

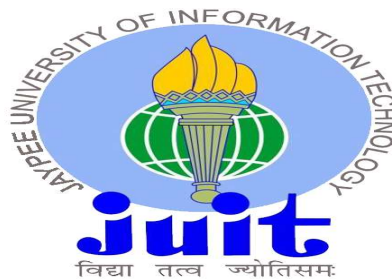
PHISHING DETECTION USING MACHINE LEARNING

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
Pankaj Sharma (201258)
Shubham Sharma (201486)

Under the guidance & supervision of
Dr. Ekta Gandotra



**Department of Computer Science & Engineering and
Information Technology**
Jaypee University of Information Technology,
Waknaghat, Solan - 173234 (India)

Candidate's Declaration

We hereby declare that the work presented in this report entitled '**PHISHING DETECTION USING MACHINE LEARNING**' 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. Ekta Gandotra** (Associate Professor, 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: Pankaj Sharma

Roll No.: 201258

(Student Signature with Date)

Student Name: Shubham Sharma

Roll No.: 201486

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. Ekta Gandotra

Designation: Associate Professor

Department: Computer Science & Engineering and Information Technology

Dated:

CERTIFICATE

This is to certify that the work which is being presented in the project report titled '**Phishing Detection Using Machine Learning**' 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 Shubham Sharma, 201486 and Pankaj Sharma, 201258 during the period from January 2024 to May 2024 under the supervision of **Dr. Ekta Gandotra** (Associate Professor, Department of Computer Science & Engineering and Information Technology). I also authenticate that I have carried out the above-mentioned project work under the proficiency stream Information Security.

Shubham Sharma 201486

Pankaj Sharma 201258

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

Dr. Ekta Gandotra

Associate Professor

Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat.

AKCNOWLEDGEMENT

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. Ekta Gandotra**, Assistant Professor, Department of CSE Jaypee University of Information Technology, Wagnaghat. 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. Ekta Gandotra**, 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.

Shubham Sharma 201486

Pankaj Sharma 201258

TABLE OF CONTENT

CANDIDATE’S DECLARATION	i
CERTIFICATE	ii
AKCNOWLEDGEMENT	iii
LIST OF TABLES	v
LIST OF FIGURES	vi
LIST OF ABBREVIATIONS	vii
ABSTRACT	viii
1. INTRODUCTION	1
1.1 INTRODUCTION	1
1.2 PROBLEM STATEMENT	2
1.3 OBJECTIVES	3
1.4 Organization of Project Report	4
2. LITERATURE SURVEY	5
2.1 OVERVIEW OF RELEVANT LITERATURE	5
2.2 KEY GAPS IN THE LITERATURE	10
3. SYSTEM DEVEPOLMENT	12
3.1 REQUIREMENTS AND ANALYSIS	12
3.2 Project Design and Architecture	13
3.3 Data Preparation	17
3.4 Implementation	22
3.5 Key Challenges	32
4. TESTING	34
4.1 Testing Strategy	30
4.2 Test Cases and Outcomes	35
5. RRESULTS AND EVALUATION	38
6. CONCLUSIONS AND FUTURE SCOPE	43
7. REFERENCES & APPENDIX	45

LIST OF TABLES

- 2.1 Literature Survey

- 3.1 Supporting python modules.
- 3.2 Description of features
- 3.3 Importing Modules
- 3.4 Importing Dataset

- 4.1 Importing python module
- 4.2 Importing data
- 4.3 Testing Case 1
- 4.4 Testing Case 2
- 4.5 Testing Case 3
- 4.6 Testing Case 4
- 4.7 Testing Case 5
- 4.8 Testing Case 6

- 5.1 Comparison

LIST OF FIGURES

- 1.1 Phishing Statistics of last five year
- 3.1 System Architecture
- 3.2 DFD of the project
- 3.3 Dataset Construction
- 3.4 Working of ML Technique
- 3.5 Importing Required files
- 3.6 Importing 're'
- 3.7 Defining the functions for feature Extraction
- 3.8 Code to generate the output file
- 3.9 Training of Decision Tree
- 3.10 Training of Random Forest tree
- 3.11 Training of SVM
- 3.12 Training of Stacking

