Detecting Phishing Websites Using Machine Learning

Submitted in partial fulfilment of the requirements for the degree of

Bachelor of Technology

in

COMPUTER SCIENCE AND ENGINEERING WITH SPECIALIZATION IN BIOINFORMATICS

by

Sivaramalingam R
17BCB0132
Sivabi1200@gmail.com

Guna Aditya Kalvagadda
17BCB0060
guna.adiyavardhan369@gmail.com

Under the guidance of

Swarnalatha P
Associate Professor Grade 2
9443630735
pswarnalatha@vit.ac

School of Computer Science & Engineering VIT, Vellore.

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

June, 2021

DECLARATION

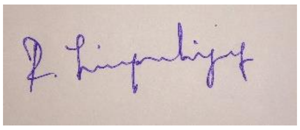
I hereby declare that the thesis entitled "Detecting Phishing Websites Using Machine Learning "submitted by me, for the award of the degree of *Bachelor of Technology in COMPUTER SCIENCE AND ENGINEERING WITH SPECIALIZATION IN BIOINFORMATICS* to VIT is a record of bonafide work carried out by me under the supervision of Swarnalatha P.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

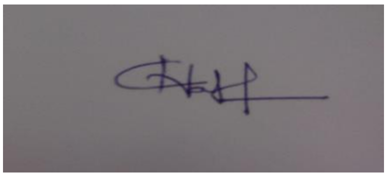
Place: Vellore

Date:8/06/21

Signature of the Candidate



R Sivaramalingam



K Guna aditya

<u>CERTIFICATE</u>

This is to certify that the thesis entitled “Detecting Phishing Websites Using Machine Learning” submitted by **Guna aditya Vardhan kalvagadda 17BCB0060 and Sivaramlingam R 17BCB0132 of School of Computer Science & Engineering**, VIT, for the award of the degree of *COMPUTER SCIENCE AND ENGINEERING WITH SPECIALIZATION IN BIOINFORMATICS,* is a record of bonafide work carried out by him / her under my supervision during the period, 01. 2. 2021 to 8.06.2021, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfils the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place : Vellore

Date :

Signature of the Guide

Swarnalatha P

11:25 PM (10 minutes ago)

to me

APPROVED FOR FURTHER PROCESS MR. SIVARAMALINGAM AND ADHITYA

Internal Examiner

External Examiner

Dr. Priya G

HOD- B.Tech CSE with spl in Bioinformatics And
B.Tech CSE &Business Systems

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to our mentor, Swarnalatha P Associate Professor, for providing us with the wonderful opportunity to work on this wonderful project on the topic Detecting Phishing Websites Using Machine Learning, which also assisted me in doing a lot of research and learning about so many new things.

Second, I'd want to thank my parents and friends for their assistance in completing this project in such a short period of time.

Student Name

K Guna aditya Vardhan(17BCB0060)

R Sivaramalingam(17BCB0132)

Executive Summary

Nowadays usage of the web is expanding step by step. Therewith cyber-theft also expanding step by step. The attacker phishing method is employed to collect the private information of innocent users. This manner is becoming more common nowadays. A phishing attack is the easiest method to get sensitive information from innocent users. The phishers aim to accumulate sensitive information from the users like username, password, bank account details, and far more sensitive information. Usually cyber intrusions are carried out via phishing assaults, when people are tricked into engaging with sites that appear to be genuine These pages are meant to appear real in order to successfully trick a person's user. People are so easily duped because they are so easily duped., robotized techniques for separating between phishing sites and their bona fide partners are required as a further line of protection. Online protection people are presently attempting to discover reliable and consistent location strategies for phishing sites discovery. This paper manages ML innovation for the identification of phishing URLs by separating and dissecting different highlights of genuine and phishing URLs. We used six classifiers algorithms: Logistic Regression, Regular boosting classifier, Decision Tree, AdaBoost Classifier, and Bagging Classifierthese are the machine-learning algorithms that are accustomed to detecting phishing websites. The paper aims to detect phishing URLs also as narrow them right down to the most effective machine learning algorithm by comparing accuracy rates. After finding the right algorithm for this problem with the very best accuracy rate. Then, weimplement the web application which algorithm provides more accuracy compared to other machine learning algorithms.

CONTENTS

.

List of Figures

List of Abbreviations

IP	Internet Protocol
URL	Uniform resource locator
HTTPS	Hypertext transfer protocol secure
SQL	Structured Query language
TLDs	Top-Level Domain

INTRODUCTION

Phishing is a strategy used by scammers to steal user information by impersonating legitimate websites. Payloads capable of sniffing important information when a user enters his information in online website that may contain infected web links given to random persons via email, messages, and other methods. People get mentally affected, and the website's illicit contents are rendered convincing, resulting in the extraction of personal information, such as passwords, usernames and bank debit/credit card information This is usually accomplished by faking reputable companies' web addresses so that the user never suspects illegalbehavior before providing their personal information.

One of the fundamental issues with creating ML-based methodologies for this issue is that not many preparing informational indexes containing phishing URLs are accessible in the public space. Accordingly, contemplates are required that assess the viability of ML approaches dependent on the informational collections that do exist. This work means to add to this need. In particular, the objective of this exploration is to analyze the presentation of the normally utilized AI calculations on the equivalent phishing informational index. In this work, we utilize an informational collection, where highlights from the information URLs have effectively been extricated, and the class marks are accessible. We going to test basic ML calculations to order URLs, for example, Logistic Regression, Regular Boosting Classifier, Decision Tree, AdaBoost Classifier and Stacking Classfier. After comparing all the algorithms, we select the best algorithm for creating web application which algorithm gives more accuracy.

Keywords – Phishing websites; classification; features; machine learning; Web application

1.1. OBJECTIVE

A phishing website is a common social engineering method to collect sensitive information from innocent users. The phishers collect critical information like username, password and banking details. We need a reliable and consistent identification procedures to identify these phishing websites. Also, build a web application.

1.2 Motivation

When new phishing strategies are developed, phishing detection solutions suffer from low detection accuracy and excessive warnings. Moreover, the most widely recognized strategy utilized, boycott based technique is wasteful in reacting to radiating phishing assaults since enlisting new space has become simpler, no thorough boycott can guarantee an ideal cutting-edge information base. Moreover, As a result, ensemble is frequently considered as a considerably superior option because it may consolidate similarities in precision and diverse blunder identification properties in chose calculations.

