DETECTION OF PHISHING WEBSITES USING MACHINE LEARNING APPROACH

A Project Report submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY (B.Tech)

in

COMPUTER SCIENCE & ENGINEERING (CSE)

by

NAME OF STUDENTS	ROLL NO.	REGISTRATION NO.
SOUHARDYA GAYEN	14400118005	181440110033
SHREYA KAYAL	14400118008	181440110030
NITISH RANJAN NASKAR	14400118024	181440110014

Under the supervision of

Prof. SUBRATA DATTA

(Assistant Professor, Department of CSE, NITMAS)

Co-Guided by

Dr. BAPPADITYA MONDAL

(Assistant Professor, Department of CSE, TNU)

At

Neotia Institute of Technology, Management and Science

(Affiliated to MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY, WEST BENGAL, INDIA and approved by AICTE, NEW DELHI, INDIA)

D.H. ROAD, SOUTH 24-PARAGANAS, WEST BENGAL, INDIA, PIN-743368

JUNE 2022

BATCH: 2018 - 2022

DECLARATION

We, hereby, declare that the project entitled "**DETECTION OF PHISHING WEBSITES USING MACHINE LEARNING APPROACH**", submitted by us to Neotia Institute of Technology, Management and Science in the partial fulfilment of the requirements for the award of the degree of B. Tech. in Computer Science & Engineering is a record of an original work done by us under the guidance of Prof. SUBRATA DATTA and Dr. BAPPADITYA MONDAL .We further declare that the project work report has not been submitted for any other degree, diploma or equivalent academic awards in MAKAUT or any other institutions.

Student 1

_____

SOUHARDYA GAYEN

Student 2

_____

SHREYA KAYAL

Student 3

_____

NITISH RANJAN NASKAR

Place: NITMAS

Date: 15/06/2022

CERTIFICATE OF RECOMMENDATION

This is to certify that the project entitled "DETECTION OF PHISHING WEBSITES USING MACHINE LEARNING APPROACH "is the original work done by SOUHARDYA GAYEN (Roll No.: 14400118005), SHREYA KAYAL (Roll No.: 14400118008) and NITISH RANJAN NASKAR (Roll No.: 14400118024) under my supervision in partial fulfilment of the degree of Bachelor of Technology in Computer Science & Engineering at Neotia Institute of Technology, Management and Science (affiliated to MAKAUT and approved by AICTE) during the academic year 2021-2022.

Wish them all success in life.

Prof. SUBRATA DATTA
Project Supervisor
Dept. of Computer Science & Engineering
Neotia Institute of Technology, Management and Science

Dr. BAPPADITYA MONDAL
Project Supervisor
Dept. of Computer Science & Engineering
The Neotia University

Place: NITMAS
Date: 15/06/2022

CERTIFICATE OF APPROVAL

The project report entitled "DETECTION OF PHISHING WEBSITES USING MACHINE LEARNING APPROACH" submitted by SOUHARDYA GAYEN (Roll No.: 14400118005, Registration No: 181440110033 of 2018-2019), SHREYA KAYAL (Roll No.: 14400118008, Registration No: 181440110030 of 2018-2019) and NITISH RANJAN NASKAR (Roll No.: 14400118024, Registration No: 181440110014 of 2018-2019) of Neotia Institute of Technology, Management and Science (NITMAS) is approved for the degree of Bachelor of Technology in Computer Science & Engineering (CSE) under MAKAUT, West Bengal, India.

_____

Dr. BAPPADITYA MONDAL

Supervisor

Dept. of Computer Science & Engineering

The Neotia University

_____

Prof. SUBRATA DATTA

SIC, Project Lab

Dept. of Computer Science & Engineering

Neotia Institute of Technology, Management and Science

_____

Prof. DEBABRATA NATH

HOD/ In-Charge

Dept. of Computer Science & Engineering

Neotia Institute of Technology, Management and Science

ACKNOWLEDGEMENT

We would like to express our profound gratitude to Prof. DEBABRATA NATH (HOD/ In-Charge in Dept. of Computer Science & Engineering of Neotia Institute of Technology, Management and Science) and Prof. SUBRATA DATTA (SIC, Project Lab in Dept. of Computer Science & Engineering of Neotia Institute of Technology, Management and Science) for their contributions to the completion of my project titled "DETECTION OF PHISHING WEBSITES USING MACHINE LEARNING APPROACH". I would like to express my special thanks to our Project Mentor Dr. BAPPADITYA MONDAL (Supervisor in Dept. of Computer Science & Engineering of The Neotia University), for his time and efforts he provided throughout the year. Your useful advice and suggestions were really helpful to us during the project's completion. In this aspect, I am eternally grateful to you.

We would like to acknowledge our parents for their vital cooperation and help in ensuring the successful completion of our project. Finally, we would want to convey our sincere thanks to our friends and supporters without whom, the task would not have been accomplished in such a timely manner.

Signature:

1. _____

SOUHARDYA GAYEN

2. _____

SHREYA KAYAL

3. _____

NITISH RANJAN NASKAR

Place: NITMAS

Date: 15/06/2022

CONTENTS

ABSTRACT

The predefined anti-phishing approaches based on machine learning techniques extract few features from other sources which detects the fraudulent websites comparatively slower and unfit for real-time execution. This paper presents a solution to this problem by deeply examining various characteristics of phishing as well as legitimate websites and analysing thirty outstanding features to distinguish phishing websites from legitimate URLs. These thirty features scan the URLs thoroughly and extract their values from the URLs of the websites and are thus independent of any third-party presence which gives an accuracy of 93%.

This results in a more realistic, reliable, resourceful, well-informed computational approach.

INTRODUCTION

Phishing is a social engineering attack that aims at exploiting the weakness found in the system at the user's end. For example, a system may be technically secure enough for password theft but the unaware user may leak his/her password when the attacker sends a false update password request through forged (phished) website. For addressing this issue, a layer of protection must be added on the user side to address this problem. A phishing attack is when a criminal sends an email or the url pretending to be someone or something he's not, in order to get sensitive information out of the victim. The victim in regard to his/her curiosity or a sense of urgency, they enter the details, like a username, password, or credit card number, they are likely to acquiesce. The recent example of a Gmail phishing scam that targeted around1 billion Gmail users worldwide.

Phishing attacks have become anxiety for the cyber world. It causes enormous problems for privacy and financial issues of internet users. Scammers, namely fishers, create false websites to feel and look like a genuine to deceive the people. They spoof emails to steal the identity of legitimate users. They gather personal covert information, password, account information, and credit card details for the transaction. Fishers always change their strategy to attack the system. Social engineering is one of the essential techniques the fishers use. Using this technique, they gather personal credentials from a trustworthy person. Phishers create false websites and spoof email in such a way that they are very similar and sometimes look like a real company website that comes from a source. Sometimes the attackers act like a real source and force the users to update the system. Moreover, they threaten the customer to suspend the account and demand ransom. Email spoofing is another technique used for phishing fraud. Customers are usually misled to disclose private information like passwords and credit card number. Thus fishing is mainly used to steal valuable information such as bank account, password, and credit card details. This type of scam is increasing rapidly, and individuals, business-people are losing their trust in online business. Thus, a negative impression of clients on online business was swarmed as they lost faith in online transactions. Even though encryption software is used to protect the information in the computers' storage, they are also vulnerable to attacks. In this paper, the detection of fishing was performed through ML.

As phishing attack allows attackers a foothold in corporate networks causing access to vital information, it is important to safeguard users from becoming victims of fraud. Thus phishing detection tools play a vital role in ensuring security. So we planned to work on this topic.

BACKGROUND STUDY:-

Before making this project, we have done the following background study -

- Good knowledge of Python Programming Language.
- Working knowledge of Jupyter Notebook software
- Working knowledge of Visual Studio Code
- Studied what is Phishing
- In what perspectives phishing is different from legitimate
- Harmful activities of Phishing Websites
- Profound knowledge of Machine Learning
- Collection of data set from the internet
- Gone through several published papers to know more about phishing website
- detection
- Collection of Features
- Good knowledge of Exploratory Data Analysis
- Well-informed about data cleaning techniques
- Algorithms based on feature selection process
- Profound knowledge of Machine Learning Algorithms

PROBLEM STATEMENT:-

Malicious links will lead to a website that often steals login credentials or financial information like credit card numbers. Attachments from phishing emails can contain malware that once opened can leave the door open to the attacker to perform malicious behaviour from the user's computer. Phishing causes social engineering attack . It occurs when an attacker, masquerading as a trusted entity, dupes a victim into opening an email, instant message, or text message.

The Harmful Effects of Phishing on Businesses:

- Loss of Data
- Damaged Reputation
- Direct Monetary Loss
- Loss of Productivity
- Loss of Customers
- Financial Penalties
- Intellectual Property Theft
- Loss of Company Value

PROPOSED WORK:-

This section reflects on how we framed our proposed work with EDA technology, data cleaning, feature selection algorithms, Machine Learning algorithms and evaluated the performance. The following figure (Fig 1) gives a brief of the steps we have followed in our proposed methodology . It aims at building a website to detect whether the websites are fraud or trustworthy.

