

**PHISHING WEBSITE DETECTION USING MACHINE LEARNING**

**MINI PROJECT REPORT**

***Submitted by***

**AADIL KHAN S (1920106002)**

**DEEPAK R (1920106016)**

**DHANUSH KUMAR S G (1920106702)**

**GAYATHRI R (1920106023)**

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF TECHNOLOGY**

**IN**

**INFORMATION TECHNOLOGY**

**SONA COLLEGE OF TECHNOLOGY, SALEM-5**

**(Autonomous)**

**NOVEMBER 2023**

**ANNA UNIVERSITY: CHENNAI 600 025**

**BONAFIDE CERTIFICATE**

Certified that this project report **“PHISHING WEBSITE DETECTION”** is the bonafide work of **“AADIL KHAN S (1920106002), DEEPAK R (1920106016), DHANUSH KUMAR S G(1920106702), GAYATHRI R (1920106023)”** who carried out the project work under my supervision.

**SIGNATURE SIGNATURE**

Dr.J.Akilandeswari Ms.J.Deepika

**HEAD OF THE DEPARTMENT SUPERVISOR**

Professor Assistant Professor

Department of Information Technology Department of Information Technology

Sona College of Technology, Sona College of Technology,

Salem-636 005. Salem-636 005

Submitted for Mini project viva voce examination held on .....................................

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ABSTRACT**

**PHISHING WEBSITE DETECTION USING MACHINE LEARNING**

With the increasing prevalence of online threats, the detection of phishing websites has become a critical component of cyber security. This study proposes a novel approach for identifying phishing websites by leveraging a comprehensive set of features extracted from various aspects of website content and behavior. The methodology involves the extraction and analysis of features such as URL structure, HTML and JavaScript content, SSL certificate details, and website traffic patterns. The research employs machine learning techniques to build a robust model capable of distinguishing between legitimate and phishing websites based on the extracted features. Feature selection methods are applied to identify the most discriminative attributes, optimizing the model's efficiency and performance. The dataset used for training and evaluation comprises a diverse collection of both legitimate and confirmed phishing websites, ensuring the model's ability to generalize across different attack scenarios.

The experimental results demonstrate the effectiveness of the proposed feature-based approach in accurately detecting phishing websites. Comparative analyses with existing detection methods showcase the superiority of the proposed model in terms of precision, recall, and overall accuracy. Furthermore, the model exhibits resilience to evasion techniques commonly employed by cybercriminals, contributing to its reliability in real-world scenarios. The findings of this research offer valuable insights into the importance of feature-based analysis for phishing website detection. The proposed methodology presents a promising solution for enhancing the security of online users by providing an advanced and adaptive defense against evolving phishing threats. As the digital landscape continues to evolve, the feature-based approach outlined in this study lays the foundation for proactive and effective countermeasures against phishing attacks.

**ACKNOWLEDGEMENT**

First and foremost, we thank to **power of almighty** for showing us inner peace

and for all blessings. Special gratitude to our parents, for showing their support and love always.

We express our sincere thanks to chairman **Sri.C.VALLIAPPA** and principal

**Dr.S.R.R.SENTHILKUMAR** for providing adequate facilities to complete the

project.

We are immensely grateful to our Head of Information Technology

Department, **Dr.J.AKILANDESWARI** for the continue encouragement to complete the project.

We are immensely grateful to our Project Coordinator **Dr.V.MOHANRAJ**

for his continuous encouragement to complete the project.

We express our heartfelt thanks to our project supervisor

**Ms.J.DEEPIKA** for her valuable guidance and fruitful discussions

throughout the course of the project work.

We feel proud in sharing this success with all our department Faculty, Staff

members and friends who helped directly and indirectly to completing this project

**TABLE OF CONTENTS**

**CHAPTER NO. TITLE PAGE NO** **ABSTRACT 3**

**LIST OF FIGURES 7**

**1. INTRODUCTION**

1.1 Background 8

1.2 Problem description 8

1.3 Aim and Objective 9

1.4 Scope 9

1.5 Challenges 9

**2. LITERATURE SURVEY 10**

2.1 Survey

**3. SYSTEM REQUIREMENTS SPECIFICATION**

3.1 Hardware requirements 13

3.2 Software requirements 13

3.3 Supporting python modules 13

**4. PROJECT DESCRIPTION**

4.1 Scope of the Project 14

4.2 Implementation 14

4.3 Integration Testing 20

4.3.1 Importing modules 20

4.3.2 Importing dataset 20

4.3.3 Importing user defined function 21

**5. RESULTS 22**

5.1 Logistic Regression 22

5.2 K-Nearest Neighbors Classifier 23

* 1. Support Vector Machine 24
  2. Naive Bayes Classifier 25
  3. Decision Tree 26
  4. Random Forest Classifier 27
  5. Gradient Boosting Classifier 28
  6. XGBoost Classifier 29

**6. CONCLUSION 30**

6.1 Conclusion 30

6.2 Future Enhancement 34

**APPENDICES 32**

Appendix 1 Sample Script

**REFERENCES**  **59**

**LIST OF FIGURES**

**FIGURE NO. TITLE PG NO.**

4.2.1 Implementation 14

4.2.2 Splitting 15

4.2.3 Preprocessing 16

4.2.4 Feature Extraction 19

**CHAPTER 1**

**INTRODUCTION**

**1.1 BACKGROUND**

Phishing attacks have become increasingly sophisticated, posing a significant threat to individuals and organizations. Phishing websites mimic legitimate sites to deceive users into disclosing sensitive information such as usernames, passwords, and financial details. Traditional methods of identifying phishing websites are often insufficient, as attackers continuously evolve their tactics. Therefore, there is a critical need for advanced techniques and tools to effectively detect and mitigate the risks associated with phishing websites.

**1.2 PROBLEM DESCRIPTION**

The primary objective of this project is to develop a robust and accurate system for the detection of phishing websites. The system should employ a combination of machine learning, data analysis, and feature engineering techniques to differentiate between legitimate and phishing websites. The detection model should be able to adapt to new and emerging phishing tactics, ensuring continuous protection against evolving threats.

**1.3 AIM AND OBJECTIVE**

* Develop a machine learning model for phishing website detection.
* Implement dynamic feature extraction and adaptation mechanisms.
* Ensure real-time processing for immediate threat response.
* Train the model on a diverse and extensive dataset of phishing websites.
* Conduct thorough testing and evaluation to measure the system's accuracy and effectiveness.

**1.4 SCOPE**

The focus of the project is on machine learning (ML) methods for network analysis of intrusion detection especially phishing websites attack.

**1.5 CHALLENGES**

The challenges faced during the project are as follows:

• Finding the appropriate dataset.

• Feature extraction required the study of various modules and understanding each module and getting the expected outcome from it.

**CHAPTER 2**

**LITERATURE SURVEY**

A literature survey is an insightful article that presents the existing information including considerable discoveries just as theoretical and methodological commitments to a specific topic.

Some of the downloaded papers are mentioned below.

**Phishing Website Detection(Adarsh Mandadi,2022)**

Phishing is an internet scam in which an attacker sends out fake messages that look to come from a trusted source. A URL or file will be included in the mail, which when clicked will steal personal information or infect a computer with a virus. Traditionally, phishing attempts were carried out through wide-scale spam campaigns that targeted broad groups of people indiscriminately. The goal was to get as many people to click on a link or open an infected file as possible. There are various approaches to detect this type of attack. One of the approaches is machine learning. The URL’s received by the user will be given input to the machine learning model then the algorithm will process the input and display the output whether it is phishing or legitimate. There are various ML algorithms like SVM, Neural Networks, Random Forest, Decision Tree, XG boost etc. that can be used to classify these URLs. The proposed approach deals with the Random Forest, Decision Tree classifiers. The proposed approach effectively classified the Phishing and Legitimate URLs with an accuracy of 87.0% and 82.4% for Random Forest and decision tree classifiers respectively.

**URL-based Phishing Websites Detection via Machine Learning(Quasem Abu- Al Haija,2021)**

Phishing is a cyber security attack that is used to trick victim users to provide sensitive information or deploy malicious software on their infrastructure. Depending on the target system and users, these attacks can inflict severe negative impacts on the system. Therefore, researchers have been working on developing phishing detection and prevention techniques to thwart these attacks. In this paper, we present an efficient phishing websites detection system that analyzes the phishing websites URL addresses to learn data patterns that can identify authentic and phishing websites. Our system employs machine learning techniques such as neural networks and decision trees to learn data patterns in websites URLs. We evaluate our system on a recent phishing websites dataset using classification accuracy as a performance indicator. Our best result shows that decision trees models provide 90 % classification accuracy on the almost balanced-class dataset.