1.3 Background

The traditional way of detecting phishing websites is to discover and update such suspicious or illegal URLs, IP’s (Internet Protocol), to the database of phishing websites, this method of identifying phishing websites is called Blacklist method. To avoid being blacklisted, attackers use a variety of ways to deceive consumers, including altering URLs to make them appear authentic, obfuscation, and a variety of other basic tactics such as fast- flux, In this method, proxies are formed automatically to host the webpage; another method is to create URLs algorithmically, and so on.

The other technique is a Heuristic based detection, it detects the attacks based on the characteristics that are discovered in phishing attacks, this method can also be used to detect zero-hour attacks which the Blacklist method fails to detect, but it is not guaranteed that these characteristics always exist in the attack furthermore, bogus positive rate in identification is high.

To conquer the downsides Security specialists are currently centred around AI methods. AI calculations needs past information to settle on a choice or expectation on future information. This method can be utilized to examine and identify different phishing sites.

1. PROJECT DESCRIPTION AND GOALS

Abstract - Nowadays usage of the web is expanding step by step. Therewith cyber-theft also expanding step by step. The attacker phishing method is employed to collect the private information of innocent users. This manner is becoming more common nowadays. A phishing attack is the easiest method to get sensitive information from innocent users. The phishers aim to accumulate sensitive information from the users like username, password, bank account details, and far more sensitive information. Most of the cyber encroachments are carried out through these phishing attacks, Such pages are intended to resemble real in order to trick a user. As people tend to be so vulnerable, they are being deceived very easily, robotized strategies for separating between phishing sites and their true partners are required as a further line of protection. Network safety people are currently attempting to discover reliable and consistent discovery methods for phishing sites identification. This paper manages ML innovation for the identification of phishing URLs by separating and breaking down different highlights of real and phishing URLs. We used six classifiers algorithms: Logistic Regression, Regular boosting classifier, Decision Tree, AdaBoost Classifier, and Stacking Classifier these are the machine-learning algorithms that are accustomed to detecting phishing websites. The paper aims to detect phishing URLs also as narrow them right down to the most effective machine learning algorithm by comparing accuracy rates. After finding the right algorithm for this problem with the very best accuracy rate. Then, we implement the web application which algorithm provides more accuracy compared to other machine learning algorithms.

2. TECHNICAL SPECIFICATION

2.1 Hardware & Software Requirements

HARDWARE

- Windows 64-bit (7,8,10 supported)
- Minimum RAM requirement- 4GB

SOFTWARE

- Python
- Anaconda Navigator

Python libraries used:

- Numpy
- Pandas
- Flask
- Sklearn
- BeautifulSoup
- Requests
- Matplotlib
- Googlesearch
- Whois
- Socket
- Ipaddress
-

Approach and Design

Design

Architecture Diagram:

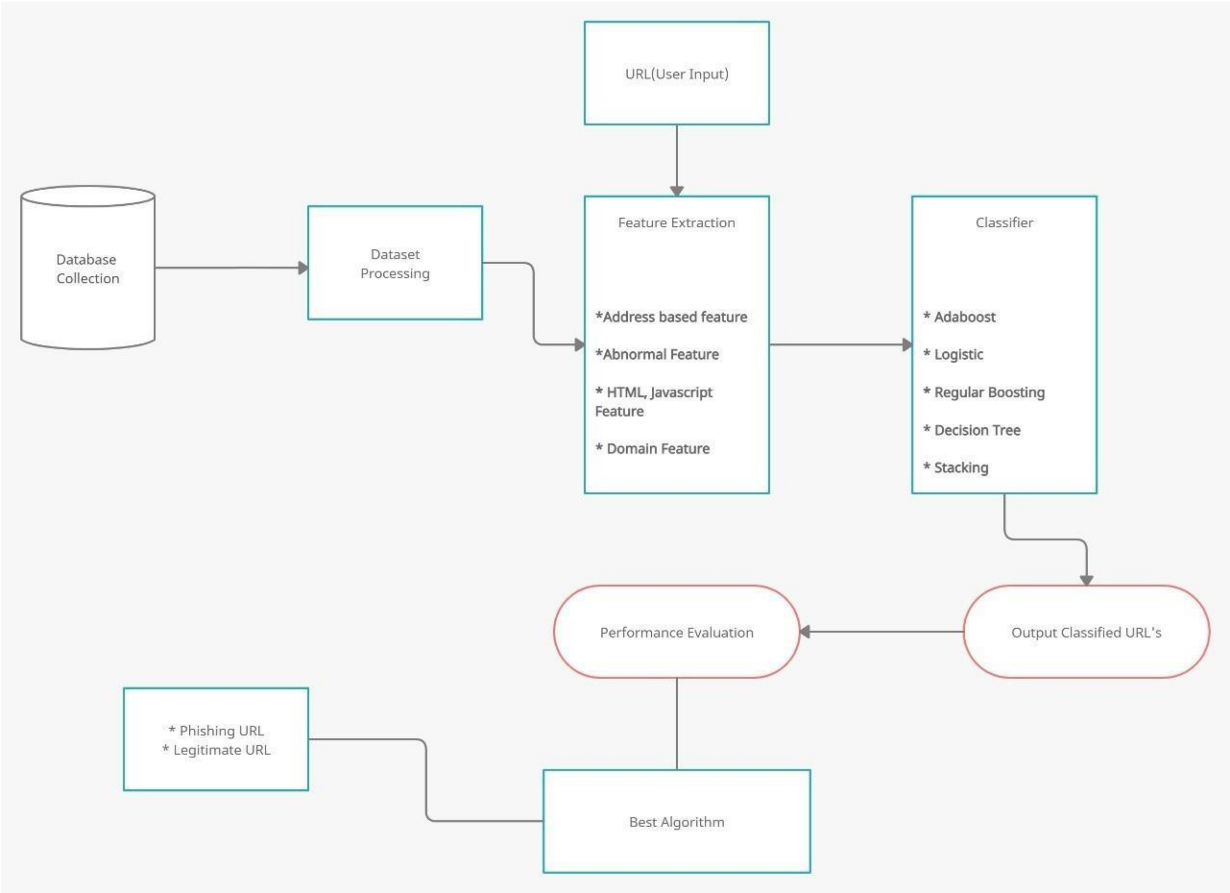


Figure 4.1 Architecture diagram

4.2 Literature review:

Table 1

Research Paper Title	Year of Publication	Methodology	Draw Backs
On Effectiveness of Source Code and SSL Based Features for Phishing Website Detection(Roopak .S, Athira P Vijayaraghavan, Tony Thomas)	2019	These Authors identify phishing webpagesbasedon Its URL and source code features To detect phishing websites, they used the RIPPER algorithm. Only 92 percent of phishing websites can be identified using webpage source code-based rules.	Black hat SEO techniques can readily overcome URL and domain-based functionality.
Phishing Websites Detection using Machine Learning	2019	The Authors Arun Kulkarni, Leonard L. Brown identify phishing webpages Using various machine learning techniques to classify	The limitation is that they only considered a small data sample of 1353 URLs, each with 9 attributes.

