

# CHAPTER 1

## INTRODUCTION

### 1.1 INTRODUCTION TO THE SYSTEM

In modern era Phishing becomes a main area of concern for security researchers due to the fact it is not tough to create the fake internet site which looks so close to legitimate internet site. Experts can discover fake web sites however not all the customers can discover the fake website and such customers become the victim of phishing attack.

Main purpose of the attacker is to steal banks account credentials. How hackers do their work, they send you just spam mail. In this mail though they will say that this email is meant to inform you that your university network password will expire in 24 hours and they have provided you to update the password and login when we click on that link we will redirect to that page which is a hacker server and they will steal your data everything which is online.

In our project we have to predict phishing websites whether they are good uniform resource locators (URLs) or bad URLs. The set of phishing URLs are gathered from open source service called Phish Tank. Benign URLs (uniform resource locator) with zero malicious detection were classified as benign and URLs with no less than eight detection were classified as malicious. It is being labeled as '0' and Phishing URL is being labeled as '1'. We study several machine learning algorithms for analysis of the characteristic in order to get a good understanding of the construction of the URLs that expand phishing.

Phishing attacks are getting a success because of lack of consumer awareness. Since phishing attack exploits the weaknesses found in customers, it's far very tough to mitigate them however it may be very vital to enhance phishing detection strategies. The general technique to discover phishing web sites through updating blacklisted URLs, Internet Protocol (IP) to the antivirus database which is also recognized as "blacklist" technique. To evade blacklists attackers make use of innovative techniques to fool customers through modifying the URL to appear valid via obfuscation and lots of other easy techniques such as: fast-flux, in which proxies are automatically generated to host the web-page; algorithmic era of recent URLs etc. To entice human beings, Phisher sends "spoofed" mails to several human beings as possible. When such emails are opened, the customers generally tend to redirect to the spoofed internet site. The internet site intuitively asks you to run a software program or download a document while you're not waiting to do so. The internet site tells you that your device is infested with malware or that your browser extensions or software program are out of date. Malicious URLs on the website could be easily recognized by examining them through machine learning

techniques.

To avoid getting phished, people should have an understanding of phishing websites and how they look if the person is using an online browser. So, the high-end companies can even blacklist phishing websites or detect phishing in their early arrival, using machine learning and deep neural network algorithm to build a model that can classify URLs as phishing. Machine learning technique is proven to be efficient than the other technique.

## **1.2 PROBLEM STATEMENT**

The major trouble is that phishing technique is bad accuracy and low adaptability to new phishing links. We plan to apply machine learning to overcome this limitation through imposing some classification algorithms and evaluating the overall performance of these algorithms on our dataset. We have decided on the Random Forest method because of its excellent performance in classification but random forest and decision tree are not good with nlp data.

## **1.3 OBJECTIVE OF THE PROJECT**

A phishing internet site is the most common social engineering approach that mimics trustful URLs and web pages. This project aims to predict phishing websites whether are good URLs or bad URLs. Both phishing and benign URLs of websites are collected to form a dataset and from them required URLs and these projects aimed functions are extracted. We have used so many different libraries and algorithms like Logistic Regression, NumPy, pandas, MultinomialNB, RegexpTokenizer many more. We gather data to create a dataset of malicious links and curate it for the ML model. The performance level of every model is measured and compared.

## **1.4 METHODOLOGY**

In this project, we have predicted phishing websites whether they are good URLs or bad URLs. Before performing a code, we have done some surfing and found some datasets. We have collected the phishing and estimated websites from open source and developed a bit of code to extract the feature. As there are different types of URLs like spam, benign, phishing, malware etc from legitimate URLs. Benign has around 35,000 URLs, so out of them; we have used 5000 URL randomly. We have examined and pre-process the dataset and divide the datasets

into training and test sets. Then we combine all the dataset into one frame. The data was containing more than 5lakhs unique approach and there were two columns which was further categories into Good and Bad.

Further we have done some classification problems and vectorize our URLs by using Count Vectorizer and tokenizer. We have used different libraries for different functions such as for visualizing most common words in good and badURLs and turn URLs into data frame. After that we have create the model and splitting the data. Then we have import links to prediction and deploy the model. The metric that we used is accuracy which is a simple one. We have compared the accuracies of the training datasets and the best one is declared.

## **1.5 ORGANIZATION**

In our project we have used a CSV file and import many Python libraries for different purposes. We have taken two algorithms which is Logistic Regression and MultinomialNB. Logistic Regression will predict the links are good or not and MultinomialNB work well with nlp data (natural language process). Then we have used some classification problems by using CountVectorizer and tokenizer. We have used some another visualization. We can show that what is the hidden link in the phishing site which will redirect to another server. Then we have networkx it is creating a data structure, dynamic function and more.

We are combining three datasets which we collected from several sites then we combine this dataset into one frame. The usability of this dataset is 10.0 which means very good. The data size is approx 30 mb. The data contains more than 5lakhs unique approach. The label column means that its prediction column in which there were two categories first is good and second is bad. After that we have checked the imbalanced of target column. Now we have a data, we convertURLs into vector form.

We have used regular expression tokenizers which divide the string using regular expression. So, in our code we are just splitting only alphabets and some URLs have numbers, dots , slash etc which are not important our data. So we only gather the string and simultaneously we have transformed this in all the rows. After converting into words we used snowball it's an nltk API (natural language toolkit) which is used to string words.

It will remove all the English words and create some root words. Root words means that it will combine the common words like pictures, photos for this two words it will create the one word. Phishing data text streamer is equal to all the holder that list is words lists are converted into streamers. Then we join all the lists words into single sentence. We also use word cloud. In our code we have used this to convert most repeated word into the word cloud form. Then we use chrome webdriver. This will create a newwindow of that chrome. So to this newchrome we will pass that link. Then by using BeautifulSoup, we gather the all html code from its page source and it is getting all the anchor tags. So we will get all the hidden link which will hacker use to redirect any users to this server and we create a data frame of this links. So it will give a two links: first is what we passed to this and second what we are getting from this link. Logistic regression object and we fit it by trainX, trainY. After that we checked the scoreand we are getting very good score which is 90.96. After that we just created theconfusion matrix to see the actual prediction and normal prediction. Using Logistic Regression we are creating a pipeline. Then we are saving this pipeline model using pickle and we check the accuracy of it and it is giving very good accuracy.

## **CHAPTER 2**

### **LITERATURE SURVEY**

#### **1. Chunlin Liu, Bo Lang : Finding effective type for malicious URL detection**

**: In ACM,2018**

Chunlin et al. proposed method that primarily consciousness on individual frequency features. In this they've mixed statistical evaluation of URL with machine learning method to get end result this is more accurate for category of malicious URLs. Also they've compared six machine learning algorithms to confirm the effectiveness of proposed algorithm which offers 99.7% precision with fake positive rate much less than 0.4%.

#### **2. Ankit Kumar Jain, B. B. Gupta : Towards detection of phishing websites on client-side using machine learning based approach :In Springer Science+Business Media, LLC, part of Springer Nature 2017**

Gupta et al. [11] put forward a novel anti phishing method that extracts features from client-side only. Suggested method is fast and reliable as it is now no longer depending on third party however it extracts capabilities only from URL and source code. In this paper, they have reached 99.09% of overall detection accuracy for phishing website. This paper has concluded that this method has limitation as it can detect website written in HTML. Non-HTML website can't detect through this method.

#### **3. A Prior-based transfer learning techniques for the Phishing Detection:-**

A logistic regression is the basis of a concern based transferrable learning method, which is supplied right here for our classification of statistical machine learning. It is used for the detection of the phishing web sites relying on our decided on traits of the URLs. Due to the divergence in the allotment of the

capabilities in the distinct phishing areas, several models are proposed for distinctive regions. It is sort of impractical to accumulate enough information from a new region to repair the detection model and use the transfer learning algorithm for adjusting the present version. A suitable manner for phishing detection is to apply our URL based technique. To deal with all of the conditions of failure of detecting traits, we need to undertake the shifting method to generate a extra effective version.

#### **4. Ahmad Abunadi, AnazidaZainal ,OluwatobiAkanb: Feature Extraction Process: A Phishing Detection Approach :In IEEE,2013**

Ahmad et al. [17] proposed three new capabilities to enhance accuracy rate for phishing internet site detection. In this paper, Author used both type of capabilities as usually recognized and new features for category of phishing and non-phishing site. At the cease author has concluded this work can be improve by the usage of this novel capabilities with decision tree machine learning classifiers.

#### **5. Sahingoz, O.K., Buber, E., Demir, O. and Diri, B., 2019. Machine learning primarily based totally phishing detection from URLs. Expert Systems with Applications, 117, pp.345- 357.**

Attempts to detect phishing site the use of URL for preventing customer sensitive information. Computer customers fall for phishing because of the 5 major reasons: -

- Users don't know, which internet pages may be trusted,
- Users don't see the entire address of the internet page, because of the redirection or hidden URLs,
- Users don't have plenty time for consulting the URL, or by chance enter a few internet pages.

# **CHAPTER 3**

## **SYSTEM ANALYSIS**

### **1. EXISTING SYSTEM**

- Malicious Web sites are the basis of most of the criminal activities over the internet.
- The dangers that arise due to the malicious sites are enormous and the end-users must be prohibited from visiting such sites.
- The users should prohibit themselves from clicking on such Uniform Resource Locator (URL).

### **2. DISADVANTAGES**

- Low Accuracy Due to Training Loss
- Many Website features not included for the consideration

### **3. PROPOSED SYSTEM**

- Collect dataset containing phishing and legitimate websites from the open-source platforms.
- Write a code to extract the required features from the URL database.
- Analyze and preprocess the dataset by NLTK Tools.
- Divide the dataset into training and testing sets.
- Run selected Logistic Regression and MultinomialNB on the dataset.
- Compare the obtained results for trained models and specify which is better.

### **4. ADVANTAGES**

- Provide clear idea about the effective level of each classifier on phishing email detection
- High level of accuracy by taking the advantages of classifiers many
- Fast in classification process fast, less consuming memory, high accuracy, Evolving with time, online working.

# CHAPTER 4

## SYSTEM REQUIREMENTS

### 1. HARDWARE REQUIREMENTS

- Processor : Intel i3
- RAM : 4 GB
- Hard Disk : 160 GB

### 2. SOFTWARE REQUIREMENTS

1. **Python:** The code is written in Python, so you need to have Python installed on your system.
2. **Pandas:** The code uses the pandas library for data manipulation and analysis.
3. **NumPy:** The NumPy library is used for numerical computations.
4. **NLTK (Natural Language Toolkit):** This library is used for stemming. Make sure to install it using ``pip install nltk``.
5. **scikit-learn:** The code utilizes scikit-learn for text vectorization (Count Vectorizer) and cosine similarity calculations. Install it using ``pip install scikit-learn``.
6. **Seaborn:** You'll need to have the Seaborn library installed. You can install it using the following command: `pip install seaborn`.
7. **Matplotlib:** You'll need to have the Matplotlib library installed. You can install it using the following command: `pip install matplotlib`.

You might also need to install other dependencies that these libraries rely on.