**Logistic Regression based Machine Learning Technique for Phishing Website Detection(T.R Soumya ,2022)**

Nowadays, many people start switching from offline to online to save their precious time. They started buying products online and made their payments through online transactions across websites. These online buyers are asked to provide details such as their name, address, location, passwords, and other essential bank details on that particular website. The unaware online buyer got caught in these sites, which leads to a process of phishing. They are called phishing websites. This research work has proposed an efficient prediction method based on the machine learning technique to analyse and predict these phishing websites. Novel classification algorithm and techniques are used to analyse and extract the datasets that might maliciously cause phishing. The essential traits are helpful to identify these types of phishing sites such as domain, URL and encryption technique of a website while detecting malicious data. This research work will use a logistic regression algorithm for detecting the phishing website. A logistic regression algorithm is used to provide better performance than the traditional classification algorithm. To protect user sensitive information and for effective, secure transaction payments, many E-commerce enterprises are using this application to stay on the safer side.

**CHAPTER 3**

**SYSTEM REQUIREMENTS SPECIFICATION**

**3.1 Hardware Requirements:**

• Processor CPU – Intel i5 or Intel i7

• Hard Disk capacity – 256 GB Hard Disk.

• RAM - 4GB minimum

**3.2 Software Requirements :**

• Programming language - Python

• Operating system - Windows 10 or Windows 11

• IDE - Anaconda , iPython version 3.x

**3.3 Supporting Python Modules:**

1. Ipaddress
2. Re
3. urllib.request
4. BeautifulSoup
5. Socket
6. Requests
7. Whois

**CHAPTER 4**

**PROJECT DESCRIPTION**

**4.1 Scope of the Project**

1. Investigate and understand prevalent phishing techniques, including email-based, social engineering, and malicious websites.
2. Analyse how phishing attacks evolve and adapt over time.
3. Implement and compare various machine learning models, such as Gradient Boosting Classifier, Random Forests, Support Vector Machines, and Logistic Regression, for phishing website detection.
4. Experiment with ensemble methods to improve overall model performance.

**4.2 Implementation**

This chapter of the report illustrates the approach employed to classify the URLs as either phishing or legitimate. The methodology involves building a training set. The training set is used for training a machine learning model, i.e., the classifier. Fig 4.2.1 shows the diagrammatic representation of the implementation.

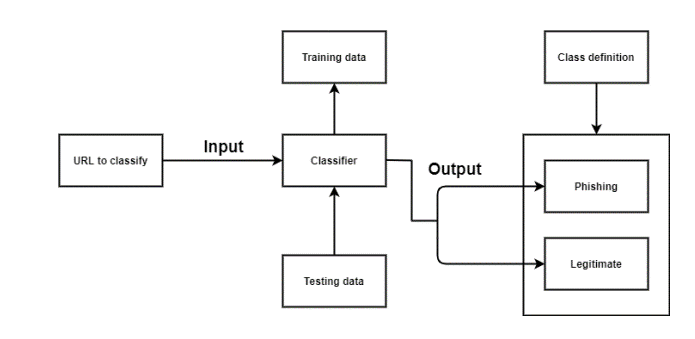


Figure 4.2.1: Implementation

* The data set is large, so working with it is intriguing
* The number of features in the data set is 30 giving a wide range of features making the predictions a little more accurate.

• **Splitting**: the dataset into training part of dataset and testing part of dataset. The dataset was split into training and testing dataset with 80% for training and 20% for testing using the “train test split” method. The splitting was done after assigning the dependent variables and independent variables. Fig 4.2.2 shows the diagrammatic representation of the implementation.

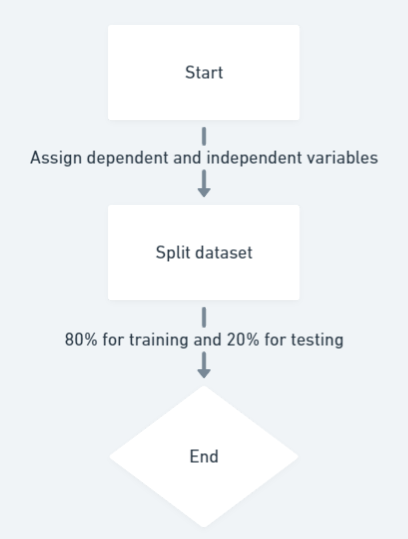


Figure 4.2.2 : Splitting

• **Preprocessing**: Preprocessing involves filling the missing data or removing the missing data and getting a clean dataset .But the dataset chosen was already preprocessed and did not require any further preprocessing from my end. The only step to be performed in preprocessing was feature scaling. Fig 4.2.3 shows the diagrammatic representation of the .

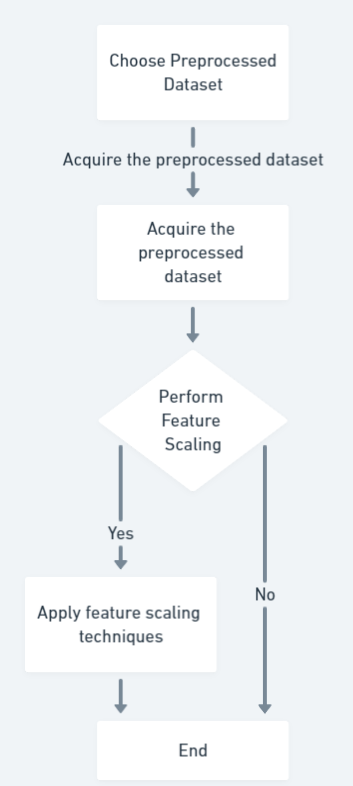


Figure : 4.2.3 : Preprocessing

• **Feature scaling**: Feature Scaling is a procedure to normalize the independent variable present in the information in a fixed range. It is performed during the data pre-processing to deal with varying magnitudes. There are two ways of feature scaling – Normalization and Standardization. The project uses standardization feature scaling methods. The variables should be put in the same scale, else one variable might dominate others hence might affect the result.

Standardization: Standardization is another scaling procedure where the values are based on the mean with a unit standard deviation. This implies the mean of that attribute gets zero and the resultant distribution has a unit standard deviation.

Xstd = (x – mean(x))/ standard deviation(x)

Normalization: Normalization is a scaling method where values are moved and rescaled so they wind up going somewhere in the range of 0 and 1. It is otherwise called Min-Max scaling.

Xnorm = (x – min(x))/(max(x) – min(x))

The project uses StandardScaler. It fits and transforms only the independent variables. The dependent variables need not be scaled in classification method. The dummy variables which we get from categorical data may or may not be scaled depending on context.

• **Feature extraction**: Phishing website detection relies on extracting features that distinguish between legitimate and malicious URLs. Various characteristics of the URL we can use to classify the URL is legitimate or phishing.

**Using IP :** This feature checks whether the given URL contains an IP address directly. If an IP address is found, it suggests of phishing, returning -1; otherwise, it returns 1.

**Long URL :** This feature assesses the length of the URL. If the URL is shorter than 54 characters, it returns 1. If it falls between 54 and 75 characters, it returns 0. URLs longer than 75 characters are considered suspicious, returning -1.

**Short URL :** This feature uses regular expressions to check if the URL uses a URL shortening service. If a match is found, indicating a short URL, it returns -1; otherwise, it returns 1.

**Symbol@ :** This will check the occurrence of the “@” symbol in the URL

return -1: If the '@' symbol is found in the URL, the method returns -1

return 1: If the '@' symbol is not found in the URL, the method returns 1

**Redirecting// :** This line checks if the URL contains more than six consecutive forward slashes ('//').

return -1: If the condition is met (i.e., more than six consecutive slashes are found), the method returns -1, suggesting a potential phishing indicator.

return 1 : If the condition is not met, meaning there are six or less consecutive slashes, the method returns 1.

**AgeofDomain :** AgeofDomain method calculates the age of the domain in months

If the age of the domain is under six months, it return -1 and suggest phishing website

If the alert( function is found in the HTML content, the method returns 1ge of the domain is greater than or equal to six months, it return 1.

**AbnormalURL :** This method compares the HTML content of the webpage .

If there is a match, it returns 1, indicating a potential abnormality.

