Cloud Based Phishing Detection

A major project report submitted in partial fulfilment of the requirement for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering

Submitted by

Paras Sharma (201133)

Rahul (201451)

Under the guidance & supervision of

Dr. Anita

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology,

Waknaghat, Solan - 173234 (India)

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: .....

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: _____ **Department:** _____ **Enrolment No** _____

Contact No. _____ **E-mail.** _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

_____

_____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at .....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

<table>
<tr><th>Copy Received on</th><th>Excluded</th><th>Similarity Index (%)</th><th colspan="2">Generated Plagiarism Report Details (Title, Abstract & Chapters)</th></tr>
<tr><td rowspan="2"></td><td rowspan="4">• All Preliminary Pages
• Bibliography/Images/Quotes
• 14 Words String</td><td rowspan="2"></td><td>Word Counts</td><td></td></tr>
<tr><td rowspan="2">Report Generated on</td><td>Character Counts</td><td></td></tr>
<tr><td rowspan="2"></td><td>Submission ID</td><td>Total Pages Scanned</td><td></td></tr>
<tr><td></td><td>File Size</td><td></td></tr>
</table>

Checked by
Name & Signature **Librarian**

.....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

CERTIFICATE

This is to certify that the work which is being presented in the project report titled **"Cloud Based Phishing Detection"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science And Engineering** and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by **Paras Sharma, 201133** and **Rahul, 201451** during the period from August 2023 to May 2024 under the supervision of **Dr. Anita**, Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.

I also authenticate that I have carried out the above-mentioned project work under the proficiency stream **Information Security.**

Paras Sharma 201133

Rahul 201451

The above statement made is correct to the best of my knowledge.

Assistant Professor (Senior Grade)

Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat.

Candidate's Declaration

I hereby declare that the work presented in this report entitled **'Cloud Based Phishing Detection'** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering / Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from August 2023 to May 2024 under the supervision of **Dr.Anita** (Assistant Professor (Senior Grade), Department of Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature with Date)

Student Name: Paras Sharma
Roll No.: 201133

(Student Signature with Date)

Student Name: Rahul
Roll No.: 201451

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature with Date)
Supervisor Name: Dr.Anita
Designation: Assistant Professor
Department: CSE
Dated:

ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to almighty God for His divine blessing makes us possible to complete the project work successfully.

I really grateful and wish my profound my indebtedness to Supervisor **Dr. Anita , Assistant Professor (Senior Grade),** Department of CSE Jaypee University of Information Technology, Waknaghat. Deep Knowledge & keen interest of my supervisor in the field of **"Information Security"** to carry out this project. Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stage have made it possible to complete this project.

I would like to express my heartiest gratitude to **Dr. Anita** Department of CSE, for his kind help to finish my project.

I would also generously welcome each one of those individuals who have helped me straight forwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patients of my parents.

Paras Sharma 201133

Rahul 201451

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

List of Abbreviations, Symbols or Nomenclature

URL	Uniform Resource Locators
IP	Internet Protocol
HTTPS	Hypertext Transfer Protocol Secure
GSB	Google Safe Browsing
LSTM	Long Short-Term Memory
CNN	Convolutional Neural Network
KNN	K-Nearest Neighbor
WC-PAD	Web Crawler based Phishing Attack Detector
RFC	Random Forest Classifier
DT	Decision Tree
DFD	Data flow diagram
CM	Confusion Matrix

ABSTRACT

In today's digital environment, more and more people are communicating online, which has increased the threat of phishing attacks. This type of cybercrime involves fraudulent attempts to obtain personal information from unknown victims, including passwords, credit card numbers, and personal information which can lead to serious problems like identity theft and loss of funds and can be prevented in real time Due to modern phishing sophistication methods, traditional code breaking and signature based signature detection methods are not applicable better at effectively blocking malicious organizations against ever-changing tactics, including email spoofing, fake websites and social engineering

The goal of project is to develop a robust and efficient system that can detect and mitigate phishing attacks in real time. Rule-based solutions are not an effective enough strategy to combat the rapidly changing phishing techniques. Machine learning enables increased detection accuracy by learning from historical attack data and identifying subtle patterns that may not be apparent to human observer

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

Phishing is a form of social engineering attack [1] commonly used to steal user data, including login credentials and credit card numbers. It occurs when an attacker pretends to be trustworthy and tricks the victim into opening an email and url , instant messaging, or text message. The recipient is then tricked into clicking on a malicious link, potentially installing malware, freezing the system as part of a ransomware attack or revealing sensitive information

An attack can have devastating results. In the case of an individual, this can involve identity theft, money theft, or unlawful purchasing.

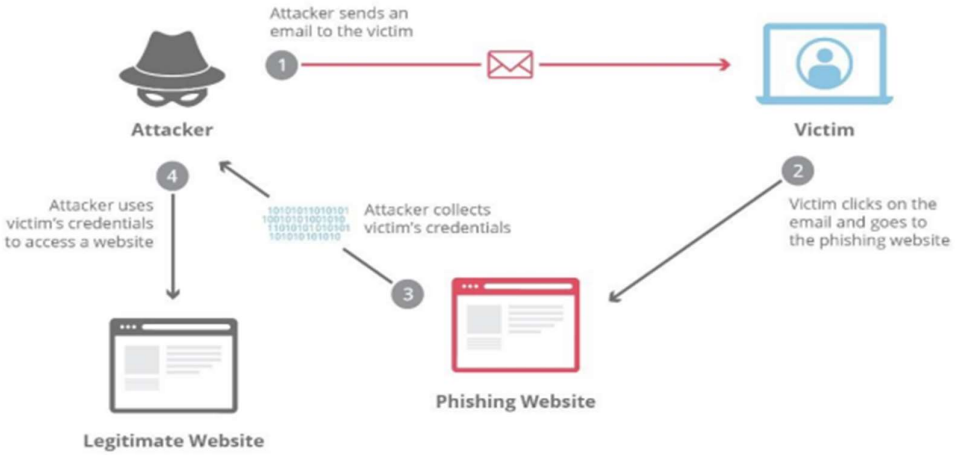


Figure 1.1 Overview of Phishing Attack

Attacks can be dangerous This covers identity theft, money laundering, and unlawful transactions for private individuals. Phishing attacks have reached unprecedented levels primarily due to emerging technologies such as mobile and social media [2].

According to a report from Microsoft, COVID-19-related cyber-attacks increased to an unprecedented level in August, with the majority of these scams being non-COVID-19 websites true, according to security firm RiskIQ (2020). A study KeepnetLABS (2018)

confirmed that over 91% of system breaches result from attacks initiated by email. Cybercriminals use email as a primary vector for their attacks. As shown in Figure 1.2, online stores were at the top of the list of targets (18.12%) followed by global networks (16.44%), while social networks followed (13.07). %) . Overall, the most imitated products in the first quarter of 2020 were Apple, Netflix,WhatsApp,PayPal, Chase, Facebook, Microsoft eBay, and Amazon .

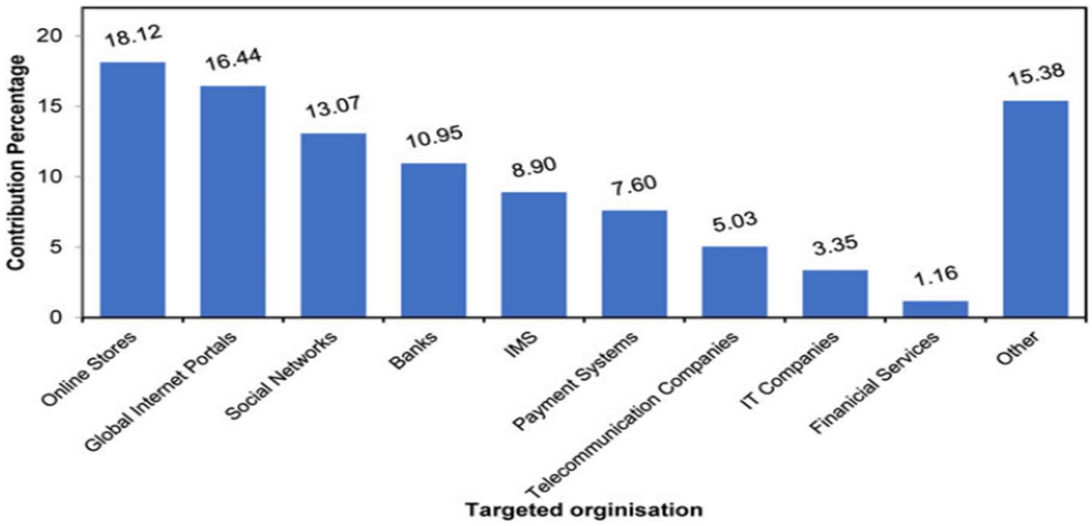


Figure 1.2 Phishing Attack Targeted Organization

1.2 PROBLEM STATEMENT

In today's digital environment, more and more people are communicating online, which has increased the threat of phishing attacks. This type of cybercrime involves fraudulent attempts to obtain personal information from unknown victims, including passwords, credit card numbers, and personal information, which can lead to serious consequences like identity theft and loss of funds and can be prevented in real time Due to modern phishing sophistication methods, traditional code breaking and signature based signature detection methods are not applicable better at effectively blocking malicious organizations against ever-changing tactics, including email spoofing, fake websites and social engineering

The goal of this project is to develop a robust and efficient system that can detect and mitigate phishing attacks in real time. Rule-based solutions are not an effective enough strategy to

combat the rapidly changing phishing techniques. Machine learning enables increased detection accuracy by learning from historical attack data and identifying subtle patterns that may not be apparent to humans.

The focus of this activity is to collect relevant and misleading URLs, as well as any other relevant data needed for recruitment. The URLs will be formatted into a database, where they will then be used to train and identify dangerous websites. Data will be collected from various sources. The fraudulent URLs category will include examples of URLs sent via malicious email. Also, this summary requires fake phishing emails with suspicious URLs. The goal of this framework is to facilitate the building of a reliable algorithm for identifying and isolating malicious sites on the Internet.

By developing and implementing more effective systems to detect phishing attempts, this project seeks to play an important role in the ongoing fight against cyber threats The outcome of this project the results of this will impact individual users, but benefit businesses, organizations and the larger digital communities.

1.3 OBJECTIVE

The project's objectives are as follows:

1. To create a dataset for phishing from different sources.
2. To develop and implement a real-time phishing detection system.
3. To improve phishing detection accuracy by comparing different machine learning algorithms.