		them using their URL. They employed four Classifier algorithms: 1. the decision tree, 2. the naive Bayesian classifier, 3. the support vector machine (SVM), and 4. the neural network.	The models were successful in differentiating between real and fake Only 90percent of overall of the time can you determine the difference between legitimate and fraudulent websites.
Phishing Websites Detection Using Machine Learning **(**R. Kiruthiga, D. Akila)	2019	These Authors identify phishing webpages using Various machine learning approaches are being used to classify websites based on their URLs. They employed four Classifier algorithms: i. the decision tree, ii. the naive Bayesian classifier, iii. the support vector machine (SVM), and iv. the random forest.	This research focuses solely on detecting phishing website URLs based on domain name attributes. Only 96 percent of the time were the classifiers successful in discriminating between legitimate and bogus websites. Website URLs contains Many features but in this paper they used only domain name features only.

Detecting Phishing Websites Using Machine Learning	**2020**	**(**Sagar Patil, Yogesh Shetye, Nilesh Shendage) The Author used Traditional approach to distinguish phishing site has been to either to utilize a boycott of known phishing joins or heuristically assessthe properties in a suspected phishing page to recognize the presence of noxious pages. The heuristic capacity depends on experimentation to characterize the edge which is utilized to order malevolent connections from favorable ones. They used Four Machine Learning Algorithms:	The downside to this methodology is helpless precision and low versatility to new phishing joins. The classifiers were only successful in differentiating between legitimate and bogus websites 90% of the time.

		1. Logistic Regression, 2. SVM, 3. Decision Trees 4. Neural Networks	
Fuzzy Rough Set Feature Selection to Enhance Phishing Attack Detection **(**Mahdieh Zabihimayvan , Derek Doran)	2019	The author employs the Fuzzy Rough Set (FRS) hypothesis as a tool for selecting the finest highlights from three benchmarked informational indexes. To assess the FRS highlight determination in fostering a generalizable	The universal feature set does not include any features from third-party services, signalling that more inquiry from outside sources is required. The universal feature set does not include any features from third-party services, meaning that without further study from outside sources.

		phishing location, the classifiers are prepared by a different out-of-test informational index of 14,000 site tests.	

3. Methodology

3.1 Dataset:

To assess our ML strategies, we have utilized the 'Phishing Websites Dataset' from UCI Machine learning store. It comprises of 11,055 URLs (occasions) with 6157 phishing examples and 4898 authentic cases. Each occasion contains 30 features. Each component is related with a standard. In the event that the standardfulfills, it is named as phishing. In the event that the standard doesn't fulfill, it is named as legitimate. The features take three discrete values. If the condition is satisfied ‘1’, If the condition is partially satisfied ‘0’,If the condition is not satisfied ‘-1’.

The features represented by the training dataset can be classified into four categories;

- ✓ Address Bar based features
- ✓ Abnormal based features
- ✓ HTML and JavaScript based features
- ✓ Domain based features

Features based on Address Bar:

- IP Address

$$\begin{cases} \text{Domain Part has an IP Address} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

- Long URL

$$\begin{cases} \text{If the URL length} < 54 \text{ char} \rightarrow \text{feature = Then it is legitimate} \\ \text{Else if length} \geq 54 \text{ and} \leq 75 \rightarrow \text{feature = Then it is suspicious} \\ \text{otherwise} \rightarrow \text{feature = for sure it is Phishing(bogus)} \end{cases}$$

- Making the URL's shot using "TinyURL"

$$\begin{cases} \text{If it is a TinyURL} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

- The URL's that contain "@" symbol

$$\begin{cases} \text{If URL contains @} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

- Using "*"//" to redirect*

$$\begin{cases} \text{If the last Occurrence of "//" in the URL} > 7 \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

- *Attaching a Affix or Postfix to the Domain distinct by (-)*

$$\begin{cases} \text{contains } (-) \rightarrow \text{ Phishing} \\ \text{Otherwise} \rightarrow \text{ Legitimate} \end{cases}$$

- *Sub-Domains*

$$\begin{cases} \text{Dots present Part} = 1 \rightarrow \text{ Legitimate} \\ \text{Dots present Part} = 2 \rightarrow \text{ Suspicious} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$$

- *HTTPS*

$$\begin{cases} \text{The issuer and HTTPS Is Trusted and Age of Certificate} \geq 1 \text{ Years} \rightarrow \text{ Legitimat} \\ \text{The issuer and HTTPS Is Not Trusted} \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$$

- *Length of Domain enrolling*

$$\begin{cases} \text{If the Domains Expiring} \leq 1 \text{ years} \rightarrow \text{ Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

- *Favicon*

$$\begin{cases} \text{If Loaded from External Domain} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

- The usage of non-standard ports
- *The presence of the "HTTPS" Token*

$$\begin{cases} \text{Using HTTP Token} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

Features based on Abnormal:

- *A Request URL*

$$\begin{cases} \text{percentage of Request URL} < 22 \text{ percent } \rightarrow \text{ Legitimate} \\ \text{if percentage is} \geq 22\% \text{ and } 61\% \rightarrow \text{ Suspicious} \\ \text{Otherwise} \rightarrow \text{feature} = \text{Phishing} \end{cases}$$

- *Anchor of URL*

$$\begin{cases} \text{percentage Of Anchor } < 31\% \rightarrow \mathit{Legitimate} \\ \text{percentage of Anchor } \geq 31\% \text{ And} \leq 67\% \rightarrow \text{ Suspicious} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$$

