FALL SEMESTER 2021-22

CSE4003- CYBER SECURITY (J COMPONENT) – Review 2 and 3

Title: Detecting Phishing Websites Using Machine Learning

Slot: E2+ TE2

Faculty name: Mrs. Gayathri P

Group members:

Deva Dharshan D – 20BCE0282

Akshaya P – 20BDS0365

Karthik Ajith – 20BDS0261

Om Mane – 20BCE2632

Theo Daniel – 20BCB0122

1.Abstract:

With the goal of fooling the users, the cyber attackers will create a software which will be looking as same as the real websites and many basic functionalities of the real websites will be available in the fake ones too and we will believe it as the real websites and we will be giving our personal details in the malicious website and they will be misusing it. In order to check whether the website is verified and legitimate we are going to make a prototype which checks the phishing website URL in different stages and make conclusion whether the website is fake or legitimate.

Our primary aim is to check the websites and to verify using machine learning whether it is safe and advisable to use by the netizens or it contains the malicious software that steals the details of the users through the particular phishing website.

2.Key Words:

Phishing detection and Phishing website

Features of url,

Machine learning ,

Algorithms.

3.Introduction:

While cybersecurity attacks continue to escalate in both scale and expertise, social engineering methods still work some of the easiest and most effective ways to reach sensitive or confidential information. the United States Computer Emergency Readiness Team (US-CERT) explains crime of stealing sensitive information as a means of social engineering using emails or malicious websites to request personal information from a person or company by pretending to be an honest organization or organization.

Simply phishing is a strategy used by scammers to steal user information by impersonating official websites in which the innocent people. Payloads that can hold important information when a user uploads information to an online website that may contain infected web links provided to random people via email, messages and other means. People are mentally disturbed and illegal website content is provided convincingly, leading to the release of personal information such as usernames and password details, bank / credit card details. This often leads to fraudulent web address companies so that the user will never suspect illegal behavior before providing his or her personal information.

One of the key issues with creating ML-based approaches to this issue is that there are not many corrections for information indexes containing stolen URLs available in the public space. Similarly, indicators are needed to evaluate the effectiveness of ML methods based on existing data sets. This wok means adding to this need. In particular, the purpose of this study is to analyze the presentation of AI statistics commonly used in the context of information theft of equally sensitive information. For this function, we use a database where the highlights from the information URLs have been successfully extracted and class marks are obtained. We will test basic ML statistics for ordering URLs, for example Logistic Regression Method, Classified Boosting Classifier Method, Random Forest Method, Ada boost Classifier Method and Stacking Classifier Method and by comparing all those models, will be finding the most accurate website model.

Objective: A phishing website is a common social engineering method to collect the sensitive or personal information from the users who are not aware of it. The target of this project is to train various machine learning models on the dataset created to predict the phishing websites with 90%> accuracy. The performance of every feature to check the fake websites are compared and taken into account.

4.Literature review:

Ref no.	Paper title	Journal name and year of publication	Work done	Techniques used	Disadvantages	Student name
1	Detecting Phishing Website Using Machine Learning (Mohammed Hazim Alkawaz, Stephanie Joanne Steven, Asif Iqbal Hajamydeen)	Scholar works 2020	In this research they have collected the information how the malicious software are being injected into the host PC and how they can prevent it by providing	Agile Unified Process (AUP) Model.	Some time while sending to multiple accounts the mail may stored in the spam folders in the mail. The website may wrongly think as a malicious message and send irrelevant notifications to the users.	Deva Dharshan D (20BCE0282)
2	Phishing Website Detection Using Machine Learning. (Leonard L. Brown, Arun Kulkarni)	IEEE Explore 2019	They used two steps for finding the phishing websites. The first step is to extract features from the URLs, and the second step is to classify URLs using the model that has been developed with the help of the training set data.	Naïve Bayes' Classifier Technique, Support Vector Machine(SVM) Technique, and the Neural Network Technique	there are lakhs of phishing websites appx 17,00,000 till today and this model is proposed by using only 1300+ phishing websites which is a major drawback.	Deva Dharshan D (20BCE0282)

3	Detection of phishing attacks (Muhammet Baykara, Zahit Ziya Gürel)	IEEE Explore 2018	it will determine if the message contained phishing issues or not and also check for the phishing message in the e-mail in primitive ways and also its source of arrival and whether it is directing to another website or not	Bayesian network Method.	In the Bayesian Network method building the network automatically is quite complicated. The Bayesian Network method can't handle sequence of information.	Deva Dharshan D (20BCE0282)
4	On Effectivness of source code and SSL based features for Phishing website Detection. (Roopak.S , Athira P Vijayaraghavan, Tony Thomas)	IEEE Explore 2019	They identifies phishing web pages based on its URL and source code features using RIPPER algorithm.	RIPPER algorithm, Secure socket layering	Black hat SEO techniques can easily overcome URL and domain based funcionality	Akshaya P (20BDS0365)
5	WC-PAD : Web Crawling based Phishing Attack Detection. (Nathezhtha T , Sangeetha D, Vaidehi V)	IEEE Explore	WC-PAD performs a three phase identification mechanism. checks for the IP in DNS blacklist, If WC-PAD found any spam activity immediately it send alert messages.	Heuristic analysis, Web crawler, Web content analyzer, web traffic analyzer, Alexareputation.	Alexa's statistics are not necessarily accurate in the absolute terms. Alexa Rank is not something valuable in reference to overall SEO optimization and performance	Akshaya P (20BDS0365)

6	Detecting Phishing website using Machine Learning (Amani Alswailern , Bashayr Alabdullah, Norah Alrumayh, Dr.Aram Alsedrani.)	IEEE Explore	Authors collected 16000 phishing and legimate URLs and using features extraction and and then execute them in final classifer algorithm using Random forest technique.	Random Forest	In Random Forest, large number of trees can make the algorithm too slow and ineffective for real-time predictions..	Akshaya P (20BDS0365)
7	Detecting Phishing Websites Using Machine Learning (Sagar Patil, Yogesh Shetye, Nilesh Shendage)	International Research Journal of Engineering and Technology (IRJET)-2020	software product which uses machine learning algorithm to detect malicious URLs. To evaluate their machine learning techniques, the authors have used the 'Phishing Websites Dataset' from UCI Machine learning repository.	Random Forest, Artificial Neural Networks, Support Vector Machine(SVM)	Random Forest - A major disadvantage of random forests algorithm is it does not gives precise continous forecast. SVM algorithm is not suitable for large data sets.	Om Mane (20BCE2632)
8	Detection of phishing email based on NLP and ML (Shaikh sayema anwer, Asst.Prof. V. Karwande)	IEEE Explore 2021	logically by Naive Bayes (NB) and Multinomial Naive Bays (MNB). The approach they offered in their 'Spam Mail Collection' data set showed an average accuracy of 98.49% utilizing the Naive Bayes (NB) model.	Naive Bayes (NB), Multinomial Naive Bayes (MNB)	Naive Bayes assumes that all predictors (or features) are independent, rarely happening in real life. This limits the applicability of this algorithm in real-world use cases.	Om Mane (20BCE2632)

9	A Review on Phishing Attacks (Akarshita Shankar, Ramesh Shetty, Badari Nath K)	IEEE Explore 2019	This study provides an in-sight tophishing, the mechanism of the attack, various forms it canoccur in and the possible solutions to overcome them	Automated Individual White-List (AIWL), Natural Language Processing (NLP)	Natural Language Processing (NLP) - NLP is unable to adapt to the new domain. NLP is built for a single and specific task.	Om Mane (20BCE2632)
10	Fuzzy rough set feature selection to enhance phishing attack detection (Mahdieh Zabihimayvan , Derek Doran)	IEEE Explore 2019	FRS theory is a tool used to select most effective features from 3 benchmarked data sets. The selected features are fed into three often used classifiers for phishin detection. To evaluate the FRS feature selection in developing a generalizable phishing detection,the classifiers are trained using random 14,000 website samples.	Fuzzy rough set Algorithm	the universal features considered here contain no domain-based features. therefore,there is no inquiry from external sources.	Theo Daniel (20BCB0122)
11	Detection of phishing websites using machine learning approach (Junaid Rashid, Toqeer Mahmood, Muhammad Wasif Nisar, Tahira Nazir)	IEEE Explore 2020	This paper concentrates about different phishing attacks on URLs. to detect fake or illegal websites and notify the user in advance to prevent users from getting their Private info to be misused using Extreme Learning Machine	Blacklist And Whitelist Approach , Heuristic Based Approach,Hybrid Approach, Naive Bayes (NB)	Blacklist And Whitelist Approach : It cannot distinguish the recently created phishing websites from legal websites. Heuristic Based Approach : difficult to implement and time intensive.	Theo Daniel (20BCB0122)