- 5.1 Confusion matrix
- 5.2 Comparative plot of accuracy scores
- 5.3 Comparison between different models and their fl score, recall and precision

- 7.1 Loading of data
- 7.2 List of Features
- 7.3 Shuffling of rows
- 7.4 Accuracy of models
- 7.5 Google Colab
- 7.6 Test Result of url 1
- 7.7 Test Result of url 2
- 7.8 Test Result of url 3
- 7.9 Test Result of url 4

LIST OF ABBREVIATIONS

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 the modern digital age, where online communication and commerce are rapidly growing, cyber risks have similarly increased. One such risk is the deceptive tactic called phishing. Phishing attacks entail impersonating trustworthy entities in order to trick unwary users into disclosing sensitive information including login credentials, personal information, and financial information. Attacks on people's security and privacy can have serious consequences for organizations, including loss of reputation and data breaches.

The goal of this project, "Phishing Detection Using Machine Learning," is to develop a robust and efficient website which can automatically identify and mitigate phishing websites in real time using machine learning techniques. Rule-based solutions are not an effective enough way to combat increasingly changing phishing strategies. Machine learning offers the potential to enhance detection accuracy by learning from historical attack data and recognizing subtle patterns that might not be apparent to human observers.

This project explores the effectiveness of machine learning algorithms for detecting email phishing. We have investigated Support Vector Machines (SVM), Naïve Bayes, Decision Trees, and Random Forest classifiers separately and evaluated their performance. Additionally, a new way was put forward that is the stacked approach: combine base learners such as Random Forest and Gradient Boosting together with a meta-classifier like Logistic Regression. Our stacked ensemble model yielded an outstanding accuracy of 99.75%, surpassing individual classifier performance. Shows the results of our study indicate the merits of stacking and ensemble methods in improving false positive performance. They provide valuable suggestions on how to increase resilience for cyber security.

CHAPTER 1: INTRODUCTION

Phishing has emerged as the biggest issue, affecting people, businesses, and even entire nations. Phishing has become a major worry for security researchers in the current era since it is easy to develop a phishing website that closely resembles a legitimate website. Professionals can identify phishing websites nevertheless, some clients are unable to identify the fake website, and as a result, they fall prey to phishing attacks. Phishing can take many different forms, ranging from mass-produced misspelt demands for excessively precise details to complex spear phishing assaults that target specific personal information. In our data-intensive era, the ability of phishing attacks to stealthily obtain your private credentials can leave you severely exposed. It only takes one person to make the crucial error of clicking on a malicious link. A user can unintentionally expose themselves and their data to various risks, such as drive-by downloads, cross-site scripting attacks, and data harvesting via seemingly innocent web forms.

Emails and websites are the two primary delivery channels for phishing attacks. The most common of them, emails, are traditionally connected to phishing. Malicious URLs that point users to maliciously created content are frequently included in these. A few straightforward modifications can quickly put consumers in an unfamiliar and uneasy situation by masking the true destination of a URL.

Phishing attacks represent a significant and growing threat in today's digital world, accounting for 36% of all US data breaches in 2023 alone. These attacks target individuals and organizations alike, with 1339 brands falling victim to phishing schemes in the fourth quarter of 2023. The number of unique phishing sites soared to 5 million in the same year, highlighting the scale of this pervasive cyber threat. These attacks aren't just common; they're costly. In 2023, phishing became the second most expensive source of compromised credentials, impacting various sectors, including healthcare, which has consistently borne the brunt of data breaches over the past 13 years. Understanding the types of phishing, such as email, spear, and whaling, is vital, as is recognizing the staggering economic toll, with losses exceeding \$10.3 billion reported in 2022. [1]



Figure 1.: Phishing Statistics of last five years [1]

The number of reported phishing assaults has climbed by 1,139% between 2018 and 2022, indicating a sharp rise in phishing frauds.

1.2 PROBLEM STATEMENT

The purpose of this project, "Phishing Detection Using Machine Learning," is to create a reliable and effective system that uses machine learning techniques to automatically detect and mitigate phishing assaults in real-time. Rule-based solutions are not an effective enough way to combat increasingly changing phishing strategies. Machine learning offers the potential to enhance detection accuracy by learning from historical attack data and recognizing subtle patterns that might not be apparent to human observers.