- *Links in <Script>, <Meta>, <Link> tags*

$$\begin{cases} \text{percentage of Links in} < 17\% \rightarrow \text{ Legitimate} \\ \text{percentage of Links in} \geq 17\% \text{ And} \leq 81\% \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$$

- Server Form Handlers

  Server Form Handlers is "about: blank" Or Is Empty → Phishing
  Server Form Handlers Refers To A Different Domain → Suspicious
  Otherwise → Legitimate

- *Using Email to Submit Information*

  Presence of “mail ()" or "mailto:" → Phishing
  Otherwise → Legitimate

- *Curious URL*

  Host Name Is Not Included→ Phishing
  Otherwise → Legitimate

Features based on HTML && JavaScript:

- *Assist of websites*

  Percentage of Redirecting ≤ 1 → Legitimate
  Percentage of Redirecting ≥ 2 And < 4 → Suspicious
  Otherwise → Phishing

- *Customization of Grade Bar*

  on Mouseover Changed →Phishing
  Doesn't Change Status Bar → Legitimate

- *Right click is disabled*

$$\begin{cases} \text{Right Click is Disabled} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

- *The Popup Window*

$$\begin{cases} \text{Popup Contains Text Fields} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

- *Redirection of Iframe*

$$\begin{cases} \text{iframe} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

Features based on Domain:

- *Life of Domain*

$$\begin{cases} \text{Age} \geq 6 \text{ months} \rightarrow \text{Legitimate} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$$

- *DNS Log*

$$\begin{cases} \text{No Record for The Domain} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

➢ *Traffic of the Website*

$$\begin{cases} \text{If Rank} < 100{,}000 \rightarrow \text{Legitimate} \\ \text{If Rank} > 100{,}000 \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$$

➢ *Page Grade*

$$\begin{cases} \text{PageRank} < 0.2 \quad \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

➢ *The Google Indices*

➢ *Number of Links That are Redirecting to Page*

➢ Feature Based On Statistical-Reports

Features	**Malicious Website**	**Safe Website**
IP Address	Website Contains IP	Website not contains IP
Long URL	URL length ≥ 54 and ≤ 75	URL length <54
Making the URL's shot using "TinyURL"	Tiny URL	Not Tiny URL
The URL's that contain "@" Symbol	URL contains @ Symbol	URL not contains @ symbol
Using "//" to redirect	If last instance of // in the URL >7	Else

Attaching a Affix or Postfix to the Domain distinct by (-)	Affix or Postfix (-) symbol contains	Not Contains
Sub-Domains	(.) symbol count >2	URL has no sub-Domains
HTTPS	else	HTTPS is trusted && life certification ≥ 12 months
Length of Domain enrolling	Domains Expiring ≤ 12 months	Else
Favicon	Loaded from external resource	Else
The Usage of Non-Standard ports	Entire port are exposed	Else
The presence of the "HTTPS" Token	Using HTTP Token	Else
A Request URL	Percentage of appeal URL larger than 22 percentage	Percentage of appeal URL less than 22 percentage
Anchor of URL	Else	Percentage of Anchor less than 31 Percentage
Links in <Script> <Meta> <Link> tags	Else	Percentage of Splice in less than 17 percentage
SFH	Server Form Handlers is "about: blank"	Else
Using Email to Submit Information	Presence of "mail()" or "mailto:"	Else
Curious URL	Host name is not involved	Else
Assist of websites	Percentage of Redirecting ≥1	Percentage of Redirecting ≤ 1
Customization of Grade Bar	on Mouseover Changed	Doesn't remake Grade Bar
Right Click is disabled	Right Click is disabled	Else
The Blooper Window	Blooper hold Text Fields	Else

Redirection of IFrame	IFrame is a tag that displays an affixed webpage within the one that is now displayed. Attackers can hide the "iframe" element, for example, by removing the frame boundaries	Else
Life Of Domain	Else	Life of Domain ≥ half year
DNS log	No Log For The Domains	Else
Traffic of the Website	Website Grade greater than 100000	Website Grade less than 100000
Page Grade	Page Grade less than 0.2	Else
The Google Indices	This trait determines in case a website is in Google's cache. When a location is filed by Google, it appears on query items. Because phishing websites are typically only available for a limited time	Else
Number of Links That are Redirecting to Page	Quantity Number of connections highlighting website page demonstrates the respective authenticity level, irrespective of whether a few connections are of a similar domain. We discovered that ninety-eight percent of malicious dataset items have no links linking to them in our datasets, owing to their short lifespan	Legitimate websites, on the other hand, have at least two outer links placing to them.

Feature Based On Statistical-Reports	A few gatherings like PhishTank , and StopBadware figure various factual reports on phishing sites at each given timeframe; some are month to month Others are done on a quarterly basis	Else

3.2 Implementation:

Python Packages:

Pandas:

Pandas is basically utilized for data investigation. Pandas permits bringing in data from different document organizations, for example, comma-isolated qualities, JSON, SQL, Microsoft Excel. Pandas permits different data control tasks like combining, reshaping, choosing, just as data cleaning, and data fighting highlights.

Numpy:

NumPy includes multidimensional arrays and lattice data structures. It is commonly used to perform many numerical procedures on arrays, such as geometrical, factual, and mathematical scheduling. As a result, the library contains a massive amount of numerical, mathematical, and change capabilities.

Scikit-learn:

Scikit-learn is one of most valuable and robust machine learning library in Python. It provides a set of professional tools for ml and quantifiable presentation, such as categorization, regression, grouping, and dimensionality reduction, via a Python consistency interface.

Matplotlib:

Matplotlib is a plotting library for making static, enlivened, and intelligent representations in Python.

Flask:

A Python API that permits developers to develop web-based applications.

Beautiful Soup:

To extract information from webpages like HTML, XML and other markup languages, A

python library Beautiful soup is used

Requests:

Requests will permit you to send HTTP/1.1 requests utilizing Python. With it, you can add content like headers, structure data, multipart documents, and boundaries through straightforward Python libraries. It additionally permits you to get to the reaction data of Python similarly.

GoogleSearch:

Googlesearch is a Python library for looking through Google, without any problem. Googlesearch utilizes solicitations and BeautifulSoup4 to scratch Google.