12	Phishing Detection using Random Forest, SVM and Neural Network with Backpropagation (Smita Sindhu , Arya Sreevalsan , Sunil Parameshwar Patil , Faiz Rahman)	IEEE Explore 2020	The paper explains Random Forest classification method, SVM classification algorithm and Neural Network using the libray "sklearn" . SVM is chosen as final classifier algorithm use to its better accuracy comparitively.	Machine Learning with Wrappers Features Random Forest Neural network with backpropagation	Machine Learning with Wrappers Features: Selection time-consuming and involves extra computational overhead.	Theo Daniel (20BCB0122)
13	Detection and prevention of phishing websites using machine learning, (vaibhav patel)	International Research Journal of Engineering and Technology (IRJET)-2018	These authors in thier previous work the blacklist and whitelist are implemented which has a drawback of not being updated in longtime. This idea solves this problem by using all three approaches i.e blacklist and whitelist,heuristics and visual similarity	random forest	it had a slight deviation from expected output. it detects some minimal false positive and false negative results	Karthik Ajith (20BDS0261)

14	A new method for detection of phishing websites: URL Detection (shraddha prakash,dhwanil parkh,srusthi kotak,smita sanke)	IEEE Explore 2018	it is improve phishing detection efficiency for detecting phishing websites. a lot of research was conducted on various algorithm for finding the best model. here idea is to improve the efficiency by using random forest as our classificational algorithm with the help of Rstudio tool that helps us in better analysis	Random forest	time consumed and complexity involved,overheads involved and the performace was effected	Karthik Ajith (20BDS0261)
15	Detecting Phishing website using Machine Learning (R kiruthiga, D.akila)	IEEE Explore 2019	an approach to detect phishing email attack using natural language processing and machine learning it is to perform seasmic analysis of the text malicious talent. A natural language process techniqe is ued to parse each sentence and finds seasmic jobs of words in the sentence is an inquiry or an order	Random forest,gradient boosting,generalized linear model, generalized additive model	In Random Forest, large number of trees can make the algorithm too slow and not effective for real-time predictions. SVM algorithm achieved best performance over LR algorithm	Karthik Ajith (20BDS0261)

5.Observation and gaps identified in Literature survey:

Traditional method to detect phishing websites is by identifying and updating blacklisted URL's, Internet Protocol (IP) to the antivirus which is commonly known as "Blacklist" method. Attackers to evade getting blacklisted use creative techniques to fool users by modifying URL so that it appears legitimate via obfuscation and many other simple techniques like fast-flux, in this technique the proxies are automatically generated to host the webpage, other technique is to generate URL's algorithmically and etc.

[1] Fuzzy rough set feature selection to enhance phishing attack detection (Mahdieh Zabihimayvan Derek Doran) - It is evident from the paper that on Comparisons with other feature selection methods utilized by the related work indicate the outperformance of FRS in selecting efficient features. The maximum F-measure gained by FRS feature selection is 95% using Random Forest classification. The F-measure value using this universal feature set is approximately 93%.

[2] Detection of phishing websites using machine learning approach (Junaid Rashid, Toqeer Mahmood, Muhammad Wasif Nisar, Tahira Nazir)-The accuracies achieved using Random Forest, SVM and Neural Network with backpropagation classification algorithms are 97.369%,97.451% and 97.259% respectively. SVM is found to be the best classifier among the three as it gives better frequency. Even though Random Forest classifier gives greater accuracy rates than SVM classifier most of the time, the accuracy rates are not constant hence proving SVM to better than random forest classifier.

[3] Phishing Detection using Random Forest, SVM and Neural Network with Backpropagation (Smita Sindhu, Arya Sreevalsan, Sunil Parameshwar Patil, Faiz Rahman) - In this study ELM is performed derived from different 30 main components which are categorized using the ML approach. Most of the phishing URLs use HTTPS to avoid getting detected. There are three ways for the detection of website phishing. The primitive approach evaluates differentitems of URL, the second approach analyzing the authority of a website and calculating whether the website is introduced or not and it also analyzing who is supervising it, the third approach checking the genuineness of the website.

[4] Detecting Phishing Websites Using Machine Learning (Sagar Patil, Yogesh Shetye, Nilesh Shendage) -It can be seen that SVM outperforms all the other algorithms based on accuracy in detection of Phishing URL.It can also be seen that SVM has the highest sensitivity among all the other classifiers. It is evident that SVM has the least False positive rate among the three.Hence, SVM works best in classifying thephishing URL from the legitimate URLs.

[5] Detection of phishing email based on NLP and ML (Shaikh sayema anwer, Asst.Prof. V. Karwande)- NB model withregularization parameter 1 and a linear kernel was reported.The Naive Bayes model with a smoothing parameter hasbeen published

[6] A Review on Phishing Attacks (Akarshita Shankar, Ramesh Shetty, Badari Nath K)- Periodical updating of anti-phishing tools and platforms canprove to be very powerful. This study

provides an in-sight to phishing, the mechanism of the attack, various forms it can occur in and the possible solutions to overcome them.

[7] WC-PAD : Web Crawling based Phishing Attack Detection. Author : Nathezhtha T , Sangeetha D, Vaidehi V-WC-PAD performs a three phase identification mechanism. the DNS blacklist - provies a list of IP address which invloves spam activity The web crawler - starts crawling to the websites interconnected pages and links heuristic analysis - analyses the web content ,URL ,Web traffic based features,also analyses copyright contents.WC-PAD checks for the IP in DNS blaclist,If WC-PAD found any spam activity immedietly it send alert messages.

[8] In Random Forest, large number of trees can make the algorithm too slow and not effective for real-time predictions. SVM algorithm achieved best performance over LR algorithm. Machine Learning with Wrappers Features: Selection time-consuming and involves extra computational overhead.

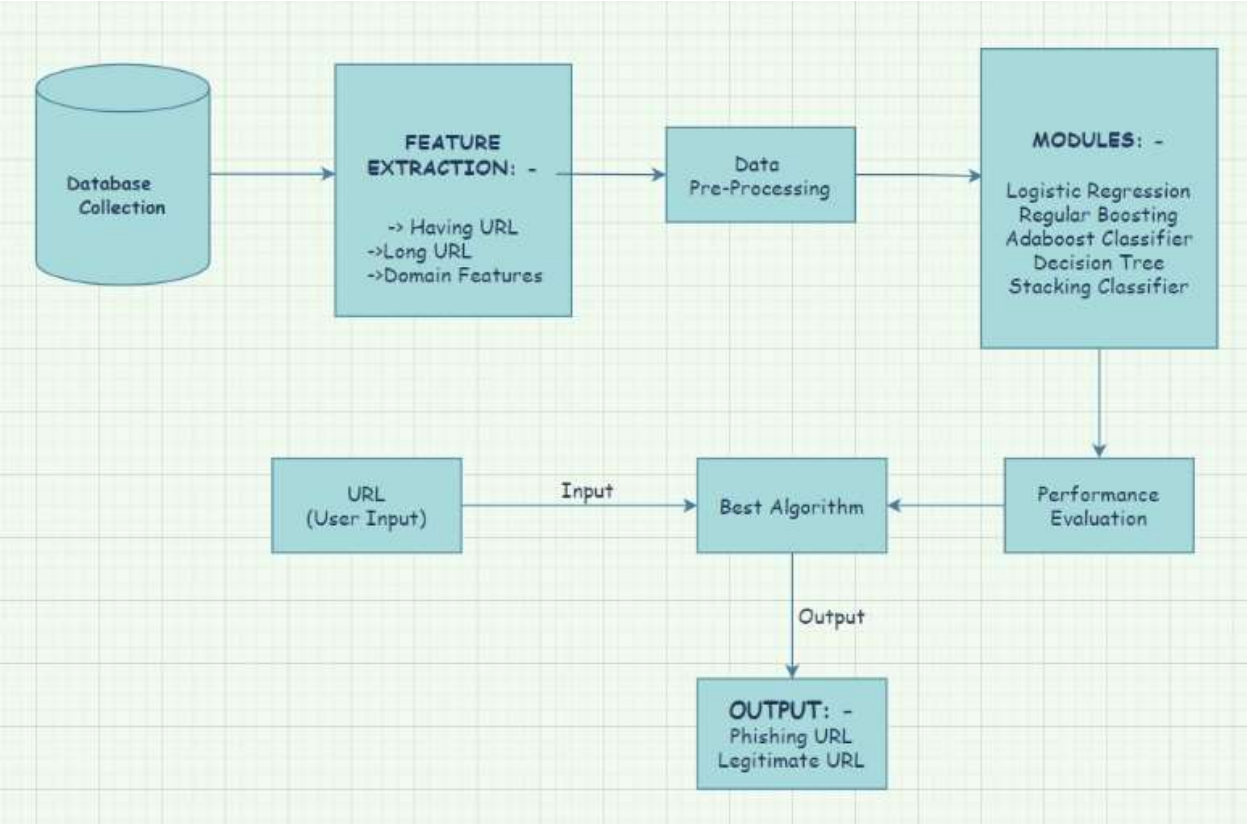
Blacklist And Whitelist Approach: It cannot distinguish the recently created phishing websites from legal websites. Heuristic Based Approach: difficult to implement and time intensive. Natural Language Processing (NLP) - NLP is unable to adapt to the new domain. NLP is built for a single and specific task.