STEP 1 :- DATA COLLECTION

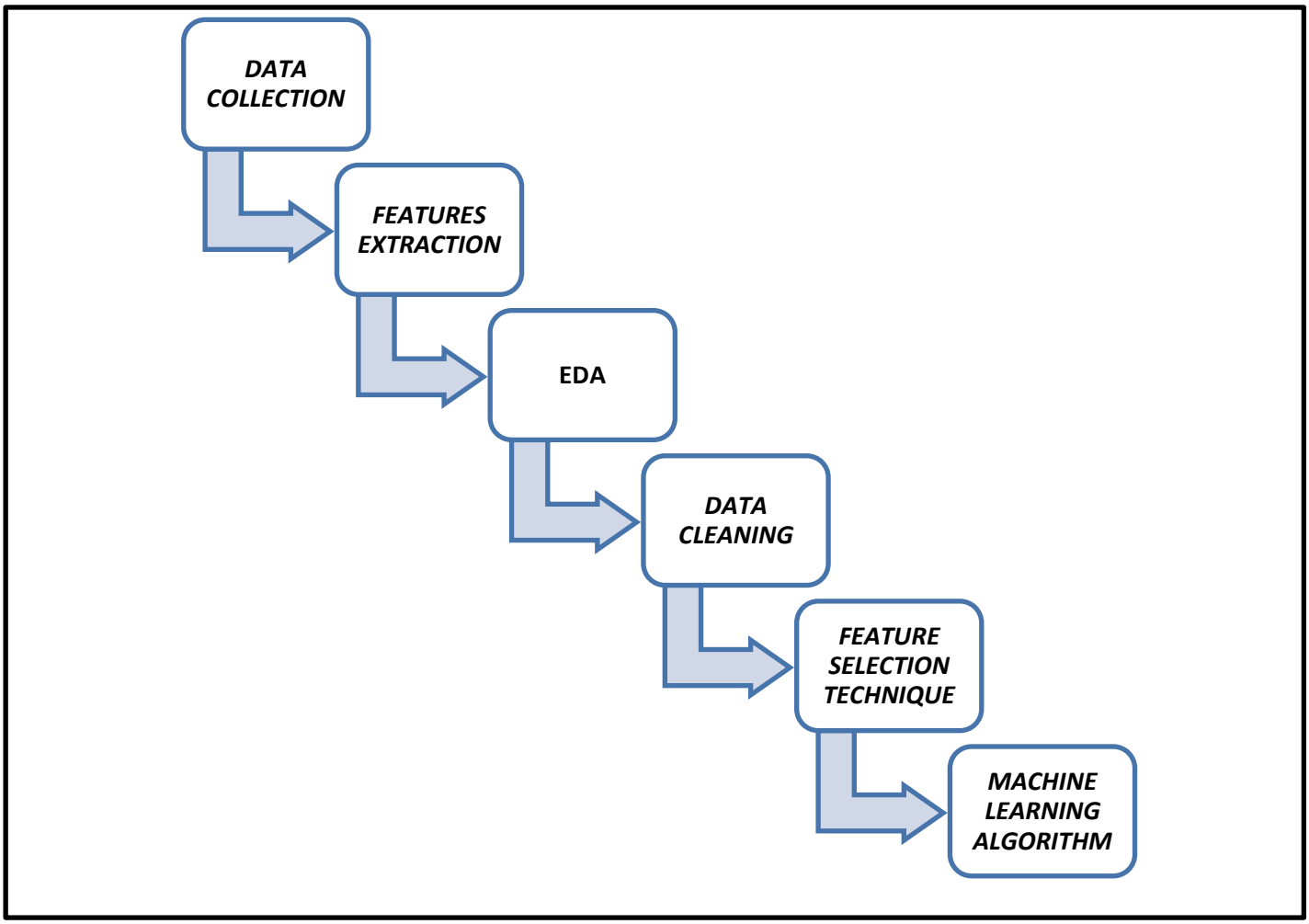
STEP 2 :- FEATURES EXTRACTION

STEP 3 :- EXPLORATORY DATA ANALYSIS

STEP 4 :- DATA CLEANING

STEP 5 :- FEATURE SELECTION TECHNIQUE

STEP 6 :- MACHINE LEARNING ALGORITHM



STEP I

1. DATA COLLECTION:-

In this step we have collected two data sets – one data set having phishing URLs and other having legitimate URLs from open source in csv (comma-separated values) format. From these datasets, 4898 phishing URLs out of 13110 phishing websites and 6157 legitimate URLs out of 35377 legitimate websites are randomly selected to train and test the ML models for our convenience. We have undertaken the above method since we know that if we train the model for 11055 data, it will be sufficient enough for rest of the websites.

The set of phishing URLs are collected from open source service called Phish Tank. The legitimate URLs are obtained from the open datasets consisting of the benign URL dataset.

The set of phishing URLs are collected from opensource service called PhishTank. This service provide a set of phishing URLs in multiple formats like csv, json etc. that gets updated hourly. To download the data: https://www.phishtank.com/developer_info.php. From this dataset, 4898 random phishing URLs are collected to train the ML models.

```
#Collecting 5,000 Phishing URLs randomly

phishurl = dataset.sample(n=4898 , replace = False, random_state=12, axis=0).copy()
phishurl = phishurl.reset_index(drop=True)
```

```
phishurl.to_csv('phishing5000.csv', index= False)
```

```
phishurl["url"][100]
```

```
'http://owiagbn.wmsistseg.com.br/'
```

```
phishurl.shape
```

```
(4898, 8)
```

The legitimate URLs are obatined from the open datasets of the University of New Brunswick, https://www.unb.ca/cic/datasets/url-2016.html. This dataset has a collection of benign, spam, phishing, malware & defacement URLs. Out of all these types, the benign url dataset is considered for this project. From this dataset, 6157 random legitimate URLs are collected to train the ML models.

```
dataset1.shape
```

```
(35349, 1)
```

```
legiurl = dataset1.sample(n=6157, replace = False, random_state=12,  axis=0).copy()
legiurl = legiurl.reset_index(drop=True)
legiurl.head()
```

	URLs
0	http://smallseotools.com/blog/how-to-create-a-...
1	http://sourceforge.net/directory/communication...
2	https://twitter.com/home?status=%E3%83%8C%E3%8...
3	http://extratorrent.cc/torrent_download/419094...
4	http://abcnews.go.com/US/emt-proposes-tampa-ba...

```
legiurl.to_csv('legitimate5000.csv', index= False)
```

```
legiurl["URLs"][100]
```

```
'http://hdfcbank.com/personal/ways-to-bank/bank-with-your-phone/mobilebanking/mobilebanking-app-for-iphone'
```

```
legiurl.shape
```

```
(6157, 1)
```

STEP II

2.FEATURES EXTRACTION:-

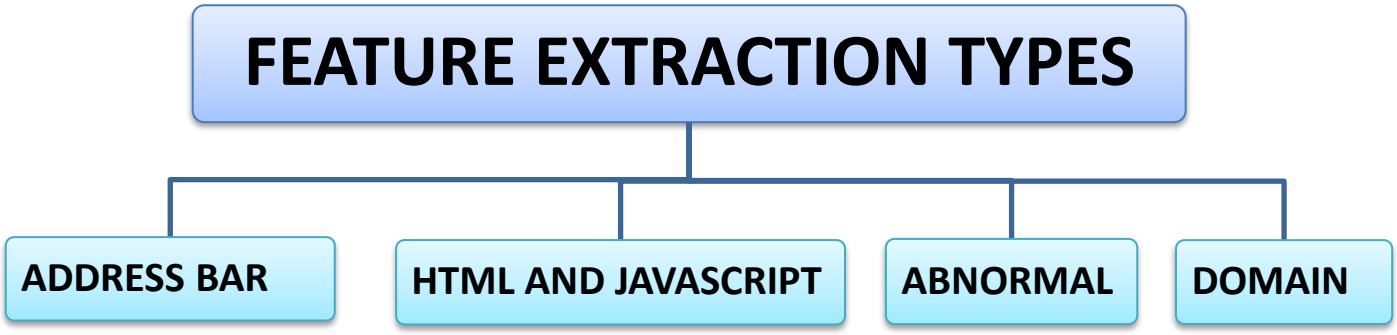
In this section, we have applied 30 features on the two data sets to transform the data sets into our desired form for Machine Learning modelling. The 30 features we have implemented, are categorized into four types of features which are as following ———

A) ADDRESS BAR BASED FEATURES

B) HTML AND JAVASCRIPT BASED FEATURES

C) ABNORMAL BASED FEATURES

D) DOMAIN BASED FEATURES



A) ADDRESS BAR BASED FEATURES: -

The features that are based on the address of the website containing Uniform Resource Locator (URL) and also allowing to navigate to a desired website are known as Address Bar Based Features.

The Address Bar Based Features are as follows -

1. IP Address – If an IP address is present in place of domain name such as http://125.94.5.122/fraud.html, users can be sure that someone is phishing into their computer.

RULE:

{If (The Domain Part has an IP Address)

Then Phishing}

{Otherwise

Legitimate}

2. URL Length – Phish attackers can use long URL to hide the fraud or suspicious part in the address bar. For example:

http://federmacedoadv.com.br/3f/aze/ab51e2e319e51502f416dbe46b773a5e/?cmd=_home&dispatch=11004d58f5b74f8dc1e7c2e8dd4105e811004d58f5b74f8dc1e7c2e8dd4105e8@phishing.website.html

Here we calculate the length of the URL and if it is found to be greater than or equal to 54 characters, then The URL is considered to be phishing.

RULE:

{If (URL length<54)

Then Legitimate}

{else if (URL length≥54 and ≤75)

Then Suspicious}

{Otherwise

Phishing}

3. Shortening service - URL shortening service is a process through which a URL may be made considerably smaller in length on the "World Wide Web". This is implemented by an "HTTP Redirect" on a domain name that is short but on clicking the short URL redirects to the long URL webpage. For example, the URL "http://portal.hud.ac.uk/" can be shortened to "bit.ly/19DXSk4".

RULE:

{IF (TinyURL)

Then Phishing}

{Otherwise

Legitimate}

4. @ symbol - If the URL contains "@" symbol, it leads the browser to ignore everything preceding the "@" symbol and the real address often follows the "@" symbol.

RULE:

{IF (URL Having @ Symbol)

Then Phishing

{Otherwise

Legitimate}

5. redirecting using //- Whenever the URL path has double slash redirection "//",the user is generally redirected to another website. An example of such URL's is: "http://www.legitimate.com//http://www.phishing.com". We have examined that "//" appears in the sixth position if the URL begins with "HTTP" and in seventh position if the URL begins with "HTTPS".

RULE:

{IF (The Position of the Last Occurrence of "//\" " in the URL > 7)

Then Phishing}

{Otherwise

Legitimate}

6. Prefix-Suffix - The statistics says that dash (-) symbol is rarely found in legitimate URLs. Phishers have a tendency to add a dash between prefixes or suffixes to the domain name so that users are unable to distinguish fraud websites from a legitimate webpage. For example, http://www.Confirme-paypal.com/.

RULE:

{IF (Domain Name Part Includes (-) Symbol)

Then Phishing}

{Otherwise

Legitimate}

7. Sub Domain - Let us take an example - link: http://www.hud.ac.uk/students/. A domain name may contain the ccTLD (country-code top-level domains).Here ccTLD is "uk". The "ac" denotes "academic", the "ac.uk" together is a SLD (second-level domain) and "hud" is the domain name. To extract this feature, first we have to omit the (www.) from the URL since it is a sub domain in itself. Then, we need to remove the (ccTLD) if it persists. Finally, we count the remaining dots. If the number of dots is calculated more than one, then the URL is categorized as "Suspicious" as it has one sub domain. However, if the dots are found to be more than two, it is classified as "Phishing" as it will consist of multiple sub domains. Otherwise, if the URL has no sub domains, it is a feature of "Legitimacy".

RULE:

{IF ({Dots in Domain Part=1)

Then Legitimate}

{else if (Dots in Domain Part=2)

Then Suspicious}

{Otherwise

Phishing}

8. SSL final state - Hyper Text Transfer Protocol with Secure Sockets Layer feature focus on checking the certificate assigned with HTTPS along with the trust certificate issuer, and the certificate age. Furthermore, we determined that a reputable certificate has a minimum age of two years.

RULE:

{IF(Use https and Issuer Is Trusted &and Age of Certificate≥ 1 Year)

Then Legitimate}

{else (if Using https and Issuer Is Not Trusted)

Then Suspicious

{Otherwise

Phishing}

9. Domain registration length – On the basis of facts, a phishing website stays for a short span of time whereas trustworthy domains are regularly paid for many years. In our dataset, we executed and found that the fraud domains have been maximum used for one year only.

RULE:

{IF (Domains Expires on≤ 1 year)

Then Phishing}

{Otherwise

Legitimate}

10. Favicon - A favicon is a graphic tiny image (icon) associated with a definite webpage allowing many user agents like graphical browsers and newsreaders to show favicon as a pictorial reminder of the website identification in the address bar. If the favicon is loaded from an external domain i.e. different from the one shown in the address bar, then the webpage has a probability of Phishing attempt.

RULE:

{IF (Favicon Loaded From External Domain)

Then Phishing}

{Otherwise
Legitimate}

11. Non-Standard Port - This feature aims at figuring out the validity of a particular service (e.g. HTTP) based on whether it is up or down on a specific server. Generally, various firewalls by default block all or most of the ports and only make available the ones selected. The objective of the phishers is to open all ports so that they can run almost any service they desire and as a result, target (user) information is threatened and at risk.