### 3. ALGORITHMS

- **CLASSIFICATION**

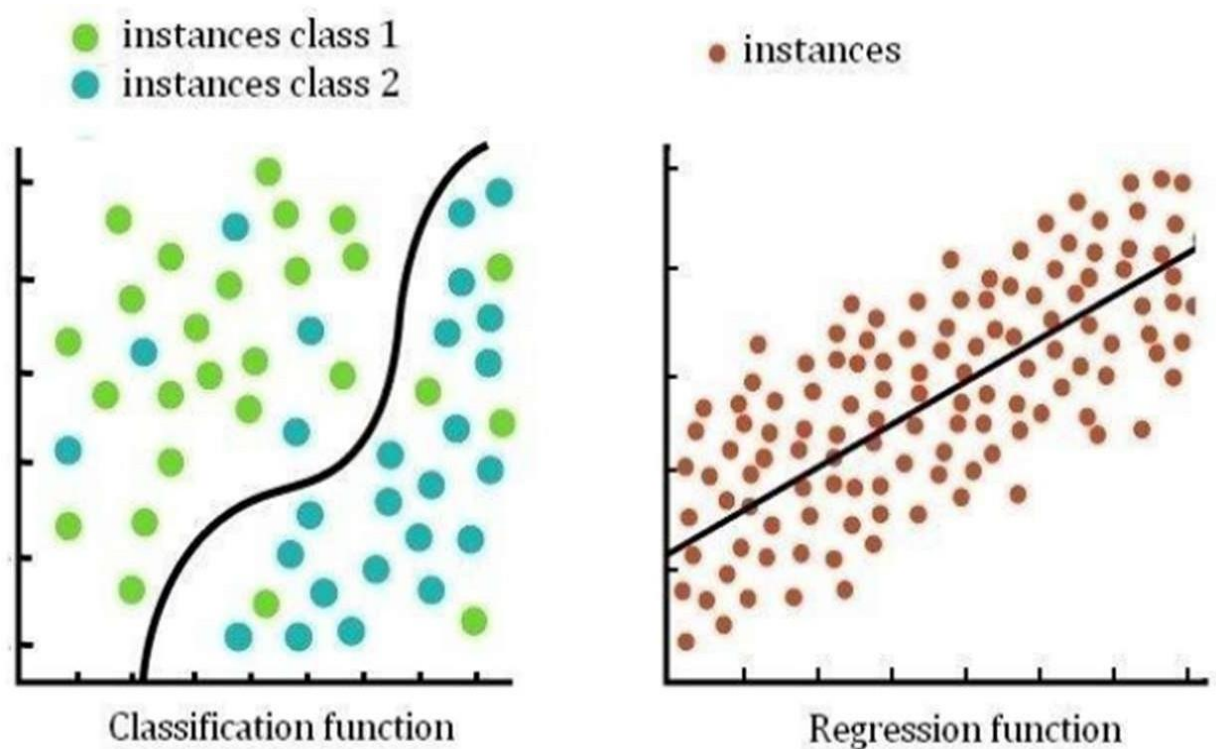
It helps find things that we can search by keywords, but it actually helps you find our own invention that's very close to our own. An area that is grouped in subject areas called classes and subclasses. Used to classify characteristics of invention. The type predictive modelling is the challenges of approximating the mapping characteristic from enter variables to discrete output variables.

- **REGRESSION**

It is a supervised learning method which enables in finding the correlation among variables and allows us to be looking ahead to the non-stop output variable primarily based totally at

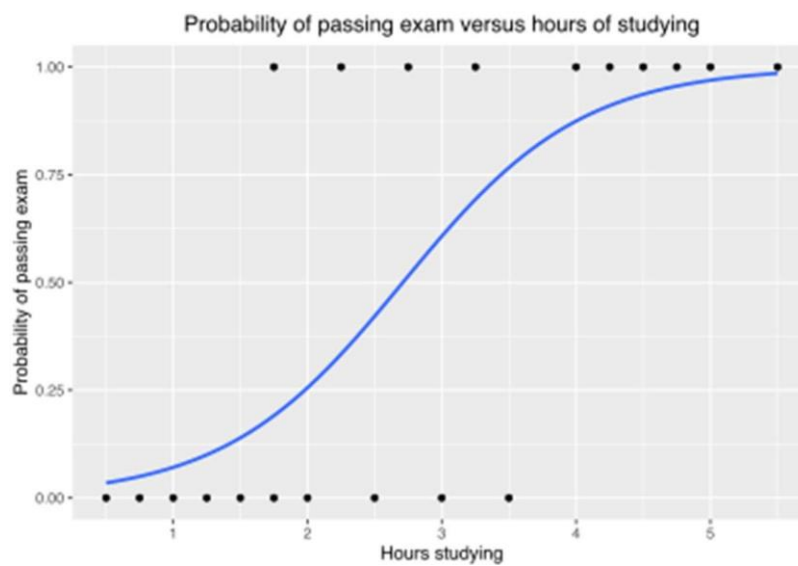


the simplest or greater predictor variables. It is specifically used for prediction, forecasting, time collection modeling, and figuring out the causal-impact relationship among variables. In Regression, we plot a graph among the variables which satisfactory suits the given data points, the use of this plot, the machine learning version could make predictions about the data.



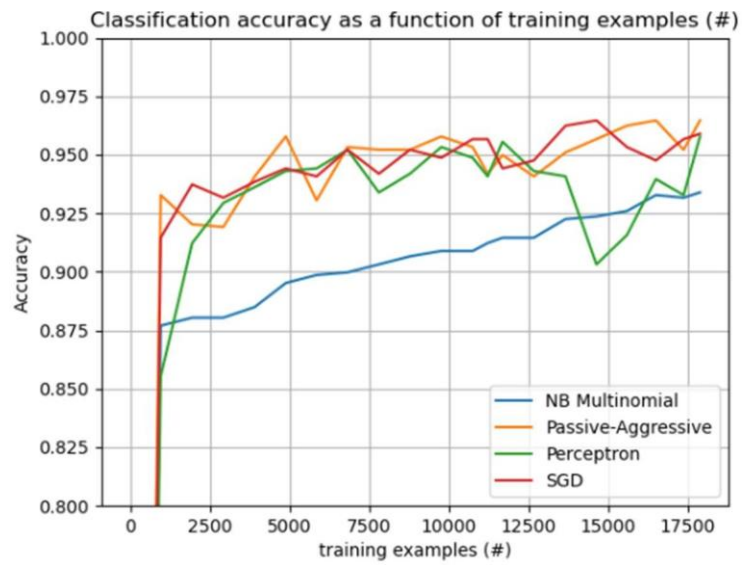
- **LOGISTIC REGRESSION**

Logistic regression is a statistical analysis method to predict a binary outcome, such as yes or no, based on prior observations of a data set. A logistic regression model predicts a dependent data variable by analysing the relationship between one or more existing independent variables. These sorts of estimation allow that we are expecting the probability of an occasion taking place or a desire being made.



- **MULTINOMIALNB**

Multinomial Naïve Bayes algorithm is a probabilistic learning method that is mostly used in Natural Language Processing (NLP). For example: As in this definition how many times the word is coming or what is the frequency of word in this definition and it is a machine learning method. Naïve Bayes is also called conditional probability in the world of statistics. Multinomial is described discrete frequency counts in other words word counts or something like that.



## CHAPTER 5

### SYSTEM DESIGN

#### 1. SYSTEM ARCHITECTURE:

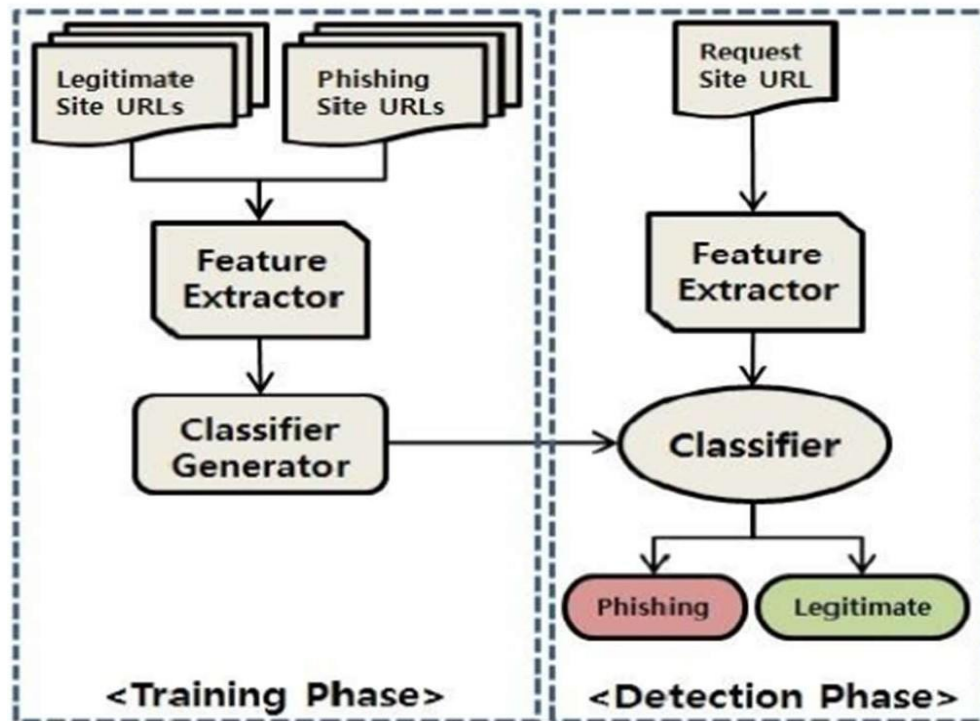
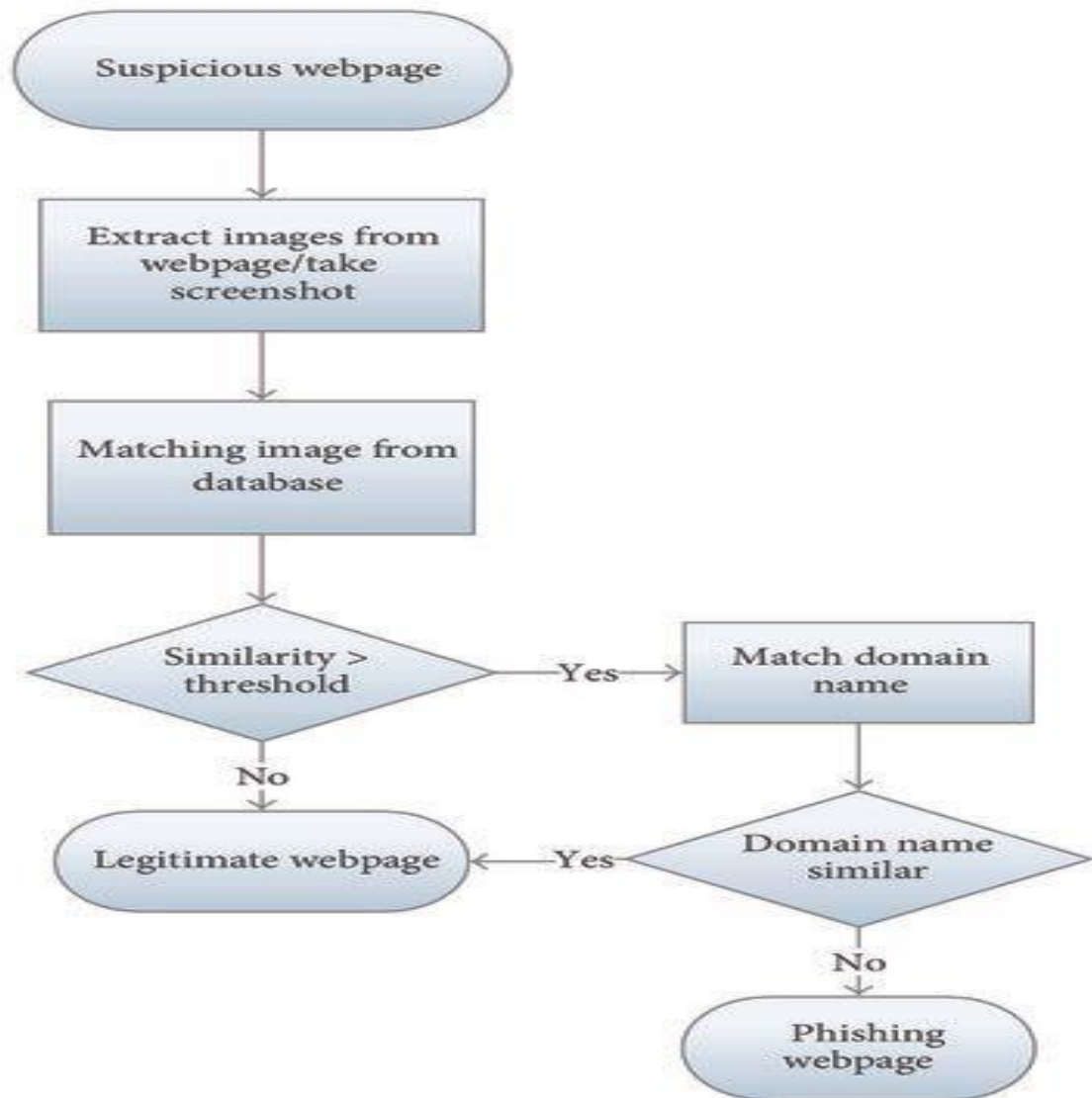