Naive Bayes assumes that all predictors (or features) are independent, rarely happening in real life. This limits the applicability of this algorithm in real-world use cases.

Alexa's statistics are not necessarily accurate in the absolute terms. Alexa Rank is not something valuable in reference to overall SEO optimization and performance

6.Proposed model:

Architecture:



Features Used: -

To develop our project, we have used the 'Phishing Websites Dataset' which was in the UC Irvine Machine Learning Store. The csv file consists of of 11,055 URLs (occasions) (6157 phishing examples + 4898 authentic cases). Each occasion contains 31 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 which will be in the csv file to differentiate the feature output with corresponding machine learning algorithm If the condition is satisfied then '1', If the condition is partially satisfied then '0', If the condition is not satisfied then '-1'.

The Features with the conditions are as follows: -

- IP Address: -
  - If the domain part has IP address then Phishing
  - Otherwise it is considered to be a legitimate one.

- Long URL: -
  - If the URL length < 54 characters then it is a legitimate one.
  - If the URL length >= 54 characters and <=75 then it is a suspicious one
  - If the URL length >75 characters then it is a Phishing one.

- Making Short URL using TinyURL: -
  - If it is a tiny URL then it is a Phishing one
  - Otherwise, it is Legitimate one.

- URL has @ symbol: -
  - If it has URL then it is Phishing one.
  - Else it is a legitimate one.

- Using '//' to redirect: -
  - If the last occurrence of '//' in the URL > 7 then it is Phishing one.
  - Otherwise, it is Legitimate one.

- Attaching a Afiix or Posting to the Domain distinct by (-) : -
  - If the URL contains (-) then it is a phishing one.
  - Otherwise, it is a Legitimate one.

- Sub Domains: -
  - If there are 1 dot part then it is Legitimate one.
  - If there are 2 dot part then it is Suspicious one.
  - If there are 1 dot part then it is Phishing one.

- HTTPS: -
  - If the issuer and the HTTPS is truested and Age of Certificate >= 1 Year then it is legitimate one.
  - If the issuer and the HTTPS is not trusted then it is suspicious one.
  - Otherwise, It is a phishing one.

- Length of Domain enrolling: -
  - If the domain is expiring <=1 Year then it is a phishing one.
  - Otherwise it is a suspicious one.

- Favicon: -
  - If the URL is loaded from external domain then it is phishing one.

- Otherwise, it a legitimate one

- The presence of the HTTPS token: -
  - If there is HTTP token then it is Phishing one.
  - Otherwise, it is a legitimate one.

- A Request URL: -
  - If the percentage of the request URL < 22% then it is legitimate
  - If the percentage is >=22% and < 61% then it is suspicious
  - Otherwise, it is a phishing one.

- Anchor of URL: -
  - If the percentage of the anchor < 31% then it is legitimate.
  - If the percentage of anchor >= 31% and <= 67% then it is suspicious.
  - Otherwise, it is a phishing one.

- Links in <Script>, <Meta>, <Links> tags: -
  - If the percentage of the links is < 17% then is legitimate
  - Otherwise, it is a Phishing one.

- Number of links 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.

- Server from Handlers: -
  - If the server from the handlers is “about: blank” or empty then is Phishing one.
  - If the server from the handlers to a different domain then it is suspicious one.
  - Otherwise, it is Legitimate one.

- Using Email to Submit Information: -
  - If the “mail()” or “mailto: ” is there then it is Phishing one.
  - Otherwise, it is Legitimate one.

- Curious URL: -
  - If the host name is not included then it is Phishing one.
  - Otherwise, it is legitimate one.

- Features 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
- Otherwise, it is legitimate one.

- Assist of websites: -
  - If the percentage of redirecting <= 1 then it is legitimate.
  - If the percentage of redirecting >= 2 and < 4 then it is suspicious
  - Otherwise, it is Phishing one.

- Customization of the Grade Bar: -
  - If the on mouseover changed then it is Phishing.
  - If it doesn't change the status bar then it is legitimate.

- Right Click: -
  - If the right click is disabled then it is Phishing one.
  - Otherwise, it is a legitimate one.

- Popup Window: -
  - If the popup contains the text fields then it is phishing.
  - Otherwise, it is legitimate.
- Redirection of Iframe: -
  - If redirection of Iframe is there then in it is Phishing one.
  - Otherwise, it is legitimate.

- Life of Domain: -
  - If the age of the domain >= 6 months then it is legitimate one.
  - Otherwise, it is phishing one.

- DNS Log: -
  - If there are no Log for the domain then it is Phishing one.
  - Otherwise, it is legitimate one.

- 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
  - Otherwise, it is legitimate one.

- Page Grade: -
  - If the page rank < 0.2 then it is Phishing
  - Otherwise it is Legitimate.

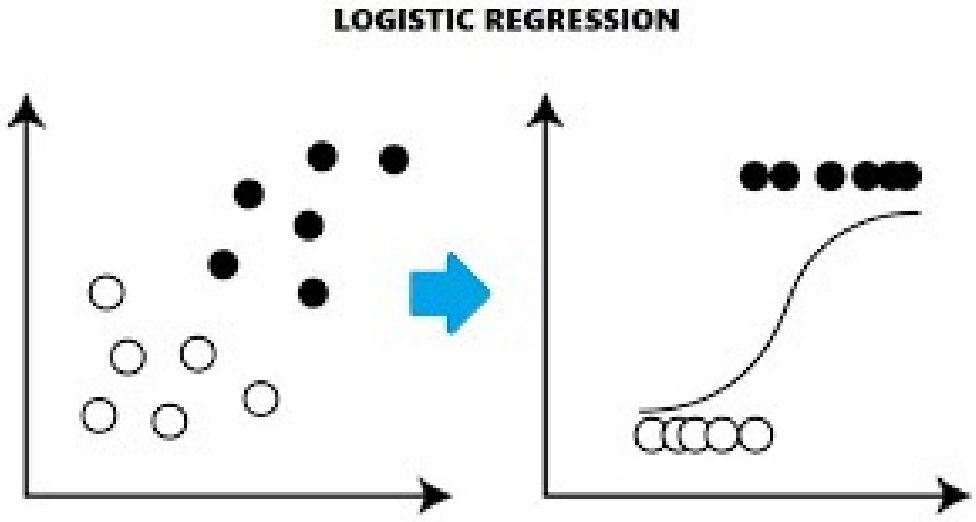
- Usage of Non-standard ports: -
  - Entire port are exposed then it is Phishing.
  - Otherwise, it is legitimate one.

- Traffic of the Website: -
  - If the rank < 100000 then it is legitimate.
  - If the rank > 100000 then it is suspicious
  - Otherwise, it is Phishing.