RULE:
{IF ("Port # is of the " preferred Status)
Then Phishing}

{Otherwise
Legitimate}

12. HTTPS Token - The phishers easily trick the users by adding the "HTTPS" token in the domain part of a URL. For example, http://https-www-paypal-it-webapps-mpp-home.soft-hair.com/.

RULE:
{IF ("Using " HTTP Token in Domain Part of The URL)
Then Phishing
{Otherwise
Legitimate

B) HTML AND JAVASCRIPT BASED FEATURES:-

The features based on scripting language JavaScript (abbreviated as JS) and markup language Hypertext Markup Language (abbreviated as HTML) used for web development are called HTML and JavaScript Based Features respectively.

The HTML and JavaScript Based Features are as follows –

1. Website Forwarding- This is one of the important features to distinguish phishing websites from legitimate ones where we look on how many times a website have been redirected. In our dataset, we have acknowledged that legitimate websites are redirected one time at max whereas, phishing websites are redirected at least 4 times. So, higher the number of forwarding or redirecting of website, higher is the probability of phishing.

Rule:

{IF (Number of Redirect Page ≤ 1)

Then Legitimate}

{else IF (Number of Redirect Page ≥ 2 and < 4)

Then Suspicious}

{Otherwise

Phishing}

2. Status Bar Customization- Phish attackers probably use JavaScript to show an unauthorized URL in the status bar to the target. To evoke this feature, we must pull-out the webpage source code, especially the "onMouseOver" event to check whether it makes any changes to the status bar. If it is capable enough to alter Status Bar, it is prone to phishing. Otherwise, it is quite liable to be legitimate.

Rule:

{IF (onMouseOver Changes Status Bar)

Then Phishing}

{else if (It Doesn't Change Status Bar)

Then Legitimate}

3. Disabling Right Click- Phishers use this JavaScript feature to disable the right-click function, so that the webpage source code is hidden from users. This feature is similar to "Using onMouseOver to hide the Link". Here we will search if the webpage source code consists of event "event.button==2" and also disability of the right click.

Rule:

{IF (Right Click Disabled)

Then Phishing

{Otherwise

Legitimate}

4. Using Pop-up Window- Generally, the user fills the personal information and finally clicks on submit button. A pop-up window is shown on screen broadcasting a welcome announcement. This is a prevalent feature of legitimate websites. The fraudulent activities are carried out by asking the users to refill the personal details through these pop-up windows.

Rule:

{IF (Popoup Window Contains Text Fields)

Then Phishing}

{Otherwise

Legitimate}

5. IFrame Redirection- IFrame is an HTML tag used to display an additional webpage into a currently shown webpage. Phishers can use the "iframe" tag by making it invisible i.e. without frame borders. In this way, phishers implement the "frameBorder" attribute causing the browser to render a visual depiction.

Rule:

{IF (Using iframe)

Then Phishing}

{Otherwise

Legitimate}

C) ABNORMAL BASED FEATURES: -

The features based on identifying the data which are unfit for the normal patterns necessary to protect the system against fraudulent activities are called Abnormal Based Features.

The Abnormal Based Features are as follows –

1. Request URL- Request URL examines the external objects within a webpage such as images, videos and sounds and detects whether loaded from another domain or same domain. The legit webpages are characterised if the web address and most of the objects are enclosed within the webpage of the same domain.

Rule:

{IF (% of Request URL <22%)

Then Legitimate}

{else if (%of Request URL≥22% and 61%)

Then Suspicious}

{Otherwise

Phishing}

2. URL of Anchor- An anchor is an HTML element defined by the <a> tag used to define a hyperlink to connect a page with another. This feature is same as The "Request URL" feature. This feature is accomplished by checking if the <a> tags and the website belong to different domain names. If the anchor does not link to any webpage, then it is considered to be a legit URL. e.g.:

<a href="#">

<a href="#content">

<a href="#skip">

<a href="JavaScript ::void(0)">

Rule:

{IF (% of URL Of Anchor <31%)

Then Legitimate}

{else IF (% of URL Of Anchor ≥31% and ≤67%)

Then Suspicious

{Otherwise

Phishing}

3. Links in <Meta>, <Script> and <Link> tags- To create our dataset, we have executed this feature for legitimacy check as we know <Meta> tag defines metadata about the HTML document, <Script> tag reflects on client side script and <Link> tag helps to retrieve other web sources. It is contemplated that these are related to the same domain of the webpage.

Rule:

{IF (% of Links in <Meta>, <Script> and <Link><17%)

Then Legitimate}

{else if (% of Links in <Meta>, <Script> and <Link> ≥17% And≤81%)

Then Suspicious}

{Otherwise

Phishing}

4. Server Form Handler (SFH) - Sometimes it is found “about:blank” when the page stop working and produce a blank page . This is considered to be fishy because some action needs to be executed after clicking on the URL or submitting the information. When the information submitted is handled by external domains i.e the domain name in SFHs and webpage are different , it is detected to be phishing.

Rule:

{IF (SFH is ""about: blank\"" Or Is Empty)

Then Phishing}

{else if (SFH "Refers To” A Different Domain)

Then Suspicious}

{Otherwise

Legitimate}

5. Submitting Information to Email- If mail () as a server- side script language or mailto() as a client-side script language are utilised for submitting user's personal information, it is prone to phishing characteristics.

Rule:

{IF (Using ""mail ()\" or \"mailto:\" Function to Submit User Information")

Then Phishing}

{Otherwise

Legitimate}

6. Abnormal URL- Host name is an identification of a legitimate URL. This feature is implemented using WHOIS library.

Rule:

{IF (The Host Name Is Not Included In URL)

Then Phishing}

{Otherwise

Legitimate}

D) DOMAIN BASED FEATURES: -

The features based on domain of a website which enables to train our model on source distribution to produce the target distribution are called Domain Based Features.

The Domain Based Features are as follows –

1. Age of Domain- Generally a phishing website has a short tenure of 6 months since it has a tendency of being caught. So its domain age is short. This feature comes into existence by using WHOIS database.

Rule:

{IF (Age of Domain≥6 months)

Then Legitimate}

{Otherwise

Phishing}

2. DNS Record- The recognition of the website by the Domain Name System (DNS) record provides confirmation for legitimacy. Thus if the given website is not identified by the WHOIS database or the hostname is not claimed in the records, then it will be undoubtedly a phishing website.

Rule:

{IF (no DNS Record For the Domain)

Then Phishing}

{Otherwise

Legitimate}

3. Website Traffic- This feature is determined by the number of visitors to the website. If the viewers are more, the website rank will be less. The statistical study says that most of the legitimate websites rank among the top 1, 00,000 whereas fraudulent websites rank more than 1, 00,000. Thus phish URLs have either 0 traffic or are not recognized by the database.

Rule:

{IF (Website Rank<100,000)

Then Legitimate}

{else if (Website Rank>100,000)

Then Suspicious}

{Otherwise

Phishing}

4. PageRank- PageRank figures out the importance of a webpage on the internet ranging from '0' to '1'. The greater the value of the PageRank, the higher the importance of the webpage. Generally, fake web pages have almost no PageRank. Sometimes, it is found that a few fraud webpages have a PageRank value of about 0.2.

Rule:

{IF (PageRank<0.2)

Then Phishing}

{Otherwise

Legitimate}

5. Google Index- Google Index is a vital feature depicting whether a site is indexed by the Google or not. Phishing sites have an accessibility for a small period of time .As a result many fake sites are not available on the Google index.

Rule:

{IF (Webpage Indexed by Google)

Then Legitimate}

{Otherwise

Phishing}

6. Number of Links Pointing to Page- This is characterized by the number of links pointing to the webpage. Legitimacy is indicated by at least 2 external links pointing to them, even if belonging to the same domain. On the other hand, the phishing datasets have no links pointing to them.

Rule:

{IF (Number of Link Pointing to the Webpage=0)

Then Phishing}

{else if (Number of Link Pointing to The Webpage>0 and ≤ 2)

Then Suspicious}

{Otherwise

Legitimate}

7. Statistical-Reports Based Feature- In this feature, we have gone through some of the top statistical-reports published from where we have extracted top 10 domains and top 10 IPS. Also, we have acknowledged PhishTank to frame numerical based statistical report on phishing websites.

Rule:

{IF (Host Belongs to Top Phishing IPs or Top Phishing Domains)

Then Phishing}

{Otherwise

Legitimate}

Legitimate

```
# All this details form url
legi_features1=[]
label=0

for i in range(0, 6157):
    url = legiurl['URLs'][i]
    legi_features1.append(generate_data_set(url))
```

```
[1, -1, 1, 1, 1, 1, 1, 1, -1, 1, 1, -1, -1, -1, -1, 1, -1, 0, -1, -1, -1, 1, 1, -1, 1, -1, 1, -1, 1]
[1, -1, 1, 1, 1, 1, 1, 1, -1, 1, 1, -1, -1, -1, 1, 0, 1, -1, 0, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1]
[1, -1, 1, 1, 1, 1, 1, 1, -1, -1, 1, -1, 1, -1, 1, 0, 1, -1, -1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1]
[1, -1, 1, 1, 1, 1, 0, 1, 1, -1, 1, 1, 1, 1, 1, -1, 1, -1, -1, -1, -1, -1, 1, -1, 1, -1, 1, 1, 1]
[1, -1, 1, 1, 1, 1, 1, 1, -1, 1, 1, -1, -1, 0, -1, 0, 1, -1, -1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1]
[1, -1, 1, 1, 1, -1, 1, 1, -1, -1, 1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1]
[1, -1, 1, 1, 1, 1, 1, 1, -1, 1, 1, -1, 0, -1, 0, 0, 1, -1, -1, -1, -1, 1, 1, 1, -1, 1, 1, 1, -1, 1]
Error trying to connect to socket: closing socket
[1, -1, 1, 1, 1, 1, -1, 1, -1, 1, 1, -1, -1, 0, -1, 1, -1, -1, -1, -1, 1, 1, 1, -1, 1, 1, 1, -1, 1]
[1, -1, 1, 1, 1, 1, 1, 1, -1, 1, 1, -1, -1, 0, 0, 0, 1, -1, -1, -1, -1, -1, 1, 1, -1, 1, -1, 1, -1, 1]
[1, -1, 1, 1, 1, 1, 0, 1, -1, 1, -1, 1, 1, 0, -1, 1, -1, -1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1]
[1, -1, 1, 1, 1, 1, 1, 1, -1, -1, 1, -1, 1, -1, 1, 0, 1, -1, -1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1]
[1, -1, 1, 1, 1, 1, 0, 1, 1, 1, 1, -1, -1, -1, 0, 0, 1, -1, -1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1]
[1, -1, 1, 1, 1, -1, 0, -1, -1, -1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, 1]
[1, -1, 1, 1, 1, 1, 0, 1, -1, 1, -1, 1, 1, 0, -1, 1, -1, -1, -1, -1, -1, 1, 1, -1, 1, -1, 1, -1, 1]
[1, -1, 1, 1, 1, 1, 0, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 0, -1, -1, -1, 1, 1, -1, 0, 1, 1, 0, 1]
[1, -1, 1, 1, -1, 1, -1, 1, -1, 1, 1, -1, -1, 1, 0, 0, 1, -1, 0, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1]
[1, -1, 1, 1, 1, 1, 0, 1, 1, -1, 1, -1, 0, 1, 0, 0, 1, -1, 0, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1]
```