If there is no match or if there are issues in accessing the response text or WHOIS information, it returns -1, suggesting a higher suspicion level.

**NonStdPort** : NonStdPort method checks if a non-standard port is specified in the domain

If a non-standard port is detected, it returns -1, indicating a potential suspicious activity.

If no port or a standard port is detected, it returns 1**.**

**-**From these characteristics of the URL we can classify the URL as legitimate or phishing website.

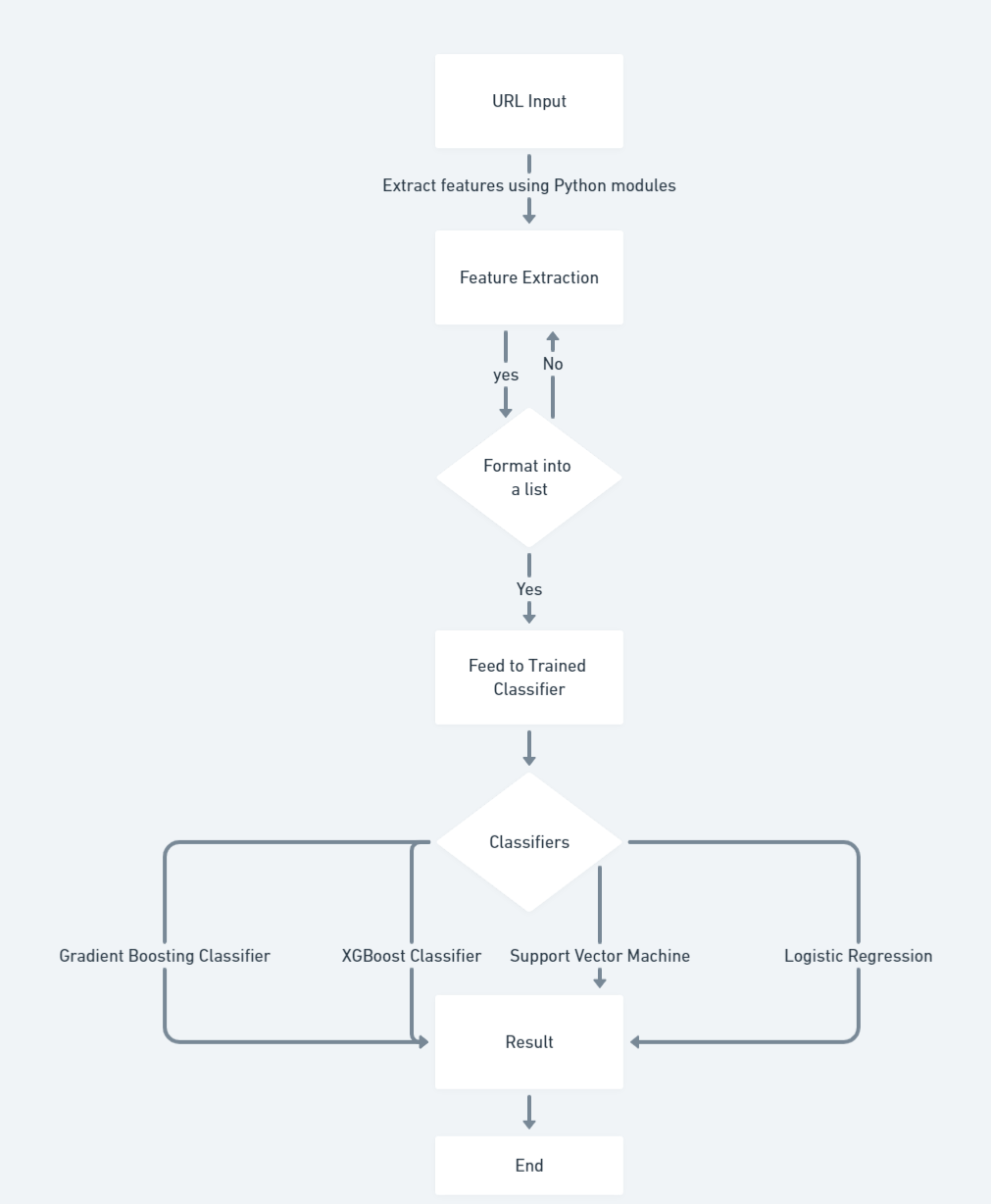
****

Figure 4.2.4: Feature Extraction

**4.3 Integration Testing**

Integration Testing is a testing approach where the units of the modules are integrated and then investigated to check regardless of whether they are fit to be utilized.

|  |  |
| --- | --- |
| Test Name | “Importing modules” |
| Input | Import “module” statements |
| Expected output | The module to be imported |
| Actual Output | The module was imported and could be used |
| Remark | Success |

**4.3.1 Importing modules**

**4.3.2 Importing dataset**

|  |  |
| --- | --- |
| Test Name | “Importing dataset” |
| Input | Import “dataset” statement |
| Expected output | The dataset to be imported |
| Actual output | The dataset was imported and could be used |
| Remark | Success |

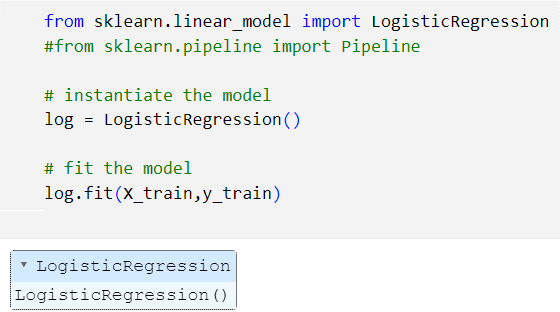
**4.3.2 Importing user defined function**

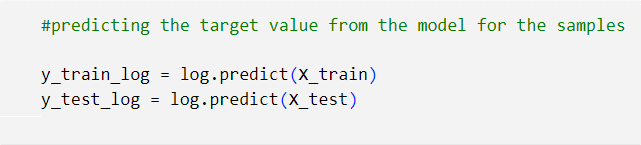
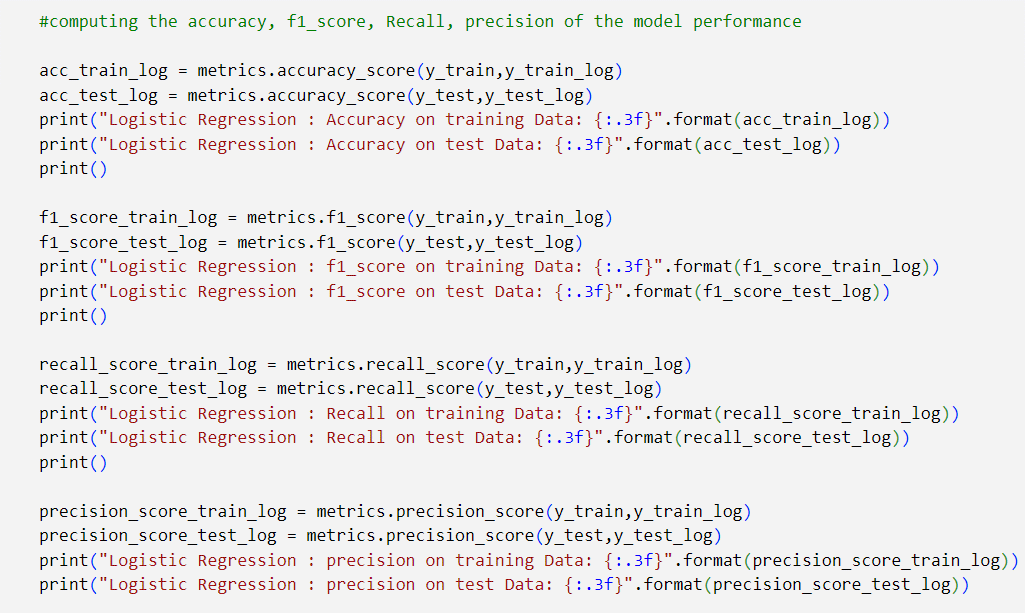
|  |  |
| --- | --- |
| Test Name | “Importing user defined function” |
| Input | Import “extraction” function |
| Expected output | The function to be imported that returns a list |
| Actual output | The function was imported and returned the list as expected |
| Remark | Success |