Modules (List of Algorithms): -

- Logistic Regression
- Regular Boosting
- Decision Tree
- AdaBoost Classifier
- Stacking Classifier

Logistic Regression:

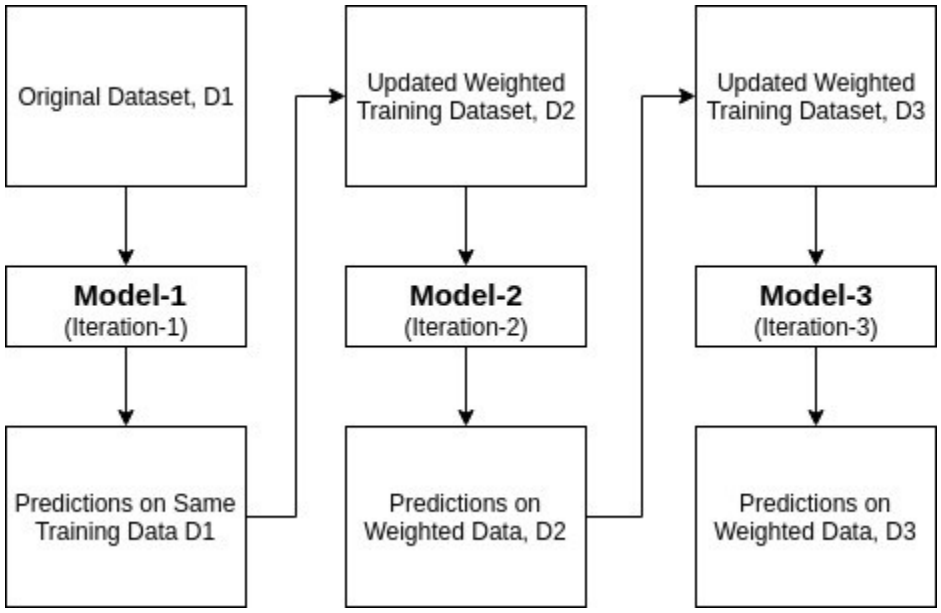


Logistic regression is a fundamental classification technique. It belongs to the group of linear classifiers and is somewhat similar to polynomial and linear regression. Logistic regression is fast and relatively uncomplicated, and it's convenient for you to interpret the results. Although it's essentially a method for binary classification, it can also be applied to multiclass problems.

The concept of the logistic regression is in the statistical software to understand the relationship between the dependent and the one or more independent variables. Here, this concept is used to estimate the probability of the legitimate, suspicious and phishing websites using the data sets provided for the training as this concept yields the binary output as true or false.

For example, since it is a statistical model it is used in the prediction of the lifetime of the patients due to the data set of diseases like diabetes, cancer, cardiac diseases.

AdaBoost Classifier: -

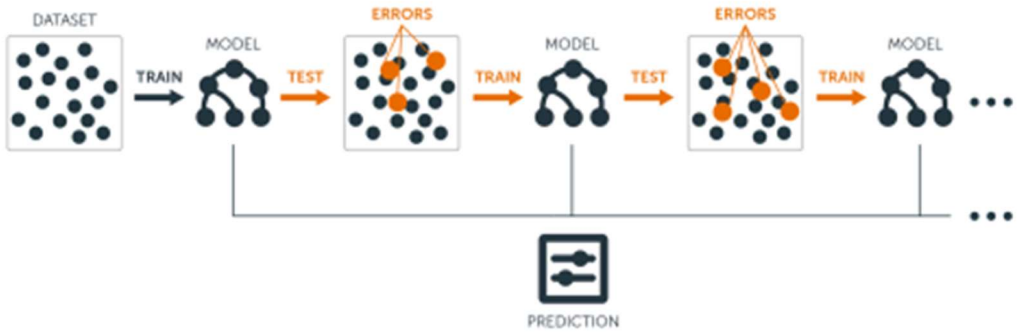


AdaBoost is best used to boost the performance of decision trees on binary classification problems. AdaBoost can be used to boost the performance of any machine learning algorithm. It is best used with weak learners.

Boosting algorithms combine multiple low accuracy (or weak) models to create a high accuracy(or strong) models. It can be utilized in various domains such as credit, insurance, marketing, and sales. AdaBoost is an example of one such algorithm. It combines multiple classifiers to increase the accuracy of classifiers. AdaBoost is an iterative ensemble method. AdaBoost classifier builds a strong classifier by combining multiple poorly performing classifiers so that you will get high accuracy strong classifier.

The basic concept behind Adaboost is to set the weights of classifiers and training the data sample in each iteration such that it ensures the accurate predictions of unusual observations. The condition of the adaboost is that the classifier should be trained interactively on various weighed training examples and in each iteration, it tries to provide an excellent fit for these examples by minimizing training error.

Regular Boosting:



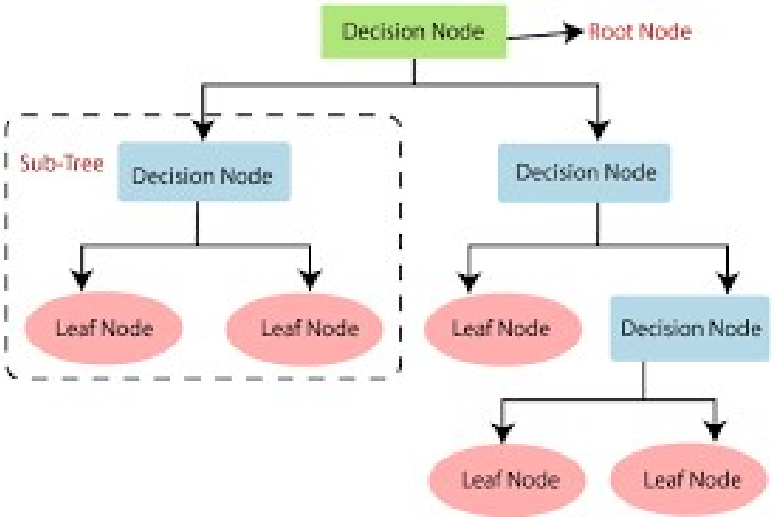
Regular boosting classifiers or gradient boosting are a gathering of ML calculations that join numerous feeble learning models together to make a solid prescient model. Decision trees are normally utilized while doing inclination boosting. Regular boosting models are turning out to be well known due to their adequacy at arranging complex datasets, and have as of late been utilized to win numerous Kaggle information science rivalries.

In simple words, we can define as the gradient boosting method itself defines it as we can combine many weak models together and we can create a strong predictive model. We use predictive model here as we are using the detection website to predict the correctness.

The model ultimately corrects the errors in the previous dataset test and correct it when it loads next time. The many weak models make up into a strong prediction model that may increase the accuracy level

For example, the prediction of sales prices using regression. (Residue)

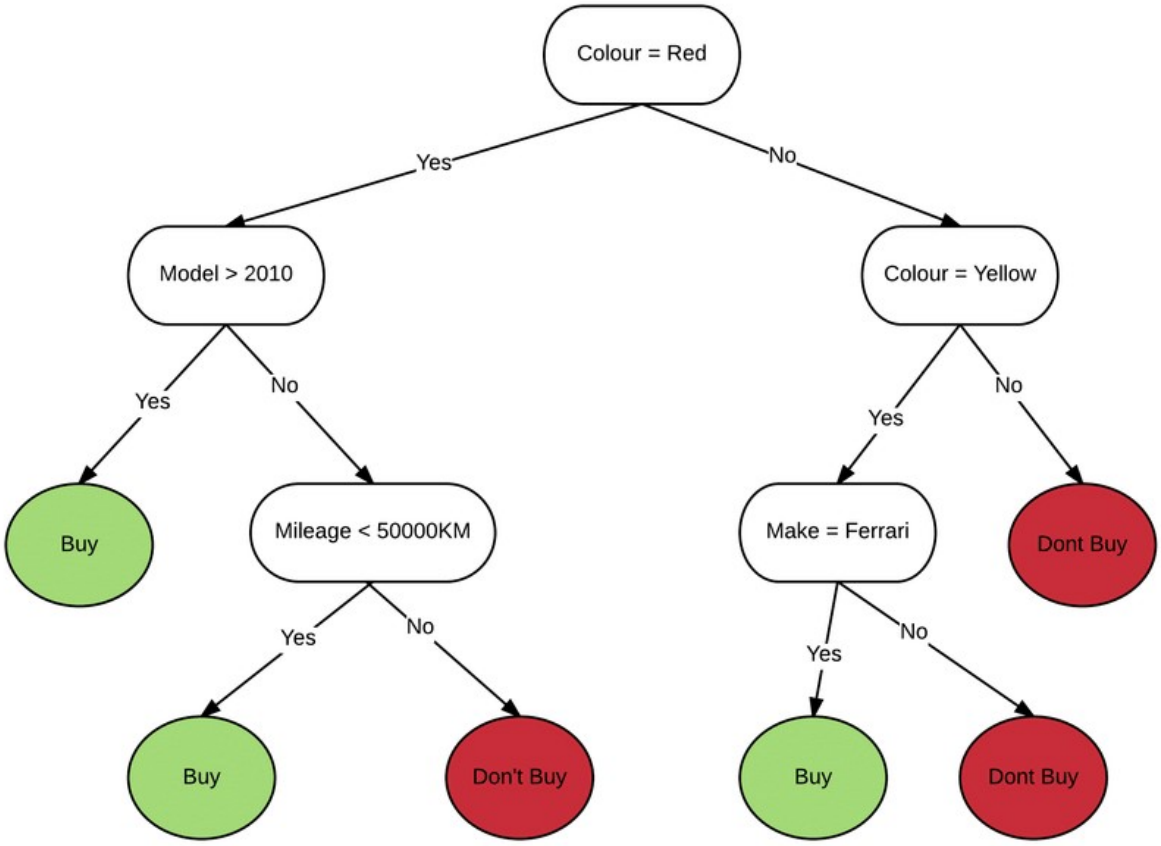
Decision Tree:



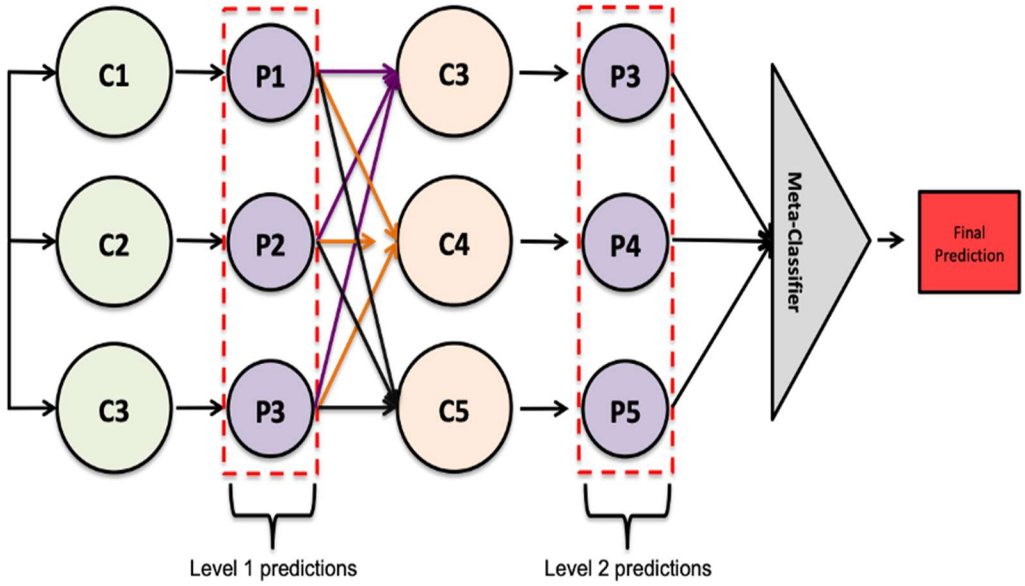
Decision tree analysis is a supervised learning model as the model uses a tree-shaped diagram to chart out a course of action or a statistical probability analysis. This method is mainly follow divide and conquer approach with the statistical model of training data and it is used to break down a large complex problem into smaller problems or branches using the given parameter in the training dataset where each branch of the decision tree could be a possible outcome.

The main goal to use this model in the project is to create a training model and that model can be used to predict the result using a simple decision rules given in the problem. It provides all the possible solution to a decision with the conditions as reference.

For example.



Stacking Classifier:



Stacking is a way to ensemble multiple classifications or regression model. There are many ways to ensemble models, the widely known models are Bagging or Boosting.

Bagging allows multiple similar models with high variance are averaged to decrease variance. Boosting builds multiple incremental models to decrease the bias, while keeping variance small. The idea is that you can attack a learning problem with different types of models which are capable to learn some part of the problem, but not the whole space of the problem

Steps in stack classifier: -

1. Getting the Stacking ensemble of models from 4 algorithmic module
2. Defining the meta learner model at level 1 of logistic regression module
3. Defining the stacking ensemble with stacking classifier
4. Getting the list of models (4 modules) to evaluate
5. Evaluating the given model using cross – validation
6. Evaluating the models and storing the results

.

Implementation Tools: -

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.

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 and 20% is the test set.

Steps to implement algorithm: -

1. Import the pandas, matplolib and numpy libraries.
2. Read the csv file which has the 11055 training set values using its location in the device.
3. Data pre-processing has to be done where we have to remove the unnecessary columns and we have to check if there are any null values.
4. Start implementing the Logistic Regression, Regular Boosting Classifier, Decision Tree, Ada Boost Classifier using the functions from the sklearn library.
5. After implementing all the algorithms, use the stacking classifier compute a meta-learning computation combine the outcomes of the 4 algorithms where the meta-model is built on the models produced by the test data given as input.

Importing the libraries and Collection of data \: -

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data = pd.read_csv("/CyberSecurity_Data.csv")
data.head()
```

	id	having_IP_Address	URL_Length	Shortining_Service	having_At_Symbol	double_slas
0	1	-1	1	1	1	
1	2	1	1	1	1	
2	3	1	0	1	1	
3	4	1	0	1	1	
4	5	1	0	-1	1	

Data Pre-Processing: -

```
[ ] data.drop(["id"],axis=1,inplace=True)
    data.columns
```

```
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')
```

```
[ ] data.shape
```

```
(11055, 31)
```

```
[ ] data.isnull().values.any()
```

```
False
```

Finding the Training and the Test Accuracies of each module: -

Logistic Regression: -

1. Having IP address: -

+ Code + Text

```
#1
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('having_IP_Address',axis=1)
y=data['having_IP_Address']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
lr=LogisticRegression(random_state=0)
lr.fit(x_train,y_train)
y_predict=lr.predict(x_test)
print("train set accuracy: ",100*lr.score(x_train,y_train))
print("Test set accuracy : ",100*lr.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
train set accuracy:  84.24920850293984
Test set accuracy :  84.84848484848484
              precision    recall  f1-score   support

          -1       0.82      0.72      0.76       762
           1       0.86      0.92      0.89      1449

    accuracy                           0.85      2211
   macro avg       0.84      0.82      0.83      2211
weighted avg       0.85      0.85      0.85      2211
```

2. URL Length: -

+ Code + Text

```
#2
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('URL_Length',axis=1)
y=data['URL_Length']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
lr=LogisticRegression(random_state=0)
lr.fit(x_train,y_train)
y_predict=lr.predict(x_test)
print("train set accuracy: ",100*lr.score(x_train,y_train))
print("Test set accuracy : ",100*lr.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
train set accuracy:  87.46042514699232
Test set accuracy :  86.47670737222975
              precision    recall  f1-score   support

          -1       0.90      0.95      0.92      1800
           0       0.19      0.10      0.13        30
           1       0.70      0.54      0.61       381

    accuracy                           0.86      2211
   macro avg       0.60      0.53      0.55      2211
weighted avg       0.85      0.86      0.86      2211
```

3. Shortening Services: -

```
#3
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('Shortining_Service',axis=1)
y=data['Shortining_Service']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
lr=LogisticRegression(random_state=0)
lr.fit(x_train,y_train)
y_predict=lr.predict(x_test)
print("train set accuracy: ",100*lr.score(x_train,y_train))
print("Test set accuracy : ",100*lr.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
train set accuracy:  97.87426503844414
Test set accuracy :  97.64812302125735
              precision    recall  f1-score   support

          -1       0.96      0.86      0.90       287
           1       0.98      0.99      0.99      1924

    accuracy                           0.98      2211
   macro avg       0.97      0.93      0.95      2211
weighted avg       0.98      0.98      0.98      2211
```

4.Having At Symbol: -

```
#4
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('having_At_Symbol',axis=1)
y=data['having_At_Symbol']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
lr=LogisticRegression(random_state=0)
lr.fit(x_train,y_train)
y_predict=lr.predict(x_test)
print("train set accuracy: ",100*lr.score(x_train,y_train))
print("Test set accuracy : ",100*lr.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
train set accuracy:  90.85255540479422
Test set accuracy :  90.41157847127997
              precision    recall  f1-score   support