Figure 5.1. System Architecture

#### 2. DATA FLOW DIAGRAM:

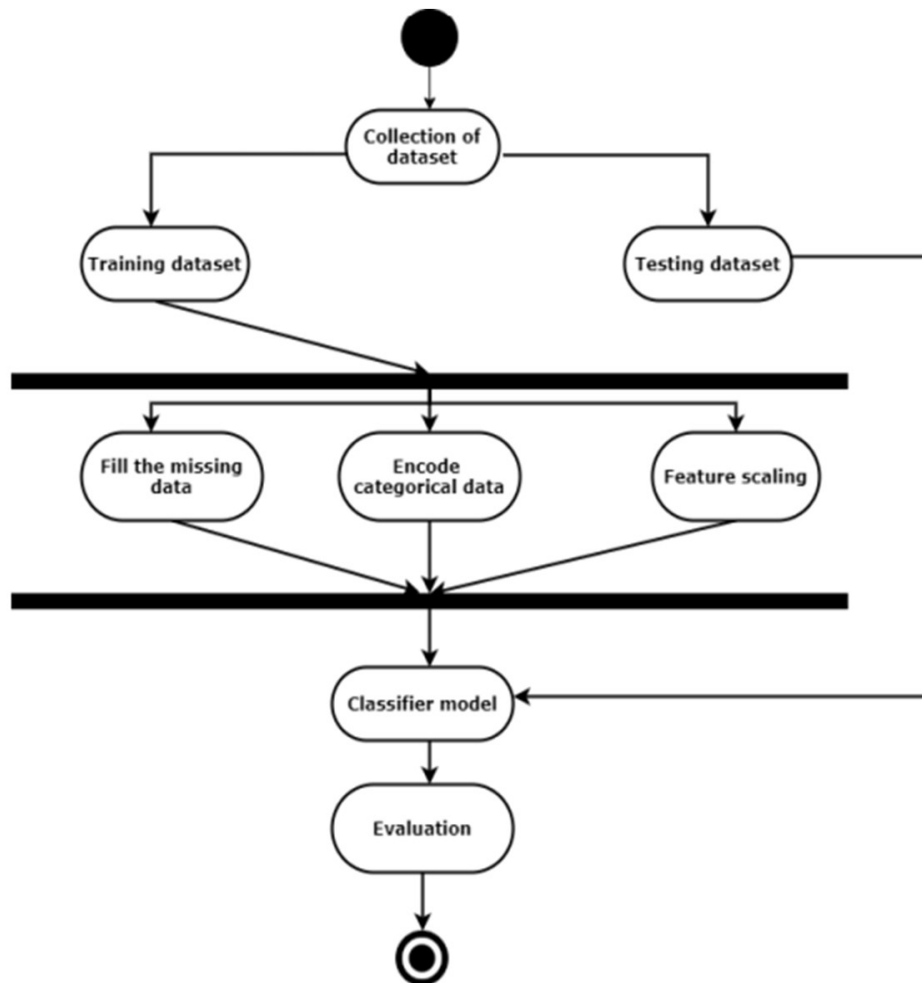
The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.



**Figure 5.2. Data Flow Diagram**

### 3. UML DIAGRAMS

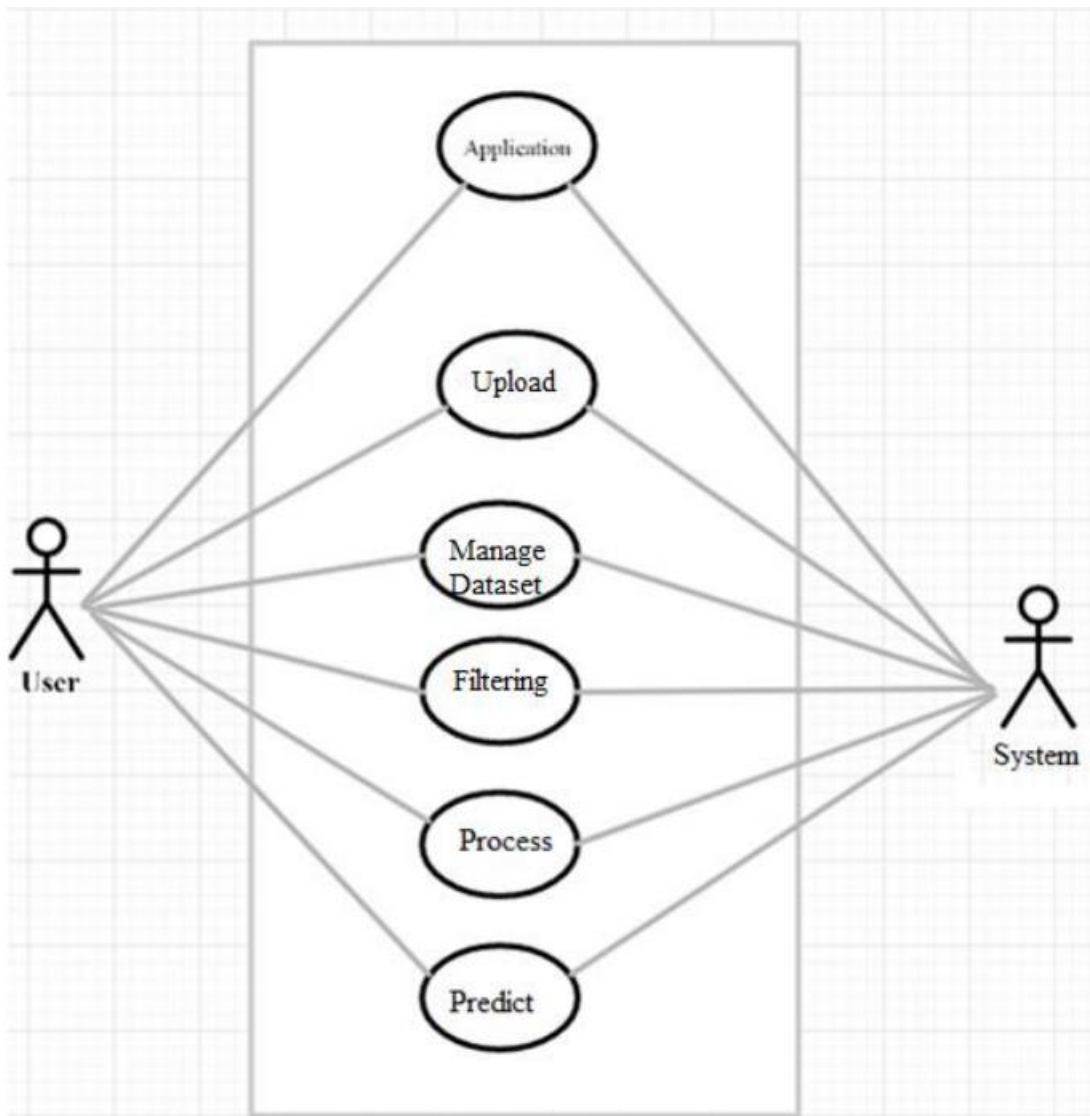
4. Activity diagram is a behavioural diagram. The fig 5.3 shows the activity diagram of the system. It depicts the control flow from a start point to an end point showing various paths which exists during the execution of the activity.



