the results with problem statement and existing systems

The results we have found that the existing system maximum uses the random forest or the decision tree classifier and we have proven more accuracy with XGBoost, ensemble models and comparing the results. The data set we are using have much precision because of the extra attributes we have used and because of that we got better results

REFERENCES

[1] Samuel Marchal, Jérôme François, Radu State, and Thomas Engel, "PhishStorm: Detecting Phishing With Streaming Analytics," IEEE Transactions on Network and Service Management, vol. 11 , issue: 4 , pp. 458-471, December 2014

[2] Mohammed Nazim Feroz,Susan Mengel, "Phishing URL Detection Using URL Ranking," IEEE International Congress on Big Data, July 2015

[3] Mahdieh Zabihimayvan, Derek Doran, "Fuzzy Rough Set Feature Selection to Enhance Phishing Attack Detection," International Conference on Fuzzy Systems (FUZZ-IEEE), New Orleans, LA, USA, June 2019

[4] Moitrayee Chatterjee,Akbar-Siami Namin, "Detecting Phishing Websites through Deep Reinforcement Learning," IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), July 2019

[5] Chun-Ying Huang,Shang-Pin Ma,Wei-Lin Yeh,Chia-Yi Lin,ChienTsung Liu, "Mitigate web phishing using site signatures," TENCON 2010-2010 IEEE Region 10 Conference, January 2011

[6] Aaron Blum,Brad Wardman,Thamar Solorio,Gary Warner, "Lexical feature based phishing URL detection using online learning," 3rd ACM workshop on Artificial intelligence and security, Chicago, Illinois, USA, pp. 54-60, August 2010

[7] Mohammed Al-Janabi,Ed de Quincey,Peter Andras, "Using supervised machine learning algorithms to detect suspicious URLs in online social networks," IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017, Sydney, Australia, pp. 1104-1111, July 2010

[8] Erzhou Zhu,Yuyang Chen,Chengcheng Ye,Xuejun Li,Feng Liu, "OFSNN:An Effective Phishing Websites Detection Model Based on Optimal Feature Selection and Neural Network," IEEE Access(Volume:7), pp. 73271-73284, June 2019

[9] Ankesh Anand,Kshitij Gorde,Joel Ruben Antony Moniz,Noseong Park,Tanmoy Chakraborty,Bei-Tseng Chu, "Phishing URL Detection with Oversampling based on Text Generative Adversarial Networks," IEEE International Conference on Big Data (Big Data), December 2018

[10] Justin Ma,Lawrence K. Saul,Stefan Savage,Geoffrey M. Voelker, "Learning to detect malicious URLs," ACM Transactions on Intelligent Systems and Technology (TIST) archive Volume 2 Issue 3, April 2011

[11] Youness Mourtaji,Mohammed Bouhorma,Alghazzawi, "Perception of a new framework for detecting phishing web pages," Mediterranean Symposium on Smart City Application Article No. 11, Tangier, Morocco, October 2017

[12] T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer, "Social phishing," Communications of the ACM, vol. 50, no. 10, pp. 94–100, 2007.

[13] G. Xiang, J. Hong, C. P. Rose, and L. Cranor, "Cantina+: A feature-rich machine learning framework for detecting phishing web sites," ACM Transactions on Information and System Security (TISSEC), vol. 14, no. 2, p. 21, 2011.

[14] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Learning to detect malicious urls," ACM Transactions on Intelligent Systems and Technology (TIST), vol. 2, no. 3, p. 30, 2011.

[15] G. Tan, P. Zhang, Q. Liu, X. Liu, C. Zhu, and L. Guo, “Malfilter: A lightweight real-time malicious url filtering system in large-scale networks,” in 2018 IEEE ISPA/IUCC/BDCloud/SocialCom/SustainCom. IEEE, 2018, pp. 565–571.

[16] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, “Machine learning based phishing detection from urls,” Expert Systems with Applications, vol. 117, pp. 345–357, 2019. [17] L. Page, S. Brin, R. Motwani, and T. Winograd, “The pagerank citation ranking: Bringing order to the web.” Stanford InfoLab, Tech. Rep., 1999.

[18] M. S. I. Mamun, M. A. Rathore, A. H. Lashkari, N. Stakhanova, and A. A. Ghorbani, “Detecting malicious urls using lexical analysis,” in International Conference on Network and System Security. Springer, 2016, pp. 467–482.

[19] “2019 Threat Report,” https://wwwcdn.webroot.com/9315/5113/6179/2019 Webroot Threat Report US Online.pdf, Webroot, 2019.

[20] P. Zhao and S. C. Hoi, “Cost-sensitive online active learning with application to malicious url detection,” in Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2013, pp. 919–927.