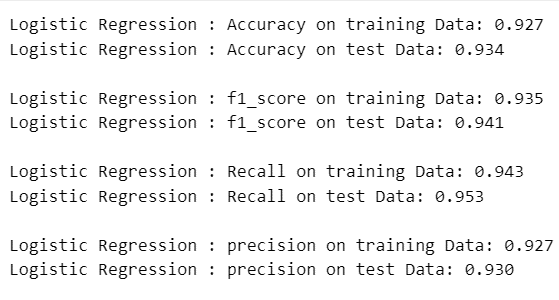
**CHAPTER 5**

**RESULTS**

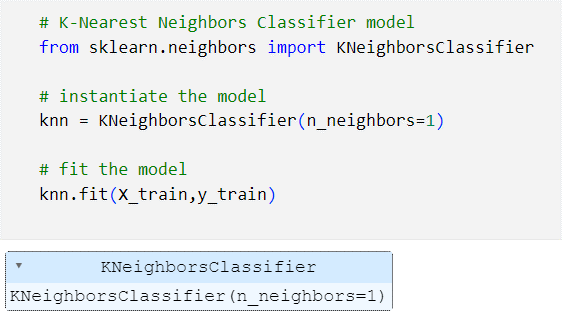
**5.1 Logistic Regression**

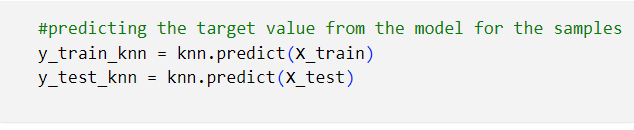
****

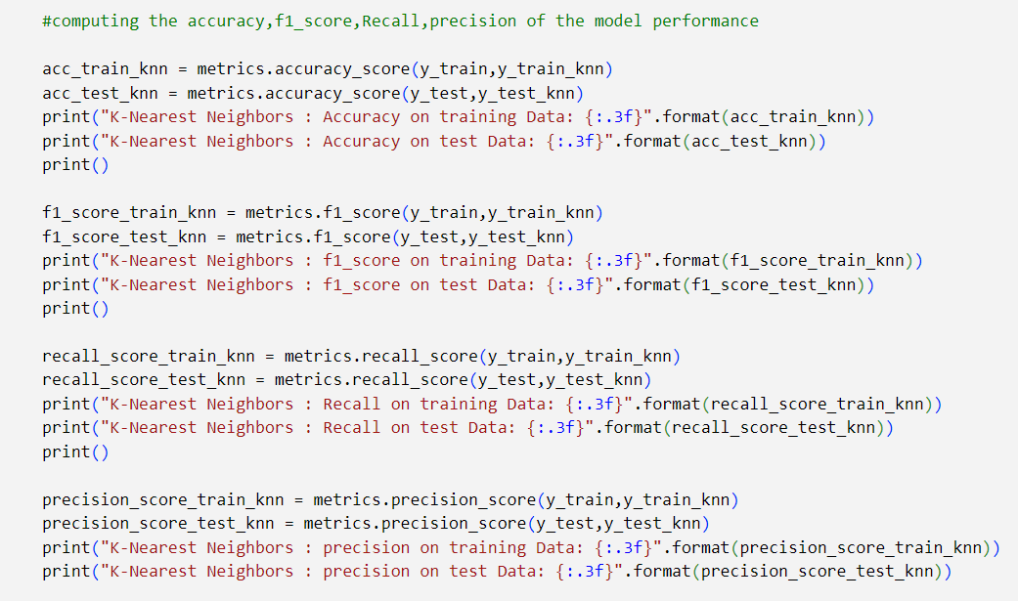
****

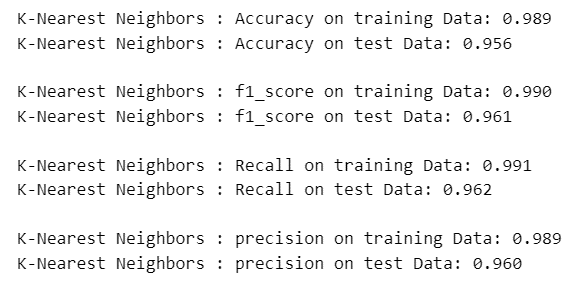
****

**5.2 K-Nearest Neighnors Classifier**

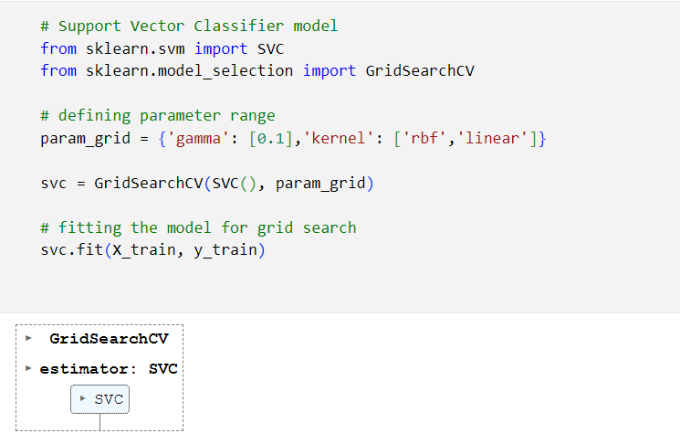
****

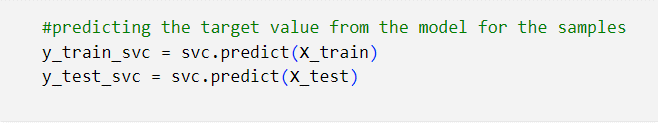
****

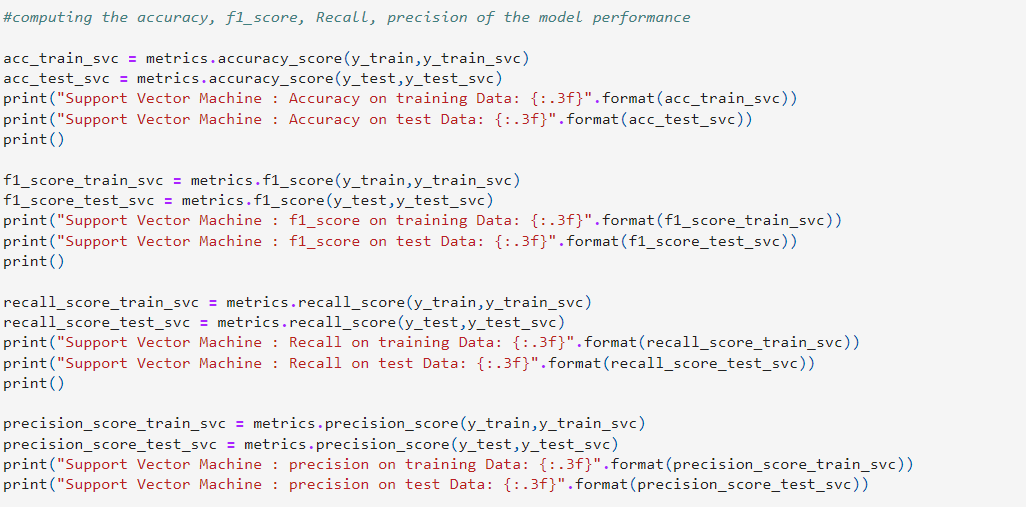
****

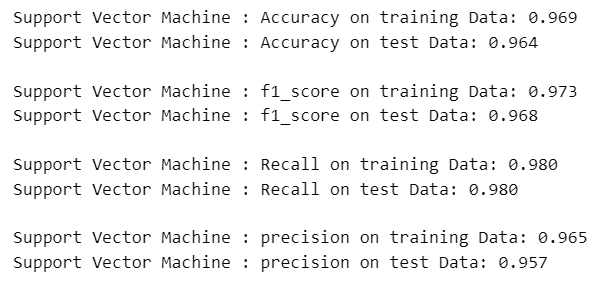
****

**5.3 Support Vector Machine**

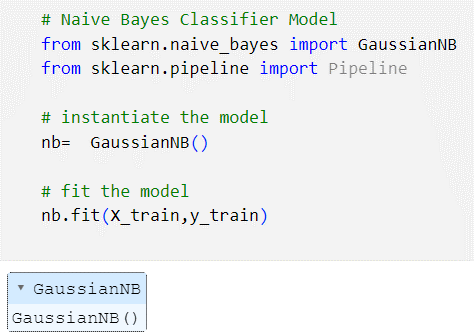
****

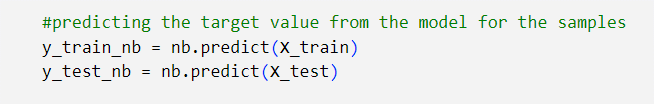
****

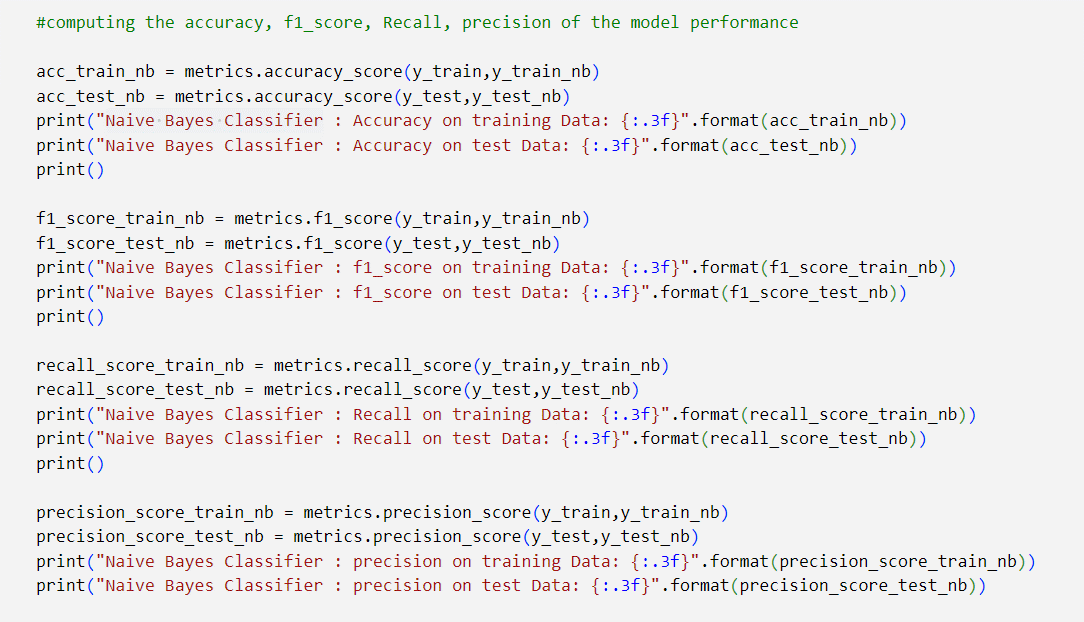
****

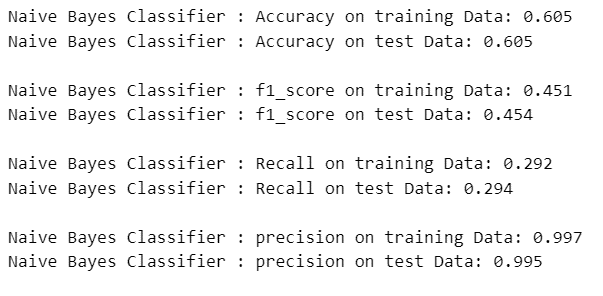
****

**5.4 Naive Bayes Classifier**

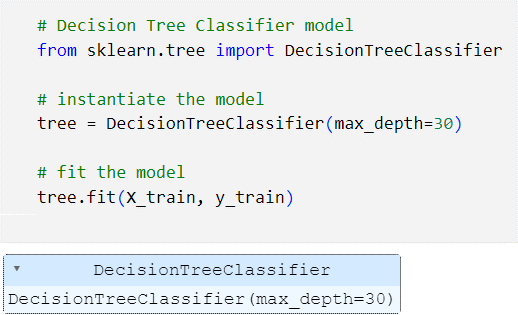
****

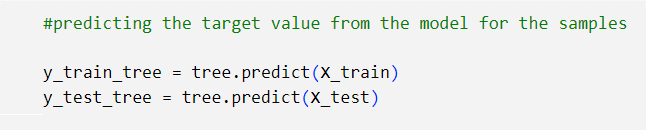
****

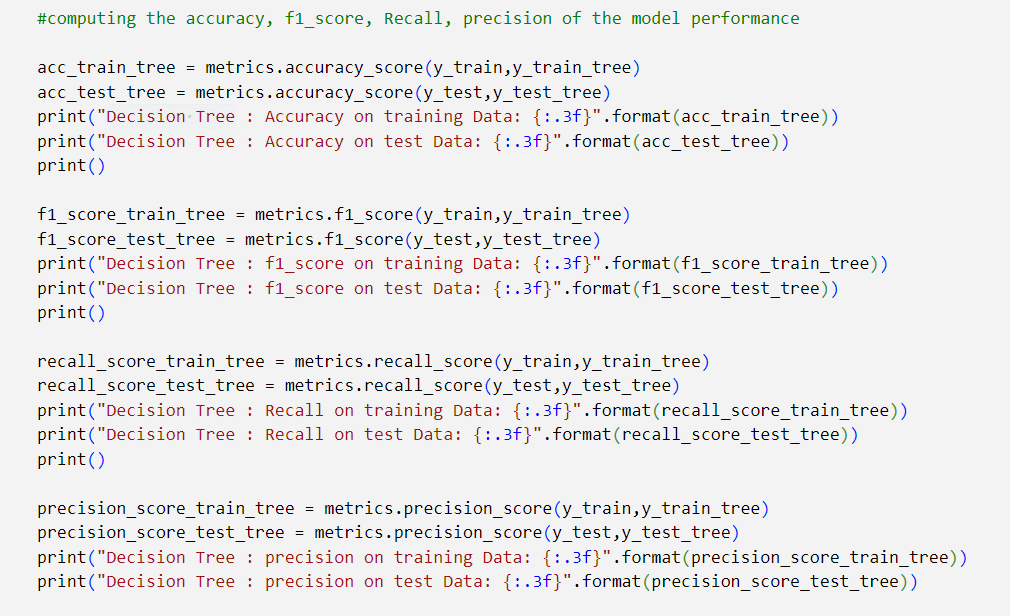
****

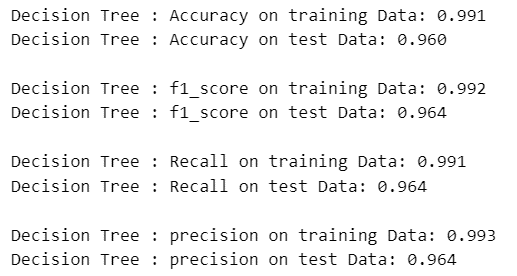
****

**5.5 Decision Tree**

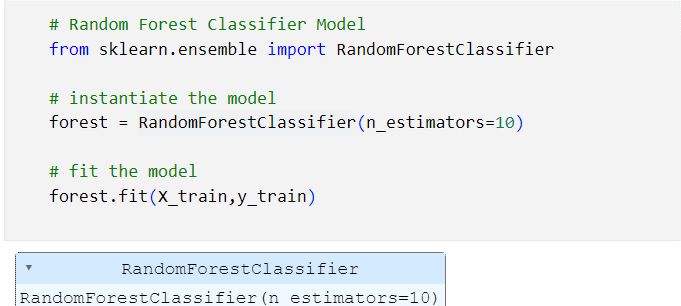
****

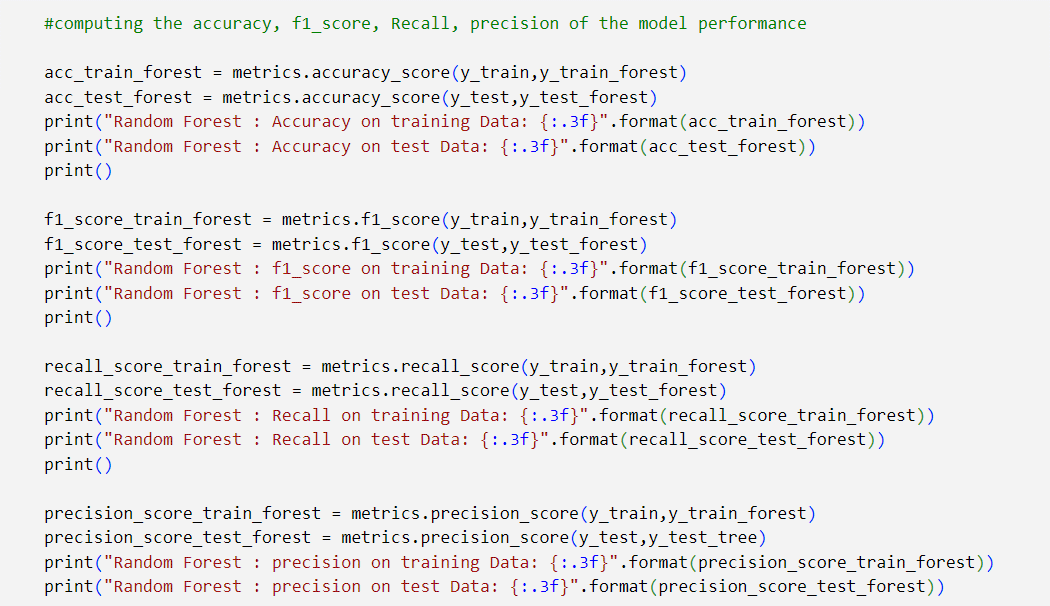
****

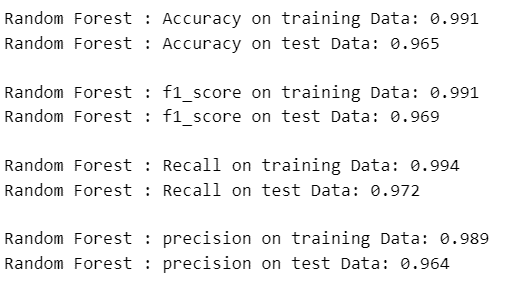
****

****

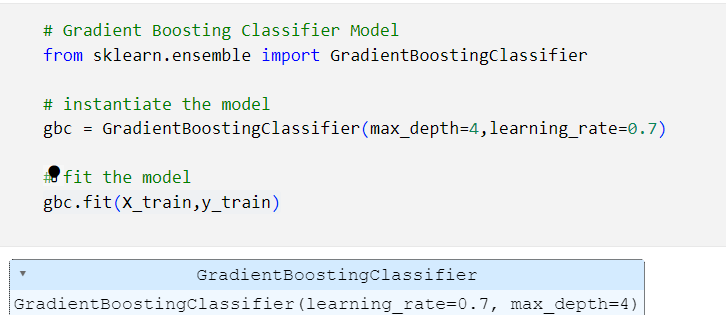
**5.6 Random Forest Classifier**

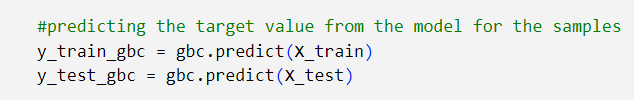
****

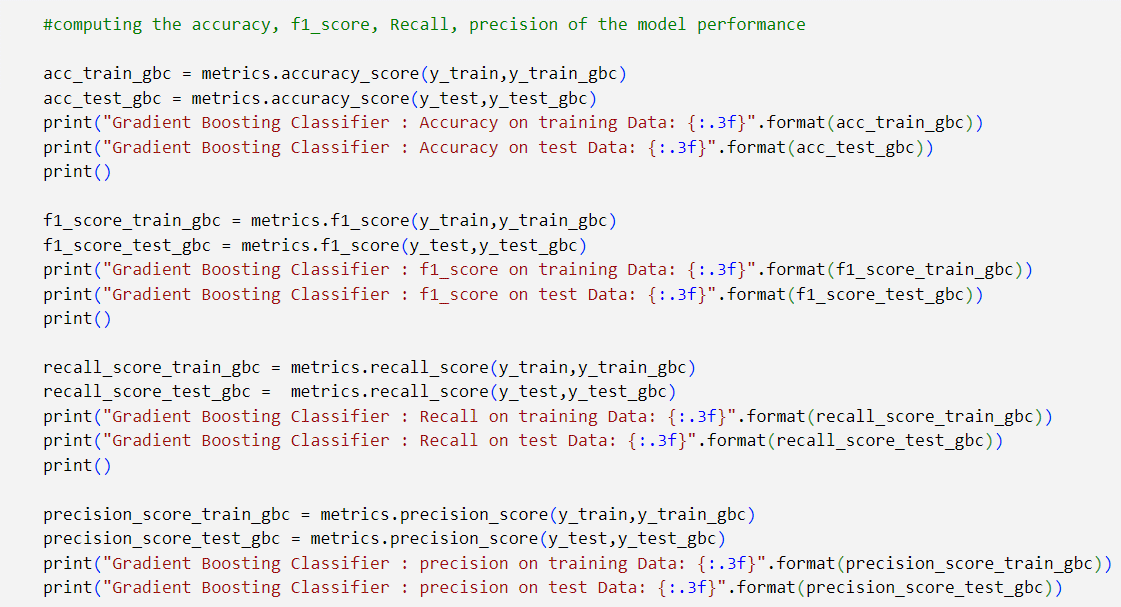
**-**

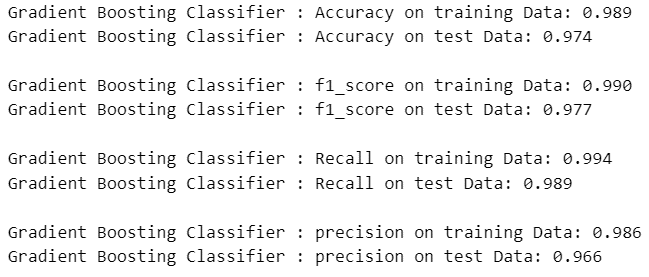
****

**5.7 Gradient Boosting Classifier**

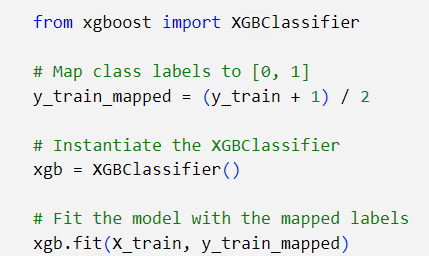
****

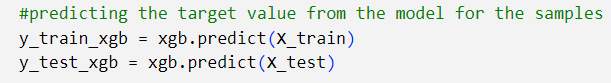
****

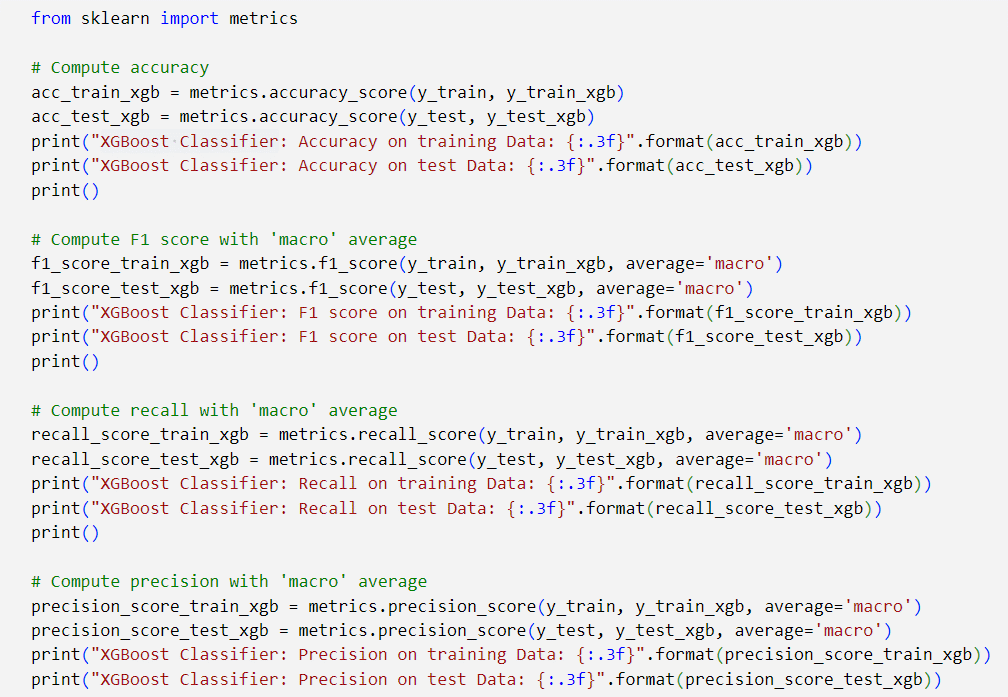
****

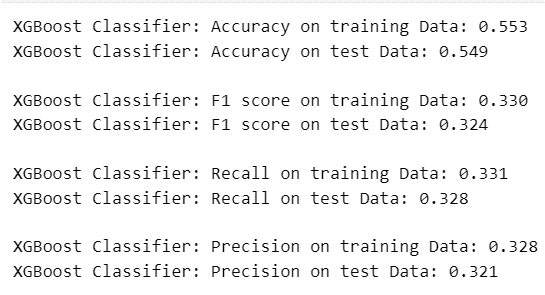
****

**5.8 XGBoost Classifier**

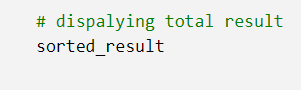
****

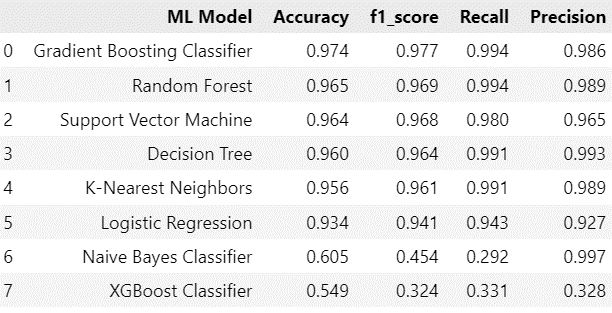
****

****

****

**Displaying total result**

****

****

**CHAPTER 6**

**CONCLUSION AND FUTURE ENHANCEMENTS**

**6.1 Conclusion**

In conclusion, the phishing website detection project has successfully addressed the critical need for a robust and adaptive system to combat the escalating threat of phishing attacks. The comprehensive approach, integrating machine learning, real-time processing, user interface design, and adaptability, has resulted in a sophisticated solution capable of providing effective protection to users and organizations.