1.4 Significance and Motivation of the Project Work

The motivation behind this project comes from the dynamic and changing nature of phishing attacks. As cybercriminals continue to refine their tactics, it will be important to adjust to the system. The goal is to empower users by turning them into active participants in maintaining a secure online environment, providing effective tools to detect and crack down on attempts to report phishing the program also emphasizes reducing response times, machine learning algorithms recognizing the critical importance of acting quickly to mitigate the impact of phishing attacks -and leveraging real-time analytics, the program the goal is to stay ahead of emerging threats. In addition, the incentives extend scalability and cost effectiveness, ensure

that the system can adapt well to different types of projects and reduce the financial burden on users and organizations in particular, the project is not just a response to current threats but a proactive move to protect the digital landscape and empower users.

Phishing attacks have become an ever-present threat in the digital landscape, targeting individuals and organizations with increasing frequency. The inability of traditional security systems to prevent these attacks calls for a robust cloud-based phishing detection system. By leveraging the power of cloud computing, this project addresses the broader threat of phishing, provides scalable and efficient solutions Protection of sensitive data is paramount, and method stable cloud ensures efficient use of resources without compromising system performance. Additionally, it facilitates management and, in a world characterized by global connectivity, cloud-based solutions allow for seamless collaboration, enabling global sharing of threat intelligence

1.5 Organization of Project Report

This report describes our progress. The remaining 5 chapters are as follows:

Chapter 2: It presents past methods of phishing, their advantages, and how they compare to my method.

Chapter 3: It describes how we built the system, including the components of the tool and the completed system, and it includes the system requirements gathering and the redesign process to build the tool based on the requirements of a needs to be addressed

Chapter 4: Evaluates the overall system implementation and methodology.

Chapter 5: Discusses the results in terms of the design requirements established in Chapter 3 and compares them with existing solutions.

Chapter 6: Project conclusion and summary so far. described another important task for this project.

CHAPTER 2: LITERATURE SURVEY

2.1 OVERVIEW OF RELEVANT LITERATURE

A literature is an informative essay that summarizes the body of knowledge, including noteworthy discoveries and theoretical and methodological commitments to a certain subject.

S. Sree Vidhyai et al. [3] this paper provides an intelligent way to effectively identify phishing emails. It tests the difference between Random Forests, and SVM. The goal is to find the most effective intelligent classification model for email phishing detection.

Sasirekha CI et al. [4] uses the phish tank dataset for to train the model. They use the Random Forest Classifier as it can manages nonlinear interactions between features, performs well with datasets, and offers improved accuracy.

Boddupalli Phani Kumar et al. [5] this includes teaching users how to properly use the technology and making them feel safe when using it.. They used different machine learning algorithm for the better results.

Ejaz et al. [6] they compare the different dataset. They use CNN, Word2Vec and ML algorithms and get the accuracy around 93-95% but it increases the model's complexity.

Mohammed M.Alani et al. [7] presented a cloud based phishing URL detector .The detector relies on features form the URL itself. This research mostly focused on accuracy and uses only random forest classifier and get the accuracy of 97.5%.

P. Amba Bhavani A et al. [8] uses Kaggle dataset. They use logistic regression, XGBoost and CNN for phishing website detection using machine learning. XGBoost gives highest accuracy, CNN gives lowest accuracy.

Roshan Ravi I et al. [9] presented URL based email phishing detection application. They focused on accuracy and use less technology. They only use random forest classifier and get the accuracy of 98.4%.

Ishita Saha et al. [10] proposed a phishing detection solution based on deep learning that combines universal resource station and website content such as images, text, frames and Convolution Neural Network (CNN) and Long Short-Term Memory (LSTM). use the right. The problem of large data set and high classification prediction performance was solved.

Vahid Shahrivari et al. [11] applied and assessed twelve classifiers on the dataset of phishing websites, which comprises 4898 phishing websites and 6157 authentic websites. Logistic Regression, Decision Trees, Support Vector Machines, Ada Boost, Random Forests, Neural Networks, KNN, Gradient Boosting, and XGBoost are the classifiers that are explored. XGBoost and random forest achieve highest accuracy.

Sweta Mittal et al. [12] uses machine learning algorithm for detecting phishing attacks. SVM achieve accuracy of 95.66%.

S.No.	Paper Title [Cite]	Journal/ Conference (Year)	Tools/ Techniques/ Dataset	Results	Limitations
1.	Efficient Email phishing using machine learning. [3]	(IJCRT) (2023)	SVM classifier, Random Forest / Phish Tank Dataset.	Accuracy is 95.05%	SVM performance has not been measured across various benchmark datasets.
2.	Email phishing detection using machine learning. [4]	(IJCRT) (2023)	Random Forest Classifier/ Phish Tank.	Using Random Forest classifier for higher accuracy.	Deep learning will be incorporated in the future for greater effectiveness.
3.	Phishing site detection using machine learning Techniques. [5]	(IRJ) (2023)	Random Forest, XGBoost, SVM /Phish Tank.	Using the different algorithm for the better results.	Some new phishing website that has not yet been added to the blacklist

					may be harmful as well because refreshing the list might take a while.
4.	Life-Long phishing attacks detection using continual learning. [6]	(ISSN) (2023)	Word2Vec, CNN, Fast Text, ML.	Accuracy around to 93-95%.	New tasks will be added in the future, which will eventually increase the model's complexity.
5.	A Cloud-Based machine - learning approach to phishing URL Detection. [7]	(IJCRT) (2022)	Random Forest / Phish Tank.	Highest accuracy at 97.5%.	This research focuses mostly on accuracy.
6.	Phishing Websites detection using machine learning. [8]	(IJCRT) (2022)	Logistic regression, XGBoost / kaggle.com	Highest accuracy at XGBoost is 92%.	CNN give us lowest accuracy.
7.	URL Based Email phishing detection application. [9]	(IJCRT) (2021)	Random Forest / Phish Tank.	Highest accuracy is 98.4%.	Less technology has been used in this paper.
8.	Phishing attacks detection using deep learning. [10]	(IJCRT) (2020)	CNN, Logistic regression	Highest accuracy at Logistic	It is less effective

			and Linear regression	regression is 99.97%.	
9.	Phishing detection using machine learning technique. [11]	(ISSN) (2020)	Decision Tree, Random Forest / Phish Tank	XGBoost achieve highest accuracy 98.1%.	Adaboost there are a few parameters that need to be tuned to improve model performance.
10.	Detection of phishing attacks using analysis in the cloud. [12]	(IJCRT) (2020)	Machine Learning Algorithm/ Not Mentioned	Support Vector machine achieve accuracy of 95.66%.	This will save time only in the first case, not in the second.

Table 1. Literature Survey.

2.2 KEY GAPS IN THE LITERATURE

Some key gaps in the literature are:

1. **Limited Comparative Analysis:** The literature lacks a comprehensive comparative analysis of various machine learning techniques across different studies. A systematic evaluation of the strengths and weaknesses of these approaches could provide insights into the most effective methods for phishing detection.

2. **Scarcity of Real-time Evaluation:** Few studies address real-time evaluation of phishing detection methods. Given the dynamic nature of phishing attacks, there is a

need for research that assesses the effectiveness of models in real-time scenarios, considering evolving tactics used by attackers.

3. **In-depth Evaluation of Feature Engineering**: While feature engineering is acknowledged as crucial, there is a gap in providing a detailed analysis of the impact of specific features on model performance. Understanding the relevance and contribution of individual features can guide more informed feature selection strategies.

4. **Limited Exploration of Hybrid Models:** The literature primarily focuses on individual machine learning or deep learning models. Exploring hybrid model that combines the strength of different algorithms may lead to improved accuracy and robustness in phishing detection systems.

5. **Insufficient Exploration of Phishing URL Structures:** There is a gap in exploring the structure and characteristics of phishing URLs. A more in-depth analysis of the features within URLs that contribute to phishing attacks could enhance the development of targeted detection mechanisms.

6. **Validation of Models on Diverse Datasets:** Many studies use specific datasets, and there is a need for research that evaluates the generalizability of proposed models across diverse datasets. This would ensure that the models are effective in various contexts and against different types of phishing attacks.

7. **Exploration of Explainability and Interpretability**: The literature lacks emphasis on the explainability and interpretability of the proposed models. Understanding how models make decisions is crucial for gaining trust and acceptance in practical cybersecurity applications.

Addressing these gaps could contribute to the advancement of phishing detection methods and the development of more robust and effective cybersecurity solutions.

In addition to these key gaps, the literature review also highlights the need for more research on the following topics:

- The use of natural language processing (NLP) for phishing detection.
- The use of social engineering techniques in phishing attacks.
- The development of more effective phishing detection tools and techniques.

CHAPTER 3: SYSTEM DEVEPOLMENT

3.1 REQUIREMENTS AND ANALYSIS

3.1.1 HARDWARE REQUIREMENT

• Processor: AMD Ryzen 3 or higher for the CPU

• Hard disk capacity: 512 MB of minimum space needed.

• RAM: 4GB at least.

3.1.2 SOTWARE REQUIREMENT

• Programming Languages: Python and Flask

• IDE: Python version 3x, Jupyter Lab, or Google Colab;

• Operating system: Windows 10 or higher.

• Docker.

3.1.3 Supporting Python modules

Python offers a method for putting definitions in a document and using them in a content or interpreter's intuitive scenario. Module definitions are files of this type that can be imported into the core module or into other modules. Table 2 lists a few of the modules that were utilized for the project.

No.	Python Modules	Description
1.	Ipaddres	The ability to create, manage, and operate IPv4 and IPv6 addresses and networks is provided by ipaddress.
2.	re	Regular expression matching tasks similar to those in Perl are provided by this module.
3.	urllib.request	The functions and classes that aid in opening URLs (mostly HTTP) in a complicated world are characterized by the urllib.request module.

4.	beautifulSoup	A Python module called BeautifulSoup is used to parse HTML and XML documents. For online scraping purposes, it creates a parse tree for parsed pages that can be used to extract information from HTML.
5.	socket	This module grants access to the socket's BSD interface.
6.	requests	This module uses Python to allow the sending of HTTP requests.
7.	WHOIS	A thorough protocol for addressing databases that hold the chosen users or trustees of an online resource is the WHOIS query and answer standard. for instance, a domain name, an independent framework, or an IP address block that is also concurrently utilized for a large amount of data.