*Figure 5.3 UML Activity diagram of Detecting Phishing Websites*

## 5.USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



**Figure 5.4 Use Case Diagram**

## 6. SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

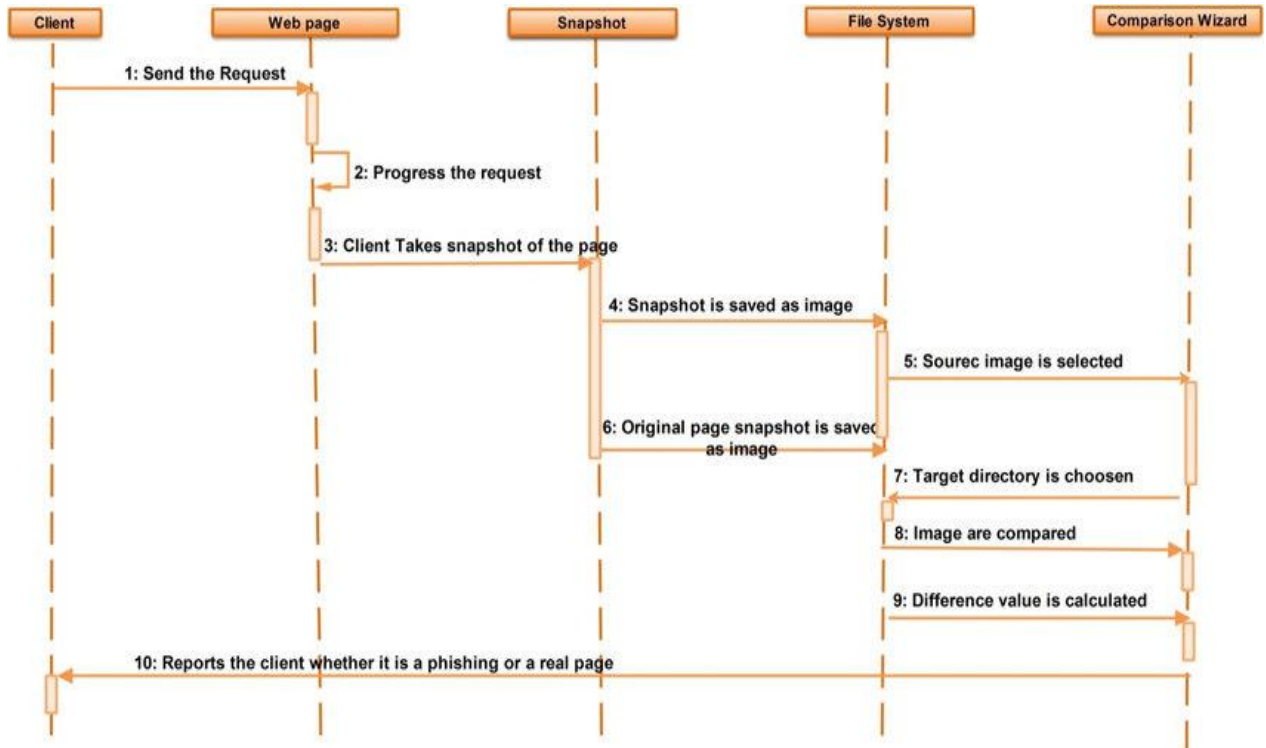


Figure 5.5 Sequence Diagram



## CHAPTER 6

### IMPLEMENTATION

#### ➤ **MODULES:**

- ❖ Data Loading
- ❖ Class Distribution Visualization
- ❖ Preprocessing
- ❖ Visualization
- ❖ Creating Models
- ❖ Sklearn Pipeline
- ❖ Testing the Model

#### ➤ **MODULES DESCRIPTION:**

##### **1. Data Loading:**

1. The code loads a dataset from a CSV file called 'phishing\_site\_urls.csv' using pandas.
2. It displays the first few rows of the dataset using the head() function.
3. It displays the last few rows of the dataset using the tail() function.
4. It provides information about the dataset using the info() function.
5. It checks for missing values in the dataset using the isnull().sum() function.

##### **2. Class Distribution Visualization:**

The code creates a bar plot to visualize the distribution of classes in the dataset (e.g., 'bad' and 'good') using the value counts() function from pandas and the bar() function from Plotly Express.

##### **3. Preprocessing:**

- **Tokenization:** The code uses a regular expression tokenizer (Regexp Tokenizer) from the Natural Language Toolkit (NLTK) to split the URLs into words, keeping only alphabetical characters.
- **Stemming:** The code applies stemming using the Snowball stemmer (Snowball Stemmer) from NLTK to reduce words to their root form.
- **Joining Words:** The code joins the stemmed words back into a single string.

##### **4. Visualization:**

- **Word Cloud:** The code defines a function (plot word cloud) to generate a word cloud

visualization of important words from the URL text. It uses the Word Cloud and Image Color Generator classes from the Word cloud library.

- The code creates word clouds for URLs labeled as 'bad' and 'good', using the text sent column.

## 5. Creating Models:

- **Count Vectorizer:** The code creates a Count Vectorizer object to convert the text data into a matrix of token counts. It uses the fit transform() function to transform the preprocessed text data into a feature matrix.
- **Logistic Regression:** The code creates a logistic regression model (Logistic Regression) and trains it on the training data. It then evaluates the model's accuracy on the testing data and displays a confusion matrix and classification report.
- **Multinomial Naive Bayes:** The code creates a multinomial Naive Bayes model (MultinomialNB) and trains it on the training data. It evaluates the model's accuracy on the testing data and displays a confusion matrix and classification report.

## 6. Sklearn Pipeline:

- The code creates a pipeline using make pipeline from scikit-learn, combining the Count Vectorizer and Logistic Regression steps into a single entity.
- The pipeline is trained and evaluated on the URL and label data, and the accuracy, confusion matrix, and classification report are displayed.
- The trained pipeline is serialized and saved to a file using pickle.

## 7. Testing the Model:

1. The code loads the saved pipeline model using pickles.load().
2. It provides some sample URLs categorized as 'bad' and 'good' and predicts their labels using the loaded model.

## ➤ METHODS OF IMPLEMENTATION:

Below is the Procedure to implement the algorithm

Import all necessary libraries

```

import pandas as pd # use for data manipulation and analysis
import numpy as np # use for multi-dimensional array and matrix

import seaborn as sns # use for high-level interface for drawing attractive and informative statistical graphics
import matplotlib.pyplot as plt # It provides an object-oriented API for embedding plots into applications
%matplotlib inline
# It sets the backend of matplotlib to the 'inline' backend:
import plotly.express as px
import time # calculate time

from sklearn.linear_model import LogisticRegression # algo use to predict good or bad
from sklearn.naive_bayes import MultinomialNB # nlp algo use to predict good or bad

from sklearn.model_selection import train_test_split # splitting the data between feature and target
from sklearn.metrics import classification_report # gives whole report about metrics (e.g, recall, precision, f1_score, c_m)
from sklearn.metrics import confusion_matrix # gives info about actual and predict
from nltk.tokenize import RegexpTokenizer # regexp tokenizers use to split words from text
from nltk.stem.snowball import SnowballStemmer # stemmes words
from sklearn.feature_extraction.text import CountVectorizer # create sparse matrix of words using regextokenizes
from sklearn.pipeline import make_pipeline # use for combining all prerocessors techniuges and algos

from PIL import Image # getting images in notebook
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator# creates words colud

from bs4 import BeautifulSoup # use for scraping the data from website
from selenium import webdriver # use for automation chrome
import networkx as nx # for the creation, manipulation, and study of the structure, dynamics, and functions of complex net

import pickle# use to dump model

import warnings # ignores pink warnings
warnings.filterwarnings('ignore')

```

```

# Loading the dataset
phish_data = pd.read_csv('phishing_site_urls.csv')

```

```
phish_data.head()
```

	URL	Label
0	nobell.it/70ffb52d079109dca5664cce6f317373782/...	bad
1	www.dghjdjgf.com/paypal.co.uk/cycgi-bin/webscr...	bad
2	serviciosbys.com/paypal.cgi.bin.get-into.herf....	bad
3	mail.printakid.com/www.online.americanexpress....	bad
4	thewhiskeydregs.com/wp-content/themes/widescre...	bad

```
phish_data.tail()
```

	URL	Label
549341	23.227.196.215/	bad
549342	apple-checker.org/	bad
549343	apple-iclods.org/	bad
549344	apple-uptoday.org/	bad
549345	apple-search.info	bad

```
phish_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 549346 entries, 0 to 549345
Data columns (total 2 columns):
#   Column  Non-Null Count  Dtype
---  -
0    URL      549346 non-null   object
1    Label    549346 non-null   object
dtypes: object(2)
memory usage: 8.4+ MB
```

```
phish_data.isnull().sum() # there is no missing values
```

```
URL      0
Label    0
dtype: int64
```

Since it is classification problem, so checking whether the classes are balanced or imbalances

```
#create a dataframe of classes counts
label_counts = pd.DataFrame(phish_data.Label.value_counts())
```

```
#visualizing target_col
fig = px.bar(label_counts, x=label_counts.index, y=label_counts.Label)
fig.show()
```

## Preprocessing -

Now we have to vectorize our URLs. We used CountVectorizer and gather words using tokenizer, since there are words in urls that are more important than other words e.g 'virus', '.exe', '.dat' etc. Lets convert the URLs into a vector form.

## RegexpTokenizer -

A tokenizer that splits a string using a regular expression, which matches either the tokens or the separators between tokens.

```
tokenizer = RegexpTokenizer(r'[A-Za-z]+') # to getting alpha only
```

```
phish_data.URL[0]
```

```
'nobell.it/70ffb52d079109dca5664cce6f317373782/login.SkyPe.com/en/cgi-bin/verification/login/70ffb52d079109dca5664cce6f317373/index.php?cmd=_profile-ach&outdated_page_tmpl=p/gen/failed-to-load&nav=0.5.1&login_access=1322408526'
```

```
# this will be pull letter which matches to expression  
tokenizer.tokenize(phish_data.URL[0]) # using first row
```

```
['nobell',  
'it',  
'ffb',  
'd',  
'dca',  
'cce',  
'f',  
'login',  
'SkyPe',  
'com',  
'en',  
'cgi',  
'bin',  
'verification',  
'login',  
'ffb',  
'd',  
'dca',  
'cce',  
'f',  
'index',  
'php',  
'cmd',  
'profile',  
'ach',  
'outdated',  
'page',  
'tmpl',  
'p',  
'gen',  
'failed',  
'to',  
'load',  
'nav',  
'login',  
'access']
```

```
print('Getting words tokenized ...')
t0= time.perf_counter()
phish_data['text_tokenized'] = phish_data.URL.map(lambda t: tokenizer.tokenize(t)) # doing with all rows
t1 = time.perf_counter() - t0
print('Time taken',t1 , 'sec')
```

Getting words tokenized ...  
Time taken 2.218552499999987 sec

```
phish_data.sample(5)
```

	URL	Label	text_tokenized
150157	blixtech.com/stories/0WnMn8aFJ4Cs	good	[blixtech, com, stories, WnMn, aFJ, Cs]
27886	welchmediadevelopers.com/wp-content/themes/twe...	bad	[welchmediadevelopers, com, wp, content, theme...
467412	wwooofjapan.com/main/index.php?option=com_conte...	good	[wwooofjapan, com, main, index, php, option, co...
183951	entertainment.howstuffworks.com/kid-nichols-ho...	good	[entertainment, howstuffworks, com, kid, nicho...
395543	mylife.com/c-1442108366	good	[mylife, com, c]

## SnowballStemmer -

Snowball is a small string processing language, gives root words.

```
stemmer = SnowballStemmer("english") # choose a language
```

```
print('Getting words stemmed ...')
t0= time.perf_counter()
phish_data['text_stemmed'] = phish_data['text_tokenized'].map(lambda l: [stemmer.stem(word) for word in l])
t1= time.perf_counter() - t0
print('Time taken',t1 , 'sec')
```

Getting words stemmed ...  
Time taken 23.08872600000001 sec

```
phish_data.sample(5)
```

	URL	Label	text_tokenized	text_stemmed
274662	amazon.com/Dance-Fleetwood-Mac/dp/B000002NHG	good	[amazon, com, Dance, Fleetwood, Mac, dp, B, NHG]	[amazon, com, danc, fleetwood, mac, dp, b, nhg]
17863	huarui-tec.com/js/?ref=fnxlefqus.battle.net/d3...	bad	[huarui, tec, com, js, ref, fnxlefqus, battle,...	[huarui, tec, com, js, ref, fnxlefqus, battl, ...
442063	tcoasttalk.com/2011/09/06/woman-in-jensen-beac...	good	[tcoasttalk, com, woman, in, jensen, beach, wo...	[tcoasttalk, com, woman, in, jensen, beach, wo...
183610	english.turkcebilgi.com/Lillian+Moller+Gilbreth	good	[english, turkcebilgi, com, Lillian, Moller, G...	[english, turkcebilgi, com, lillian, moller, g...
405079	nick.com/videos/penguins-of-madagascar-videos	good	[nick, com, videos, penguins, of, madagascar, ...	[nick, com, video, penguin, of, madagascar, vi...

```
print('Getting joiningwords ...')
t0= time.perf_counter()
phish_data['text_sent'] = phish_data['text_stemmed'].map(lambda l: ' '.join(l))
t1= time.perf_counter() - t0
print('Time taken',t1 , 'sec')
```