Phishing

```
#Extracting the feautres & storing them in a list

phish_features=[]


for i in range(0, 4898):
    url=phishurl['url'][i]
    phish_features.append(generate_data_set(url))
```

```
Connection problem. Please check your internet connection!
[1, 1, 1, 1, 1, 1, -1, -1, -1, -1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, -1]
[1, 1, 1, 1, 1, 1, 0, 1, -1, -1, 1, 1, 1, -1, 0, 1, -1, -1, -1, -1, -1, 1, 1, -1, -1, 1, 0, 1]
Error trying to connect to socket: closing socket
[1, 1, 1, 1, 1, 1, 1, -1, -1, -1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, -1, 1]
[1, 1, 1, 1, 1, 1, 0, 1, -1, -1, 1, 1, 1, -1, 1, 1, -1, -1, -1, -1, 1, 1, -1, 1, -1, 1, -1, 1]
Connection problem. Please check your internet connection!
[1, 1, 1, 1, 1, -1, 1, -1, -1, -1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, -1]
[1, 1, -1, 1, 1, 1, 1, -1, -1, -1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, 1]
[1, -1, 1, 1, 1, -1, -1, 1, -1, 1, 1, -1, 1, 1, 1, -1, 1, -1, -1, -1, -1, -1, 1, 1, -1, -1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 0, 1, 1, -1, 1, 1, 1, 0, 1, 1, 1, -1, 0, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1]
[1, 1, 1, 1, 1, 1, 1, 1, -1, 1, 1, 1, 1, 1, 1, -1, 1, -1, -1, -1, -1, -1, 1, 1, -1, 1, -1, 1, 1, 1]
[1, 1, 1, 1, 1, -1, 0, 1, -1, -1, 1, -1, 1, -1, 1, 1, -1, -1, -1, -1, -1, 1, 1, -1, -1, 1, 1, -1, 1]
[1, 0, 1, 1, 1, 1, -1, 1, -1, 1, 1, -1, -1, 0, -1, 1, -1, -1, -1, -1, -1, 1, 1, -1, -1, 1, 1, -1, 1]
[1, 0, 1, 1, 1, -1, 0, 1, -1, -1, 1, -1, 1, -1, 1, 0, 1, -1, -1, -1, -1, -1, 1, 1, -1, -1, 1, 1, -1, 1]
[1, 0, 1, 1, 1, 1, 0, 1, -1, 1, 1, 1, 1, 1, 1, -1, 1, -1, -1, -1, -1, -1, 1, 1, -1, 1, -1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 0, 1, -1, 1, -1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, 1, 1, -1, -1, 1, 1, 1, 1]
[1, -1, -1, 1, 1, 1, -1, 1, -1, -1, 1, 1, 1, -1, 1, 1, -1, -1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1]
```

After applying these 30 features on those two datasets (phishing and legitimate), we have got numerical based two datasets. Now we concatenate the two numerical based datasets into one dataset which is as following-

```
phishing_legitimate = pd.read_csv("DATASET PROJECT.csv")
```

```
phishing_legitimate
```

	index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain
0	1	-1	1	1	1	-1	-1	-1
1	2	1	1	1	1	1	-1	0
2	3	1	0	1	1	1	-1	-1
3	4	1	0	1	1	1	-1	-1
4	5	1	0	-1	1	1	-1	1
...	...	...	...	...	...	...	...	...
11050	11051	1	-1	1	-1	1	1	1
11051	11052	-1	1	1	-1	-1	-1	1
11052	11053	1	-1	1	1	1	-1	1
11053	11054	-1	-1	1	1	1	-1	-1
11054	11055	-1	-1	1	1	1	-1	-1

11055 rows × 32 columns

STEP III

3.EXPLORATORY DATA ANALYSIS:-

Exploratory Data Analysis (EDA) is a method to analyse the data based on visual techniques to have a thorough knowledge of trends, patterns and also to check assumptions through statistical and graphical studies.

The EDA Methods we have undertaken are as follows –

1. Variable Identification- In this section we have identified the predictors i.e., input variables and also the target i.e., output variable from the dataset. We have further identified the data type of the variables.

```
In [12]: #print the target and input variables
         print("input variables", list(x.columns))
         print("output variables", "['Result']")
```

```
input variables ['having_IPhaving_IP_Address', 'URLURL_Length', 'Shortining_Service', 'having_At_Symbol', 'double_slash_redirec
ting', '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', 'Rig
htClick', 'popUpWidnow', 'Iframe', 'age_of_domain', 'DNSRecord', 'web_traffic', 'Page_Rank', 'Google_Index', 'Links_pointing_to
_page', 'Statistical_report']
output variables ['Result']
```

```
In [13]: #Identify the data type of the variables
         df_train.info()
```

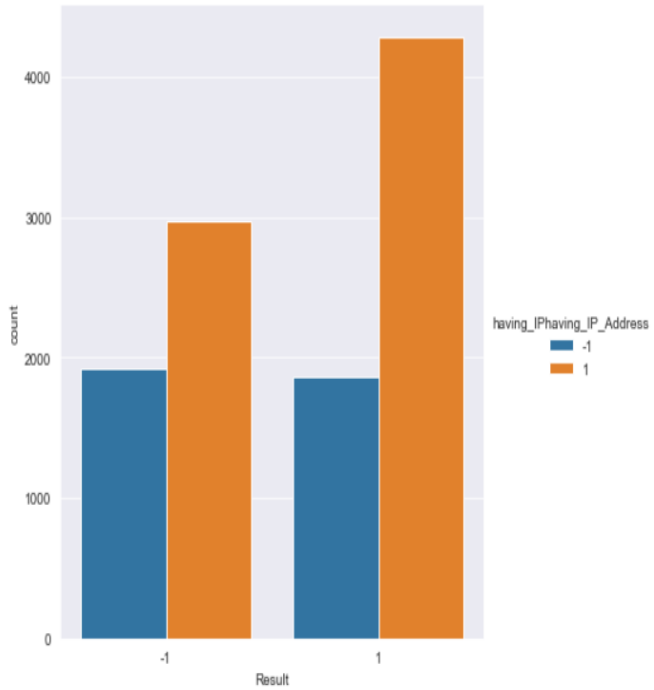
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11055 entries, 0 to 11054
Data columns (total 31 columns):
 #    Column                        Non-Null Count  Dtype
---   ------                        --------------  -----
 0    having_IPhaving_IP_Address    11055 non-null  int64
 1    URLURL_Length                 11055 non-null  int64
 2    Shortining_Service            11055 non-null  int64
 3    having_At_Symbol              11055 non-null  int64
 4    double_slash_redirecting      11055 non-null  int64
 5    Prefix_Suffix                 11055 non-null  int64
 6    having_Sub_Domain             11055 non-null  int64
 7    SSLfinal_State                11055 non-null  int64
 8    Domain_registeration_length   11055 non-null  int64
 9    Favicon                       11055 non-null  int64
 10   port                          11055 non-null  int64
 11   HTTPS_token                   11055 non-null  int64
 12   Request_URL                   11055 non-null  int64
 13   URL_of_Anchor                 11055 non-null  int64
 14   Links_in_tags                 11055 non-null  int64
 15   SFH                           11055 non-null  int64
 16   Submitting_to_email           11055 non-null  int64
 17   Abnormal_URL                  11055 non-null  int64
 18   Redirect                      11055 non-null  int64
 19   on_mouseover                  11055 non-null  int64
 20   RightClick                    11055 non-null  int64
 21   popUpWidnow                   11055 non-null  int64
 22   Iframe                        11055 non-null  int64
 23   age_of_domain                 11055 non-null  int64
 24   DNSRecord                     11055 non-null  int64
 25   web_traffic                   11055 non-null  int64
 26   Page_Rank                     11055 non-null  int64
 27   Google_Index                  11055 non-null  int64
 28   Links_pointing_to_page        11055 non-null  int64
 29   Statistical_report            11055 non-null  int64
 30   Result                        11055 non-null  int64
dtypes: int64(31)
memory usage: 2.6 MB
```

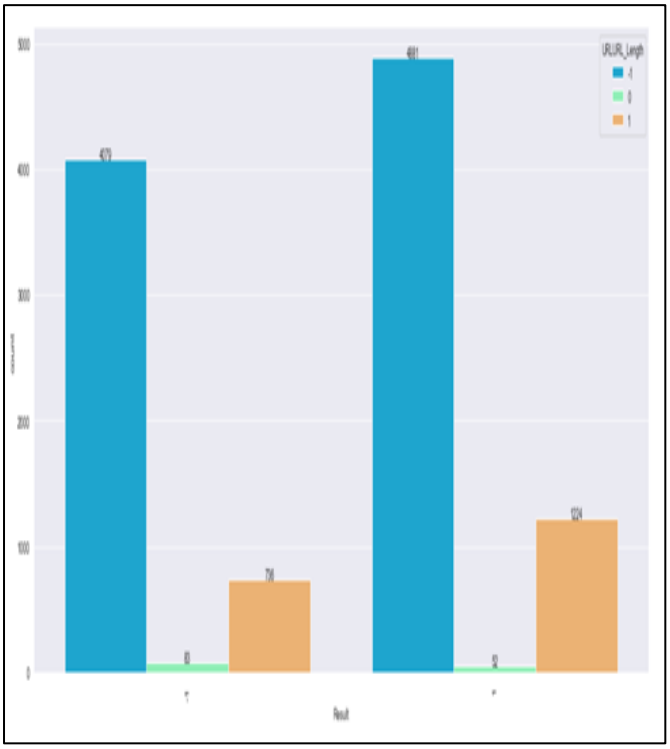
2.Univariate Analysis- At first, we need to explore all the variables one by one. Then we follow the Univariate Analysis. Since we have categorical data, we focus on frequency distribution of each category, Bar Chart, Pie Chart and so on. A count plot is based on a histogram across categorical, instead of quantitative, variable.

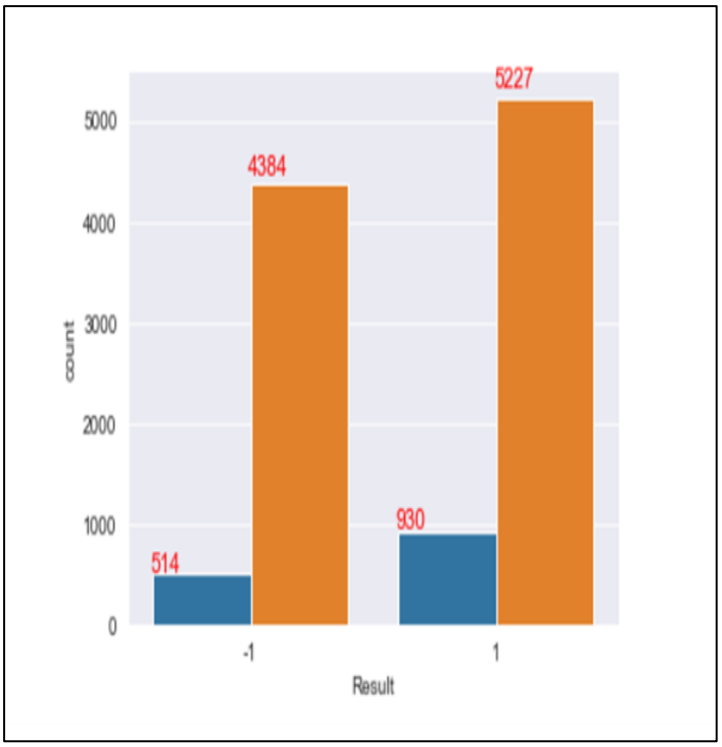
```
<matplotlib.axes._subplots.AxesSubplot at 0x292067302b0>
```

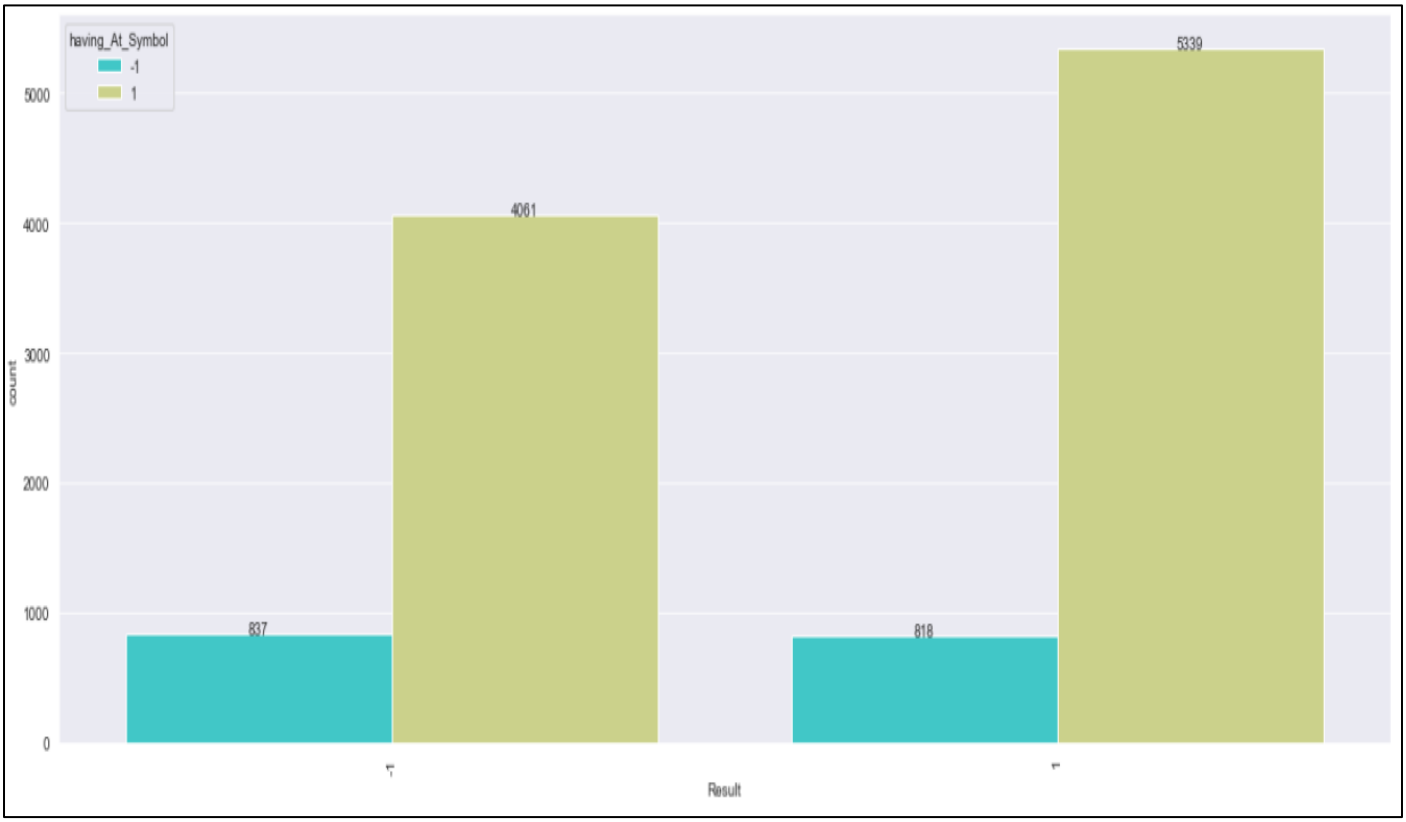
3. Bivariate Analysis- We perform bivariate analysis with 2 variables for any combination of categorical and continuous variables. The combination can be: Categorical & Categorical, Categorical & Continuous and Continuous & Continuous. Our dataset consists of categorical data only. So, we have implemented only categorical & categorical combination. We have analysed the target variable with input attributes.