Table 2: Python modules

3.2 Project Design and Architecture

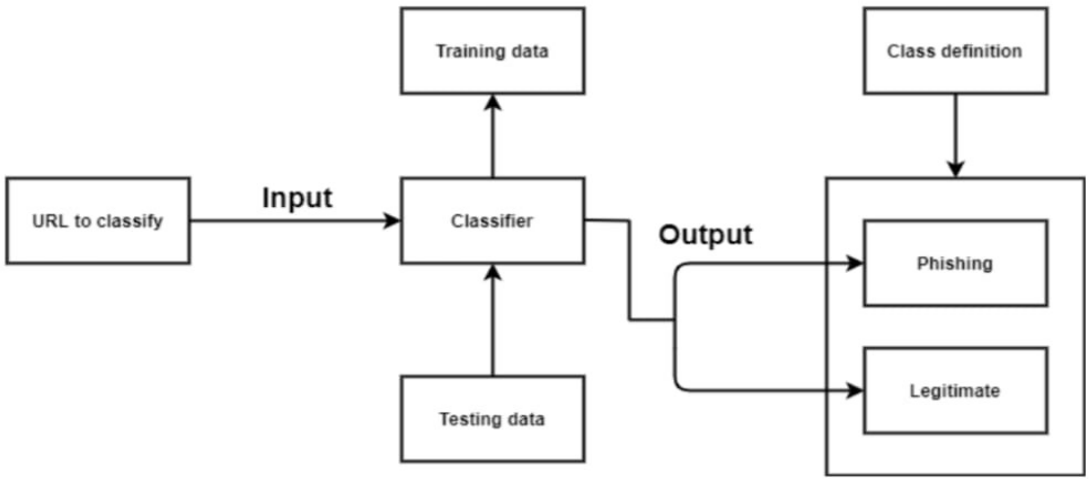


Figure 3.1 System Architect

The diagram of the system is as shown in the fig. 3.1 URLs to be classified as legitimate or phishing are provided as input to the appropriate classification. A classifier trained to classify URLs from the training dataset as phishing or legitimate then uses the patterns it learns to classify the newly provided input by attributes such as IP address, URL length, domain, favicon which has from the URL and creates a list of its values. The list is provided for classifiers such as SVM, Decision tree, and Random Forest classifiers. The performance of these models is then analyzed for accuracy scores. Using the generated list, the trained classifier determines whether the URL is legitimate or phishing. There are 24 features being considered in this project.

The data flow in the system is graphically represented using DFDs [14]. It explains how the system works all the time, from input to report generation. A represents any possible path from one part of the system to another. Three sequences designated as 0, 1, and 2 can be used to represent information in a data flow diagram. Data flow diagrams can be represented by different letters, the most common being Yourdon Coad and Gane-Sarson methods.

Then, determining set of features that are prominent in the phishing URL's. The URL'S are downloaded from various sources and a CSV dataset is constructed by extracting the feature set values from the mails. The dataset is now subjected to various visualization plots and dimensionality reduction techniques. Feature Importance is also plotted to show case the important features in the process of classification. Various classification algorithms are then implemented upon the dataset. Top 4 classification algorithms are selected on the basis of the cross-validation error. The selected models are then integrated [15] to form an ensemble model to get better performance. The steps of the proposed work are presented in Fig. 3.2.

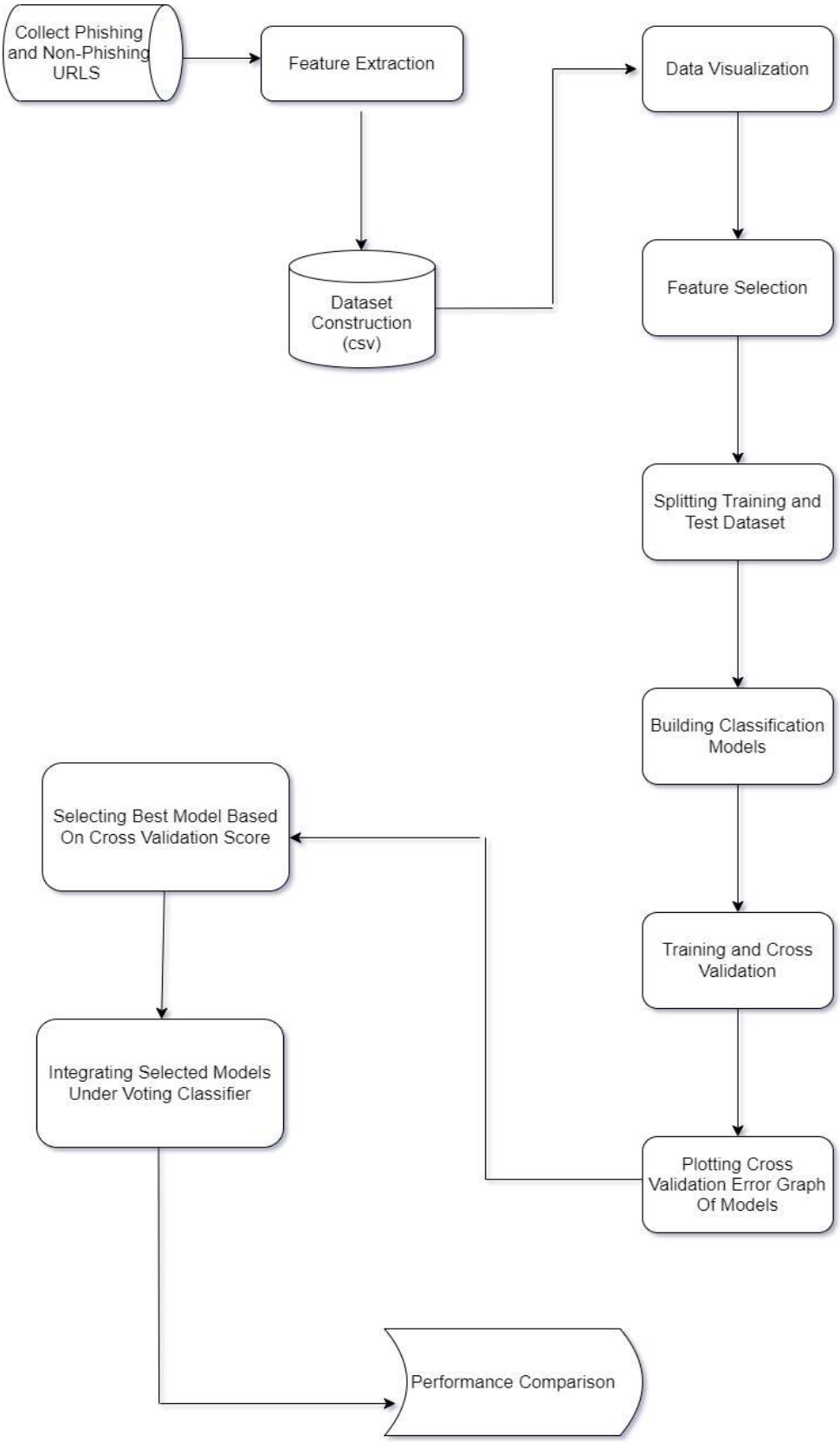


Figure 3.2 Data Flow Diagram

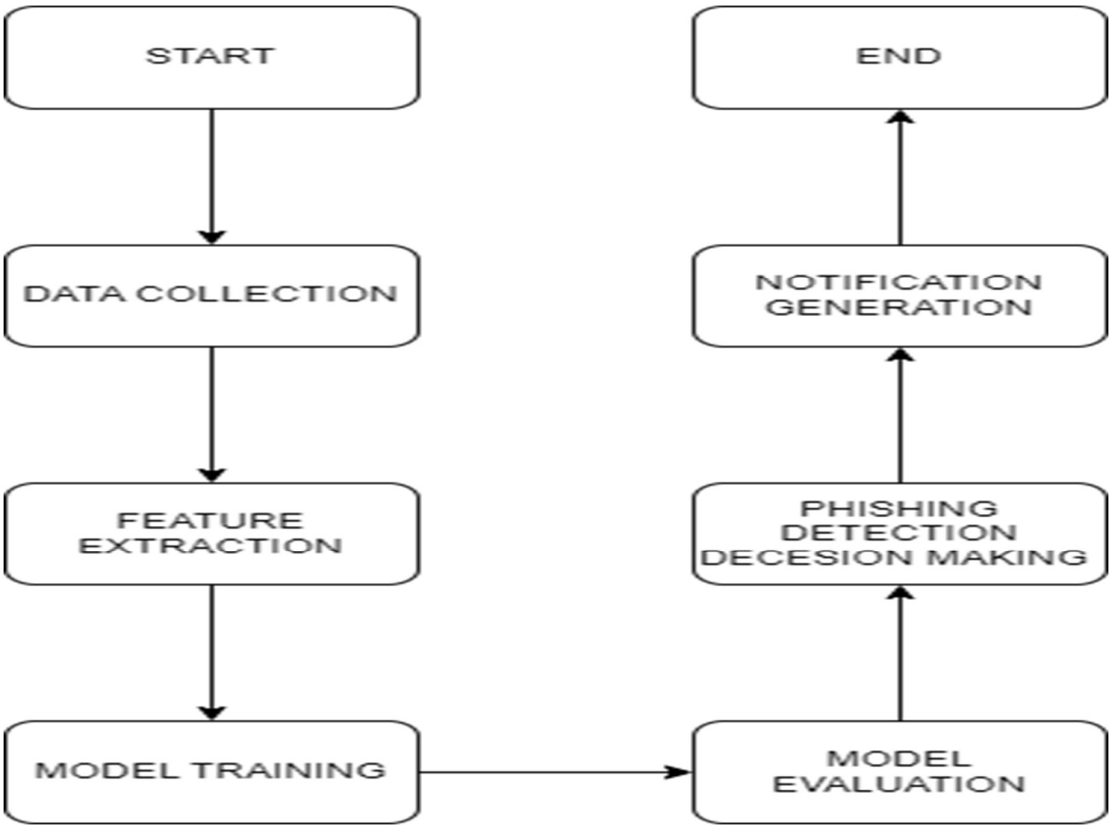


Figure 3.3 Work Flow Diagram

3.3 DATA PREPARATION

A thorough data preparation procedure was used to guarantee the efficacy of our phishing detection algorithm. Phish Tank [16], Kaggle [17], and the University of New Brunswick [18] dataset of URL is utilized in this project. In the data preparation stage, a number of crucial actions were taken before the machine learning models were trained:

1. **Data Collection:** The dataset of URL consisting of phishing and legitimate URL are collected from different sources like Phish Tank, Kaggle, and University of New Brunswick. We labeled phishing URL's as '1' and '0' for legitimate URL.
2. **Merging of Data:** After collecting legitimate and phishing URL into different file, we merged two files to make the required dataset. By this we get the both dataset in single file for further steps.

3. **Feature Extraction:** Features such as Ip address, Length of URL, depth of URL, count @ etc. are extracted from the URL. We are extracting 24 features from the URL.

4. **Data Preprocessing:** In data preprocessing for this project, we cleaned the dataset by addressing missing values, removing duplicates, and handling outliers. Features that are insignificant are removed from the dataset. The dataset, labeled for phishing and legitimate URLs, underwent a balanced split for training and evaluation.