Getting joiningwords ...  
Time taken 0.16629320000001258 sec

```
phish_data.sample(5)
```

	URL	Label	text_tokenized	text_stemmed	text_sent
218594	music.yahoo.com/tsuyoshi-nagabuchi/photos/	good	[music, yahoo, com, tsuyoshi, nagabuchi, photos]	[music, yahoo, com, tsuyoshi, nagabuchi, photo]	music yahoo com tsuyoshi nagabuchi photo
438735	startrek.com/database_article/year-of-hell-par...	good	[startrek, com, database, article, year, of, h...	[startrek, com, databas, articl, year, of, hel...	startrek com databas articl year of hell part ii
240858	soundcloud.com/late-nite-tuff-guy	good	[soundcloud, com, late, nite, tuff, guy]	[soundcloud, com, late, nite, tuff, guy]	soundcloud com late nite tuff guy
522517	gfdkotriam.fo4j4wnq51hepa.com/img/bitcoin.png	bad	[gfdkotriam, fo, j, wnq, hepa, com, img, bitco...	[gfdkotriam, fo, j, wnq, hepa, com, img, bitco...	gfdkotriam fo j wnq hepa com img bitcoin png
5228	islander.it/images/cani/frida/gdf5gdf4gd5g4d6f...	bad	[islander, it, images, cani, frida, gdf, gdf, ...	[island, it, imag, cani, frida, gdf, gdf, gd, ...	island it imag cani frida gdf gdf gd g d f g d...

## Visualization –

### 1. Visualize some important keys using word cloud

```
#sliceing classes
bad_sites = phish_data[phish_data.Label == 'bad']
good_sites = phish_data[phish_data.Label == 'good']
```

```
bad_sites.head()
```

	URL	Label	text_tokenized	text_stemmed	text_sent
0	nobell.it/70ffb52d079109dca5664cce6f317373782/...	bad	[nobell, it, ffb, d, dca, cce, f, login, Skype...	[nobel, it, ffb, d, dca, cce, f, login, skype,...	nobel it ffb d dca cce f login skype com en cg...
1	www.dghjdgf.com/paypal.co.uk/cycgi-bin/websrcr...	bad	[www, dghjdgf, com, paypal, co, uk, cycgi, bin...	[www, dghjdgf, com, paypal, co, uk, cycgi, bin...	www dghjdgf com paypal co uk cycgi bin websrcr...
2	serviciosbys.com/paypal.cgi.bin.get-into.herf....	bad	[serviciosbys, com, paypal, cgi, bin, get, int...	[serviciosbi, com, paypal, cgi, bin, get, into...	serviciosbi com paypal cgi bin get into herf s...
3	mail.printakid.com/www.online.americanexpress....	bad	[mail, printakid, com, www, online, americanex...	[mail, printakid, com, www, onlin, americanexp...	mail printakid com www onlin americanexpress c...
4	thewhiskeydregs.com/wp-content/themes/widescre...	bad	[thewhiskeydregs, com, wp, content, themes, wi...	[thewhiskeydreg, com, wp, content, theme, wide...	thewhiskeydreg com wp content theme widescreen...



## 2.create a function to visualize the important keys from url

```
def plot_wordcloud(text, mask=None, max_words=400, max_font_size=120, figure_size=(24.0,16.0),
                  title = None, title_size=40, image_color=False):
    stopwords = set(STOPWORDS)
    more_stopwords = {'com','http'}
    stopwords = stopwords.union(more_stopwords)

    wordcloud = WordCloud(background_color='white',
                          stopwords = stopwords,
                          max_words = max_words,
                          max_font_size = max_font_size,
                          random_state = 42,
                          mask = mask)
    wordcloud.generate(text)

    plt.figure(figsize=figure_size)
    if image_color:
        image_colors = ImageColorGenerator(mask);
        plt.imshow(wordcloud.recolor(color_func=image_colors), interpolation="bilinear");
        plt.title(title, fontdict={'size': title_size,
                                   'verticalalignment': 'bottom'})
    else:
        plt.imshow(wordcloud);
        plt.title(title, fontdict={'size': title_size, 'color': 'green',
                                   'verticalalignment': 'bottom'})

    plt.axis('off');
    plt.tight_layout()
d = '../input/masks/masks-wordclouds/'
```

```
data = good_sites.text_sent
data.reset_index(drop=True, inplace=True)
```

```
data = bad_sites.text_sent
data.reset_index(drop=True, inplace=True)
```

## Creating Model -

- **CountVectorizer** - CountVectorizer is used to transform a corpora of text to a vector of term / token counts.

```
#create cv object
cv = CountVectorizer()
```

```
help(CountVectorizer())
```

Converts a collection of text documents to a matrix of token counts.

```
feature = cv.fit_transform(phish_data.text_sent) #transform all text which we tokenize and stemmed
```

```
feature[:5].toarray() # convert sparse matrix into array to print transformed features
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```



## Splitting the data –

```
trainX, testX, trainY, testY = train_test_split(feature, phish_data.Label)
```

## LogisticRegression -

Logistic Regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.). In other words, the logistic regression model predicts  $P(Y=1)$  as a function of  $X$ .

```
# create lr object
lr = LogisticRegression()
```

```
lr.fit(trainX, trainY)
```

```
LogisticRegression()
```

```
lr.score(testX, testY)
```

```
0.9633529201890241
```

```
*** Logistic Regression is giving 96% accuracy, Now we will store scores in dict to see which model perform best**
```

```
Scores_ml = {}
Scores_ml['Logistic Regression'] = np.round(lr.score(testX, testY), 2)
```

```
print('Training Accuracy :', lr.score(trainX, trainY))
print('Testing Accuracy :', lr.score(testX, testY))
con_mat = pd.DataFrame(confusion_matrix(lr.predict(testX), testY),
                        columns = ['Predicted:Bad', 'Predicted:Good'],
                        index = ['Actual:Bad', 'Actual:Good'])
```

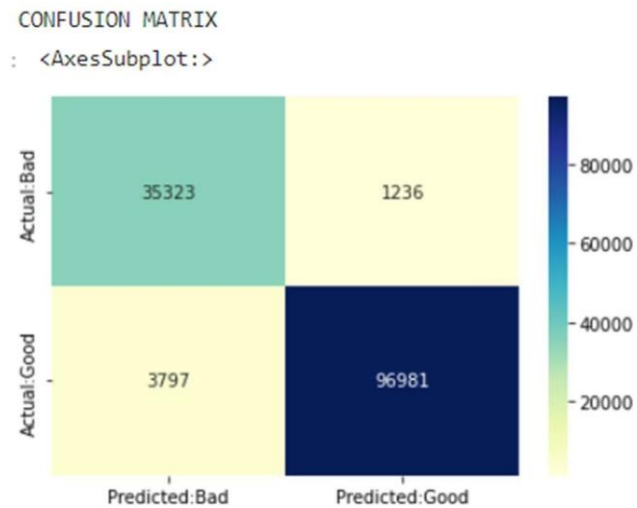
```
print('\nCLASSIFICATION REPORT\n')
print(classification_report(lr.predict(testX), testY,
                            target_names = ['Bad', 'Good']))
```

```
print('\nCONFUSION MATRIX')
plt.figure(figsize = (6,4))
sns.heatmap(con_mat, annot = True, fmt = 'd', cmap = "YlGnBu")
```

Training Accuracy : 0.9767602164030398

Testing Accuracy : 0.9633529201890241

	pr eci sio n	r e c a l l	f 1 - s c o r e	s u p p o r t
Bad	0.9 0	0 . 9 7	0 . 9 3	3 6 5 5 9
Good	0.9 9	0 . 9 6	0 . 9 7	1 0 0 7 7 8
accuracy			0 . 9 6	1 3 7 3 3 7
macro avg	0.9 5	0 . 9 6	0 . 9 5	1 3 7 3 3 7
weighted avg	0.9 6	0 . 9 6	0 . 9 6	1 3 7 3 3 7



## MultinomialNB

Applying Multinomial Naive Bayes to NLP Problems. Naive Bayes Classifier Algorithm is a family of probabilistic algorithms based on applying Bayes' theorem with the “naive” assumption of conditional independence between every pair of a feature.

```
# create mnb object
mnb = MultinomialNB()
```

```
mnb.fit(trainX,trainY)
```

```
MultinomialNB()
```

```
mnb.score(testX,testY)
```

```
0.9580812162782062
```

\*\*\* MultinomialNB gives us 95% accuracy\*\*

```
Scores_ml['MultinomialNB'] = np.round(mnb.score(testX,testY),2)
```

```
print('Training Accuracy :',mnb.score(trainX,trainY))
print('Testing Accuracy :',mnb.score(testX,testY))
con_mat = pd.DataFrame(confusion_matrix(mnb.predict(testX), testY),
                        columns = ['Predicted:Bad', 'Predicted:Good'],
                        index = ['Actual:Bad', 'Actual:Good'])

print('\nCLASSIFICATION REPORT\n')
print(classification_report(mnb.predict(testX), testY,
                            target_names =['Bad','Good']))

print('\nCONFUSION MATRIX')
plt.figure(figsize= (6,4))
sns.heatmap(con_mat, annot = True,fmt='d',cmap="YlGnBu")
```

```
Training Accuracy : 0.97402483926322
Testing Accuracy : 0.9580812162782062
```

CLASSIFICATION REPORT

	precision	recall	f1-score	support
Bad	0.92	0.94	0.93	38285
Good	0.97	0.97	0.97	99052
accuracy			0.96	137337
macro avg	0.95	0.95	0.95	137337
weighted avg	0.96	0.96	0.96	137337

CONFUSION MATRIX

]: <AxesSubplot:>



```

print('Training Accuracy :',mnb.score(trainX,trainY))
print('Testing Accuracy :',mnb.score(testX,testY))
con_mat = pd.DataFrame(confusion_matrix(mnb.predict(testX), testY),
                             columns = ['Predicted:Bad', 'Predicted:Good'],
                             index = ['Actual:Bad', 'Actual:Good']))

print('\nCLASSIFICATION REPORT\n')
print(classification_report(mnb.predict(testX), testY,
                             target_names =['Bad','Good']))

print('\nCONFUSION MATRIX')
plt.figure(figsize= (6,4))
sns.heatmap(con_mat, annot = True,fmt='d',cmap="YlGnBu")

```

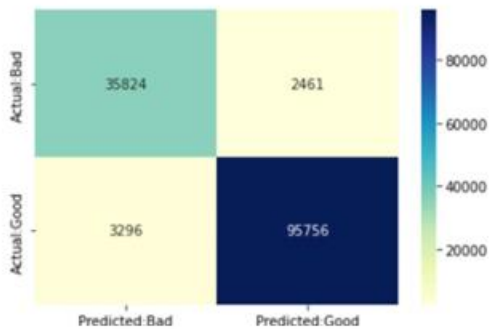
Training Accuracy : 0.97402483926322  
Testing Accuracy : 0.9580812162782062

#### CLASSIFICATION REPORT

	precision	recall	f1-score	support
Bad	0.92	0.94	0.93	38285
Good	0.97	0.97	0.97	99052
accuracy			0.96	137337
macro avg	0.95	0.95	0.95	137337
weighted avg	0.96	0.96	0.96	137337

#### CONFUSION MATRIX

]: <AxesSubplot:>