          -1       0.81      0.49      0.61       342
           1       0.91      0.98      0.95      1869

    accuracy                           0.90      2211
   macro avg       0.86      0.74      0.78      2211
weighted avg       0.90      0.90      0.89      2211
```

5.Double slash redirecting: -

+ Code + Text

```
#5
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('double_slash_redirecting',axis=1)
y=data['double_slash_redirecting']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
lr=LogisticRegression(random_state=0)
lr.fit(x_train,y_train)
y_predict=lr.predict(x_test)
print("train set accuracy: ",100*lr.score(x_train,y_train))
print("Test set accuracy : ",100*lr.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
train set accuracy:  97.98733604703754
Test set accuracy :  97.55766621438264
              precision    recall  f1-score   support

          -1       0.93      0.87      0.90       278
           1       0.98      0.99      0.99      1933

    accuracy                           0.98      2211
   macro avg       0.95      0.93      0.94      2211
weighted avg       0.98      0.98      0.98      2211
```

ADABoost Classifier: -

1. Having IP address: -

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('having_IP_Address',axis=1)
y=data['having_IP_Address']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
adc=AdaBoostClassifier(random_state=0)
adc.fit(x_train,y_train)
y_predict=adc.predict(x_test)
print("train set accuracy: ",100*adc.score(x_train,y_train))
print("Test set accuracy : ",100*adc.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

DataFrame: data
View

```
train set accuracy:  84.96155585707824
Test set accuracy :  84.71279963817278
              precision    recall  f1-score   support

          -1       0.83      0.70      0.76       762
           1       0.85      0.93      0.89      1449

    accuracy                           0.85      2211
   macro avg       0.84      0.81      0.82      2211
weighted avg       0.85      0.85      0.84      2211
```

2. URL Length: -

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('URL_Length',axis=1)
y=data['URL_Length']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
adc=AdaBoostClassifier(random_state=0)
adc.fit(x_train,y_train)
y_predict=adc.predict(x_test)
print("train set accuracy: ",100*adc.score(x_train,y_train))
print("Test set accuracy : ",100*adc.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
train set accuracy:  85.2894617819991
Test set accuracy :  84.62234283129806
              precision    recall  f1-score   support

          -1       0.87      0.95      0.91      1800
           0       0.24      0.20      0.22        30
           1       0.68      0.40      0.50       381

    accuracy                           0.85      2211
   macro avg       0.60      0.52      0.54      2211
weighted avg       0.83      0.85      0.83      2211
```

3. Shortening Services: -

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('Shortining_Service',axis=1)
y=data['Shortining_Service']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
adc=AdaBoostClassifier(random_state=0)
adc.fit(x_train,y_train)
y_predict=adc.predict(x_test)
print("train set accuracy: ",100*adc.score(x_train,y_train))
print("Test set accuracy : ",100*adc.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

DataFrame: data

View

```
train set accuracy:  97.42198100407056
Test set accuracy :  97.60289461781998
              precision    recall  f1-score   support

          -1       0.95      0.86      0.90       287
           1       0.98      0.99      0.99      1924

    accuracy                           0.98      2211
   macro avg       0.96      0.93      0.94      2211
weighted avg       0.98      0.98      0.98      2211
```

4. Having At Symbol: -

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('having_At_Symbol',axis=1)
y=data['having_At_Symbol']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
adc=AdaBoostClassifier(random_state=0)
adc.fit(x_train,y_train)
y_predict=adc.predict(x_test)
print("train set accuracy: ",100*adc.score(x_train,y_train))
print("Test set accuracy : ",100*adc.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
train set accuracy:  90.06105834464043
Test set accuracy :  89.68792401628222
              precision    recall  f1-score   support

          -1       0.77      0.47      0.59       342
           1       0.91      0.97      0.94      1869

    accuracy                           0.90      2211
   macro avg       0.84      0.72      0.76      2211
weighted avg       0.89      0.90      0.89      2211
```

5. Double slash redirecting: -

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('double_slash_redirecting',axis=1)
y=data['double_slash_redirecting']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
adc=AdaBoostClassifier(random_state=0)
adc.fit(x_train,y_train)
y_predict=adc.predict(x_test)
print("train set accuracy: ",100*adc.score(x_train,y_train))
print("Test set accuracy : ",100*adc.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
train set accuracy:  97.85165083672547
Test set accuracy :  97.3767526006332
              precision    recall  f1-score   support

          -1       0.91      0.88      0.89       278
           1       0.98      0.99      0.99      1933

    accuracy                           0.97      2211
   macro avg       0.95      0.93      0.94      2211
weighted avg       0.97      0.97      0.97      2211
```

Regular Boosting Classifier: -

1. Having IP address: -

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('having_IP_Address',axis=1)
y=data['having_IP_Address']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  87.44911804613297
Test set accuracy :  87.01944821347807
              precision    recall  f1-score   support

          -1       0.83      0.78      0.81       762
           1       0.89      0.92      0.90      1449

    accuracy                           0.87      2211
   macro avg       0.86      0.85      0.85      2211
weighted avg       0.87      0.87      0.87      2211
```

2. URL Length: -

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('URL_Length',axis=1)
y=data['URL_Length']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  93.31750339213026
Test set accuracy :  92.8991406603347
              precision    recall  f1-score   support

          -1       0.93      0.98      0.96      1800
           0       1.00      0.50      0.67        30
           1       0.89      0.71      0.79       381

    accuracy                           0.93      2211
   macro avg       0.94      0.73      0.81      2211
weighted avg       0.93      0.93      0.93      2211
```

3. Shortening Services: -

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('Shortining_Service',axis=1)
y=data['Shortining_Service']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  98.60922659430122
Test set accuracy :  98.23609226594301
              precision    recall  f1-score   support

          -1       0.97      0.89      0.93       287
           1       0.98      1.00      0.99      1924

    accuracy                           0.98      2211
   macro avg       0.98      0.94      0.96      2211
weighted avg       0.98      0.98      0.98      2211
```

4. Having At Symbol: -

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('having_At_Symbol',axis=1)
y=data['having_At_Symbol']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  92.34509271822705
Test set accuracy :  92.08502939846224
              precision    recall  f1-score   support

          -1       0.87      0.57      0.69       342
           1       0.93      0.99      0.95      1869

    accuracy                           0.92      2211
   macro avg       0.90      0.78      0.82      2211
weighted avg       0.92      0.92      0.91      2211
```

5. Double slash redirecting: -

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('double_slash_redirecting',axis=1)
y=data['double_slash_redirecting']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  98.92582541836273
Test set accuracy :  98.37177747625509
              precision    recall  f1-score   support

          -1       0.97      0.90      0.93       278
           1       0.99      1.00      0.99      1933

    accuracy                           0.98      2211
   macro avg       0.98      0.95      0.96      2211
weighted avg       0.98      0.98      0.98      2211
```

Decision Tree: -

1. Having IP address: -

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('having_IP_Address',axis=1)
y=data['having_IP_Address']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
dt=DecisionTreeClassifier(random_state=0)
dt.fit(x_train,y_train)
y_predict=dt.predict(x_test)
print("train set accuracy: ",100*dt.score(x_train,y_train))
print("Test set accuracy : ",100*dt.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
train set accuracy:  96.69832654907282
Test set accuracy :  86.2053369516056
              precision    recall  f1-score   support

          -1       0.79      0.81      0.80       762
           1       0.90      0.89      0.89      1449

    accuracy                           0.86      2211
   macro avg       0.85      0.85      0.85      2211
weighted avg       0.86      0.86      0.86      2211
```

2. URL Length: -

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('URL_Length',axis=1)
y=data['URL_Length']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
dt=DecisionTreeClassifier(random_state=0)
dt.fit(x_train,y_train)
y_predict=dt.predict(x_test)
print("train set accuracy: ",100*dt.score(x_train,y_train))
print("Test set accuracy : ",100*dt.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
train set accuracy:  99.19719583898689
Test set accuracy :  95.2962460425147
              precision    recall  f1-score   support