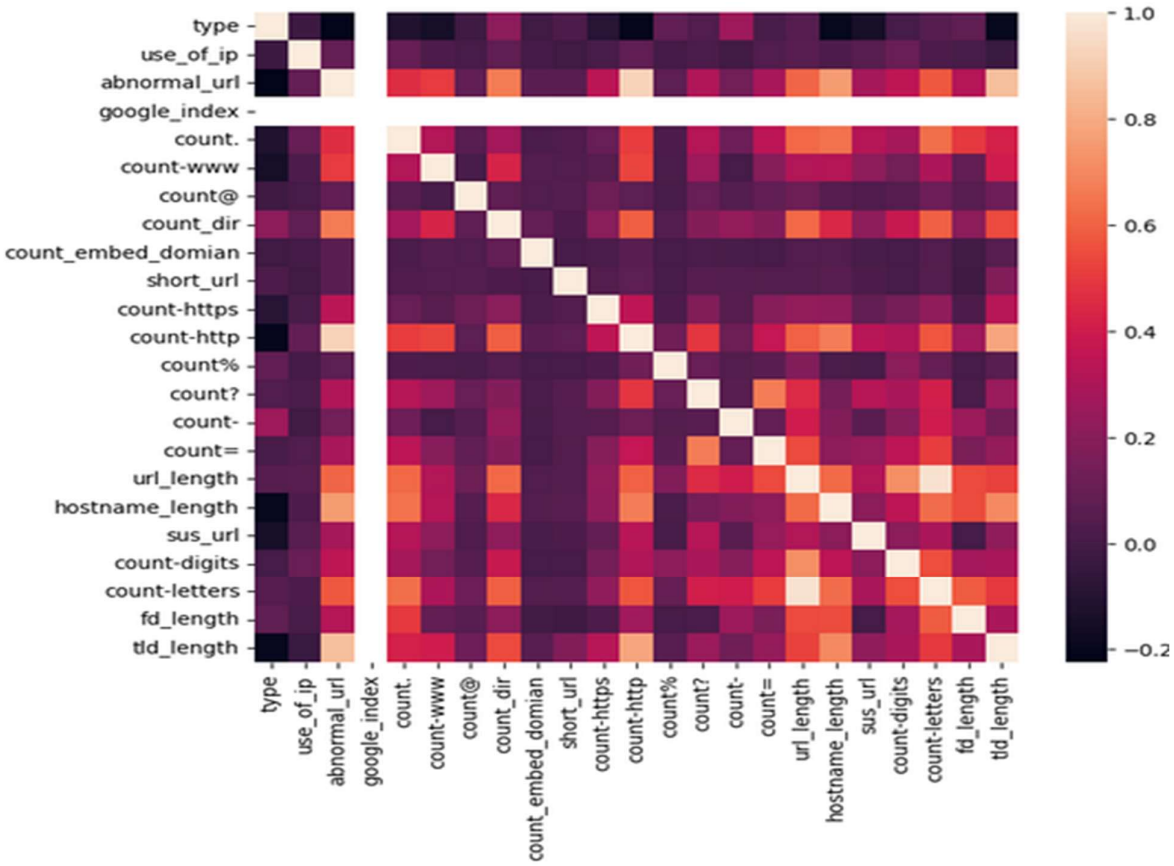


Figure 3.4 Heatmap

5. **Feature Selection:** The process of automatically or manually selecting the features that contribute the most to your predictor variable or output of interest. Having irrelevant features in your data can reduce the accuracy and properties of the model your model can learn based on redundancy.

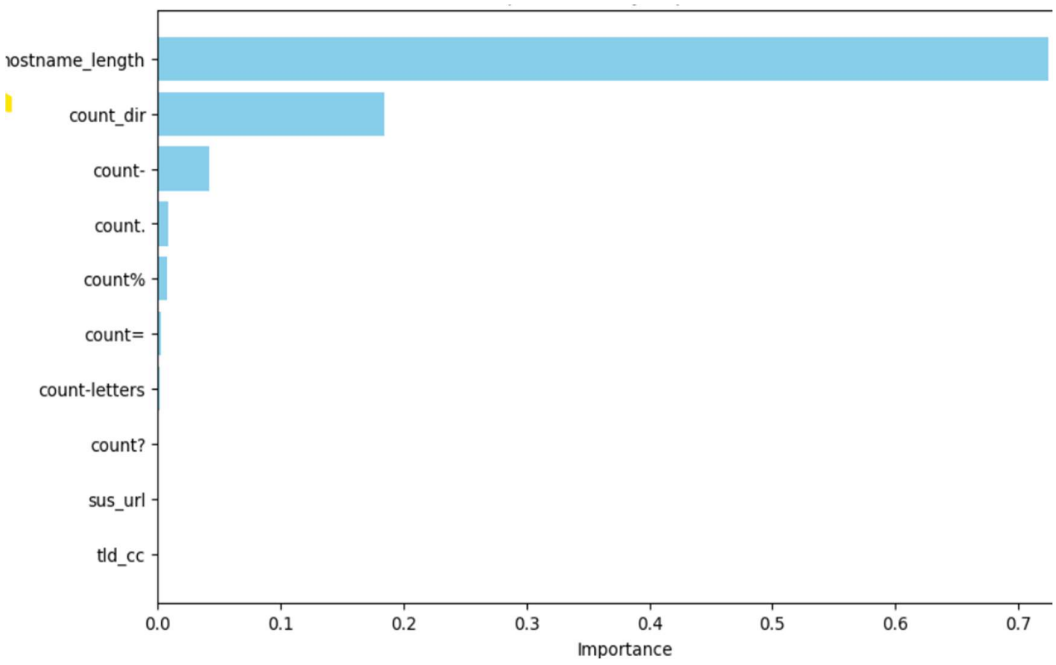


Figure 3.5 Features Importance

By implementing above steps, we prepared the dataset.

3.4 IMPLENTATION

This section of the report outlines the process for classifying URLs as phishing or legitimate. The method enables the development of a training program. The machine learning model is trained with the training set, i.e., the distribution of the. Figure 6 shows a diagram of the implementation.

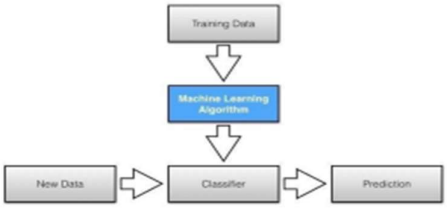


Figure 3.6 Implementation of Algo's

Feature	Description	Data Type
use_of_ip	Presence of an IP address in the URL	Integer
abnormal_url	Detection of abnormal URL patterns	Integer

google_index	Google index status of the URL	Integer
count	Generic count feature	Integer
count-www	Count of 'www' in the URL	Integer
count@	Count of '@' in the URL	Integer
count_dir	Count of directory-related elements	Integer
count_embed_domain	Count of embedded domains in the URL	Integer
short_url	Presence of a short URL	Boolean
count-https	Count of 'https' in the URL	Integer
count-http	Count of 'http' in the URL	Integer
count%	Count of '%' in the URL	Integer
count?	Count of '?' in the URL	Integer
count=	Count of '=' in the URL	Integer
url_length	Length of the URL	Integer
hostname_length	Length of the hostname in the URL	Integer
sus_url	Suspicious URL indicator	Boolean
count-digits	Count of digits in the URL	Integer
count-letters	Count of letters in the URL	Integer
fd_length	Length of the first directory in the URL	Integer
tld	Top-level domain of the URL	Integer
tld_length	Length of the top-level domain	Integer

Table 3 URL Features

3.4.1 TOOLS

- Programming Languages: Python and Flask
- IDE: Python version 3x, Jupyter Lab, or Google Colab;
- Operating system: Windows 10 or higher.
- Docker.

3.4.2 TECHNIQUES

We used Machine learning algorithms such as

- Decision Tree: Used mostly to solve classification issues, decision trees are a supervised learning method. This can also be used to solve regression problems. With inner nodes for data set properties, branches for decision rules, and leaf nodes for each outcome, this classification is represented as a hierarchical tree.[19]

- Random Forest Classifier: Moving on to another popular machine learning algorithm called Random Forest Classifier it belongs to the category of learning methods. This versatile algorithm can be utilized for both classification and regression problems, within Machine Learning. It is based on the concept of group learning, which is the process of combining multiple classifiers to solve a complex problem and improve the performance of the model.[20]

- SVM: A machine learning algorithm called the support vector machine draws boundaries between data points using preset outputs, labels, or classes. It solves challenging issues with classification, regression, and outlier detection using supervised learning models.[21]

- Stacking: One technique for grouping several classifications or regression models is stacking. Stacking, also known as Stacked Generalization, represents an alternative framework. Investigating a range of alternative models for a given issue is the goal of stacking. The concept is to use various models that can learn specific aspects of a problem but not the entire area of the problem to tackle learning problems. Thus, you can construct a variety of learners and utilize them to construct an intermediate prediction, one for every model that has been learnt.[22]

- XGBoost: XGBoost is an ensemble [23] learning method. XGBoost is a distributed, scalable machine learning framework gradient boosted decision tree (GBDT). It is

the top machine learning when it comes to regression, classification and ranking and provides parallel tree enhancement.[24]

3.4.2 CODE SNIPPETS: FEATURE EXTRACTION AND MODELS

```
import pandas as pd
import itertools
from sklearn.metrics import classification_report,confusion_matrix, accuracy_score
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import xgboost as xgb
from lightgbm import LGBMClassifier
import os
import seaborn as sns
from wordcloud import WordCloud
```

In the fig 3.7 we have imported all the libraries that are required to extract the features form the dataset and then we have uploaded the dataset file ‘malicious_phish.csv’ that contains legitimate and phishing URL’s, form this file feature is extracted.

```
[ ] df=pd.read_csv('malicious_phish.csv')

    print(df.shape)
    df.head()
```

Figure 3.7 Importing Required Files

```
[ ]  import re
     #Use of IP or not in domain
     def having_ip_address(url):
         match = re.search(
             '(([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.'
             '([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\/)|'  # IPv4
             '((0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\/)' # IPv4 in hexadecimal
             '(?:[a-fA-F0-9]{1,4}:){7}[a-fA-F0-9]{1,4}', url)  # Ipv6
         if match:
             #print(match.group())
             return 1
         else:
             #print('No matching pattern found')
             return 0
     df['use_of_ip'] = df['url'].apply(lambda i: having_ip_address(i))

     from urllib.parse import urlparse

     def abnormal_url(url):
         hostname = urlparse(url).hostname
         hostname = str(hostname)
         match = re.search(hostname, url)
         if match:
             # print match.group()
             return 1
         else:
             # print 'No matching pattern found'
             return 0


     df['abnormal_url'] = df['url'].apply(lambda i: abnormal_url(i))
```

```
[ ]  from urllib.parse import urlparse
     from tld import get_tld
     import os.path

     #First Directory Length
     def fd_length(url):
         urlpath= urlparse(url).path
         try:
             return len(urlpath.split('/')[1])
         except:
             return 0

     df['fd_length'] = df['url'].apply(lambda i: fd_length(i))

     #Length of Top Level Domain
     df['tld'] = df['url'].apply(lambda i: get_tld(i,fail_silently=True))


     def tld_length(tld):
         try:
             return len(tld)
         except:
             return -1

     df['tld_length'] = df['tld'].apply(lambda i: tld_length(i))
     df.to_csv('output_with_features.csv', index=False)
```

Figure 3.8 Feature Extraction Code

In Fig 3.8, we have imported ‘re’, which is use to evaluate the regular expression. Through this we will find the ip address of the url by ‘re.search’ module present in the ‘re’. With this it will find the match present in the list and give us the result. This figure shows us different functions for the feature extraction such as: having_ip_address, use_of_ip, abnormal_ip, count_dot etc.