It involves collecting both legitimate and phishing URLs, as well as any other information required for the task. Some features like URL length, IP address, Domain age, etc will be extracted from the collected URLs. The URLs will be organized in a database and used for training and detecting malicious websites. Various machine learning algorithms, such as support vector machines, random forest tree, Stacking etc. will be used and compared to determine the most effective approach for detecting phishing attempts. Feature extraction

will play a crucial role in extracting relevant information from, text content, and URL structures, enabling the models to make right decisions.

By successfully developing and implementing an accurate phishing detection system, it aims to contribute significantly to the ongoing battle against cyber threats. The outcomes of it will extend beyond individual users, benefiting businesses, organizations, and the broader digital ecosystem by improving cyber security measures and fostering a safer online environment.

It aims to use machine learning to detect phishing attempts, providing a proactive and intelligent solution to this issue. It combines the power of machine learning algorithms, extensive feature engineering, and real-time responsiveness to create a comprehensive defence mechanism against an ever-evolving array of phishing tactics.

1.3 OBJECTIVES

The following are the project's goals:

1. To prepare our own dataset selected for a phishing website.
2. To Implement and compare various Machine Learning (Ensemble) algorithms for detection of phishing websites.
3. To Implement a Real-Time Phishing Detection System.

1.4 Significance and Motivation of the Project Work

We can stay safe from phishing websites by using a variety of anti-phishing strategies. Google Safe Browsing (GSB) is a service that Mozilla Firefox, Safari, and Google Chrome employ to block phishing websites. There are numerous other commonly used tools of this type, like McAfee Site Advisor, Quick Heal, Avast, and Netcraft. The blacklist method is used by GSB to analyze a URL. The primary shortcoming of GSB was its inability to identify phishing websites because the blacklist was not kept up to date. In the case of Netcraft, a phishing website was flagged as such even if it wasn't blocked. Netcraft only blocks websites when it is very certain that they are phishing. Only when the user selects the right button on the icon to view the risk rating does the warning appear. The danger arises when someone decides to use it without first examining the rating or decides to use it after doing so. Software such as QuickHeal and Avast provide protection against internet

security assaults. Following installation, the Avast antivirus's functionality was examined. The phisher URL, which Netcraft and GSB had correctly identified, was not found by the Avast browser. The previously mentioned premise acknowledges the need for sophisticated anti-phishing solutions. It is important to note that each of these tools needs to be installed separately. If a layperson is unaware of techniques such as phishing, he may never install tools. If so, then GSB service is the only one that people use. Therefore, it is crucial to be aware of anti-phishing tools and phishing. Furthermore, no one should rely solely on tools because it is known that doing so could result in misclassification.

1.5 Organization of Project Report

The progress we accomplished is covered in this report. The following is the format of the final five chapters:

Chapter 2: Explains earlier phishing strategies, their benefits, and how they vary from my approach.

Chapter 3: Explains the process of gathering requirements for the system and the iterative design process that was used to construct the tool based on the necessary needs. It also covers the features included in the tool and the final design.

Chapter 4: Compares the findings with existing solutions and analyzes the findings in light of the design requirements set forth in Chapter 3.

Chapter 5: This chapter presents the key findings of the project, analyzes the results, and identifies areas needing further research.

Chapter 6: This chapter summarizes the project's overall findings, concludes the study, and outlines important tasks and potential directions for future research.

CHAPTER 2: LITERATURE SURVEY

2.1 OVERVIEW OF RELEVANT LITERATURE

A competent report that covers the knowledge currently available, including noteworthy discoveries as well as theoretical and methodological commitments to a certain issue, is called a literature overview.

Aldakheel et al. have suggested an efficient model for phishing website identification that centers on the best feature selection technique [2]. By identifying phony websites and emails, they suggested a method to stop phishing assaults online. This demonstrated the value of machine learning in improving cybersecurity and protecting consumers from possible risks. They tested their dataset using a variety of techniques. The results indicate that they use CNN to get the best accuracy.