```
In [23]: sns.catplot(x='Result',hue="having_IPhaving_IP_Address",data=df_train,kind = "count",height=6, aspect=1)
Out[23]: <seaborn.axisgrid.FacetGrid at 0x2920619dac0>
```






5227
4384
5000
4000
3000
2000
1000
0
count
930
514
-1
1
Result


having_At_Symbol
-1
1
5339
4061
5000
4000
3000
2000
1000
0
count
837
818
Result

STEP IV

4. DATA CLEANING: -

Data cleaning technique plays a vital role in discarding the incorrect, incomplete, irrelevant, duplicated and improperly formatted data.

The Data Cleaning Methods we have undertaken are as follows –

1.Handling Duplicate Data- First we have determined the number of duplicated rows. Then we have dropped those rows. Then we re-executed the code of duplicity to check whether the duplicated rows are discarded or not.

```
In [55]: # If you use the method sum() along with it, then it will return the total number of the duplicates in the dataset
         df.duplicated(keep=False)

Out[55]: 0         True
         1         True
         2         True
         3         True
         4         True
                  ...
         11050     True
         11051     True
         11052     True
         11053     True
         11054    False
         Length: 11055, dtype: bool

In [56]: df.duplicated(keep=False).sum()

Out[56]: 7843

In [57]: df.duplicated().sum()
         # keep = First

Out[57]: 5206
```

```
df1 = df.drop_duplicates(keep='first')
df1
```

9]:

	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain	SSLfir
0	-1	1	1	1	-1	-1	-1	
1	1	1	1	1	1	-1	0	
2	1	0	1	1	1	-1	-1	
3	1	0	1	1	1	-1	-1	
4	1	0	-1	1	1	-1	1	
...	...	...	...	...	...	...	...	
11037	1	-1	-1	1	-1	-1	-1	
11045	1	-1	1	1	1	-1	1	
11048	1	-1	1	1	1	-1	-1	
11049	-1	-1	1	1	-1	-1	1	
11054	-1	-1	1	1	1	-1	-1	

5849 rows × 31 columns

2. Handling Missing Value- We follow this step to eliminate the null values from the dataset as they cause incorrect result. However our categorical dataset contains no null values.

```
# lets find out the percentage of misssing vale in each column
percent_missing = df_train.isnull().sum() * 100 / len(df_train)
missing_value_df = pd.DataFrame({'column_name': df_train.columns,
                                 'percent_missing': percent_missing})
missing_value_df
```

	column_name	percent_missing
having_IPhaving_IP_Address	having_IPhaving_IP_Address	0.0
URLURL_Length	URLURL_Length	0.0
Shortining_Service	Shortining_Service	0.0
having_At_Symbol	having_At_Symbol	0.0
double_slash_redirecting	double_slash_redirecting	0.0
Prefix_Suffix	Prefix_Suffix	0.0
having_Sub_Domain	having_Sub_Domain	0.0
SSLfinal_State	SSLfinal_State	0.0
Domain_registeration_length	Domain_registeration_length	0.0
Favicon	Favicon	0.0
port	port	0.0
HTTPS_token	HTTPS_token	0.0
Request_URL	Request_URL	0.0
URL_of_Anchor	URL_of_Anchor	0.0
Links_in_tags	Links_in_tags	0.0

Request_URL	Request_URL	0.0
URL_of_Anchor	URL_of_Anchor	0.0
Links_in_tags	Links_in_tags	0.0
SFH	SFH	0.0
Submitting_to_email	Submitting_to_email	0.0
Abnormal_URL	Abnormal_URL	0.0
Redirect	Redirect	0.0
on_mouseover	on_mouseover	0.0
RightClick	RightClick	0.0
popUpWidnow	popUpWidnow	0.0
Iframe	Iframe	0.0
age_of_domain	age_of_domain	0.0
DNSRecord	DNSRecord	0.0
web_traffic	web_traffic	0.0
Page_Rank	Page_Rank	0.0
Google_Index	Google_Index	0.0
Links_pointing_to_page	Links_pointing_to_page	0.0
Statistical_report	Statistical_report	0.0
Result	Result	0.0

STEP V

5. FEATURE SELECTION TECHNIQUE: -

The feature selection techniques are essential to remove the redundant, irrelevant or less important features to train and build the machine learning model and thus help to enhance the overall performance and accuracy of the model.

The Feature Selection Techniques we have undertaken are as follows –

1. VARIANCE THRESHOLD-

Variance Threshold is a statistical filter method implemented to remove constant and quasi-constant features. We simply compute the variance of each feature, and select the subset of features based on a user-specified threshold assuming that features with a higher variance may contain more useful information.

```
#select those columns whose variance is greater than or equal to 1
#variance threshold = 1%
phishleg_var1 = phishlegVt.loc[:,phishlegVt.var() >= 0.01]
phishleg_var1.head(10)
```

	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain	SSLfinal_S
0	-1	1	1	1	-1	-1	-1	
1	1	1	1	1	1	-1	0	
2	1	0	1	1	1	-1	-1	
3	1	0	1	1	1	-1	-1	
4	1	0	-1	1	1	-1	1	
5	-1	0	-1	1	-1	-1	1	
6	1	0	-1	1	1	-1	-1	
7	1	0	1	1	1	-1	-1	
8	1	0	-1	1	1	-1	1	
9	1	1	-1	1	1	-1	-1	

10 rows × 30 columns

- **To remove the Constant Features:-**

Those features which contain constant values (only one value for all the outputs or target values) in the dataset i.e. threshold = 0 are termed as constant features. These features have no information to train the ML models and predict the target.

- **To remove the Quasi-Constant Features:-**

Those features that are almost constant since they are having the same values for a very large subset of the outputs are termed as quasi-constant features. These too have no contribution towards the prediction of the output. Since there is no predefined protocol to decide the value of threshold for the variance of quasi-constant features, we have assumed threshold = 0.01 which denotes that approximately 99% values are similar. Therefore, if the variance of the values in a column is less than 0.01, we will simply omit that column.

```
print("Total features : ", len(X_train.columns))
print("Total constant features : ", len(constantselected_cols))
print("Total non-quasi constant features : ", len(X_train.columns)-len(quasi_cols))
```

```
Total features :  30
Total constant features :  0
Total non-quasi constant features :  30
```

```
frame = X_train.var().to_frame()
```

```
frame = frame.rename(columns={0:"Variance"})
```

```
print("Total features : ", sum(frame.Variance.values>=0))
print("Total constant features : ", sum(frame.Variance.values==0))
print("Total non-constant and non-quasi constant features : ", sum(frame.Variance.values>0.01))
```

```
Total features :  30
Total constant features :  0
Total non-constant and non-quasi constant features :  30
```

```
#lets transfer the train and test data using fit selector
X_train_aq = quasiconstant_selector.transform(X_train)
X_test_aq = quasiconstant_selector.transform(X_test)
X_train_aq.shape, X_test_aq.shape
```

```
((4094, 30), (1755, 30))
```

2. REMOVE THE DUPLICATE FEATURES-

It is quite essential to eliminate the redundant features to avoid redundancy in the dataset. We have followed the method of removing the duplicate features for categorical data since we have categorical features.