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Assuming 'data' is your original DataFrame
# Sample a fraction of the dataset (adjust the fraction as needed)
data_sample = data.sample(frac=0.5, random_state=42)

# Assuming 'type' is the target variable
X = data_sample.drop('type', axis=1)  # Features
y = data_sample['type']  # Target variable

# One-hot encode categorical variables
X_encoded = pd.get_dummies(X)

# Splitting the dataset into train and test sets: 80-20 split
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=12)

# Instantiate the Decision Tree model
tree_model = DecisionTreeClassifier(random_state=12)

# Fit the model on the training data
tree_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = tree_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

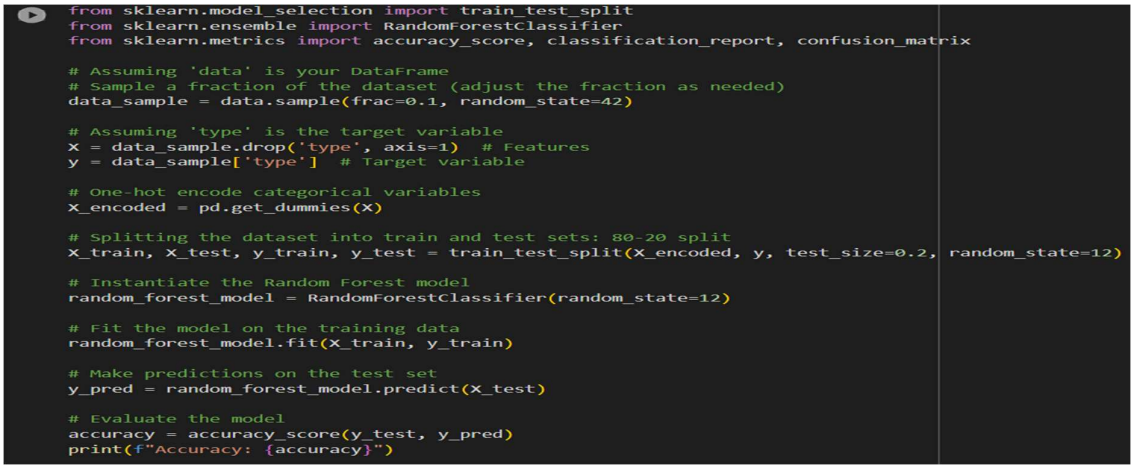
Figure 3.9 Decision Tree

In figure 3.9, we are using Decision Tree to train our dataset. We follow following steps to create the required code:

- **Sampling Data**: A fraction (50%) of the original dataset (‘data’) is sampled randomly to create a new DataFrame called ‘data_sample’.
- **Feature and Target Variable Setup**: ‘X’ is created by dropping the column 'type' from the sampled data, representing the features.
  ‘y’ is created as the 'type' column, representing the target variable.
- **Splitting:** The data set is splits into training and test sets(80-20 splits), and the partitions are randomly seed for regeneration.
- **Decision Tree Model Initialization:** The Decision Tree distribution is modelled using the specified random results.
- **Model Training:** The decision tree model is embedded in the training data.
- **Model Prediction:** The test set is used to make predictions.
- **Model Evaluation:** Model accuracy is calculated and published.
- **Additional Metrics and Confusion Matrix:** Classification reports and confusion matrices are published to provide additional evaluation metrics.

- **Visualizing Confusion Matrix:** The heatmap of the confusion matrix is plotted using seaborn and matplotlib.

This code provides a detailed example of constructing, training, and evaluating decision tree models for a binary multiclass classification problem. The confusion matrix heat map visually represents the performance of the model in the testing process.



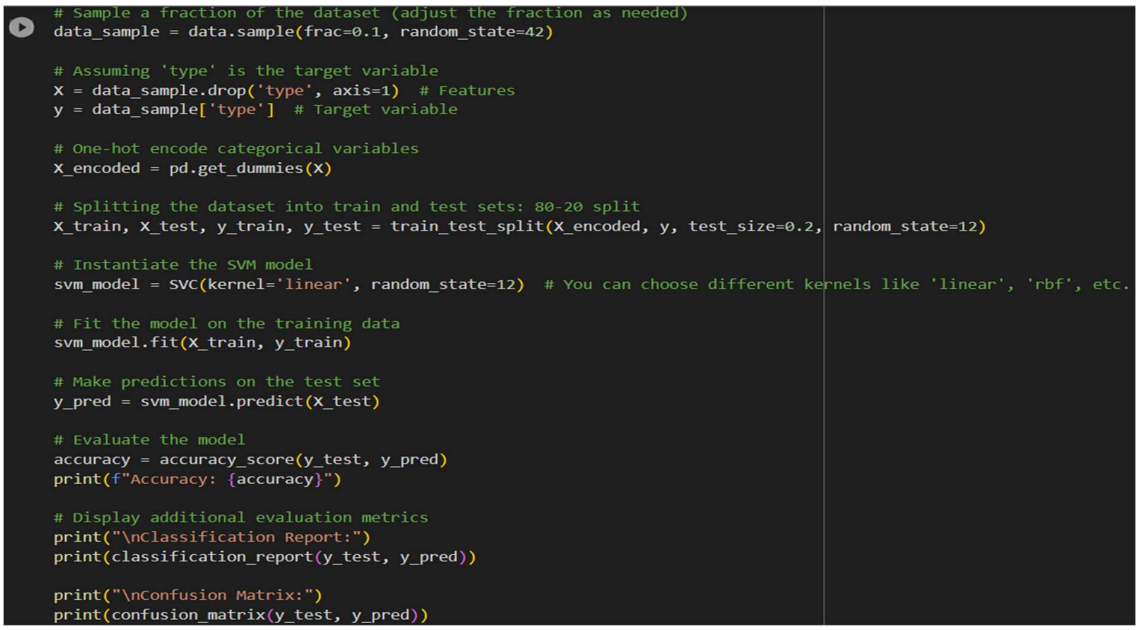
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Assuming 'data' is your DataFrame
# Sample a fraction of the dataset (adjust the fraction as needed)
data_sample = data.sample(frac=0.1, random_state=42)

# Assuming 'type' is the target variable
X = data_sample.drop('type', axis=1)  # Features
y = data_sample['type']  # Target variable

# One-hot encode categorical variables
X_encoded = pd.get_dummies(X)

# Splitting the dataset into train and test sets: 80-20 split
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=12)

# Instantiate the Random Forest model
random_forest_model = RandomForestClassifier(random_state=12)

# Fit the model on the training data
random_forest_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = random_forest_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

Figure 3.10 Random Forest Classifier

In figure 3.10, we are using Random Forest Tree to train our dataset. We follow following steps to create the required code:

- **Train-Test Split:** To ensure consistency, the dataset is divided into training and testing sets (80-20 split) using a random seed.
- **Random Forest Model Initialization:** Using a given random seed, an instance of the Random Forest classifier is constructed.
- **Training the Model:** When we fit the Random Forest model to our training dataset it is referred to as model training.
- **Model Prediction:** The test set is used to make predictions.
- **Model Evaluation:** A computation and print of the model's accuracy are made.
- **Additional Metrics and Confusion Matrix:** We generate evaluation metrics by producing a categorization report and printing a confusion matrix.
- **Visualizing the Confusion Matrix:** We make use of Seaborn and matplotlib libraries to create a heatmap that visualizes the confusion matrix.

This code structure is similar, to the example. Instead of using a Decision Tree we employ a Random Forest classifier. The Random Forest method combines decision trees predictions to achieve accurate results. The heatmap representation of the confusion matrix gives us insights, into how our model performs on the test dataset.



```
# Sample a fraction of the dataset (adjust the fraction as needed)
data_sample = data.sample(frac=0.1, random_state=42)

# Assuming 'type' is the target variable
X = data_sample.drop('type', axis=1)  # Features
y = data_sample['type']  # Target variable

# One-hot encode categorical variables
X_encoded = pd.get_dummies(X)

# Splitting the dataset into train and test sets: 80-20 split
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=12)

# Instantiate the SVM model
svm_model = SVC(kernel='linear', random_state=12)  # You can choose different kernels like 'linear', 'rbf', etc.

# Fit the model on the training data
svm_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = svm_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")

# Display additional evaluation metrics
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

Figure 3.11 SVM

In figure 3.11, we are using SVM to train our dataset. We follow following steps to create the required code

- **SVM Model Initialization:** To initialize the SVM model we use a kernel and a predetermined random seed. This approach helps in generating an instance of the SVM classifier.
- **Model Training:** The training set of data is used to fit the SVM model.
- **Model Prediction:** The test set is used to make predictions.
- **Model Evaluation:** A computation and print of the model's accuracy are made.

The provided code outlines the step, by step procedure for applying linear SVM to solve classification problems. It covers stages such as data generation, model training, prediction and analysis. One useful tool is the confusion matrix heatmap which visually represents how well the model performs on the test set. The linear kernel used in SVM is particularly effective for dealing with data that can be separated linearly.