          -1       0.97      0.97      0.97      1800
           0       0.67      0.67      0.67        30
           1       0.88      0.89      0.88       381

    accuracy                           0.95      2211
   macro avg       0.84      0.84      0.84      2211
weighted avg       0.95      0.95      0.95      2211
```

3. Shortening Services: -

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('Shortining_Service',axis=1)
y=data['Shortining_Service']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
dt=DecisionTreeClassifier(random_state=0)
dt.fit(x_train,y_train)
y_predict=dt.predict(x_test)
print("train set accuracy: ",100*dt.score(x_train,y_train))
print("Test set accuracy : ",100*dt.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
train set accuracy:  99.81908638625056
Test set accuracy :  98.95974672094076
              precision    recall  f1-score   support

          -1       0.95      0.98      0.96       287
           1       1.00      0.99      0.99      1924

    accuracy                           0.99      2211
   macro avg       0.97      0.98      0.98      2211
weighted avg       0.99      0.99      0.99      2211
```

4. Having At Symbol: -

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('having_At_Symbol',axis=1)
y=data['having_At_Symbol']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
dt=DecisionTreeClassifier(random_state=0)
dt.fit(x_train,y_train)
y_predict=dt.predict(x_test)
print("train set accuracy: ",100*dt.score(x_train,y_train))
print("Test set accuracy : ",100*dt.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
train set accuracy:  98.28132066938036
Test set accuracy :  92.9895974672094
              precision    recall  f1-score   support

          -1       0.76      0.79      0.78       342
           1       0.96      0.96      0.96      1869

    accuracy                           0.93      2211
   macro avg       0.86      0.87      0.87      2211
weighted avg       0.93      0.93      0.93      2211
```

5. Double slash redirecting: -

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('double_slash_redirecting',axis=1)
y=data['double_slash_redirecting']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
dt=DecisionTreeClassifier(random_state=0)
dt.fit(x_train,y_train)
y_predict=dt.predict(x_test)
print("train set accuracy: ",100*dt.score(x_train,y_train))
print("Test set accuracy : ",100*dt.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
train set accuracy:  99.90954319312529
Test set accuracy :  98.73360470375395
              precision    recall  f1-score   support

          -1       0.94      0.96      0.95       278
           1       0.99      0.99      0.99      1933

    accuracy                           0.99      2211
   macro avg       0.97      0.98      0.97      2211
weighted avg       0.99      0.99      0.99      2211
```

7.Result and discussion:

The Training accuracy Values for all the 31 features: -

Module	Name	having_IP_Address	URL_Length	Shortining_Service	having_At_Symbol
Logistic Regression	Akshaya P	84.2492085	87.46042515	97.87426504	90.8525554
Regular Boosting	Deva Dharshan D	87.44911805	93.31750339	98.60922659	92.34509272
Decision Tree	Om Mane	96.69832655	99.19719584	99.81908639	98.28132067
AdaBoost Classifier	Theo Daniel	84.96155586	85.28946178	97.421981	90.06105834

double_slash_redirecting	Prefix_ Suffix	having_Sub_ Domain	SSLfinal_State	Domain_registeration_length
97.98733605	87.20036183	53.38082316	81.07191316	81.00407056
98.92582542	89.15649028	64.54093171	84.96155586	83.87607417
99.90954319	97.78380823	91.06739032	96.56264134	96.23473541
97.85165084	87.05336952	55.46132972	79.69244686	80.68747173

Favicon	port	HTTPS_token	Request_URL	URL_of_Anchor
98.81275441	98.79014021	94.54997739	81.5468114	69.7987336
99.36680235	99.36680235	96.55133424	82.41745816	74.90954319
100	99.9660787	99.62686567	95.61284487	94.47082768
98.30393487	98.7336047	94.47082768	81.70511081	66.79104478

Links_in_tags	SFH	Submitting_to_email	Abnormal_URL	Redirect
50.93848937	80.04296698	95.72591588	96.24604251	93.61148801
60.36861149	84.36227951	96.82270466	98.24739937	94.92311171
89.5296246	96.24604251	99.87562189	99.79647218	98.85798281
47.86295794	78.33559475	95.16056083	95.88421529	93.13658978

on_mouseover	RightClick	popUpWidnow	Iframe	age_of_domain	DNSRecord
95.60153777	96.80009046	98.6883763	96.96969697	65.7620986	85.68521031
97.72727273	98.23609227	98.97105382	98.11171416	77.71370421	8.794663048
99.86431479	99.89823609	99.9773858	99.9886929	95.07010403	95.20578924
95.16056083	96.40434193	98.37177748	96.42695613	69.26729986	85.87743103

web_traffic	Page_Rank	Google_Index	Links_pointing_to_page	Statistical_report	Result
60.10854817	75.94979647	87.14382632	76.24378109	90.94301221	93.19312528
68.09136137	79.01402081	88.37630032	81.614654	92.28855721	95.33016735
92.75214835	95.55630936	95.43193125	97.3880597	98.29262777	99.02758933
59.55450023	75.62189055	86.94029851	75.0226142	89.80099502	93.81501583

Average of all 31 Features: -

Module	Name	Average Training set Accuracy
Logistic Regression	Akshaya P	92.29785092
Regular Boosting	Deva Dharshan D	95.96088473
Decision Tree	Om Mane	**97.35450314**
AdaBoost Classifier	Theo Daniel	93.87511125

Creating the model and fitting the data into the model

Prepare models:

```
models=[lr,rb,dt,adc]
a=["Logistic Regression","Regular Boosting","Decision Tree","AdaBoost Classifire"]
names=["LR","RB","DT","ADC"]


test=[]
train=[]
for model in models:
    model.fit(x_train,y_train)
    train.append(model.score(x_train,y_train))
    test.append(model.score(x_test,y_test))


results = pd.DataFrame({ 'ML Model': a,
    'Train Accuracy': train,
    'Test Accuracy': test})


results
```

	ML Model	Train Accuracy	Test Accuracy
0	Logistic Regression	0.928313	0.928539
1	Regular Boosting	0.953528	0.950701
2	Decision Tree	0.989484	0.965626
3	AdaBoost Classifire	0.936680	0.938037

Sorting order by Train accuracy:

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

	ML Model	Train Accuracy	Test Accuracy
2	Decision Tree	0.989484	0.965626
1	Regular Boosting	0.953528	0.950701
3	AdaBoost Classifire	0.936680	0.938037
0	Logistic Regression	0.928313	0.928539

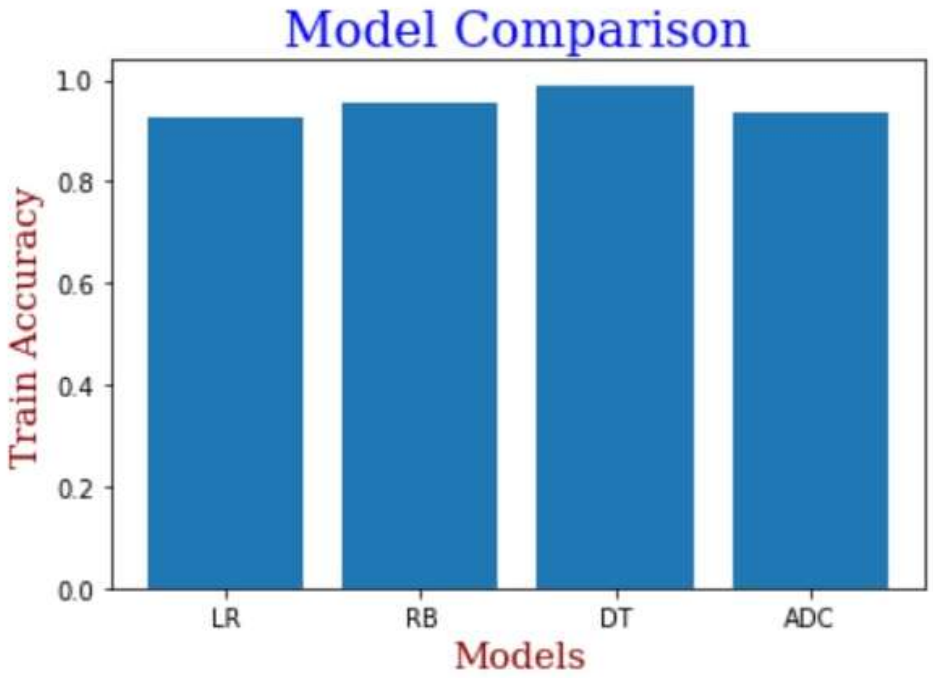
```
fig=plt.figure()
ax=fig.add_subplot(111)
font1 = {'family':'serif','color':'blue','size':20}
font2 = {'family':'serif','color':'darkred','size':15}
plt.xlabel('Models',fontdict = font2)
plt.ylabel('Train Accuracy',fontdict = font2)
plt.title('Model Comparison',fontdict = font1)
plt.bar(names,train)
plt.show()
```