Whois:

Make a basic importable Python module that will deliver parsed WHOIS information for an offered domain. Able to extract information for all the well-known TLDs (com, organization, net, …)

Socket:

Socket writing computer programs is a method of associating two hubs on an organization to communicate with one another.

Ipaddress:

The python module ipaddress is utilized widely to approve and classify IP address to IPV4 and IPV6 type. It can likewise be utilized to do examination of the IP address esteems just as IP address number-crunching for controlling the ip addresses.

Dataset Collections:

To assess our ML strategies, we have utilized the 'Phishing Websites Dataset' from UCI Machine learning store. It comprises of 11,055 URLs (occasions) with 6157 phishing examples and 4898 authentic cases. Each occasion contains 30 features. Each component is related with a standard. In the event that the standard fulfills, it is named as phishing. In the event that the standard doesn't fulfill, it is named as legitimate. The features take three discrete values. If the condition is satisfied '1', If the condition is partially satisfied '0', If the condition is not satisfied '-1'.

Importing Libraries

```
In [1]: import pandas as pd
        import numpy as np
        import  matplotlib.pyplot as plt
```

Collection of Data

```
In [2]: data=pd.read_csv("C:/Users/H4CK3R/Desktop/Phishing/detect_phishing_website.csv")
```

```
In [3]: data.head()
```

Out[3]:

	id	having_IP_Address	URL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain	SSLfinal_State	Dor
0	1	-1	1	1	1	-1	-1	-1	-1	
1	2	1	1	1	1	1	-1	0	1	
2	3	1	0	1	1	1	-1	-1	-1	
3	4	1	0	1	1	1	-1	-1	-1	
4	5	1	0	-1	1	1	-1	1	1	

5 rows × 32 columns

Figure 5.1 data collection

Data Pre-processing:

We removed null values and unwanted columns from the datasets to get better accuracy.

Data PreProcessing

```
In [5]: #Removing Unnecessary columns
        data.drop(["id"], axis = 1, inplace = True)
```

```
In [6]: data.columns
```

```
Out[6]: Index(['having_IP_Address', 'URL_Length', 'Shortining_Service',
               'having_At_Symbol', 'double_slash_redirecting', 'Prefix_Suffix',
               'having_Sub_Domain', 'SSLfinal_State', 'Domain_registeration_length',
               'Favicon', 'port', 'HTTPS_token', 'Request_URL', 'URL_of_Anchor',
               'Links_in_tags', 'SFH', 'Submitting_to_email', 'Abnormal_URL',
               'Redirect', 'on_mouseover', 'RightClick', 'popUpWidnow', 'Iframe',
               'age_of_domain', 'DNSRecord', 'web_traffic', 'Page_Rank',
               'Google_Index', 'Links_pointing_to_page', 'Statistical_report',
               'Result'],
              dtype='object')
```

```
In [7]: data.shape
```

```
Out[7]: (11055, 31)
```

```
In [8]: #Checking Data can contain null values
        data.isnull().values.any()
```

```
Out[8]: False
```

Figure 5.2 Data preprocessing

Data Splitting:

To fit the models over the dataset the dataset is parted into training and testing sets. The split proportion is 80-20. Where in 80% records to training set.

Machine Learning Algorithms:

On the training dataset, we developed and tested supervised machine learning algorithms. The methods listed below were chosen based on their performance on classification problems. 80% of the dataset is taken for Training sets and remaining 20% is taken into test sets. The outcomes of our experiment aredetailed in the results section.

- ✓ Logistic Regression
- ✓ Regular Boosting
- ✓ Decision Tree
- ✓ Adaboost Classifier
- ✓ Stacking Classifier

Logistic Regression:

Logistic Regression is a key characterization procedure. It has a place with the gathering of direct classifiers and is to some degree like polynomial and straight regression. Logistic Regression is quick and generally straightforward, and it's helpful for you to decipher the outcomes. In spite of the fact that it's basically a strategy for parallel grouping, it can likewise be applied to multiclass issues.

Logistic Regression

In [23]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
```

In [19]:
```
lr=LogisticRegression(random_state = 0)
```

In [20]:
```
lr.fit(x_train,y_train)
```

Out[20]:
```
LogisticRegression(random_state=0)
```

In [21]:
```
#Predicting the result for test data
y_predict=lr.predict(x_test)
```

In [24]:
```
print("Train Accuracy : ",100*lr.score(x_train,y_train))
print("Test Accuracy : ",100*lr.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train Accuracy :  92.93306196291272
Test Accuracy :  92.08502939846224
              precision    recall  f1-score   support

          -1       0.92      0.91      0.91      1007
           1       0.92      0.93      0.93      1204

    accuracy                           0.92      2211
   macro avg       0.92      0.92      0.92      2211
```

Figure 5.3 Logistic regression

Regular Boosting:

Regular boosting classifiers are a collection of AI algorithms that combine multiple inadequate learning models to create a strong predictive model. When undertaking inclination boosting, choice trees are commonly used. Regular boosting models are gaining popularity because to their ability to organize complex datasets, and have recently been used to win multiple Kaggle information science competitions.

Regular Boosting Classifier

```
In [25]: from sklearn.ensemble import GradientBoostingClassifier

In [26]: rb=GradientBoostingClassifier()

In [27]: rb.fit(x_train,y_train)

Out[27]: GradientBoostingClassifier()

In [28]: #Predicting the result for test data
         y_predict=rb.predict(x_test)

In [29]: print("Train Accuracy : ",100*rb.score(x_train,y_train))
         print("Test Accuracy : ",100*rb.score(x_test,y_test))
         print(metrics.classification_report(y_test,y_predict))

         Train Accuracy :  95.26232473993667
         Test Accuracy :  94.5273631840796
                       precision    recall  f1-score   support

                  -1       0.95      0.93      0.94      1007
                   1       0.94      0.96      0.95      1204

            accuracy                           0.95      2211
           macro avg       0.95      0.94      0.94      2211
        weighted avg       0.95      0.95      0.95      2211
```

Figure 1.4 Regular boosting classfier

Decision Tree:

Decision tree is quite possibly the most remarkable and famous calculation. Decision tree calculation falls under the class of administered learning calculations. It works for both nonstop just as straight out yield factors. Officially a Decision tree is a graphical portrayal of all potential answers for a decision. Nowadays, tree-based calculations are the most ordinarily utilized calculations on account of administered learning situations. They are simpler to decipher and picture with extraordinary versatility. We can utilize tree-based calculations for both relapse and order issues .However, more often than not they are utilized for characterization issue.

Decision Tree