```
# Print the duplicate features
print("Duplicate Features")
print(df_t.duplicated(keep = 'first')) # keep : {'first', 'last', False}, default 'first'
duplicate_feat = df_t.duplicated().sum()
print("Count of duplicate_feature :",duplicate_feat)
```

```
Duplicate Features
having_IPhaving_IP_Address     False
URLURL_Length                  False
Shortining_Service             False
having_At_Symbol               False
double_slash_redirecting       False
Prefix_Suffix                  False
having_Sub_Domain              False
SSLfinal_State                 False
Domain_registeration_length    False
Favicon                        False
port                           False
HTTPS_token                    False
Request_URL                    False
URL_of_Anchor                  False
Links_in_tags                  False
SFH                            False
Submitting_to_email            False
Abnormal_URL                   False
Redirect                       False
on_mouseover                   False
RightClick                     False
popUpWidnow                    False
Iframe                         False
age_of_domain                  False
DNSRecord                      False
web_traffic                    False
Page_Rank                      False
Google_Index                   False
Links_pointing_to_page         False
Statistical_report             False
Result                         False
dtype: bool
Count of duplicate_feature : 0
```

3. ANOVA (Analysis of Variance) or F-Test-

An ANOVA test is a way to find out if survey or experiment results are significant. In other words, they help you to figure out if you need to reject the null hypothesis or accept the alternate hypothesis.

Basically, you're testing groups to see if there's a difference between them. Examples of when you might want to test different groups:

- A group of psychiatric patients are trying three different therapies: counselling, medication and biofeedback. You want to see if one therapy is better than the others.
- A manufacturer has two different processes to make light bulbs. They want to know if one process is better than the other.
- A univariate test, Linear model for testing the individual effect of each of features with target.
- ANOVA assumes a linear relationship between the features and the target, and also that the variables are normally distributed.
- It's well-suited for continuous variables and requires a binary target, but sklearn extends it to regression problems, also.

```
print("P_Values :")
print(np.round(fs.pvalues_,4))
print("F Values :")
print(fs.scores_)
```

```
P_Values :
[0.000e+00 0.000e+00 0.000e+00 7.900e-03 5.000e-03 0.000e+00 0.000e+00
 0.000e+00 0.000e+00 5.160e-02 2.775e-01 1.000e-04 0.000e+00 0.000e+00
 0.000e+00 0.000e+00 5.400e-01 0.000e+00 4.200e-03 3.281e-01 4.777e-01
 8.590e-02 3.687e-01 0.000e+00 1.110e-02 0.000e+00 2.000e-04 0.000e+00
 0.000e+00 0.000e+00]
F Values :
[6.51218811e+01 1.98915700e+01 3.41178567e+01 7.07197919e+00
 7.88297262e+00 9.93703780e+02 5.62376190e+02 5.40947370e+03
 2.93491486e+02 3.79158361e+00 1.17936977e+00 1.54958606e+01
 3.50390596e+02 5.00510976e+03 3.69703923e+02 5.01773262e+02
 3.75536464e-01 4.77937749e+01 8.21727694e+00 9.56649146e-01
 5.04218845e-01 2.94976662e+00 8.08136488e-01 2.31506341e+01
 6.45334801e+00 5.04018152e+02 1.35815164e+01 8.28719573e+01
 2.75350910e+01 6.96335563e+01]
```

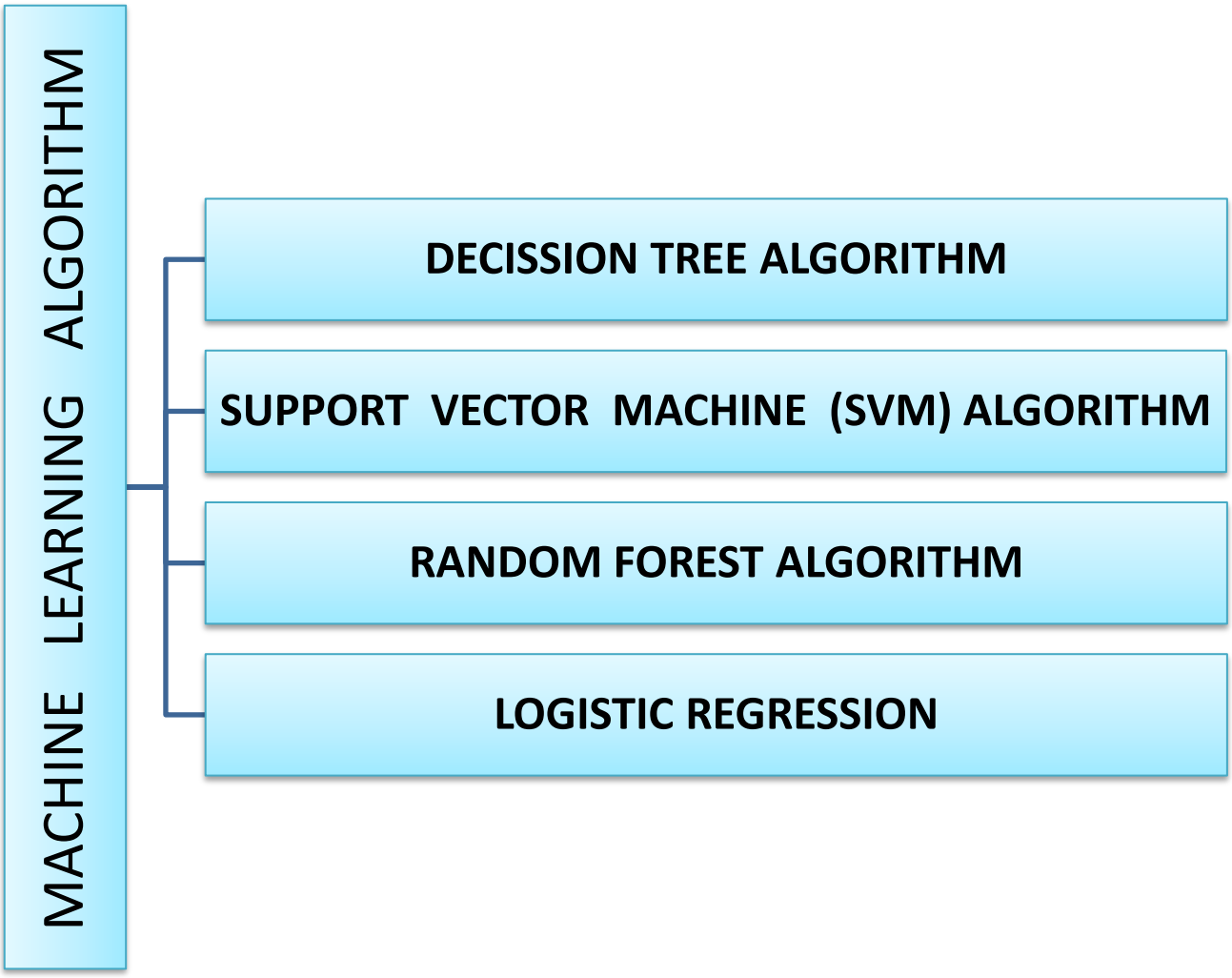
```
features_score = pd.DataFrame(fs.scores_)
features_pvalue = pd.DataFrame(np.round(fs.pvalues_,4))
features = pd.DataFrame(X.columns)
feature_score = pd.concat([features,features_score,features_pvalue],axis=1)
# Assign the column name
feature_score.columns = ["Input_Features","Score","P_Value"]
print(feature_score.nlargest(5,columns="Score"))
```

```
       Input_Features        Score  P_Value
7      SSLfinal_State  5409.473703      0.0
13      URL_of_Anchor  5005.109761      0.0
5       Prefix_Suffix   993.703780      0.0
6   having_Sub_Domain   562.376190      0.0
25        web_traffic   504.018152      0.0
```

STEP VI

6.MACHINE LEARNING ALGORITHMS: -

A Machine Learning Algorithm is defined as the methodology by which the Artificial Intelligence system trains the machine learning model by providing input data to the model for predicting the output. Machine Learning Algorithms consist of two processes - classification and regression. Our dataset is based on classification.



The Machine Learning Algorithm we have implemented are as follows –

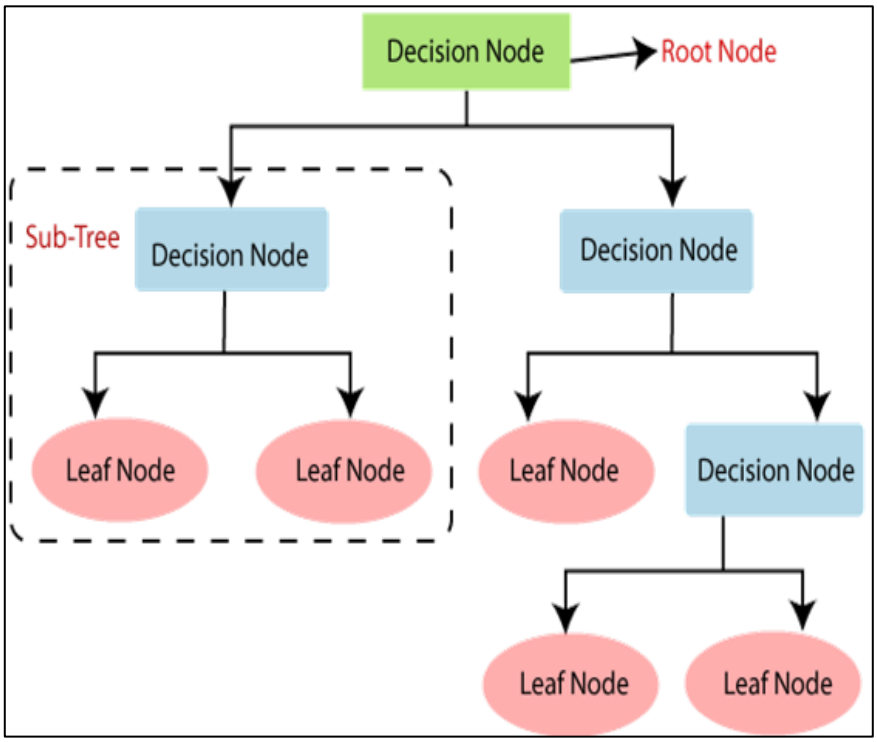
1. DECISION TREE ALGORITHM: -

This algorithm relies on a tree-structured Supervised learning technique where internal nodes denotes the features, branches represent the decision taking rules and each leaf node produces the outcome. The accuracy of Decision Tree Algorithm is 92%.

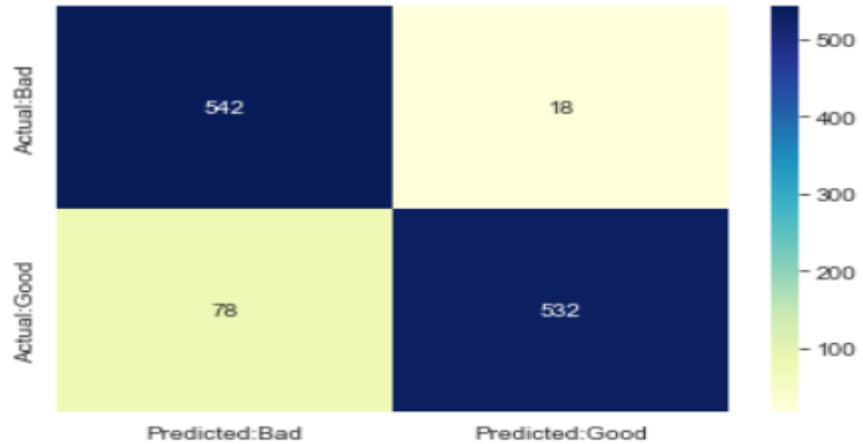
Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

Algorithm:

- **Step-1**: Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2**: Find the best attribute in the dataset using Attribute Selection Measure (ASM).
- **Step-3**: Divide the S into subsets that contains possible values for the best attributes.
- **Step-4**: Generate the decision tree node, which contains the best attribute.
- **Step-5**: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.