**6.2 Future Enhancements**

Further work can be done to enhance the model by using ensembling models to get greater accuracy score. Ensemble methods is a ML technique that combines many base models to generate an optimal predictive model. Further reaching future work would be combining multiple classifiers, trained on different aspects of the same training set, into a single classifier that may provide a more robust prediction than any of the single classifiers on their own. The project can also include other variants of phishing like smishing, vishing, etc. to complete the system. Looking even further out, the methodology needs to be evaluated on how it might handle collection growth. The collections will ideally grow incrementally over time so there will need to be a way to apply a classifier incrementally to the new data, but also potentially have this classifier receive feedback that might modify it over time.

**APPENDICES**

**feature.py**

import ipaddress

import re

import urllib.request

from bs4 import BeautifulSoup

import socket

import requests

from googlesearch import search

import whois

from datetime import date, datetime

import time

from dateutil.parser import parse as date\_parse

from urllib.parse import urlparse

class FeatureExtraction:

features = []

def \_\_init\_\_(self,url):

self.features = []

self.url = url

self.domain = ""

self.whois\_response = ""

self.urlparse = ""

self.response = ""

self.soup = ""

try:

self.response = requests.get(url)

self.soup = BeautifulSoup(response.text, 'html.parser')

except:

pass

try:

self.urlparse = urlparse(url)

self.domain = self.urlparse.netloc

except:

pass

try:

self.whois\_response = whois.whois(self.domain)

except:

pass

self.features.append(self.UsingIp())

self.features.append(self.longUrl())

self.features.append(self.shortUrl())

self.features.append(self.symbol())

self.features.append(self.redirecting())

self.features.append(self.prefixSuffix())

self.features.append(self.SubDomains())

self.features.append(self.Hppts())

self.features.append(self.DomainRegLen())

self.features.append(self.Favicon())

self.features.append(self.NonStdPort())

self.features.append(self.HTTPSDomainURL())

self.features.append(self.RequestURL())

self.features.append(self.AnchorURL())

self.features.append(self.LinksInScriptTags())

self.features.append(self.ServerFormHandler())