```
In [30]: from sklearn.tree import DecisionTreeClassifier

In [31]: dt=DecisionTreeClassifier(random_state=0)

In [32]: dt.fit(x_train,y_train)

Out[32]: DecisionTreeClassifier(random_state=0)

In [33]: #Predicting the result for test data
         y_predict=dt.predict(x_test)

In [34]: print("Train Accuracy : ",100*dt.score(x_train,y_train))
         print("Test Accuracy : ",100*dt.score(x_test,y_test))
         print(metrics.classification_report(y_test,y_predict))

         Train Accuracy :  99.10673903211217
         Test Accuracy :  95.92944369063771
                       precision    recall  f1-score   support

                   -1       0.96      0.95      0.96      1007
                    1       0.96      0.97      0.96      1204

             accuracy                           0.96      2211
            macro avg       0.96      0.96      0.96      2211
         weighted avg       0.96      0.96      0.96      2211
```

Figure 5.6 Decision tree

Adaboost Classifier:

The adaBoost strategy follows a decision tree model with a profundity equivalent to one. AdaBoost is only the woods of stumps instead of trees. AdaBoost works by putting more weight on hard-to-arrange occurrences and less on those all around dealt with well. The adaBoost calculation is created to tackle both regression and classification issues.

AdaBoost Classifier

```
In [35]: from sklearn.ensemble import AdaBoostClassifier

In [36]: adc=AdaBoostClassifier()

In [37]: adc.fit(x_train,y_train)

Out[37]: AdaBoostClassifier()

In [38]: #Predicting the result for test data
         y_predict=dt.predict(x_test)

In [39]: print("Train Accuracy : ",100*adc.score(x_train,y_train))
         print("Test Accuracy : ",100*adc.score(x_test,y_test))
         print(metrics.classification_report(y_test,y_predict))

Train Accuracy :  93.9280868385346
Test Accuracy :  93.08005427408412
              precision    recall  f1-score   support

          -1       0.96      0.95      0.96      1007
           1       0.96      0.97      0.96      1204

    accuracy                           0.96      2211
   macro avg       0.96      0.96      0.96      2211
weighted avg       0.96      0.96      0.96      2211
```

Figure 5.7 Adaboost classifier

Ensemble Methods:

Stacking is an AI calculation performed by a group. It employs a meta-learning computation to determine how to optimally combine the outcomes of at least two base AI algorithms. Stacking has the advantage of bridging the capacities of a variety of well-performing models on an order or relapse undertaking and making projections that outperform any one model in the group..

At least two base models are used in the engineering of a stacking model. The meta-model is built on the forecasts produced by basic models using out-of-test data.

Stacking Classifier

```
In [57]: from sklearn.model_selection import cross_val_score
         from sklearn.model_selection import RepeatedStratifiedKFold
         from sklearn.ensemble import StackingClassifier
         from numpy import mean
         from numpy import std
```

```
In [49]: # get a stacking ensemble of models
         level0 = list()
         level0.append(('lr', LogisticRegression()))
         level0.append(('dt', DecisionTreeClassifier()))
         level0.append(('adc', AdaBoostClassifier()))
         level0.append(('rb', GradientBoostingClassifier()))
         # define meta learner model
         level1 = LogisticRegression()
         # define the stacking ensemble
         model = StackingClassifier(estimators=level0, final_estimator=level1, cv=5)
```

```
In [50]: # get a list of models to evaluate
         models = dict()
         models['Logistic'] = LogisticRegression()
         models['Decision'] = DecisionTreeClassifier()
         models['AdaBoost'] = AdaBoostClassifier()
         models['Regular'] = GradientBoostingClassifier()
         models['Stacking'] = model
```

Figure 5.8 stacking classifier

Web Application:

Among all the single machine learning algorithms decision tree gave highest accuracy among all

The algorithms. After finding the best algorithm we saved model and developed a web application.

This web application will predict the given url whether it is phishing website or legitimate website.

Codes:

Saved machine learning model

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
import features_extraction


  def getResult(url):
     data = pd.read_csv("detect_phishing_website.csv")
     # Removing Unnecessary columns
     data.drop(["id"], axis=1, inplace=True)
     # Removing Null values
     data = data.dropna()
     # Define X && Y
     y = data.Result
     x = data.drop('Result', axis=1)
     # splitting the data into train data and test data
     x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
     # Creating the model and fitting the data into the model
     dt = DecisionTreeClassifier(random_state=0)
     dt.fit(x_train, y_train)
     # Predicting the result for test data
     y_predict = dt.predict(x_test)
     score = dt.score(x_test, y_test)
     print(100 * score)
     X_new = []
     X_input = url
     X_new = features_extraction.generate_data_set(X_input)

     X_new = np.array(X_new).reshape(1, -1)
     # print(X_new)

     try:
           prediction = dt.predict(X_new)
           if prediction ==-1:
                   return "Phishing Url"
           else:
                   return "Legitimate Url"
           except:
                   return "Phishing Url"

# getResult("
```

API program

```
import os
import decision_tree
from flask import Flask
from flask import (
    Blueprint, flash, g, redirect, render_template, request, session, url_for
)
from flask import jsonify
from werkzeug.utils import secure_filename

app = Flask(__name__)



@app.route('/result')
def result():
    urlname = request.args['name']
    result = decision_tree.getResult(urlname)
    return result



@app.route('/', methods=['GET', 'POST'])
def hello():
    return render_template("phishing_detection.html")


if __name__ == '__main__':
    app.run(debug=True)
```

4. PROJECT DEMONSTRATION

Phishing Website Detection:

This is the detection of phishing website.