```

print('Training Accuracy :',mnb.score(trainX,trainY))
print('Testing Accuracy :',mnb.score(testX,testY))
con_mat = pd.DataFrame(confusion_matrix(mnb.predict(testX), testY),
                             columns = ['Predicted:Bad', 'Predicted:Good'],
                             index = ['Actual:Bad', 'Actual:Good']))

print('\nCLASSIFICATION REPORT\n')
print(classification_report(mnb.predict(testX), testY,
                             target_names =['Bad','Good']))

print('\nCONFUSION MATRIX')
plt.figure(figsize= (6,4))
sns.heatmap(con_mat, annot = True,fmt='d',cmap="YlGnBu")

```

Training Accuracy : 0.97402483926322  
Testing Accuracy : 0.9580812162782062

#### CLASSIFICATION REPORT

	precision	recall	f1-score	support
Bad	0.92	0.94	0.93	38285
Good	0.97	0.97	0.97	99052
accuracy			0.96	137337
macro avg	0.95	0.95	0.95	137337
weighted avg	0.96	0.96	0.96	137337

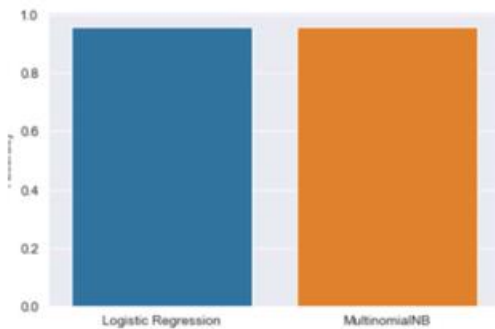
#### CONFUSION MATRIX

]: <AxesSubplot:>



```
acc = pd.DataFrame.from_dict(Scores_ml,orient = 'index',columns=['Accuracy'])
sns.set_style('darkgrid')
sns.barplot(acc.index,acc.Accuracy)
```

<AxesSubplot:ylabel='Accuracy'>



\*\*\* So, Logistic Regression is the best fit model, Now we make sklearn pipeline using Logistic Regression\*\*

```
pipeline_ls = make_pipeline(CountVectorizer(tokenizer = RegexpTokenizer(r'[A-Za-z]+').tokenize,stop_words='english'), LogisticRegression())
##(r'\b(?:http|ftp)s?:\/\/\S*\w\/\w+\/[^\w\s]+') ([a-zA-Z]+)([0-9]+) -- these tokenizers giving me low accuracy
```

```
trainX, testX, trainY, testY = train_test_split(phish_data.URL, phish_data.Label)
```

```
pipeline_ls.fit(trainX,trainY)
```

```
Pipeline(steps=[('countvectorizer',
                  CountVectorizer(stop_words='english',
                                  tokenizer=<bound method RegexpTokenizer.tokenize of RegexpTokenizer(pattern='[A-Za-z]+', flags=re.UNICODE|re.MULTILINE|re.DOTALL)>)),
                  ('logisticregression', LogisticRegression())])
```

```
pipeline_ls.score(testX,testY)
```

```
0.9650349141163707
```

```
print('Training Accuracy :',pipeline_ls.score(trainX,trainY))
print('Testing Accuracy :',pipeline_ls.score(testX,testY))
con_mat = pd.DataFrame(confusion_matrix(pipeline_ls.predict(testX), testY),
                        columns = ['Predicted:Bad', 'Predicted:Good'],
                        index = ['Actual:Bad', 'Actual:Good'])
```

```
print('\nCLASSIFICATION REPORT\n')
print(classification_report(pipeline_ls.predict(testX), testY,
                            target_names = ['Bad', 'Good']))
```

```
print('\nCONFUSION MATRIX')
plt.figure(figsize= (6,4))
sns.heatmap(con_mat, annot = True,fmt='d',cmap="YlGnBu")
```

# CHAPTER 7

## SYSTEM TESTING

In the context of a phishing website detection project using machine learning algorithms, system testing involves evaluating the entire system's functionality, performance, and behavior to ensure that it meets the specified requirements and effectively detects phishing websites. Here's how system testing might be structured for such a project.

### ➤ TYPES OF TESTS

#### 1. Functional Testing:

- **Feature Validation:** Verify that all intended features work as expected (e.g., URL analysis, content examination).
- **Algorithm Performance:** Test the accuracy of the machine learning models in identifying phishing websites against a diverse dataset.
- **User Interaction:** Ensure the system effectively communicates with users (e.g., displaying warnings/alerts).
- **Feedback Mechanism:** Test the functionality that collects and processes user feedback to improve the system.

#### 2. Performance Testing:

- **Scalability:** Assess the system's ability to handle increased loads of website analysis requests without significant performance degradation.
- **Response Time:** Measure the time taken by the system to analyze a website and provide a result, ensuring it meets acceptable thresholds.

#### 3. Security Testing:

- **Robustness:** Check the system's resilience against various attacks or attempts to bypass the phishing detection mechanisms.
- **Data Protection:** Ensure that sensitive data (user information, feedback) is securely handled and stored.

#### 4. Usability Testing:

- **User Interface:** Evaluate the interface for ease of use and clarity in displaying results and warnings.
- **Accessibility:** Ensure the system is accessible to all intended users (e.g., considering different devices or browsers)..

#### 5. Integration Testing:

- **Model Integration:** Test the integration of machine learning models within the system, checking for proper functionality and data flow.
- **Database Integration:** Verify that historical data retrieval and storage are correctly integrated and utilized for analysis.



## 6. Regression Testing:

- **Stability:** Ensure that new updates or changes do not negatively impact previously working functionalities.
- **Reusability:** Test the system's ability to maintain functionality across different environments or configurations.

## 7. Acceptance Testing:

- **User Acceptance:** Involve end-users to validate if the system meets their requirements and expectations in identifying phishing websites effectively.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

### 7.1 Source Code:

```
<html>

<!-- From https://codepen.io/frytyler/pen/EGdtg -->

<head>

  <meta charset="UTF-8">

  <title>ML API</title>

  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>

  <link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>

  <link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>

  <link          href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
rel='stylesheet' type='text/css'>

  <link rel="stylesheet" href="{ { url_for('static', filename='css/style.css') } }">

</head>

<body>

  <div class="login">
```

```
<h1>Phishing Site Detection</h1>
```

```
<!-- Main Input For Receiving Query to our ML -->
```

```
<form action="{ { url_for('predict') }}" method="post">
```

```
  <input type="text" name="EnterYourSite" placeholder="Enter your Site" required="required"
```

```
/>
```

```
  <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
```

```
</form>
```

```
<br>
```

```
<br>
```

```
  {{ prediction_text }}
```

```
</div>
```

```
</body>
```

```
</html>
```

```
@import url(https://fonts.googleapis.com/css?family=Open+Sans);
```

```
.btn {
```

```
  display: inline-block;
```

```
  *display: inline;
```

```
  *zoom: 1;
```

```
  padding: 4px 10px 4px;
```

```
  margin-bottom: 0;
```

font-size: 13px;  
line-height: 18px;  
color: #333333;  
text-align: center;  
text-shadow: 0 1px 1px rgba(255, 255, 255, 0.75);  
vertical-align: middle;  
background-color: #f5f5f5;  
background-image: -moz-linear-gradient(top, #ffffff, #e6e6e6);  
background-image: -ms-linear-gradient(top, #ffffff, #e6e6e6);  
background-image: -webkit-gradient(linear, 0 0, 0 100%, from(#ffffff), to(#e6e6e6));  
background-image: -webkit-linear-gradient(top, #ffffff, #e6e6e6);  
background-image: -o-linear-gradient(top, #ffffff, #e6e6e6);  
background-image: linear-gradient(top, #ffffff, #e6e6e6);  
background-repeat: repeat-x;  
filter: progid:dximagetransform.microsoft.gradient(startColorstr=#ffffff, endColorstr=#e6e6e6, GradientType=0);  
border-color: #e6e6e6 #e6e6e6 #e6e6e6;  
border-color: rgba(0, 0, 0, 0.1) rgba(0, 0, 0, 0.1) rgba(0, 0, 0, 0.25);  
border: 1px solid #e6e6e6;  
-webkit-border-radius: 4px;  
-moz-border-radius: 4px;  
border-radius: 4px;  
-webkit-box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0 1px 2px rgba(0, 0, 0, 0.05);  
-moz-box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0 1px 2px rgba(0, 0, 0, 0.05);

```
box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0 1px 2px rgba(0, 0, 0, 0.05);  
  
cursor: pointer;  
  
*margin-left: .3em;  
  
}
```

```
.btn:hover,  
  
.btn:active,  
  
.btn.active,  
  
.btn.disabled,  
  
.btn[disabled] {  
  
    background-color: #e6e6e6;  
  
}
```

```
.btn-large {  
  
    padding: 9px 14px;  
  
    font-size: 15px;  
  
    line-height: normal;  
  
    -webkit-border-radius: 5px;  
  
    -moz-border-radius: 5px;  
  
    border-radius: 5px;  
  
}
```

```
.btn:hover {  
  
    color: #333333;
```

```
.btn-block {  
  
    width: 100%;  
  
    display: block;  
  
}  
  
* {  
  
    -webkit-box-sizing: border-box;  
  
    -moz-box-sizing: border-box;  
  
    -ms-box-sizing: border-box;  
  
    -o-box-sizing: border-box;  
  
    box-sizing: border-box;  
  
}  
  
html {  
  
    width: 100%;  
  
    height: 100%;  
  
    overflow: hidden;  
  
}  
  
body {  
  
    width: 100%;  
  
    height: 100%;  
  
    font-family: 'Open Sans', sans-serif;  
  
    background: #092756;
```

```

color: #fff;

font-size: 18px;

text-align: center;

{

  "cell_type": "markdown",

  "metadata": {

    "id": "QYFaNT4QH9Vj",

    "papermill": {

      "duration": 0.035205,

      "end_time": "2020-11-21T06:33:13.502886",

      "exception": false,

      "start_time": "2020-11-21T06:33:13.467681",

      "status": "completed"

    },

    "tags": []

  },

}

.login {

  position: absolute;

  top: 40%;

  left: 50%;

  margin: -150px 0 0 -150px;

  width: 400px;

  height: 400px;

```

```
}
```

```
.login h1 { color: #fff; text-shadow: 0 0 10px rgba(0,0,0,0.3); letter-spacing: 1px; text-align: center;
}
```