self.features.append(self.InfoEmail())

self.features.append(self.AbnormalURL())

self.features.append(self.WebsiteForwarding())

self.features.append(self.StatusBarCust())

self.features.append(self.DisableRightClick())

self.features.append(self.UsingPopupWindow())

self.features.append(self.IframeRedirection())

self.features.append(self.AgeofDomain())

self.features.append(self.DNSRecording())

self.features.append(self.WebsiteTraffic())

self.features.append(self.PageRank())

self.features.append(self.GoogleIndex())

self.features.append(self.LinksPointingToPage())

self.features.append(self.StatsReport())

# 1.UsingIp

def UsingIp(self):

try:

ipaddress.ip\_address(self.url)

return -1

except:

return 1

# 2.longUrl

def longUrl(self):

if len(self.url) < 54:

return 1

if len(self.url) >= 54 and len(self.url) <= 75:

return 0

return -1

# 3.shortUrl

def shortUrl(self):

match = re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|' 'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|' 'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|'

'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|' 'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|'

'q\.gs|is\.gd|po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|' 'x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.me|v\.gd|tr\.im|link\.zip\.net', self.url)

if match:

return -1

return 1

# 4.Symbol@

def symbol(self):

if re.findall("@",self.url):

return -1

return 1

# 5.Redirecting//

def redirecting(self):

if self.url.rfind('//')>6:

return -1

return 1

# 6.prefixSuffix

def prefixSuffix(self):

try:

match = re.findall('\-', self.domain)

if match:

return -1

return 1

except:

return -1

# 7.SubDomains

def SubDomains(self):

dot\_count = len(re.findall("\.", self.url))

if dot\_count == 1:

return 1

elif dot\_count == 2:

return 0

return -1

# 8.HTTPS

def Hppts(self):

try:

https = self.urlparse.scheme

if 'https' in https:

return 1

return -1

except:

return 1

# 9.DomainRegLen

def DomainRegLen(self):

try:

expiration\_date = self.whois\_response.expiration\_date

creation\_date = self.whois\_response.creation\_date

try:

if(len(expiration\_date)):

expiration\_date = expiration\_date[0]

except:

pass

try:

if(len(creation\_date)):

creation\_date = creation\_date[0]

except:

pass