Rashid et al. [3] make use of the greatest methods available to gather the best possible set of information and identify the most appropriate Machine Learning techniques for phishing URL identification. They employ Random Forest and Support Vector Machine (SVM), two conventional machine methods. The maximum accuracy is achieved by the support vector machine between two algorithms.

In order to solve the issue of a large data set and higher classifier prediction performance, Adebowale et al. [4] proposed a deep learning-based phishing detection solution that made use of the universal resource locator and website content, including images, text, and frames. Convolution Neural Network (CNN) and Long Short-Term Memory (LSTM) were combined. The maximum accuracy was obtained using a hybrid of long short-term memory (LSTM) and convolution neural network (CNN).

While understanding the features will heavily influence the model's accuracy, feature engineering is essential in the search for solutions for phishing website detection. The time required to gather these features is the limitation, even though the features drawn from all of these different dimensions make sense. In order to address this shortcoming, the authors have put forth a multidimensional phishing detection feature [5] strategy that leverages deep learning (MFPD) to focus on a quick detection method.

A three-phase detection system known as the Web Crawler based Phishing Attack Detector (WC-PAD) [6] has been proposed to accurately detect instances of phishing. The URL, traffic, and content of the web are used as input features for this. Classification is now complete taking these features into account.

Phishing Net [7] is a deep learning-based method for quickly identifying phishing URLs. A technique for determining whether a webpage is phishing or authentic is parse tree validation [8]. This is a creative method for locating these websites: using Google's API, intercept each hyperlink on a current page, then create a parse tree using all of the intercepted hyperlinks. Parsing in this case starts at the root node. In order to ascertain whether any child node shares the same value as the root node, the Depth-FirstSearch (DFS) method is employed.

Tyagi et al.'s study [9] concentrated on a number of machine learning methods designed to determine if a website is authentic or fraudulent. Machine learning solutions are chosen because they can identify zero-hour phishing assaults and are more adept at handling novel forms of phishing attacks. In their application, they were able to anticipate a website's legitimacy or phishing activity with 98.1% accuracy.

A model that uses the Random Forest algorithm for URL detection to identify phishing websites was proposed by Parekh et al. [10]. They used the PhishTank dataset and a standard crawl to find both phishing and trustworthy URLs. They exclusively employ the Random Forest algorithm, which predicted 95% of websites to be either authentic or phishing.

A voting classifier was introduced by Gupta et al. [11] as a sort of ensemble learning to determine the accuracy of various combinations of classifiers. The results demonstrate that using a voting classifier yields predictions that are more accurate than those made by an individual classifier. Utilizing the two together yields an accuracy rate of 98.2%.

A method that integrates to create an online tool for the identification, verification, and gathering of phishing websites. [12]. This online tool checks and identifies phishing websites while continuously monitoring the PhishTank blacklist.

S. No.	Paper Title [Cite]	Journal/ Conference (Year)	Tools/ Techniques/ Dataset	Results	Limitations
1.	A Deep Learning Based Innovative Technique for Phishing Detection in Modern Security with Uniform Resource Locators. [2]	MDPI Journal (2023)	KNN, SVM, Random Forest Classifier, RNN, NLP, 1D CNN. PhishTank, University of California Irvine	With an accuracy rate of 98.77%, 1D CNN was the most accurate.	Limitations in this paper include the scope of the dataset.
2.	Phishing detection using machine learning techniques [3]	IEEE Conference (2020)	Random Forest, Support Vector Machine (SVM) University of California Irvine	Support Vector Machine achieve accuracy of 95.66%.	The paper does not evaluate the proposed ml model on a large and diverse dataset of phishing emails.

3.	Intelligent phishing detection scheme using deep learning algorithms. [4]	Journal (2020)	Convolution Neural Network (CNN), Long Short-Term Memory (LSTM) PhishTank and Common Crawl	Hybrid (CNN+ LSTM) gave excellent classification accuracy of 93.28%.	In this, the algorithm takes a lot of time to train and can improve overall accuracy.
4.	Phishing Website Detection Based on Multidimensional Features Driven by Deep Learning [5]	IEEE Conference (2019)	KNN, Random Forest Classifier, SVM, LSTM	It achieved 98.8% accuracy.	It's a more costly method because it needs more calculation.
5.	WC-PAD: Web Crawling based Phishing Attack Detection [6]	IEEE Conference (2019)	PhishTank and Common Crawl	It provides detection accuracy of 98.9% for both phishing and zero-day phishing attacks.	time-consuming because there are three stages involved, and each website must go through them.