Stacking classifier:

```
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


level0 = list()
level0.append(('lr', LogisticRegression()))

level0.append(('dt', DecisionTreeClassifier()))
level0.append(('adc', AdaBoostClassifier()))
level0.append(('rb', GradientBoostingClassifier()))
level1 = LogisticRegression()
model = StackingClassifier(estimators=level0, final_estimator=level1, cv=5)

models = dict()
models['Logistic'] = LogisticRegression()
models['Decision'] = DecisionTreeClassifier()
models['AdaBoost'] = AdaBoostClassifier()
models['Regular'] = GradientBoostingClassifier()
models['Stacking'] = model


def evaluate_model(model, x_train, y_train):
 cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
 scores = cross_val_score(model, x_train, y_train, scoring='accuracy', cv=cv, n_jobs=-1, e
 return scores


result, names = list(), list()
for name, model in models.items():
 scores = evaluate_model(model, x_train, y_train)
 result.append(scores)
 names.append(name)
 print('>%s %.3f (%.3f)' % (name, mean(scores), std(scores)))
```

```
>Logistic 0.927 (0.010)
>Decision 0.959 (0.007)
>AdaBoost 0.937 (0.008)
>Regular 0.949 (0.009)
>Stacking 0.964 (0.006)
```

Model comparison:



```
>Logistic 0.927 (0.010)
>Decision 0.959 (0.007)
>AdaBoost 0.937 (0.008)
>Regular 0.949 (0.009)
```

Demonstration

In [*]:

```
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
import emoji


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('The accuracy level is : ',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)
        print("The Given URL ",url," is :",end=" ")
        if prediction ==-1:
            print(emoji.emojize('Phishing Url :skull_and_crossbones:'))
        else:
            print(emoji.emojize('Legitimate Url :slightly_smiling_face:'))
    except:
            print("The Given URL ",url," is :",end=" ")
            print(emoji.emojize('Phishing Url :skull_and_crossbones:'))

url=input('Enter The URL : ')
Domain=['.com','.net','.in']
for i in Domain:
    if i in url:
        getResult(url)
        break
else:
    print(emoji.emojize('Enter Valid URL :pleading_face:'))
```

```
Enter The URL : www.google.com
The accuracy level is :  96.47218453188603
The Given URL  www.google.com  is : Legitimate Url 🙂
```

Example for Legitimate Url:

```
Enter The URL : www.facebook.com
The accuracy level is :  96.74355495251018
The Given URL  www.facebook.com  is : Legitimate Url 🙂
```

```
Enter The URL : https://stackoverflow.com/
The accuracy level is :  96.11035730438715
The Given URL  https://stackoverflow.com/  is : Legitimate Url 🙂
```

```
Enter The URL : www.onlinesbi.com
The accuracy level is :  96.15558570782451
The Given URL  www.onlinesbi.com  is : Legitimate Url 🙂
```

Example for Phishing Url:

```
Enter The URL : www.youtube.com
The accuracy level is :  96.24604251469923
The Given URL  www.youtube.com  is : Phishing Url ☠
```

As we all know we can earn through YouTube for earning we have to give our bank account details to the website, even though they won't do anything with our bank account its advised to not give bank details in other websites apart from bank website so this is the reason to consider it as phishing one. The another one is if we see a video we will be getting videos related to our previous video watched that we means it controls our feed page by providing recommendations on its own which makes a question mark our privacy. So, these are all the reasons why YouTube is showing as phishing one.

```
Enter The URL : https://www.abchina.com/en/
The accuracy level is :  97.24106739032112
Connection problem. Please check your internet connection!
The Given URL  https://www.abchina.com/en/  is : Phishing Url ☠
```

Example for invalid input:

8. 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.

Future work

Although the use of URL lexical features alone has been shown to result in high accuracy (97.35%), phishers have learned how to make predicting a URL destination difficult by carefully manipulating the URL to evade detection. Therefore, combining these features with others, such as host, is the most effective approach .For future enhancements, we intend to build the phishing detection system as a scalable web service which will incorporate online learning so that new phishing attack patterns can easily be learned and improve the accuracy of our models with better feature extraction

9.References:

[1] Roopak.S, Athira P Vijayaraghavan, Tony Thomas ' On Effectivness of source code and SSL based features for Phishing website Detection',*Indian Institute of Information technology and Management-Kerala* Thiruvananthapuram, India https://ieeexplore.ieee.org/document/9063824

[2] Nathezhtha.T, Sangeetha.D,Vaidehi.V ' WC-PAD: Web Crawling based Phishing Attack Detection, Madras Institue of Technology, Anna University, Chennai, Tamil Nadu, India'. https://ieeexplore.ieee.org/document/8888416

[3] Amani Alswailern , Bashayr Alabdullah, Norah Alrumayh, Dr.Aram Alsedrani ' Detecting Phishing website using Machine Learning', Al-Imam Muhammad Ibn Saud Islamic University Riyadh, Saudi Arabia, https://ieeexplore.ieee.org/document/9283771

[4] Patil Sagar, Shetye Yogesh, Shendage Nilesh, "Detecting Phishing Websites Using Machine Learning", International Research Journal of Engineering and Technology (IRJET),Volume: 07 Issue: 02 | Feb 2020 https://www.irjet.net/archives/V7/i3/IRJET-V7I3115.pdf

[5] Shankar Akarshita, Shetty Ramesh, K Badari Nath, "A Review on Phishing Attacks", International Journal of Applied Engineering Research ISSN 0973-4562 Volume 14, Number 9 (2019),https://www.ripublication.com/ijaer19/ijaerv14n9_15.pdf

[6] Shaikh Sayema Anwer , Asst.Prof. V. Karwande, "DETECTION OF PHISHING EMAIL BASED ON NLP AND ML TECHNIQUES", Open Access International Journal of Science & Engineering || Volume 6 || Issue 6 || June 2021 || http://www.oaijse.com/VolumeArticles/FullTextPDF/784_13.DETECTION_OF_PHISHING_EMAI L_BASED_ON_NLP_AND_ML_TECHNIQUES.pdf

[7]Mahdieh Zabihimayvan , Derek Doran, "Fuzzy rough set feature selection to enhance phishing attack detection" , IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), issue:2019, https://ieeexplore.ieee.org/document/8858884

[8]Junaid Rashid, Toqeer Mahmood, Muhammad Wasif Nisar, Tahira Nazir, "Detection of phishing websites using machine learning approach", 4th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT),issue:2020 https://ieeexplore.ieee.org/document/9114695

[9]Smita Sindhu , Arya Sreevalsan , Sunil Parameshwar Patil , Faiz Rahman, "Phishing Detection using Random Forest, SVM and Neural Network with Backpropagation", International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE),issue:2020, https://ieeexplore.ieee.org/document/9277256

[10] Vaibhav Patel, Tushar Bhat, Pritesh Thakkar, SP Godse, Chirag Shah "Detection and prevention of phishing websites using machine learning approach" Fourth International Conference on Computing Communication Control and Automation (ICCUBEA) 2018

[11] Shraddha Parekh, Srusthi Kotak, Dhwani Parikh, Smita Sankhe "A new method for detection of phishing websites : URL Detection" 2nd International Conference On Intentive Communication and Computational Technologies (ICICCT) 2018

[12] 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, 2019

[13]Mohammed Hazim Alkawaz, Stephanie Joanne Steven, Asif Iqbal Hajamydeen, " Detecting Phishing Website Using Machine Learning",16th IEEE International Colloquium on Signal Processing & Its Applications (CSPA)|2020| https://ieeexplore.ieee.org/document/9068728

[14]Muhammet Baykara, Zahit Ziya Gürel, "Detection of phishing attacks", 6th International Symposium on Digital Forensic and Security"(ISDFS)|2018|https://ieeexplore.ieee.org/document/8355389

[15] Arun D. Kulkarni, Leonard L. Brown, "Phishing Websites Detection using Machine Learning", International Journal of Advanced Computer Science and Applications (IJACSA) |2019|, https://scholarworks.uttyler.edu/cgi/viewcontent.cgi?article=1022&context=compsci_fac