```
Training Accuracy : 0.915152810429579
Testing Accuracy : 0.9179487179487179

CLASSIFICATION REPORT

              precision    recall  f1-score   support

         Bad       0.87      0.97      0.92       560
        Good       0.97      0.87      0.92       610

    accuracy                           0.92      1170
   macro avg       0.92      0.92      0.92      1170
weighted avg       0.92      0.92      0.92      1170


CONFUSION MATRIX

: <matplotlib.axes._subplots.AxesSubplot at 0x29213504490>
```



2.SUPPORT VECTOR MACHINE (SVM) ALGORITHM-

This is a supervised machine learning algorithm where we need to plot each data item as a point in n-dimensional space where n gives the number of features. The accuracy of Support Vector Machine (SVM) Algorithm is 92%.

Algorithm:

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

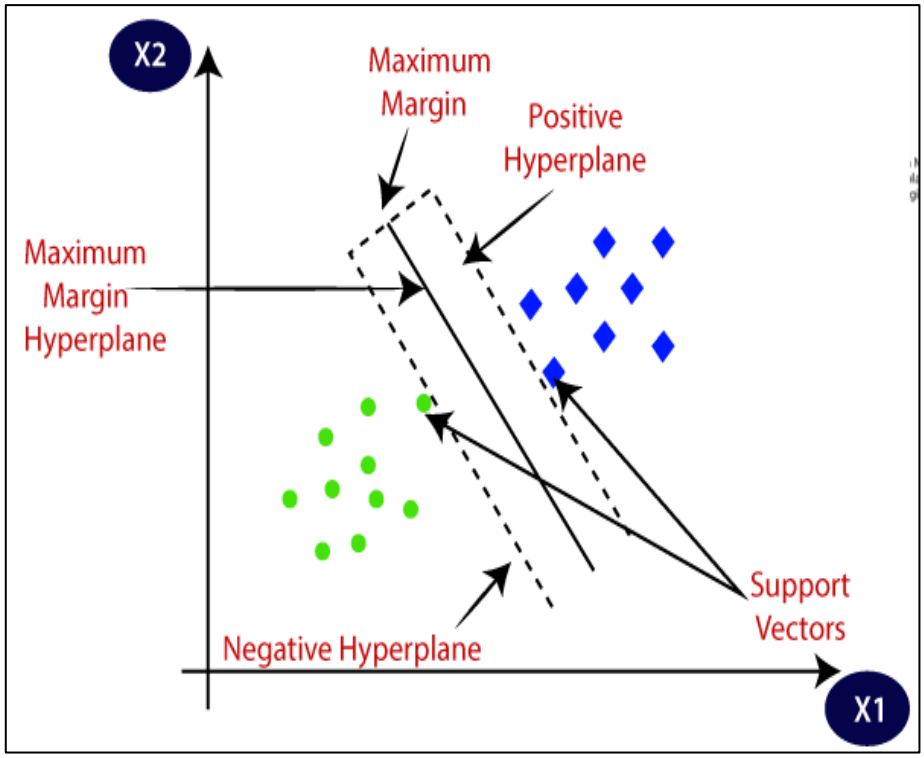
The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

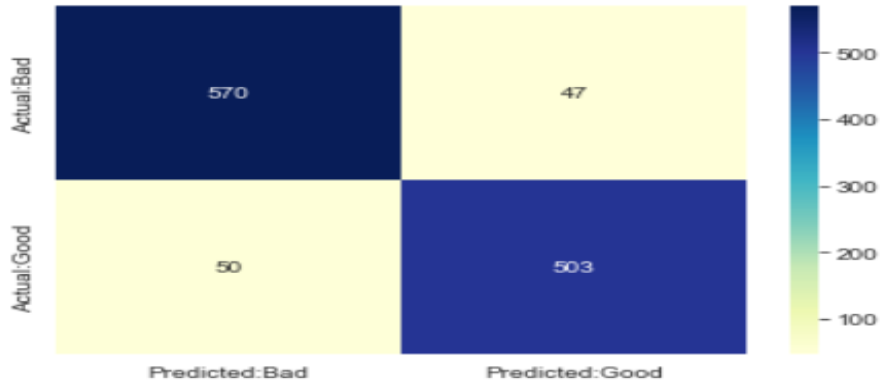
Types of SVM:

SVM can be of two types:

- Linear SVM: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- Non-linear SVM: Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.




```
Training Accuracy : 0.9258388544560804
Testing Accuracy : 0.917094017094017

CLASSIFICATION REPORT

                precision    recall  f1-score   support

           Bad       0.92      0.92      0.92       617
          Good       0.91      0.91      0.91       553

      accuracy                           0.92      1170
     macro avg       0.92      0.92      0.92      1170
  weighted avg       0.92      0.92      0.92      1170


CONFUSION MATRIX

<matplotlib.axes._subplots.AxesSubplot at 0x292136020d0>
```



3.**RANDOM FOREST ALGORITHM-**

Random Forest Algorithm is a popular machine learning algorithm based on ensemble learning, which involves combining multiple classifiers to solve a complex problem and also to enhance the performance of the model. The accuracy of Random Forest Algorithm is 93%.

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Algorithm:

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

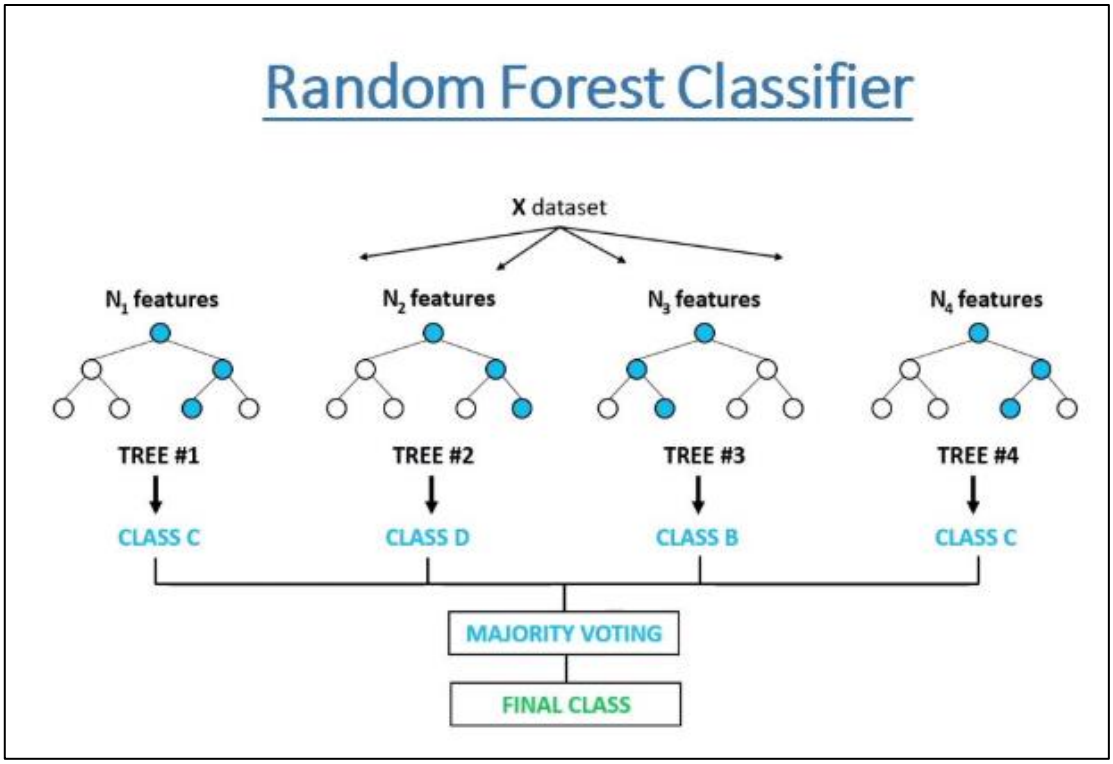
Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

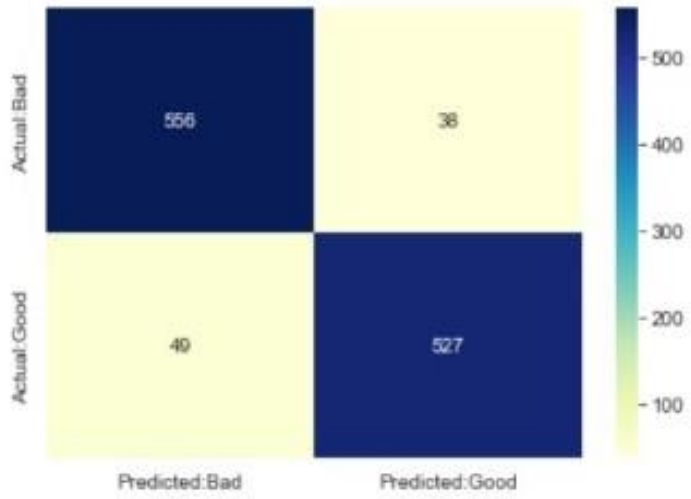
Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.



Bad	0.92	0.94	0.93	594
Good	0.93	0.91	0.92	576
accuracy			0.93	1170
macro avg	0.93	0.93	0.93	1170
weighted avg	0.93	0.93	0.93	1170

CONFUSION MATRIX