6.	Phishing URL Detection via CNN and Attention-Based Hierarchical RNN [7]	IEEE Conference (2019)	CNN and RNN PhishTank and OpenPhish	CNN and RNN achieves 98% result.	False positive rate is high
7.	Phishing Detection in Websites using Parse Tree Validation [8]	(2018)	Kstart, Random Forest, SMO, XCS and Naïve Bayes.	XCS achieves 98.41% accuracy.	The false negative and false positive rates are high.
8.	Phishing Detection Using Machine Learning Technique [9]	IEEE Conference (2018)	Decision Tree, Random Forest, KNN, Neural Network, SVM linear, logistic regression, XGBoost, Gradient Boosting.	XGBoost achieve highest accuracy 98.1%	Continuously need to update dataset to maintain detection accuracy.
9.	A New Method for Detection of Phishing Websites: URL Detection. [10]	IEEE Conference (2018)	Random Forest PhishTank and Common Crawl	It achieves highest accuracy of 95%.	It only uses one method to detect and with limited dataset

10.	Spam Detection Using Ensemble Learning. [11]	Springer Conference (2018)	Gaussian Naive Bayes, Multinomial Naive Bayes, Bernoulli Naive Bayes, and Decision tree University of California Irvine	A maximum accuracy of 98.2% is achieved by combining decision trees, Bernoulli Naive Bayes, and Gaussian Naive Bayes.	This paper does not discuss the computational cost of training and deploying the proposed ensemble learning mode
11.	PhishBox: An approach for phishing validation and detection [12]	IEEE Conference (2018)	PhishTank and Alexa Top List	They achieved 95% accuracy.	The data on the blacklist was deemed invalid when viewed at a 12-hour period.

Table 2.1: Literature Survey.

2.2 KEY GAPS IN THE LITERATURE

Some key gaps in the literature are:

1. Limited Comparative Analysis: A thorough comparison of different machine learning and deep learning techniques across different research is lacking in the literature. 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 models that combine the strengths 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 DEVELOPMENT

3.1 REQUIREMENTS AND ANALYSIS

3.1.1 HARDWARE REQUIREMENT

- Processor : AMD or Intel 3rd generation or Higher.
- Total Space - Minimum 250MB Space required
- RAM - 4GB at least.

3.1.2 SOFTWARE REQUIREMENT

- Language : Python.
- O.S : Windows 10 or higher.
- IDE - Jupyter Lab , Python ver 3.x or Google Colab.

3.1.3 Supporting Python programs

Python offers a method for putting definitions in a document and using them in a content or interpreter's intuitive scenario. A module is a file of this type; definitions from one module can be included into other modules or the core module [13]. Table 2 displays a selection of the project's modules.

Sr. No.	Python Module	Description
1.	Ipaddres	The ability to create, manage, and operate IPv4 and IPv6 addresses and networks is provided by ipaddress.
2.	Re	Similar to Perl's regular expression matching exercises, this module provides them.

3.	urllib.request	This module, urllib.request, describes classes and functions that make it easier to open URLs (mostly HTTP) in a complicated world.
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 Python-based library permits the sending of HTTP requests.
7.	Whois	A widely used protocol for queries and answers, WHOIS is used to address databases that hold the chosen users or trustees of an online resource. for instance, an IP address block, domain name, or autonomous framework that is also concurrently utilised for a large amount of information.

Table 3.1: Supporting Python programs.