```python
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.ensemble import StackingClassifier

# Assuming 'data' is your DataFrame
sample = data.sample(frac=0.5, random_state=42)

# Separate features (X) and target variable (y)
X = sample.drop('type', axis=1)  # Assuming 'type' is your target variable
y = sample['type']

# Perform one-hot encoding for categorical features
X_encoded = pd.get_dummies(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=42)

# Define the base models
base_models = [
    ('rf', RandomForestClassifier(random_state=42)),
    ('gb', GradientBoostingClassifier(random_state=42))
]

# Define the meta-model
meta_model = LogisticRegression(random_state=42)

# Create the stacking model
stacking_model = StackingClassifier(estimators=base_models, final_estimator=meta_model)
```

Figure 3.12 Stacking

In figure 3.12, we are using Stacking to train our dataset. We follow following steps to create the required code:

- **Define Base Models:** A list of tuples defines two base models: a Random Forest Classifier and a Gradient Boosting Classifier.
- **Define Meta-Model:** A Meta-model is a Logistic Regression model that receives input in the form of base model predictions.
- **Create Stacking Model:** The defined base models and meta-model are used to create the Stacking Classifier.
- **Fit Stacking Model:** Training data is used to train the stacking model.
- **Make Predictions and Evaluate:** The correctness of the stacking model is computed and printed, and predictions are made using the test set.

This code describes a simple cluster approach in which predictions from multiple base models (Random Forest and Gradient Boosting) are combined with Logistic Regression meta-model Stacking allows the model to use different algorithm properties and can drive

overall performance effectiveness. The accuracy metric is used to evaluate the stacking model in the testing process.

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.ensemble import StackingClassifier

# Assuming 'data' is your DataFrame
sample = data.sample(frac=0.5, random_state=42)

# Separate features (X) and target variable (y)
X = sample.drop('type', axis=1)  # Assuming 'type' is your target variable
y = sample['type']

# Perform one-hot encoding for categorical features
X_encoded = pd.get_dummies(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=42)

# Define the base models
base_models = [
    ('rf', RandomForestClassifier(random_state=42)),
    ('gb', GradientBoostingClassifier(random_state=42)),
    ('xgb', XGBClassifier(random_state=42))
]

# Define the meta-model
meta_model = LogisticRegression(random_state=42)
```

Fig. 3.13 XGBoost

In figure 3.13, we are using XGBoost to train our dataset. We follow following steps to create the required code:

- **Create Stacking Model:** Using the specified base models and meta-model, the Stacking Classifier is built.
- **Fit Stacking Model:** The training data is used to train the stacking model.
- **Make Predictions and Evaluate:** The correctness of the stacking model is computed and printed, and predictions are made using the test set.

This code uses the capabilities of Random Forest, Gradient Boosting, and XGBoost classifiers, extends the stacking method to include multiple base models and uses a meta-model (Logistic Regression) to combine the predictions of the base models. The accuracy metric is used to evaluate the stacking model in the testing process. The idea is to create clusters that will benefit from the complementary strengths of different algorithms.

3.5 Key Challenges

During the development process, the creation and preprocessing of the data set presented several challenges that significantly affected the performance of the phishing detection model.

1. **Integrating phishing and legitimate datasets:** One of the first challenges is to combine datasets containing phishing and legitimate URLs into a combined dataset. Integration of these sources required careful consideration of data structure, composition, and potential sources of bias. Addressing this challenge required a standardized data structure to ensure a balance of representativeness between phishing and non-phishing patterns.
2. **Feature selection and removal**: Deciding what to keep or remove from the data set during the preprocessing phase presented a subtle challenge. This decision affected the model's ability to identify relevant patterns. Meeting this challenge required careful priority analysis, consideration of the unique characteristics of phishing attacks.
3. **Data imbalance handling:** The imbalance between phishing and legitimate instances in the dataset caused difficulties in model training. Traditional machine learning models are biased towards the majority class, affecting overall accuracy. To overcome this, techniques such as oversampling, under sampling and the use of an unbalanced study design were used to ensure that the two classes were adequately represented.
4. **Dataset Quality and Model Accuracy**: Difficulties persisted during model training, as lower than expected accuracies were observed. This was primarily analysed for data set quality issues, outliers and mislabelled observations. Intensive data cleaning and additional preprocessing steps were used to improve the quality of the dataset, subsequently improving the accuracy of the model.
5. **Large data set optimization:** Balancing the size of the data set to ensure sufficient samples for efficient model training without unnecessary complexity was another challenge.

CHAPTER 4: TESTING

In this chapter, by evaluating and contrasting the algorithm's output with the real result, we can verify that the suggested system is functioning as intended. In essence, the system is being validated. The following are the outcomes of testing each algorithm using a phishing and legal URL.

4.1 Testing Strategy

Unit testing is a kind of software testing that involves testing individual program components.

Integration testing is a method of testing in which the modules constituent parts are combined and thereafter examined to determine whether or not they are suitable for use.

4.2 Test Cases and Outcomes

4.2.1 Importing of the modules

Test Case	01
Name	Importing Modules
Input	Import module statements
Output	The module was imported Successfully and usable
Remark	Success

Table 4

4.2.2 Importing of the data

Test Case	02
Name	Importing Dataset
Input	Import dataset statements
Output	The dataset was imported Successfully and usable
Remark	Success

Table 5

4.2.3 Unit Testing

Test Case	1
Input	www.facebook.com
Expected Output	Legitimate
Actual Output	Legitimate
Remark	Success

Table 6: Testing Case 1

Test Case	2
Input	www.google.com
Expected Output	Legitimate
Actual Output	Legitimate
Remark	Success

Table 7: Testing Case 2

Test Case	3
Input	https://www.juit.ac.in/
Expected Output	Legitimate
Actual Output	Legitimate
Remark	Success

Table 8: Testing Case 3

Test Case	4
Input	infonews.co.nz/news-index.cfm?l=7&t=0
Expected Output	Phishing
Actual Output	Phishing
Remark	Success

Table 9: Testing Case 4

Test Case	5
Input	123people.com/s/cathy+reynolds
Expected Output	Phishing
Actual Output	Phishing
Remark	Success

Table 10: Testing Case 5

Test Case	6
Input	youtube.com/watch?v=CM7vAP1qyXQ
Expected Output	Phishing
Actual Output	Phishing
Remark	Success

Table 11: Testing Case 6

CHAPTER 5: RESULTS AND EVALUATION

5.1 RESULTS

The confusion matrix (CM), which can be used to assess performance, is a graphical representation of the accurate and inaccurate predictions made by a classifier. In general, Figure 9 depicts the CM as follows:

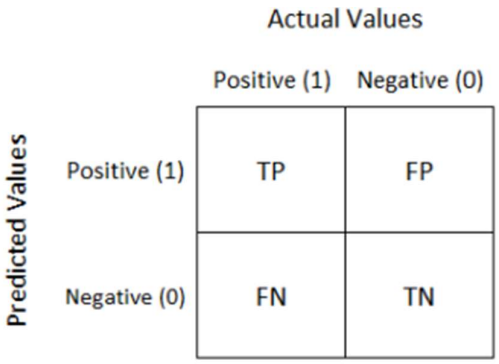


Figure 5.1 Confusion Matrix

True positives (TP), True negatives (TN), False positives (FP), and False negatives (FN) are represented in the above figure. The following is the confusion matrix for the algorithms used:

5.1.1 DECISION TREE

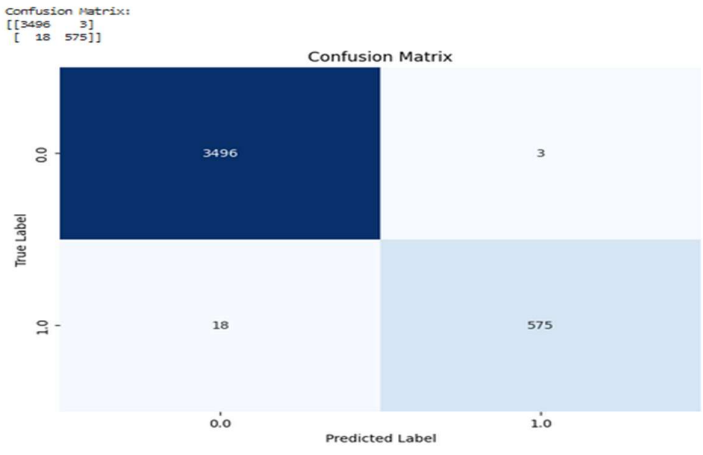


Figure 5.2 Decision Tree

5.1.2 RANDOM FOREST CLASSIFIER

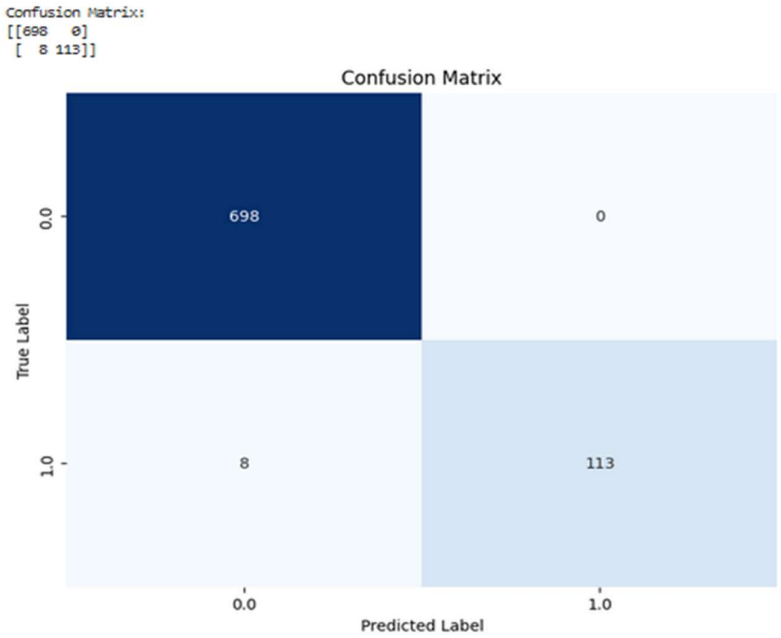


Figure 5.3 Random Forest Classifier

5.1.3 SVM

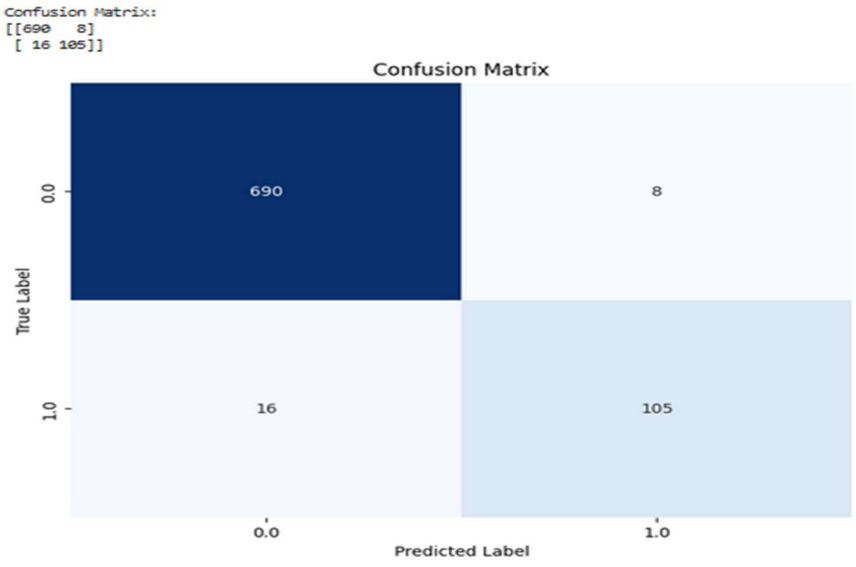


Figure 5.4 SVM

5.1.3 Frontend

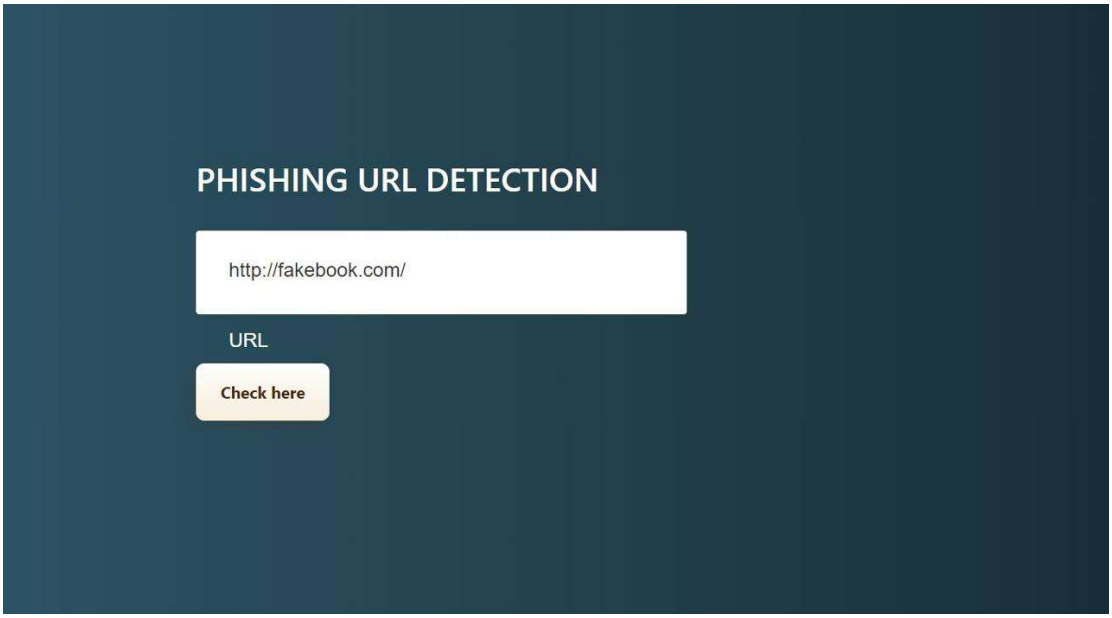


Figure 5.5 Frontend

5.2 COMPARISION

S.NO.	Model Name	Accuracy	Recall
1.	Decision Tree	91.4%	0.92
2.	Random Forest Classifier	90.02%	0.92
3.	Support Vector Machine	88%	0.89
4.	Adaboost	90%	0.91
5.	Stacking	92%	0.93

Table 12 Comparison Table

In this study, we evaluate and compare the effectiveness of different machine learning models, including Decision Trees, Random Forests, SVM, and ensemble

techniques like stacking, for the task of phishing detection based on their respective accuracy, recall.

● Decision Trees: Decision Trees are a popular supervised learning algorithm used for classification tasks. This model achieves an impressive accuracy of 91.4%, indicating its ability to correctly classify instances. Recall (0.97), demonstrating high effectiveness in identifying phishing instances while minimising false positives and false negatives.

● Random Forest: Random Forest is an ensemble learning method that constructs multiple decision trees and aggregates their predictions. This model achieves high accuracy (90.02%). Random Forests excel in handling complex datasets like phishing detection, providing robust performance with high accuracy and reliability.

● SVM (Support Vector Machine): SVM is a powerful supervised learning algorithm used for both classification and regression tasks. The SVM model in this context achieves an accuracy of 88% with recall of 0.83. While SVM performs well in accuracy and precision, its recall rate suggests some difficulty in capturing all phishing instances effectively.

● Stacking (Random Forest, Gradient Boosting) with Meta classifier of Logistic Regression: Stacking is an ensemble learning technique that combines base models such as Random Forest and Gradient Boosting [-], leveraging their strengths to achieve exceptional predictive performance. In this case, the stacking ensemble with Logistic Regression as the meta classifier achieves outstanding accuracy (92%), along with recall (0.97) demonstrating superior performance in phishing detection.

Among the stacking configurations evaluated, the ensemble model "Stacking (Random Forest, Gradient Boosting) with a meta classifier of Logistic Regression" stands out with the highest accuracy of 99.7%. This indicates superior performance

in identifying phishing attempts compared to other stacking configurations and individual models. Additionally, recall highlighting its effectiveness in leveraging the combined strengths of Random Forest and Gradient Boosting as base classifiers, guided by the Logistic Regression meta classifier. The exceptional accuracy underscores the potential of this ensemble approach for robust phishing detection in real-world applications.

5.2.1 ACCURACY

The fraction of the sample that is accurately corrected is the accuracy. Figure 5.5 is a comparison graphic that shows how accurate the five algorithms.

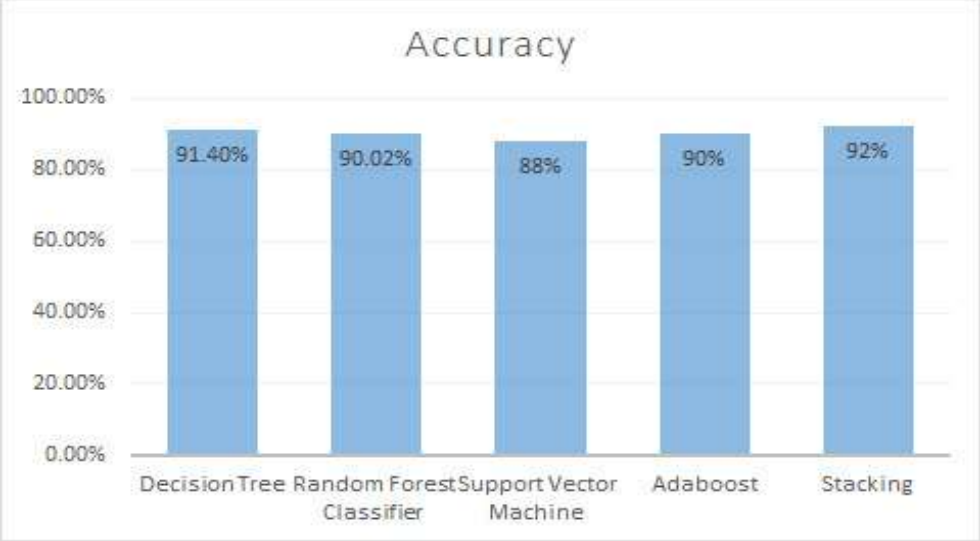


Figure 5.6

5.2.2 RECALL

Referred to as recall, sensitivity, or true positive rate, it quantifies the percentage of pertinent cases that a categorization model correctly detects. A comparative plot demonstrates the recall performance of three algorithms in the context of Figure 5.6

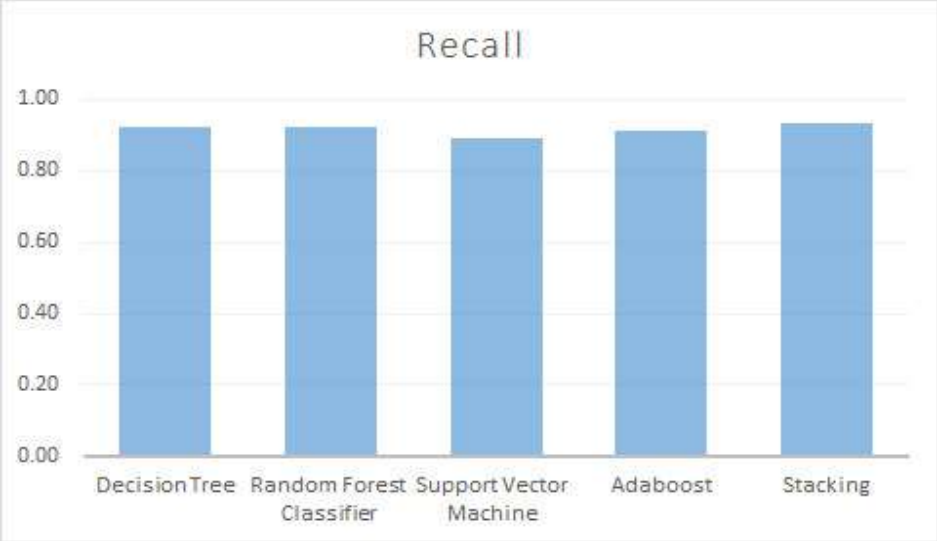

Recall
1.00
0.80
0.60
0.40
0.20
0.00
Decision Tree
Random Forest Classifier
Support Vector Machine
Adaboost
Stacking

Figure 5.7

CHAPTER 6: Conclusions and Future Scope

6.1 CONCLUSION

In this rapidly expanding technological world, phishing seems incredibly dangerous. To keep pace with the global economy, every country is focusing on cashless commerce, online shopping, paperless ticketing, and other services but phishing is starting to hinder this growth. People don't think the internet is reliable. Artificial intelligence (AI) can be used to collect data and gather sophisticated data. The average person who has no idea how to identify security risks will never take a risk by trading money online. Fishermen are targeting the installment market because of the huge benefits of the cloud.

The study mainly focus to explore this area by developing a case study on the use of machine learning to detect phishing websites. Its goal was to create an accurate, efficient and inexpensive phishing detection system using machine learning.

The project is written in Python.

To this end, the proposed method used four classifications of machine learning and compared the results of the four algorithms. In addition, high scores were obtained for accuracy. Support Vector Machine, Decision Tree, and Random Forest Classifier, Stacking, and XGBoost are the five methods used. Encouraging results were obtained in all four classifications, stacking was best with an accuracy of 92%. Accuracy scores can change as you use different data types and algorithms and can be more accurate than Stacking. This algorithm is used in real time to determine if a URL is phishing or genuine.

6.2 FUTURE WORK

Combinations of models can be used to further improve the model and achieve higher accuracy scores. Group learning methods in machine learning using base models work together to generate the best prediction model. Future research could go further by combining multiple classifiers trained on different aspects of the same training set to create a single classifier that can provide more reliable predictions than any of the individual classifiers.