```
Out[172]: <matplotlib.axes._subplots.AxesSubplot at 0x2a428c2ea00>
```



4.LOGISTIC REGRESSION-

Logistic Regression is a Machine Learning algorithm based on the concept of probability. It analyse the data based on predictions of a categorical dependent variable. It is a classification model. The accuracy of Logistic Regression Algorithm is 92%.

Logistic Regression Equation:

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

- We know the equation of the straight line can be written as:

$$y= b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

- In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

$$\frac{y}{1-y}$$; 0 for y= 0, and infinity for y=1

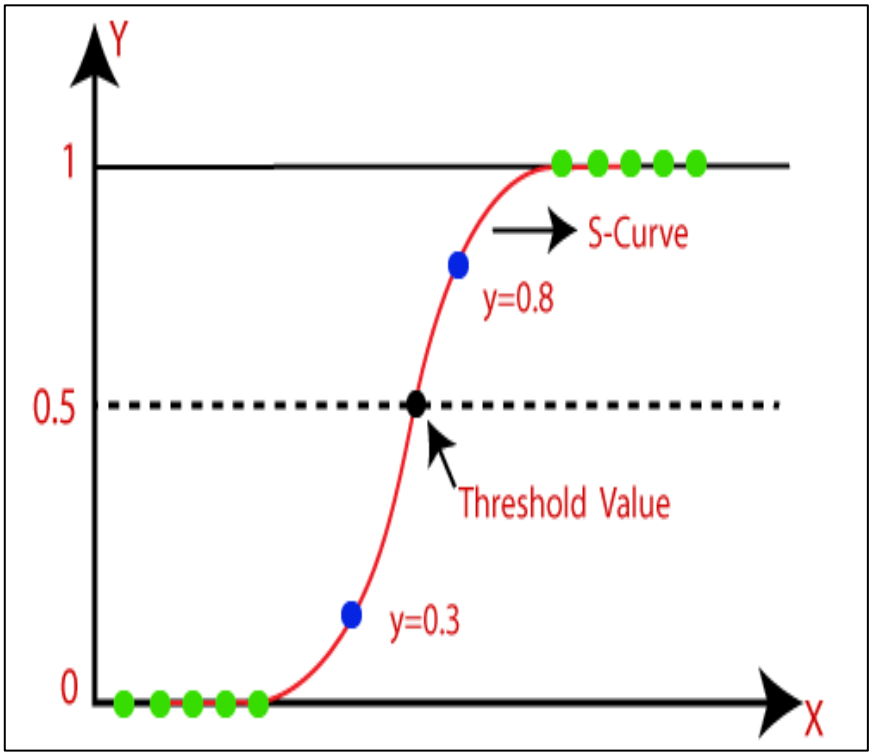
- But we need range between -[infinity] to +[infinity], then take logarithm of the equation it will become:

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

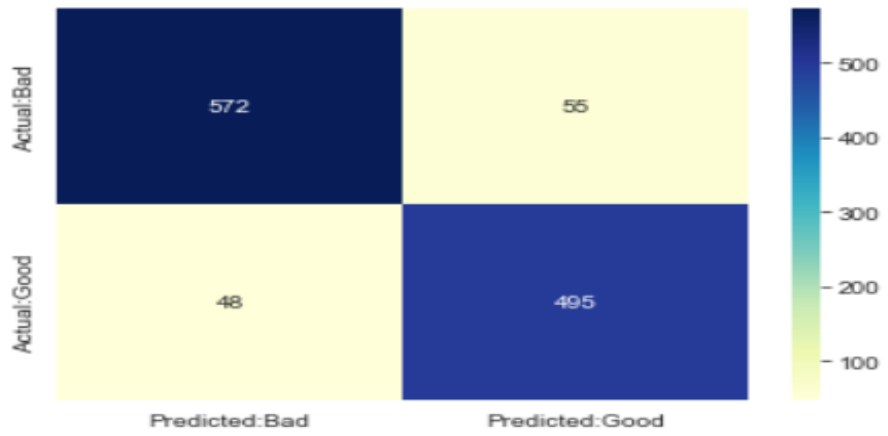
The above equation is the final equation for Logistic Regression.

Algorithm:

- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.
- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).




```
Training Accuracy : 0.9249839709339602
Testing Accuracy : 0.911965811965812

CLASSIFICATION REPORT

              precision    recall  f1-score   support

         Bad       0.92      0.91      0.92       627
        Good       0.90      0.91      0.91       543

    accuracy                           0.91      1170
   macro avg       0.91      0.91      0.91      1170
weighted avg       0.91      0.91      0.91      1170


CONFUSION MATRIX

<matplotlib.axes._subplots.AxesSubplot at 0x29213542c70>
```



SOFTWARE/ HARDWARE REQUIREMENTS

SOFTWARE REQUIREMENTS: -

- INTERNET
- PYTHON
- JUPYTER NOTEBOOK OR GOOGLE COLAB
- VISUAL STUDIO CODE

HARDWARE REQUIREMENTS: -

- PERSONAL COMPUTER OR LAPTOP

RESULT

After applying the Machine Learning Algorithms, we have trained and tested the model. We have performed model evaluation to determine which model gives the better performance. Our source code has produced the following result.

RESULTS

In [187]:
```
#creating dataframe
results = pd.DataFrame({ 'ML Model': ML_Model,
    'Train Accuracy': acc_train,
    'Test Accuracy': acc_test})
results
```

Out[187]:

	ML Model	Train Accuracy	Test Accuracy
0	Decision Tree	0.916	0.915
1	Random Forest	0.925	0.926
2	SVM	0.923	0.920
3	lr	0.921	0.923

In [188]:
```
#Sorting the dataframe on accuracy
results.sort_values(by=['Test Accuracy', 'Train Accuracy'], ascending=False)
```

Out[188]:

	ML Model	Train Accuracy	Test Accuracy
1	Random Forest	0.925	0.926
3	lr	0.921	0.923
2	SVM	0.923	0.920
0	Decision Tree	0.916	0.915

CONCLUSION

Our project work has presented phishing detection system based on machine learning. We have used 30 different features that distinguish phishing websites from legitimate websites. Our experiment results show 93% accuracy in detecting phishing websites using Random Forest Classifier. Working on this project is very knowledgeable and worth the effort. Through this project, one can know a lot about the phishing websites and how they are differentiated from legitimate ones.

We have also made a pickle file to create a website where users have to enter a URL and system based on our training will give the result whether it is a phishing or a legitimate website.

```
In [177]: # save XGBoost model to file
          import pickle
          pickle.dump(forest, open("SVM.pkl", "wb"))

In [178]: # load model from file
          loaded_model = pickle.load(open("SVM.pkl", "rb"))

In [179]: result = loaded_model.score(X_train, y_train)
          print(result)

          0.9296858303056209
```

We have taken data from the data set and checked manually whether it is giving correct and appropriate result.

```
In [181]: predict_phish = [[-1,1,1,1,-1,-1,-1,-1,-1,1,1,-1,1,-1,1,-1,-1,-1,0,1,1,1,1,-1,-1,-1,-1,1,1,-1]]
          predict_legitimate = [[1,0,-1,1,1,-1,1,1,-1,1,1,1,1,0,0,-1,1,1,0,-1,1,-1,1,-1,-1,0,-1,1,1,1]]
          loaded_model = pickle.load(open('SVM.pkl', 'rb'))
          #predict_bad = vectorizers.transform(predict_bad)
          # predict_good = vectorizer.transform(predict_good)
          result = loaded_model.predict(predict_phish)
          result2 = loaded_model.predict(predict_legitimate)
          print(result)
          if(result==1):
              print("THIS WEBSITE IS A LEGITIMATE WEBSITE.")
          else:
              print("THIS WEBSITE IS A PHISHING WEBSITE.")
          print("*"*30)
          print(result2)
          if (result2 ==1):
            print('THIS WEBSITE IS A LEGITIMATE WEBSITE.')
          else:
            print('THIS WEBSITE IS A PHISHING WEBSITE.')

          [-1]
          THIS WEBSITE IS A PHISHING WEBSITE.
          ******************************
          [1]
          THIS WEBSITE IS A LEGITIMATE WEBSITE.
```

First, we have checked for legitimate website.

Then we have checked for phishing websites.

LIST OF TABLES

- **CONFUSION MATRIX TABLE:**

Algorithms / Criteria	Decision Tree	Random Forest	SVM	Logistics Regression
True Positive	532	507	503	495
False Positive	78	47	50	48
True Negative	542	573	570	572
False Negative	18	43	47	55

Algorithms / Criteria	Decision Tree	Random Forest	Support vector machine	Logistics Regression
True Positive Rate	0.97%	0.92%	0.91%	0.90%
False Positive Rate	0.12%	0.075%	0.080%	0.077%
True Negative Rate	0.87%	0.92%	0.92%	0.92%
False Negative Rate	0.03%	0.08%	0.086%	0.1%
Accuracy	0.92%	0.93%	0.92%	0.91%

- **CLASSIFICATION REPORT:**

Algorithms / Criteria	Decision Tree	Random Forest	SVM	Logistics Regression
Bad Precision	0.87	0.92	0.91	0.92
Good Precision	0.96	0.93	0.93	0.93
Macro avg Precision	0.92	0.93	0.92	0.92
Weighted avg Precision	0.92	0.93	0.92	0.92
Bad recall	0.96	0.94	0.94	0.93
Good Recall	0.87	0.91	0.90	0.91
Macro avg recall	0.92	0.93	0.92	0.92
Weighted avg recall	0.92	0.93	0.92	0.92
Bad F1 Score	0.91	0.93	0.92	0.92
Good F1 Score	0.92	0.92	0.92	0.92
Accuracy F1 Score	0.92	0.93	0.92	0.92
Macro avg F1 Score	0.92	0.93	0.92	0.92
Weighted avg F1 Score	0.92	0.93	0.92	0.92
Bad support	548	594	578	593
Good support	622	576	583	577
Accuracy support	1170	1170	1170	1170
Macro Avg support	1170	1170	1170	1170
Weighted Avg support	1170	1170	1170	1170

- **RESULTANT TABLE:**

Index	ML Model	Train Accuracy	Test Accuracy
0	Random Forest	0.950	0.926
1	Logistics Regression	0.921	0.923
2	SVM	0.923	0.920
3	Decision Tree	0.916	0.915

COMPARATIVE STUDY

1. Ashit Kumar Dutta, Department of Computer Science and Information System, College of Applied Sciences, Almaarefa University, Riyadh, Saudi Arabia Published on October 11, 2021.

- This paper has produced only 90% accuracy whereas our code has generated an accuracy of 93 %.Also this paper has extracted 14 different features whereas our source code has extracted 30 features.

2. Ankit Kumar Jain, B. B. Gupta, National Institute of Technology Kurukshetra, Kurukshetra, India Published online on 26 December 2019.

This paper has implemented 19 features whereas our source code has worked with 30 features to produce the output.

FUTURE SCOPE

- We have created a website. However this project can be taken further by creating a browser extensions by developing a GUI.
- More machine learning algorithms can be applied for better comparison and result.
- We can apply deep learning techniques also to this project.
- In future, more features can be added to improve the accuracy of the proposed phishing detection system. Furthermore, other machine learning techniques can be used to increase the efficiency of the proposed system.

REFERENCES

- Published by Vaibhav Patil, Pritesh Thakkar Prof. S. P. Godse, Tushar Bhat and Chirag Shah of "Detection and Prevention of Phishing Websites using Machine Learning Approach", Dept. of Computer Engineering in Sinhgad Academy of Engineering Pune, India 2018.

- Ankit Kumar Jain and B. B. Gupta, "Phishing Detection Analysis of Visual Similarity Based Approaches", Hindawi 2017.

- Jian Mao, Pei Li, Kun Li, Tao Wei, and Zhenkai Liang, "Bait Alarm Detecting Phishing Sites Using Similarity in Fundamental Visual Features", INCS 2013.

- Y. Zhang, J. I. Hong, and L. F. Cranor. Cantina: a content-based approach to detecting phishing web sites. In WWW '07: Proceedings of the 16th international conference on World Wide Web, pages 639–648, New York, NY, USA, 2007. ACM.

- Haijun Zhang, Gang Liu, Tommy W. S. Chow, and Wenyin Liu, "Textual and Visual Content-Based Anti-Phishing A Bayesian Approach", IEEE 2011.

SOURCE CODE

We have created a Google drive and for the source code.

The Google drive link of our source code is as following: -

https://drive.google.com/drive/folders/1INENeNSsy3H59A97j4mSf4Xnrv_CdJ3e?usp=sharing

The github link of our source code is as following: -

https://github.com/NitishNaskar/Phishing-Website-Detection