3.2 Project Design and Architecture

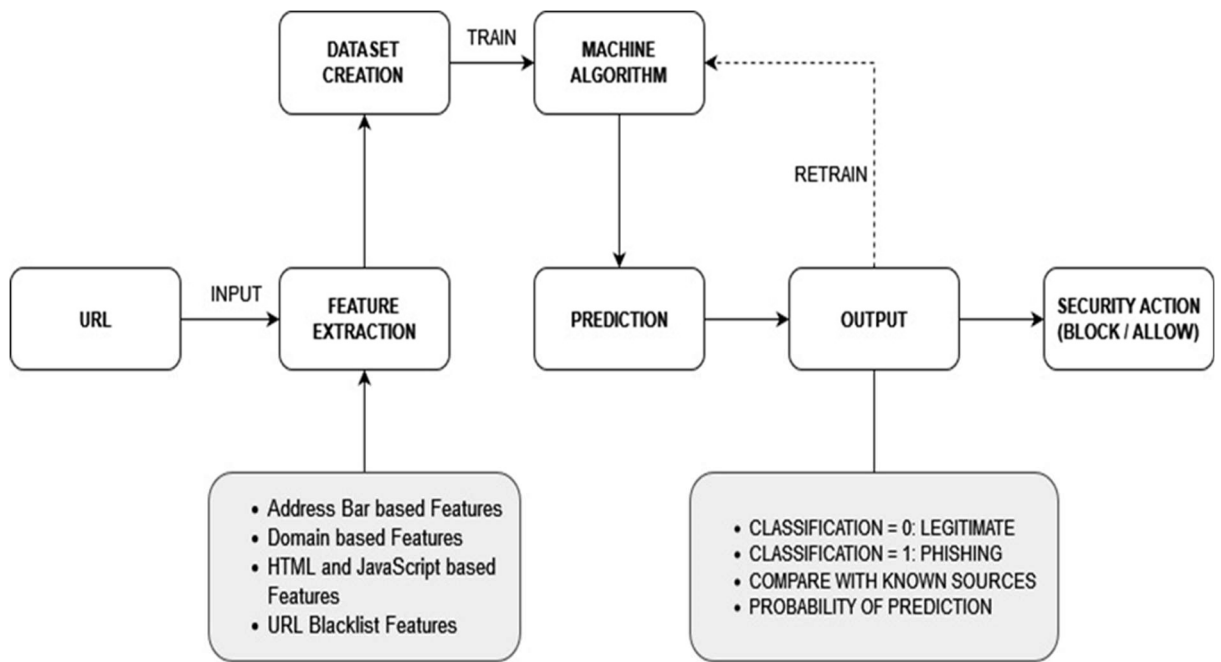


Figure 3.1: System Architecture

In this project, we have predicted phishing websites whether they are good URLs or bad URLs.

Figure 3.1 illustrates the system's architecture. The URLs are sent to the appropriate machine learning algorithm for classification as either phishing or authentic. The classifier being trained to determine whether URLs from the training dataset are phishing or legitimate then uses the pattern found in the freshly fed input to classify it.

A list of the values for the attributes that are extracted from the URL, like the IP address, domain, count '@', URL length, etc., is produced. The generated dataset is subsequently fed into classifiers like SVM, Decision Tree, Random Forest classifier, and stacking. The accuracy score is then produced when the performance of these models is assessed. The resulting list is used by the trained classifier to predict if a given URL is phishing or legitimate.

It's essential to emphasise that the dataset adopts a binary encoding, with values '1' denoting phishing URLs and '0' representing legitimate ones. This clear distinction enables the trained classifiers to produce precise forecasts generated from the acquired patterns,

contributing to the robustness of the phishing detection system. The integration of various classifiers and a feature-rich dataset underscores the sophistication of the predictive model employed in this project.

Through various evaluations of these machine learning models, we generate accuracy scores to quantify their performance. The classifier, armed with the learned patterns, efficiently predicts the legitimacy of newly encountered URLs. This approach not only enhances the precision of phishing detection but also contributes to the adaptability of our system, ensuring its efficacy against a diverse range of potential threats

It aims to use machine learning to detect phishing attempts, providing a proactive and intelligent solution to this issue. It combines the power of machine learning algorithms, extensive feature engineering, and real-time responsiveness to create a comprehensive defence mechanism against an ever-evolving array of phishing tactics.

3.2.1 DataFlow Diagram

Data Flow diagrams (DFDs) are used to show how data flows through a system. It describes the steps that are involved in a system, starting with input and ending with report generation. It displays every route that could lead from one system entity to another.