```
input {
    width: 100%;
    margin-bottom: 10px;
    background: rgba(0,0,0,0.3);
    border: none;
    outline: none;
    padding: 10px;
    font-size: 13px;
    color: #fff;
    text-shadow: 1px 1px 1px rgba(0,0,0,0.3);
    border: 1px solid rgba(0,0,0,0.3);
    border-radius: 4px;
    box-shadow: inset 0 -5px 45px rgba(100,100,100,0.2), 0 1px 1px rgba(255,255,255,0.2);
    -webkit-transition: box-shadow .5s ease;
    -moz-transition: box-shadow .5s ease;
    -o-transition: box-shadow .5s ease;
    -ms-transition: box-shadow .5s ease;
    transition: box-shadow .5s ease;
}
```

```
input:focus { box-shadow: inset 0 -5px 45px rgba(100,100,100,0.4), 0 1px 1px
rgba(255,255,255,0.2); }
```

```
from flask import Flask, request, Response, render_template
import pickle
```

```
app = Flask(__name__)

# read our pickle file and label our logisticmodel as model
phish_model_ls = pickle.load(open('phishing.pkl', 'rb'))
```

```
urlError = {
    "Please enter url field"
}
```

```
{
    "cell_type": "markdown",
    "metadata": {
        "id": "ZFR6LnIHH9Vo",
        "papermill": {
            "duration": 0.034848,
            "end_time": "2020-11-21T06:33:13.713729",
            "exception": false,
            "start_time": "2020-11-21T06:33:13.678881",
```



```

    "status": "completed"

  },

  "tags": []

},

"source": [

  "##### * Importing some useful libraries"

]

},

{

  "cell_type": "code",

  "execution_count": 12,

  "metadata": {

    "colab": {

      "base_uri": "https://localhost:8080/"

    },

    "execution": {

      "iopub.execute_input": "2020-11-21T06:33:13.789742Z",

      "iopub.status.busy": "2020-11-21T06:33:13.789113Z",

      "iopub.status.idle": "2020-11-21T06:33:22.729362Z",

      "shell.execute_reply": "2020-11-21T06:33:22.728761Z"

    },

    "id": "_WNO2nTGH9Vp",

    "outputId": "2b0fb9f3-601c-4c80-ff84-8055d72b337d",

    "papermill": {

```

```

    "duration": 8.979884,
    "end_time": "2020-11-21T06:33:22.729479",
    "exception": false,
    "start_time": "2020-11-21T06:33:13.749595",
    "status": "completed"
  },
  "tags": []
},
"tags": []
},
"outputs": [
{
  "data": {
    "text/html": [
      "<div>\n",
      "<style scoped>\n",
      "  .dataframe tbody tr th:only-of-type {\n",
      "    vertical-align: middle;\n",
      "  }\n",
      "\n",
      "  .dataframe tbody tr th {\n",
      "    vertical-align: top;\n",
      "  }\n",
      "\n",

```

```

" .dataframe thead th {\n",
"     text-align: right;\n",
" } \n",
"</style>\n",
"<table border= \"1\" class= \"dataframe\">\n",
" <thead>\n",
" <tr style= \"text-align: right;\">\n",
" <th></th>\n",
" <th>URL</th>\n",
" <th>Label</th>\n",
" </tr>\n",
" </thead>\n",
" <tbody>\n",
" <tr>\n",
" <th>0</th>\n",
" <td>nobell.it/70ffb52d079109dca5664cce6f317373782/...</td>\n",
" <td>bad</td>\n",
" </tr>\n",
" <tr>\n",
" <th>1</th>\n",
" <td>www.dghjdgf.com/paypal.co.uk/cycgi-bin/websrcr...</td>\n",
" <td>bad</td>\n",
" </tr>\n",
" <tr>\n",

```

```

"    <th>2</th>\n",
"    <td>serviciosbys.com/paypal.cgi.bin.get-into.herf....</td>\n",
"    <td>bad</td>\n",
"  </tr>\n",
"  <tr>\n",
"    <th>3</th>\n",
"    <td>mail.printakid.com/www.online.americanexpress....</td>\n",
"    <td>bad</td>\n",
"  </tr>\n",
"  <tr>\n",
"    <th>4</th>\n",
"    <td>thewhiskeydregs.com/wp-content/themes/widescre...</td>\n",
"    <td>bad</td>\n",
"  </tr>\n",
" </tbody>\n",
"</table>\n",
"</div>"

```

],

"text/plain": [

```

"
                                URL Label\n",
"0 nobell.it/70ffb52d079109dca5664cce6f317373782/... bad\n",
"1 www.dghjdgf.com/paypal.co.uk/cycgi-bin/websrcr... bad\n",
"2 serviciosbys.com/paypal.cgi.bin.get-into.herf.... bad\n",
"3 mail.printakid.com/www.online.americanexpress.... bad\n",

```

```

    "4 thewhiskeydregs.com/wp-content/themes/widescre... bad"
  ]
},
"execution_count": 15,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
  "phish_data.head()"
]
},
{
  "cell_type": "code",
  "execution_count": 16,
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/",
      "height": 206
    },
    "execution": {
      "iopub.execute_input": "2020-11-21T06:33:27.646131Z",
      "iopub.status.busy": "2020-11-21T06:33:27.645097Z",
      "iopub.status.idle": "2020-11-21T06:33:27.649188Z",

```

```

"shell.execute_reply": "2020-11-21T06:33:27.648573Z"

"  </tr>\n",

"  <tr>\n",

"    <th>549345</th>\n",

"    <td>apple-search.info</td>\n",

"    <td>bad</td>\n",

"  </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

    "          URL Label\n",

    "549341  23.227.196.215/  bad\n",

    "549342  apple-checker.org/  bad\n",

    "549343  apple-iclouds.org/  bad\n",

    "549344  apple-uptoday.org/  bad\n",

    "549345  apple-search.info  bad"

]

},

"execution_count": 16,

"metadata": {},

"output_type": "execute_result"

}

```

```
],  
  "source": [  
    "phish_data.tail()  
  ]  
},  
{  
  "cell_type": "code",  
  "execution_count": 17,  
  "metadata": {  
    "colab": {  
      "base_uri": "https://localhost:8080/"  
    },  
    "execution": {  
      "iopub.execute_input": "2020-11-21T06:33:27.778145Z",  
  
    },  
    "id": "2K2ahryEH9Vr",  
    "outputId": "d053bfb0-4e7e-45c2-de45-57d1ec4b007d",  
    "papermill": {  
      "duration": 0.103111,  
      "end_time": "2020-11-21T06:33:27.792805",  
      "exception": false,  
      "start_time": "2020-11-21T06:33:27.689694",  
      "status": "completed"
```

```

    },
    "tags": []
  },
  "outputs": [
    {
      "name": "stdout",
      "output_type": "stream",
      "text": [
        "<class 'pandas.core.frame.DataFrame'>\n",
        "RangeIndex: 549346 entries, 0 to 549345\n",
        "Data columns (total 2 columns):\n",
        "#   Column  Non-Null Count  Dtype \n",
        "---  -
        0   URL      549346 non-null  object\n",
        1   Label    549346 non-null  object\n",
        "dtypes: object(2)\n",
        "memory usage: 8.4+ MB\n"
      ]
    }
  ],
  "source": [
    "phish_data.info()"
  ]
},

```



```

{
  "cell_type": "code",
  "execution_count": 18,
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "execution": {
      "iopub.execute_input": "2020-11-21T06:33:28.007694Z",
      "iopub.status.busy": "2020-11-21T06:33:28.006670Z",
      "iopub.status.idle": "2020-11-21T06:33:28.014459Z",
      "shell.execute_reply": "2020-11-21T06:33:28.013951Z"
    },
    "id": "T3YGB9mLH9Vr",
  }
})

def home():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():

    X_predict = []

    url = request.form.get("EnterYourSite")

```

```

print(url, "000000000000000000000000")

if url:

    X_predict.append(str(url))

    y_Predict = ".join(phish_model_ls.predict(X_predict))

    print(y_Predict)

    if y_Predict == 'bad':

        result = "This is a Phishing Site"

    else:

        result = "This is not a Phishing Site"

    return render_template('index.html', prediction_text = result)

elif not url:

    return Response(

        response=urlError,

        status=400

    )

if __name__ == '__main__':