age = (expiration\_date.year-creation\_date.year)\*12+ (expiration\_date.month-creation\_date.month)

if age >=12:

return 1

return -1

except:

return -1

# 10. Favicon

def Favicon(self):

try:

for head in self.soup.find\_all('head'):

for head.link in self.soup.find\_all('link', href=True):

dots = [x.start(0) for x in re.finditer('\.', head.link['href'])]

if self.url in head.link['href'] or len(dots) == 1 or domain in head.link['href']:

return 1

return -1

except:

return -1

# 11. NonStdPort

def NonStdPort(self):

try:

port = self.domain.split(":")

if len(port)>1:

return -1

return 1

except:

return -1

# 12. HTTPSDomainURL

def HTTPSDomainURL(self):

try:

if 'https' in self.domain:

return -1

return 1

except:

return -1

# 13. RequestURL

def RequestURL(self):

try:

for img in self.soup.find\_all('img', src=True):

dots = [x.start(0) for x in re.finditer('\.', img['src'])]

if self.url in img['src'] or self.domain in img['src'] or len(dots) == 1:

success = success + 1

i = i+1

for audio in self.soup.find\_all('audio', src=True):

dots = [x.start(0) for x in re.finditer('\.', audio['src'])]

if self.url in audio['src'] or self.domain in audio['src'] or len(dots) == 1:

success = success + 1

i = i+1

for embed in self.soup.find\_all('embed', src=True):

dots = [x.start(0) for x in re.finditer('\.', embed['src'])]

if self.url in embed['src'] or self.domain in embed['src'] or len(dots) == 1:

success = success + 1

i = i+1

for iframe in self.soup.find\_all('iframe', src=True):

dots = [x.start(0) for x in re.finditer('\.', iframe['src'])]

if self.url in iframe['src'] or self.domain in iframe['src'] or len(dots) == 1:

success = success + 1

i = i+1

try:

percentage = success/float(i) \* 100

if percentage < 22.0:

return 1

elif((percentage >= 22.0) and (percentage < 61.0)):

return 0

else:

return -1

except:

return 0

except:

return -1

# 14. AnchorURL

def AnchorURL(self):

try:

i,unsafe = 0,0

for a in self.soup.find\_all('a', href=True):

if "#" in a['href'] or "javascript" in a['href'].lower() or "mailto" in a['href'].lower() or not (url in a['href'] or self.domain in a['href']):

unsafe = unsafe + 1

i = i + 1

try:

percentage = unsafe / float(i) \* 100

if percentage < 31.0:

return 1

elif ((percentage >= 31.0) and (percentage < 67.0)):