The system's operations are divided into three categories by the data flow diagram (dfd) [14], which provides a more thorough view of the system through feature scaling, classification, and preprocessing. It also provides a comprehensive overview of the system's operation by graphically illustrating each of these categories in detail.

The diagram below provides a visual representation of the system's design, the variously complex input to output processes, and the system's behaviour to aid in understanding.

In Figure 3.2 feature scaling is used, which involves normalizing the range of independent variables or data characteristics, is an essential preprocessing step in machine learning. This guarantees that every feature makes an equal contribution to the model, which is crucial when features have various sizes or units. The given workflow builds and fits a feature scaling model, such as Standard or MinMax from `sklearn.preprocessing`, using the training data after dividing the dataset into training and testing sets.

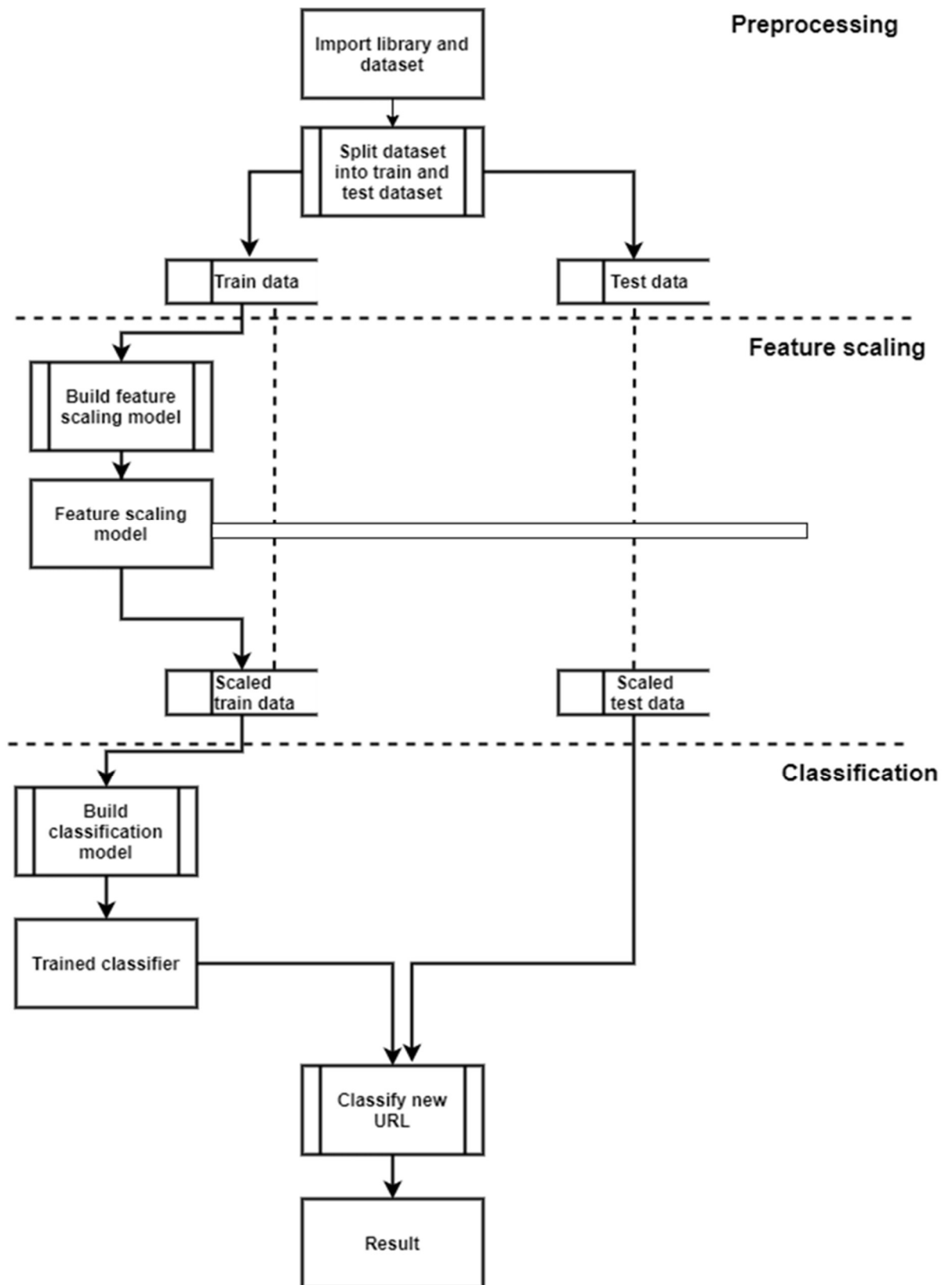


Figure 3.2: DFD of the project