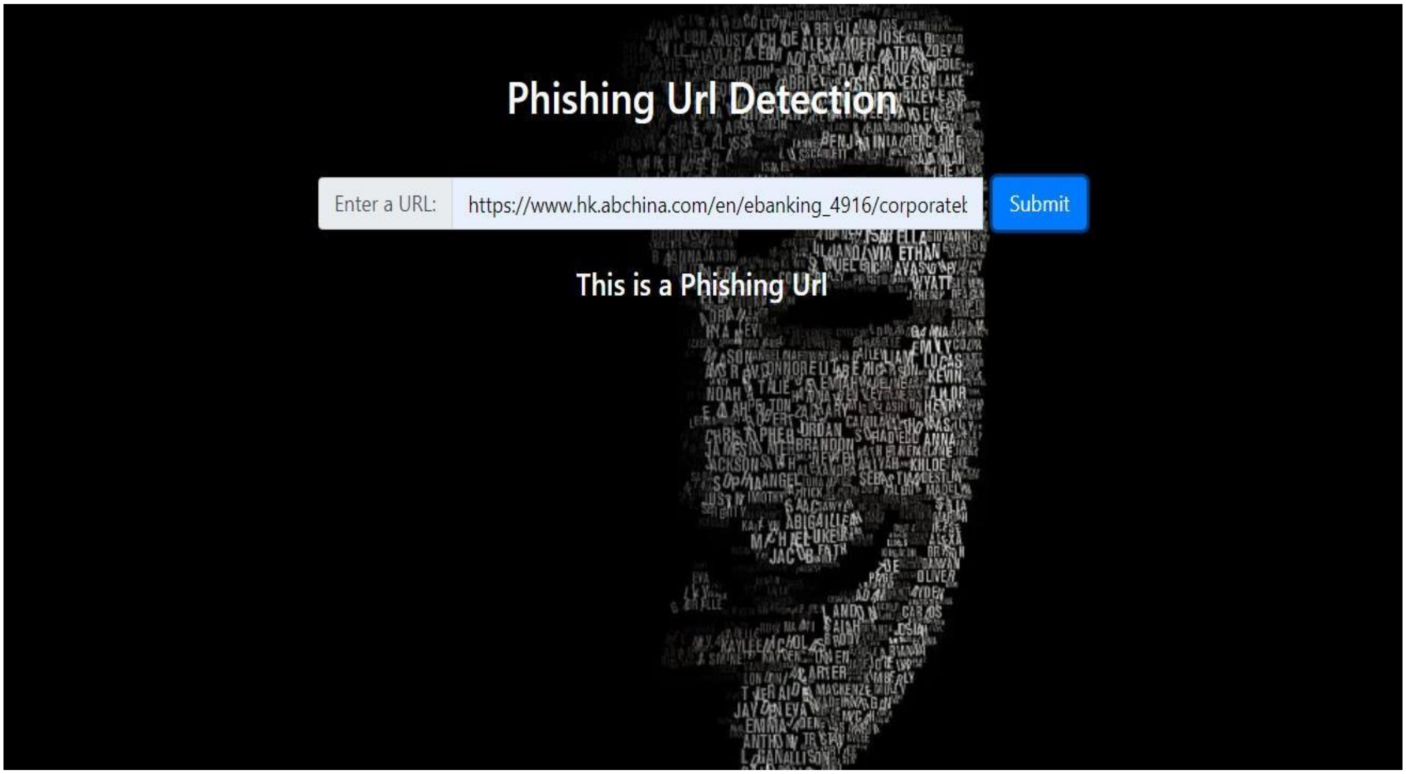


Figure 6.1 web Application

Any user with suitable internet connection, while surfing through the web if they get suspicious of a website and want to check if it’s a legitimate website or not they can simply copy and paste the website URL and our app will identify if it’s a legitimate website or not

Legitimate Website Detection:

This is the detection of Legitimate website.

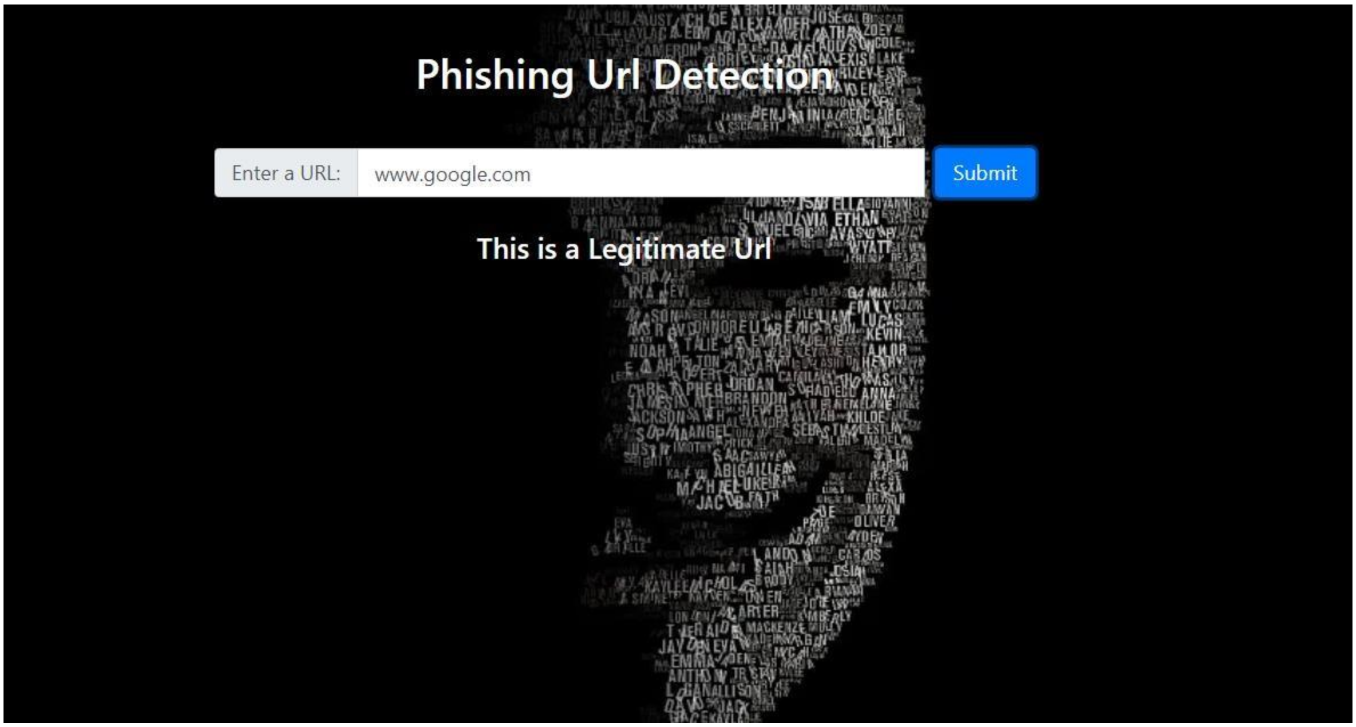


Figure 6.2 Detection

5. Results and Discussion:

To assess our ML strategies, we have utilized the 'Phishing Websites Dataset' from UCI Machine learning store. It comprises of 11,055 URLs (occasions) with 6157 phishing examples and 4898 authentic cases. Each occasion contains 30 features. Each component is related with a standard. In the event that the standard fulfills, it is named as phishing. In the event that the standard doesn't fulfill, it is named as legitimate. The features take three discrete values. If the condition is satisfied '1', If the condition is partially satisfied '0', If the condition is not satisfied '-1'.