return 0

else:

return -1

except:

return -1

except:

return -1

# 15. LinksInScriptTags

def LinksInScriptTags(self):

try:

i,success = 0,0

for link in self.soup.find\_all('link', href=True):

dots = [x.start(0) for x in re.finditer('\.', link['href'])]

if self.url in link['href'] or self.domain in link['href'] or len(dots) == 1:

success = success + 1

i = i+1

for script in self.soup.find\_all('script', src=True):

dots = [x.start(0) for x in re.finditer('\.', script['src'])]

if self.url in script['src'] or self.domain in script['src'] or len(dots) == 1:

success = success + 1

i = i+1

try:

percentage = success / float(i) \* 100

if percentage < 17.0:

return 1

elif((percentage >= 17.0) and (percentage < 81.0)):

return 0

else:

return -1

except:

return 0

except:

return -1

# 16. ServerFormHandler

def ServerFormHandler(self):

try:

if len(self.soup.find\_all('form', action=True))==0:

return 1

else :

for form in self.soup.find\_all('form', action=True):

if form['action'] == "" or form['action'] == "about:blank":

return -1

elif self.url not in form['action'] and self.domain not in form['action']:

return 0

else:

return 1

except:

return -1

# 17. InfoEmail

def InfoEmail(self):

try:

if re.findall(r"[mail\(\)|mailto:?]", self.soap):

return -1

else:

return 1

except:

return -1

# 18. AbnormalURL

def AbnormalURL(self):

try:

if self.response.text == self.whois\_response:

return 1

else:

return -1

except:

return -1

# 19. WebsiteForwarding

def WebsiteForwarding(self):

try:

if len(self.response.history) <= 1:

return 1

elif len(self.response.history) <= 4:

return 0

else:

return -1

except:

return -1

# 20. StatusBarCust

def StatusBarCust(self):

try:

if re.findall("<script>.+onmouseover.+</script>", self.response.text):

return 1

else:

return -1

except:

return -1

# 21. DisableRightClick

def DisableRightClick(self):

try:

if re.findall(r"event.button ?== ?2", self.response.text):

return 1

else:

return -1

except:

return -1

# 22. UsingPopupWindow

def UsingPopupWindow(self):

try:

if re.findall(r"alert\(", self.response.text):

return 1

else:

return -1

except:

return -1

# 23. IframeRedirection

def IframeRedirection(self):

try:

if re.findall(r"[<iframe>|<frameBorder>]", self.response.text):

return 1

else:

return -1

except:

return -1

# 24. AgeofDomain

def AgeofDomain(self):

try:

creation\_date = self.whois\_response.creation\_date

try:

if(len(creation\_date)):

creation\_date = creation\_date[0]

except:

pass

today = date.today()

age = (today.year-creation\_date.year)\*12+(today.month-creation\_date.month)

if age >=6:

return 1

return -1

except:

return -1

# 25. DNSRecording

def DNSRecording(self):

try:

creation\_date = self.whois\_response.creation\_date

try:

if(len(creation\_date)):

creation\_date = creation\_date[0]

except:

pass

today = date.today()

age = (today.year-creation\_date.year)\*12+(today.month-creation\_date.month)

if age >=6:

return 1

return -1

except:

return -1

# 26. WebsiteTraffic

def WebsiteTraffic(self):

try:

rank = BeautifulSoup(urllib.request.urlopen("http://data.alexa.com/data?cli=10&dat=s&url=" + url).read(), "xml").find("REACH")['RANK']

if (int(rank) < 100000):

return 1

return 0

except :

return -1

# 27. PageRank

def PageRank(self):

try:

prank\_checker\_response = requests.post("https://www.checkpagerank.net/index.php", {"name": self.domain}) global\_rank = int(re.findall(r"Global Rank: ([0-9]+)", rank\_checker\_response.text)[0])

if global\_rank > 0 and global\_rank < 100000:

return 1

return -1

except:

return -1

# 28. GoogleIndex

def GoogleIndex(self):

try:

site = search(self.url, 5)

if site:

return 1

else:

return -1

except:

return 1

# 29. LinksPointingToPage

def LinksPointingToPage(self):

try:

number\_of\_links = len(re.findall(r"<a href=", self.response.text))

if number\_of\_links == 0:

return 1

elif number\_of\_links <= 2:

return 0

else:

return -1

except:

return -1

# 30. StatsReport

def StatsReport(self):

try:

url\_match = re.search(

'at\.ua|usa\.cc|baltazarpresentes\.com\.br|pe\.hu|esy\.es|hol\.es|sweddy\.com|myjino\.ru|96\.lt|ow\.ly', url)

ip\_address = socket.gethostbyname(self.domain)

ip\_match = re.search('146\.112\.61\.108|213\.174\.157\.151|121\.50\.168\.88|192\.185\.217\.116|78\.46\.211\.158|181\.174\.165\.13|46\.242\.145\.103|121\.50\.168\.40|83\.125\.22\.219|46\.242\.145\.98|''107\.151\.148\.44|107\.151\.148\.107|64\.70\.19\.203|199\.184\.144\.27|107\.151\.148\.108|107\.151\.148\.109|119\.28\.52\.61|54\.83\.43\.69|52\.69\.166\.231|216\.58\.192\.225|' '118\.184\.25\.86|67\.208\.74\.71|23\.253\.126\.58|104\.239\.157\.210|175\.126\.123\.219|141\.8\.224\.221|10\.10\.10\.10|43\.229\.108\.32|103\.232\.215\.140|69\.172\.201\.153|' '216\.218\.185\.162|54\.225\.104\.146|103\.243\.24\.98|199\.59\.243\.120|31\.170\.160\.61|213\.19\.128\.77|62\.113\.226\.131|208\.100\.26\.234|195\.16\.127\.102|195\.16\.127\.157|' '34\.196\.13\.28|103\.224\.212\.222|172\.217\.4\.225|54\.72\.9\.51|192\.64\.147\.141|198\.200\.56\.183|23\.253\.164\.103|52\.48\.191\.26|52\.214\.197\.72|87\.98\.255\.18|209\.99\.17\.27|' '216\.38\.62\.18|104\.130\.124\.96|47\.89\.58\.141|78\.46\.211\.158|54\.86\.225\.156|54\.82\.156\.19|37\.157\.192\.102|204\.11\.56\.48|110\.34\.231\.42', ip\_address)

if url\_match:

return -1

elif ip\_match:

return -1

return 1

except:

return 1

def getFeaturesList(self):

return self.features

**REFERENCES**

[1] A. Mandadi, S. Boppana, V. Ravella and R. Kavitha, "Phishing Website Detection Using Machine Learning," 2022 IEEE 7th International conference for Convergence in Technology (I2CT), Mumbai, India, 2022, pp. 1-4, doi: 10.1109/I2CT54291.2022.9824801.

[2] M. M. Vilas, K. P. Ghansham, S. P. Jaypralash and P. Shila, "Detection of Phishing Website Using Machine Learning Approach," 2019 4th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT), Mysuru, India, 2019, pp. 384-389, doi: 10.1109/ICEECCOT46775.2019.9114695.

[3] T. R. Soumya, R. P, N. A. Rosy, N. Pughazendi, S. Padmapriya and R. Khilar, "Logistic Regression based Machine Learning Technique for Phishing Website Detection," 2022 4th International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2022, pp. 683-686, doi: 10.1109/ICIRCA54612.2022.9985643.

[4] J. Rashid, T. Mahmood, M. W. Nisar and T. Nazir, "Phishing Detection Using Machine Learning Technique," 2020 First International Conference of Smart Systems and Emerging Technologies (SMARTTECH), Riyadh, Saudi Arabia, 2020, pp. 43-46, doi: 10.1109/SMART-TECH49988.2020.00026.

[5] A. N. S. Charan, Y. -H. Chen and J. -L. Chen, "Phishing Websites Detection using Machine Learning with URL Analysis," 2022 IEEE World Conference on Applied Intelligence and Computing (AIC), Sonbhadra, India, 2022, pp. 808-812, doi: 10.1109/AIC55036.2022.9848895.

[6] M. H. Alkawaz, S. J. Steven, A. I. Hajamydeen and R. Ramli, "A Comprehensive Survey on Identification and Analysis of Phishing Website based on Machine Learning Methods," 2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), Penang, Malaysia, 2021, pp. 82-87, doi: 10.1109/ISCAIE51753.2021.9431794.