A task to remove the system could be another variation of phishing, such as smishing. The ability of the method to evaluate collection growth should be investigated in the future. Since

it is likely that the collection will have to be extended incrementally over time, a method must be found to apply a classification incrementally to new data. Furthermore, the classification may involve comments that may ultimately lead to conversion.

Other optimization efforts include developing real-time detection application or a webapp and moving the device to a secure cloud platform.

REFERENCES

[1]. P.Lawson, C.J. Pearson, A. Crowson, C. B. Mayhorn, “Email phishing and signal detection: How persuasion principles and personality influence response patterns and accuracy”, Applied Ergonomics 86 (2020) 103084.

[2]. M. C. Masti, R. J., Soriente, C., Kostiainen, K., and Capkun, S. (2015). “Personalized security indicators to detect application phishing attacks in mobile platforms”. Available at: http://arxiv.org/abs/1502.06824.

[3]. S. S. VIDHYA1, M. NAVEENKUMAR (2023), “Efficient Email phishing using machine learning”, The International Journal of Creative Research Thoughts (IJCRT) (2023).

[4]. Sasirekha C1, Nandhini R1, Karthiga Mai N L, Bhuvaneshwari R S, Chandra V S, “Email phishing detection using machine learning”, The International Journal of Creative Research Thoughts (IJCRT) (2023).

[5].B.P. Kumar, B.K.Chowdary, Dr. N. Sridevi, “Phishing site detection using machine learning techniques”,International Research Journal Of Modernization In Engineering Technology And Science.

[6]. Ejaz, A., Mian, A.N. & Manzoor, “Life-long phishing attack detection using continual learning”, The International Standard Serial Number (ISSN).

[7] M.M.Alani , H. Tawfik (2022), “A Cloud-Based machine learning approach to phishing URL detection”, The International Journal of Creative Research Thoughts(IJCRT) (2023).

[8]. A.Bhavani A, C.M. PinnaM ,S. Likhitha, C. P. sai, “ Phishing Websites Detection Using Machine Learning”, The International Journal of Creative Research Thoughts(IJCRT)(2022).

[9]. R. Ravi1, A.A Shillare, P.P Bhoir 3, K.S. Charumathi 4, “URL Based Email phishing detection application”, The International Journal of Creative Research Thoughts (IJCRT) (2021).

[10]. I.Saha, D.Sarma, R.J. Chakma (2020), “Phishing Attacks Detection using Deep Learning Approach”, The International Journal of Creative Research Thoughts (IJCRT) (2021).

[11]. V.S. MohammaD, Mahdi D.M. Izad (2020), “Phishing Detection Using Machine Learning Techniques”, The International Standard Serial Number (ISSN) .

[12]. S.Mittal, Jayasimha S R (2020), “Detection of Phishing Attacks using Content Analysis in the Cloud”, The International Journal of Creative Research Thoughts (IJCRT) (2021).

[13]. P. Yang, "Phishing website detection based on multidimensional features driven by deep learning," IEEE Access (2020).

[14]. D. Vaya, S. Khandelwal, and T.Hadpawat. “Visual cryptography: A review. International Journal of Computer Applications,” 174:40–43, 09 2017.

[15]. N. Agrawal and S. Singh.” Origin (dynamic blacklisting) based spammer detection and spam mail filtering approach.” In 2016 Third International Conference on Digital Information Processing, Data Mining, and Wireless Communications (DIPDMWC).

[16]. O. K. Sahingoz, "Machine learning based phishing detection from URLs," Expert Systems with Applications (2019).

[17]. Ammar Odeh, "Machine Learning Techniques for Detection of Website Phishing: A Review for Promises and Challenges," in 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC).

[18]. E. A. Aldakheel, M. Zakariah, G. A. Gashgari, F. A. Almarshad and A. A. Alzahrani, "A Deep Learning-Based Innovative Technique for Phishing Detection in Modern Security

with Uniform Resource Locators," 2023 Sensors 23, no. 9: 4403 https://doi.org/10.3390/s23094403.

[19]. 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.

[20].Decision Trees. https://scikit-learn.org/stable/modules/tree.html#:~:text=Decision%20Trees%20(DTs)%20are%20a,as%20a%20piecewise%20constant%20approximation.

[21]. M. A. Adebowale, K.T. Lwin, M. A. Hossain, "Intelligent phishing detection scheme using deep learning algorithms," 2020 Journal of Enterprise Information Management 36, no. 3 (2023): 747-766.

[22].Random Forest. https://www.ibm.com/topics/random-forest

[23]. I. Tyagi, J. Shad, S. Sharma, S. Gaur and G. Kaur, "A Novel Machine Learning Approach to Detect Phishing Websites," 2018 5th International Conference on Signal Processing and Integrated Networks (SPIN), Noida, India, 2018, pp. 425-430, doi: 10.1109/SPIN.2018.8474040.

[24].Support Vector Machines (SVM). https://scikit-learn.org/stable/modules/svm.html

[25]. Stacking https://www.javatpoint.com/stacking-in-machine-learning

[26]. Ensemble Learning. https://www.simplilearn.com/ensemble-learning-article

[27].XGBoosthttps://www.simplilearn.com/what-is-xgboost-algorithm-in-machine-learning-aricle

[28]. Data Flow Diagram (DFD). https://www.lucidchart.com/pages/data-flow-diagram

[29].PhishTank https://phishtank.org/

[30].University of New Brunswick “URL dataset (ISCXURL)

https://www.unb.ca/cic/dataset s/url-2016.html

[31]Python Modules. https://docs.python.org/3/library/ipaddress.html

APPENDIX

SNIPPETS

```
[ ]  #Loading the data
     data = pd.read_csv('output_with_features.csv')
     data.head()
```

Figure 1

```
# Make predictions on the test set
y_pred = random_forest_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")

# Display additional evaluation metrics
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))

# Visualize Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False,
            xticklabels=random_forest_model.classes_, yticklabels
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```

Figure 2 Accuracy

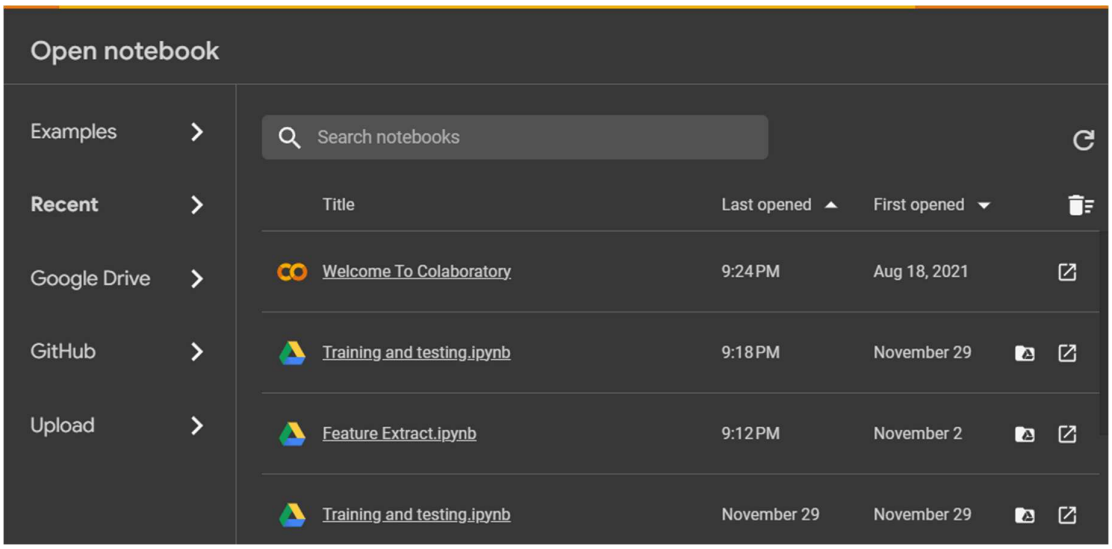


Figure 3 Google Colab

DrillBit

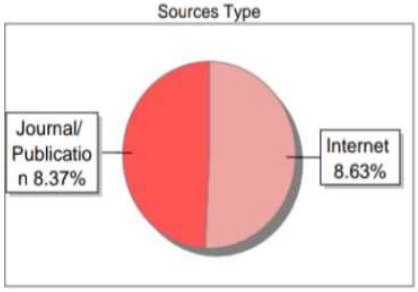
The Report is Generated by DrillBit Plagiarism Detection Software

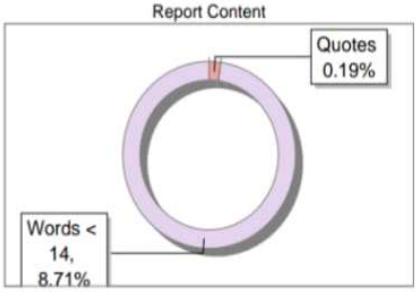
Submission Information

Author Name	Paras
Title	Phishing
Paper/Submission ID	1801073
Submitted by	anita@juit.ac.in
Submission Date	2024-05-14 11:10:24
Total Pages, Total Words	40, 6212
Document type	Project Work

Result Information

Similarity **17 %**





Exclude Information

Quotes	Not Excluded
References/Bibliography	Excluded
Source: Excluded < 14 Words	Not Excluded
Excluded Source	**0 %**
Excluded Phrases	Not Excluded

Database Selection

Language	English
Student Papers	Yes
Journals & publishers	Yes
Internet or Web	Yes
Institution Repository	Yes

A Unique QR Code use to View/Download/Share Pdf File

DrillBit

DrillBit Similarity Report

17	63	B
SIMILARITY %	MATCHED SOURCES	GRADE

A-Satisfactory (0-10%)
B-Upgrade (11-40%)
C-Poor (41-60%)
D-Unacceptable (61-100%)

LOCATION	MATCHED DOMAIN	%	SOURCE TYPE
1	www.frontiersin.org	1	Internet Data
2	blog.carninja.com	1	Internet Data
3	link.springer.com	1	Internet Data
4	bmsit.ac.in	1	Publication
5	Revealing the Unknown Real-Time Recognition of Galpagos Snake Species Using D by Patel-2020	1	Publication
6	bmsit.ac.in	<1	Publication
7	arxiv.org	<1	Publication
8	The development of a knowledge-based geographical information system for the zon by Chen-1994	<1	Publication
9	ACM Press the first ACM conference- San Antonio, TX, USA (2011.02., by Lorenzi, David Vai- 2011	<1	Publication
10	qdoc.tips	<1	Internet Data
11	www.icet.ac.in	<1	Publication
12	iabac.org	<1	Internet Data
13	mdpi.com	<1	Internet Data