    app.run(debug=True, host='127.0.0.1', port=8000, threaded=True)

```

## CHAPTER 8

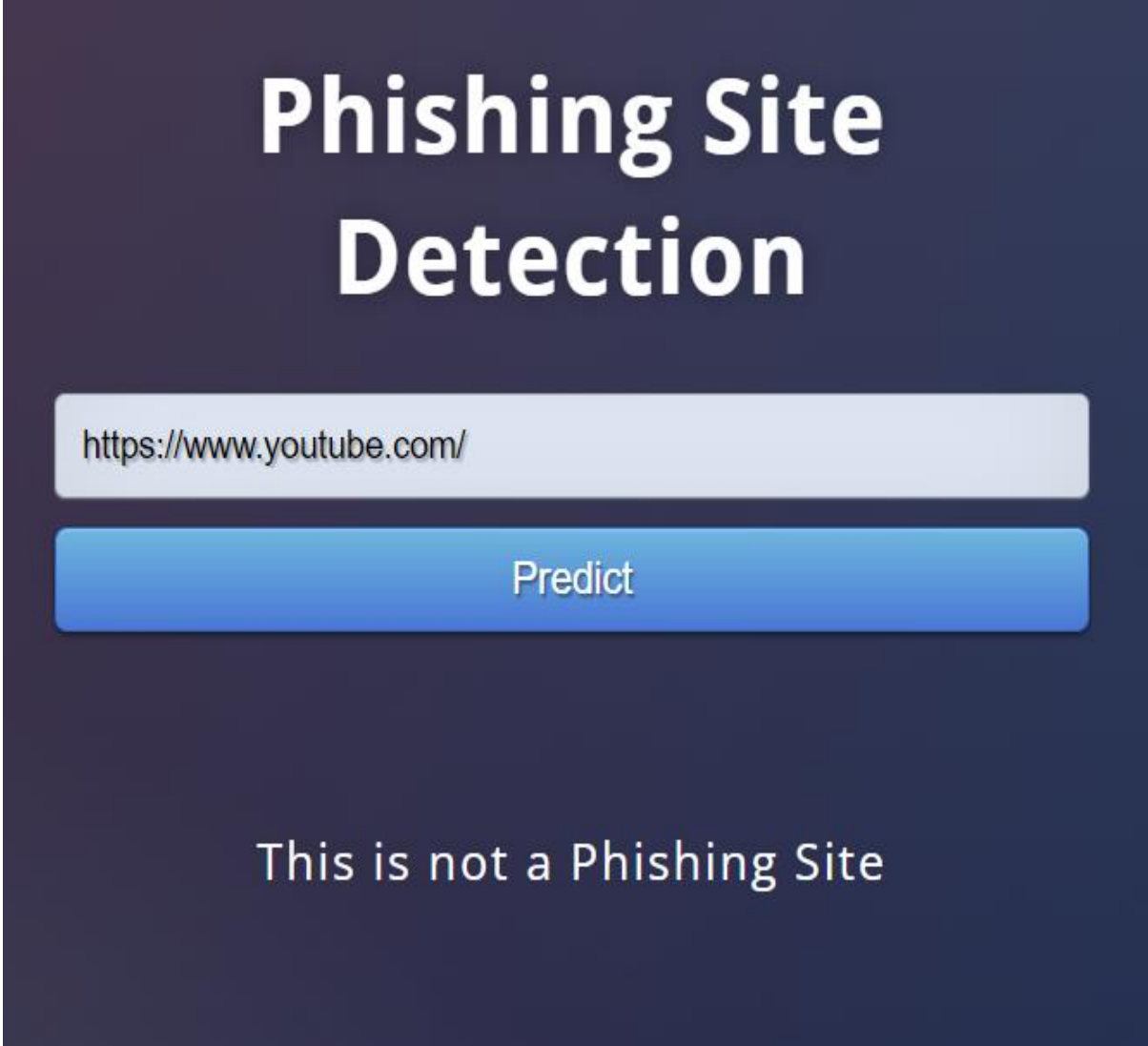
### EXPERIMENTAL RESULTS

#### 1. HOME PAGE:



**Figure 8.1 Shows the Home Page of the Project**

## 2. UPLOAD WEBSITE:



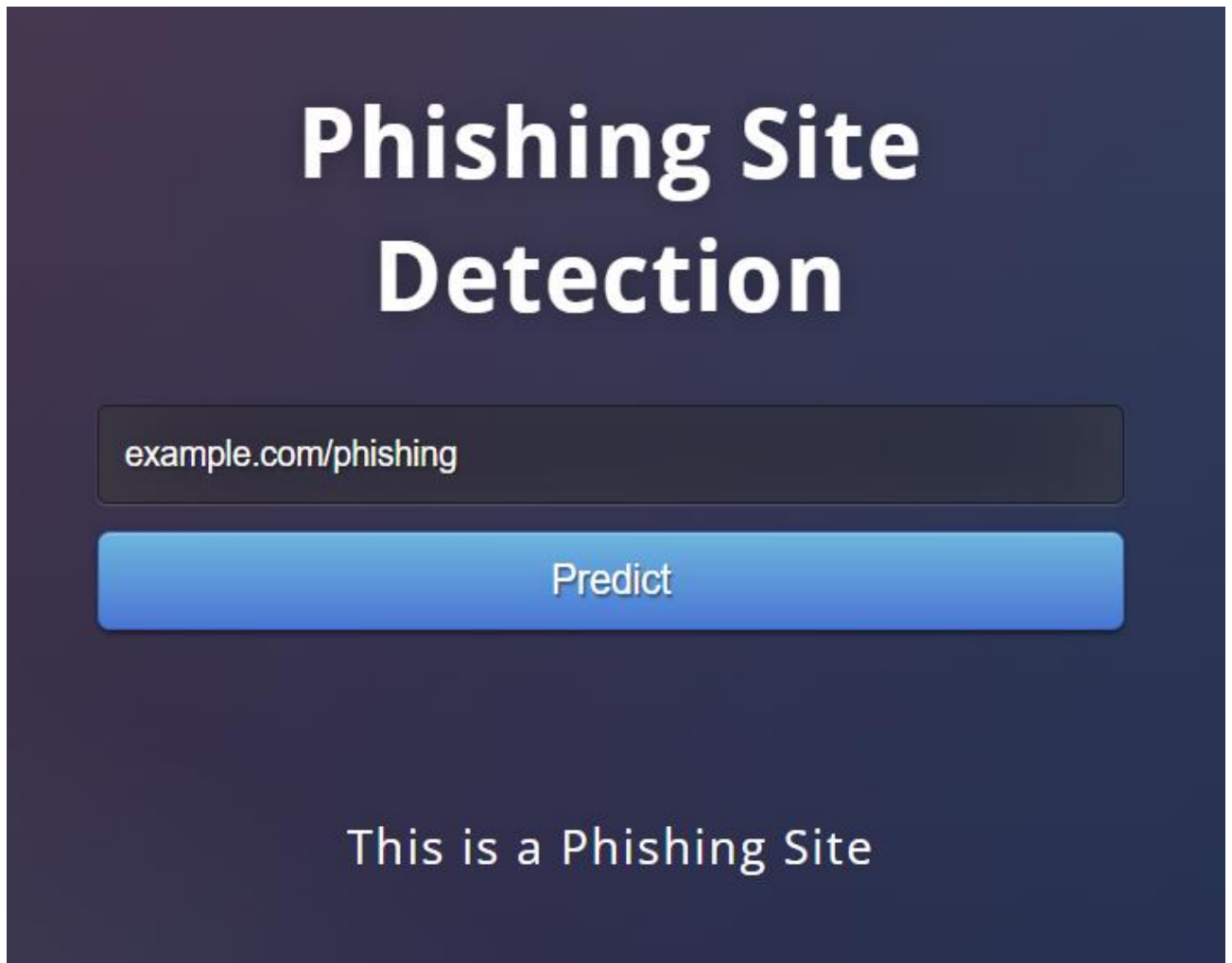
The image shows a web application interface for detecting phishing sites. It has a dark blue background. At the top, the title "Phishing Site Detection" is written in large, white, sans-serif font. Below the title is a light blue input field containing the URL "https://www.youtube.com/". Underneath the input field is a blue button with the word "Predict" in white. At the bottom of the interface, the text "This is not a Phishing Site" is displayed in white.

# Phishing Site Detection

This is not a Phishing Site

**Figure 8.2 Shows Upload Website**

### 3. PHISHING WEBSITE DETECTED:



**Figure 8.3 Shows Phishing Website Detected**

## **CHAPTER 9**

### **CONCLUSION**

Our Project demonstrates a comprehensive process for building a machine learning model to detect phishing URLs. The project covers data loading, preprocessing, visualization, model creation using logistic regression, and evaluation. Through the use of text preprocessing techniques like tokenization and stemming, as well as advanced visualization methods such as word clouds, the project effectively prepares the data. The logistic regression model is trained and evaluated, showcasing its ability to accurately classify URLs as phishing or legitimate. Multinomial Naive Bayes is also explored for comparison.

Moreover, the project employs a pipeline to streamline the preprocessing and classification steps. This holistic approach reflects how data science techniques, including NLP, machine learning, and visualization, can be synergistically applied to address real-world cybersecurity challenges like phishing detection.

## CHAPTER 10

### REFERENCES

- [https://www.researchgate.net/profile/RishikeshMahajan/publication/328541785\\_Phishing\\_Website\\_Detection\\_using\\_Machine\\_Learning\\_Algorithms/links/5d0397fd92851c9004394af4/Phishing-Website-Detection-using-Machine-LearningAlgorithms.pdf](https://www.researchgate.net/profile/RishikeshMahajan/publication/328541785_Phishing_Website_Detection_using_Machine_Learning_Algorithms/links/5d0397fd92851c9004394af4/Phishing-Website-Detection-using-Machine-LearningAlgorithms.pdf)
- Ankit Kumar Jain, B. B. Gupta : Towards detection of phishing websites on client-side using machine learning based approach : In Springer Science+Business Media, LLC, part of Springer Nature 2017
- Chunlin Liu, Bo Lang : Finding effective classifier for malicious URL detection : In ACM, 2018  
[https://www.researchgate.net/profile/ErPurviPujara/publication/331198983\\_Phishing\\_Website\\_Detection\\_using\\_Machine\\_Learning\\_A\\_Review/links/5c6bd4ae4585156b5706e727/P\\_hishingWebsite-Detection-using-Machine-Learning-A-Review.pdf](https://www.researchgate.net/profile/ErPurviPujara/publication/331198983_Phishing_Website_Detection_using_Machine_Learning_A_Review/links/5c6bd4ae4585156b5706e727/P_hishingWebsite-Detection-using-Machine-Learning-A-Review.pdf)
- Sahingoz, O.K., Buber, E., Demir, O. and Diri, B., 2019. Machine learning based phishing detection from URLs. Expert Systems with Applications, 117, pp.345-357
- Sci-kit learn, SVM library. <http://scikit-learn.org/stable/modules/svm.html>.
- 'APWG | Unifying The Global Response To Cybercrime' (n.d.) available: <https://apwg.org/>
- ] 14 Types of Phishing Attacks That IT Administrators Should Watch For [online] (2021) <https://www.blog.syscloud.com>, available: <https://www.blog.syscloud.com/types-of-phishing/>
- Lakshmanarao, A., Rao, P.S.P., Krishna, M.M.B. (2021) 'Phishing website detection using novel machine learning fusion approach', in 2021 International Conference on Artificial Intelligence and

Smart Systems (ICAIS), Presented at the 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), 1164–1169

- Vaishnavi, D., Suwetha, S., Jinila, Y.B., Subhashini, R., Shyry, S.P. (2021) ‘A Comparative Analysis of Machine Learning Algorithms on Malicious URL Prediction’, in 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), Presented at the 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), 1398–1402
- Microsoft, Microsoft Consumer safety report. <https://news.microsoft.com/en-sg/2014/02/11/microsoft-consumersafety-index-revealsimpact-of-poor-online-safety-behaviours-in-singapore/sm.001xdu50tlxsej410r11kqvks u4nz>.
- Internal Revenue Service, IRS E-mail Schemes. Available at <https://www.irs.gov/uac/newsroom/consumers-warnedof-new-surge-in-irs-email-schemes-during-2016-tax-season-tax-industry-also-targeted>.
- Abu-Nimeh, S., Nappa, D., Wang, X., Nair, S. (2007), A comparison of machine learning techniques for phishing detection. Proceedings of the Anti-phishing Working Groups 2nd Annual ECrime Researchers Summit on - ECrime '07. doi:10.1145/1299015.1299021.
- Wang Wei-Hong, L V Yin-Jun, CHEN Hui-Bing, FANG Zhao-Lin., A Static Malicious Javascript Detection Using SVM, In Proceedings of the 2nd International Conference on Computer Science and Electrical Engineering (ICCSEE 2013)
- E., B., K., T. (2015)., Phishing URL Detection: A Machine Learning and Web Mining-based Approach. International Journal of Computer Applications,123(13), 46-50. doi:10.5120/ijca2015905665.
- Ningxia Zhang, Yongqing Yuan, Phishing Detection Using Neural Net-work, In Proceedings of



International Conference on Neural Information Processing, pp. 714–719. Springer, Heidelberg (2004).

- Ram Basnet, Srinivas Mukkamala et al, Detection of Phishing Attacks: A Machine Learning Approach, In Proceedings of the International World Wide Web Conference (WWW), 2003.
- Sci-kit learn, SVM library. <http://scikit-learn.org/stable/modules/svm.html>
- Sklearn, ANN library. <http://scikit-learn.org/stable/modules/ann.html>.
- H. Chapla, R. Kotak and M. Joiser, "A Machine Learning Approach for URL Based Web Phishing Using Fuzzy Logic as Classifier", 2019 International Conference on Communication and Electronics Systems (ICCES), pp. 383-388, 2019, July
- Sclera,Randomforestlibrary.[http://scikitlearn.org/stable/modules/Randomforesets.h tml](http://scikitlearn.org/stable/modules/Randomforesets.html).
- Ahmad Abunadi, Anazida Zainal ,OluwatobiAkanb: Feature Extraction Process: A Phishing Detection Approach :In IEEE,2013