The features represented by the training dataset can be classified into four categories;

- ✓ Address Bar based features
- ✓ Abnormal based features
- ✓ HTML and JavaScript based features
- ✓ Domain based features

On the training dataset, we developed and tested supervised machine learning algorithms. The methods listed below were chosen based on their performance on classification problems. 80% of the dataset is taken for Training sets and remaining 20% is taken into test sets. The outcomes of our experiment aredetailed in the results section.

- ✓ Logistic Regression
- ✓ Regular Boosting
- ✓ Decision Tree
- ✓ Adaboost Classifier
- ✓ Stacking Classifier

Model Comparison:

Sorted Order by Train Accuracy

In [44]:
```
results.sort_values(by=['Train Accuracy','Test Accuracy'], ascending=False)
```

Out[44]:

	ML Model	Train Accuracy	Test Accuracy
2	Decision Tree	0.991067	0.959294
1	Regular Boosting	0.952623	0.945274
3	AdaBoost Classifire	0.939281	0.930801
0	Logistic Regression	0.929331	0.920850

Figure 2 Sorted order accuracy

Sorted Order by Test Accuracy

In [45]:
```
results.sort_values(by=['Test Accuracy', 'Train Accuracy'], ascending=False)
```

Out[45]:

	ML Model	Train Accuracy	Test Accuracy
2	Decision Tree	0.991067	0.959294
1	Regular Boosting	0.952623	0.945274
3	AdaBoost Classifire	0.939281	0.930801
0	Logistic Regression	0.929331	0.920850

Figure 3

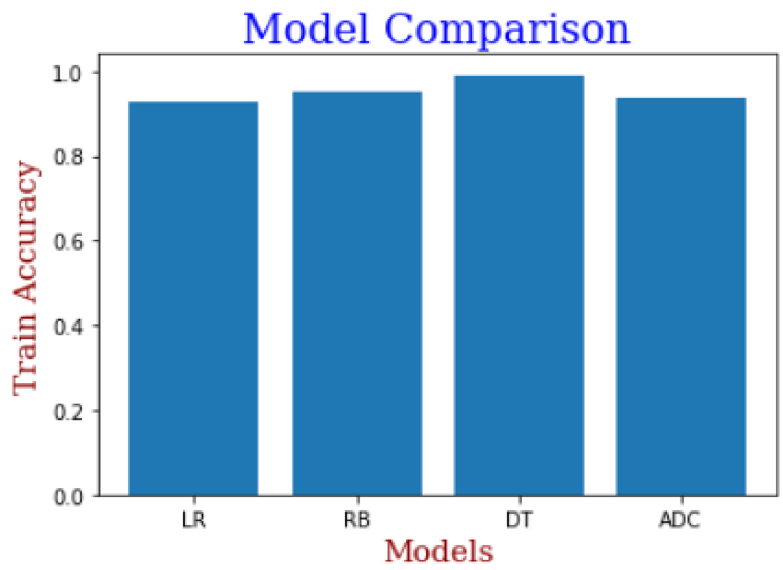


Figure 4 Model comparison

In out of four single machine learning algorithms decision tree gives highest accuracy among all.

```
>Logistic 0.928 (0.006)
>Decision 0.961 (0.007)
>AdaBoost 0.938 (0.007)
>Regular 0.949 (0.007)
>Stacking 0.964 (0.006)
```

In this situation, we can see that the stacking group hopes to beat any single model by and large, with an exactness of roughly 96.4 percent.

6. Future work

BRAND ALERT

A Brand alert program as we our users can check on our website if the URL is of a phishing website or not, Our next step is to alert the companies or any Brand that are being targeted by the attackers so that they can warn and save their respective customers from being scammed

HOST ALERT

As a hosting services provider, your customer may be hacked and host phishing. To be alerted in real-time about all hosts that are currently hosting phishing websites, please fill out the following form.

Conclusion

Phishing is an increasing crime that This is something everyone has to be mindful of. Regardless of the existence of rules, the biggest weapon against phishing is education. It's a smart option to exercise caution when using technological devices or visiting webpages. Stay updated out for similar features including a sense of danger, a desire for verification, and punctuation and grammatical issues. Feel the necessity to compare the stated URL to a scan for the company's website on your own.

REFERENCES:

[1] Roopak .S, Athira P Vijayaraghavan, Tony Thomas, “On Effectiveness of Source Code and SSL Based Features for Phishing Website Detection,” in Indian Institute of Information technology and Management-Kerala Thiruvananthapuram, India. Springer, 2019.

[2] Arun Kulkarni, Leonard L. Brown, “Phishing Websites Detection using Machine Learning, ” Department of Computer Science The University of Texas at Tyler Tyler, TX, 75799. International Journal of Advanced Computer Science and Applications, Vol. 10, No. 7, 2019.

[3] R. Kiruthiga, D. Akila, “Phishing Websites Detection Using Machine Learning ,” International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-8, Issue-2S11, September 2019.

[4] Sagar Patil, Yogesh Shetye, Nilesh Shendage, “Detecting Phishing Websites Using Machine Learning ,” in 1,2,3Department of Information Technology, Padmabhushan VasantDada Patil Pratishthan’s College of Engineering, Sion, Mumbai Maharashtra, India. International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056 Volume: 07 Issue: 02 | Feb 2020 www.irjet.net p-ISSN: 2395-0072.

[5] Mahdieh Zabihimayvan and Derek Doran ,” Fuzzy Rough Set Feature Selection to Enhance Phishing A+ttack Detection ,” Department of Computer Science and Engineering Wright State University, Dayton, OH, USA {zabihimayvan.2, derek.doran}@wright.edu. Springer,201