The training and testing datasets are then subjected to this scaling model, which brings them both down to a comparable scale. This improves the efficiency of the training process and the performance of the final classifier, which can handle features more reliably and precisely and, in turn, produces better prediction results for incoming data points.

3.3 Data Preparation

A critical aspect of our phishing detection system lies in the data preparation process. This well-defined procedure ensures the efficacy of the machine learning algorithms in differentiating between legitimate and fraudulent URLs.

3.1 URL COLLECTION

The initial and crucial stage of data preparation involves meticulously collecting URLs from dependable sources. To construct a robust dataset, we strategically gather data from reputable sources, including PhishTank [15] and the URL dataset from the University of New Brunswick [16]. These sources provide a comprehensive collection of both phishing and legitimate URLs, encompassing a total of 40,919 entries. Here, 21,236 URLs are identified as legitimate, while the remaining 19,683 are classified as phishing attempts.

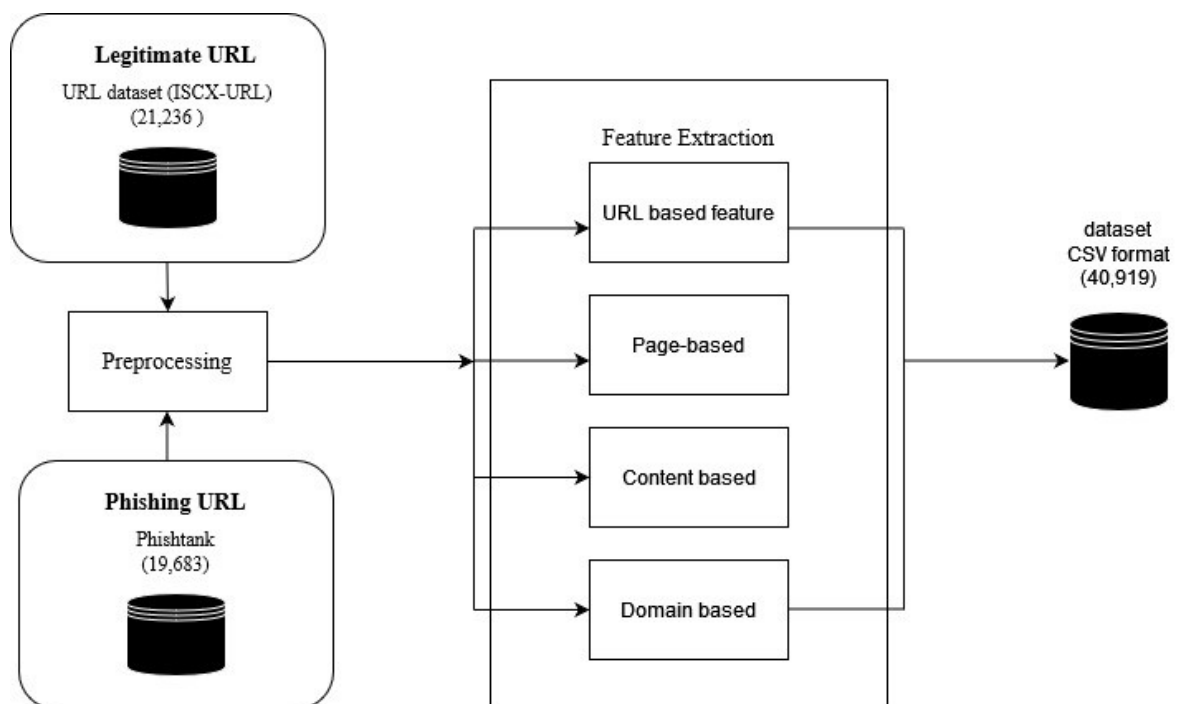


Figure 3.3: Dataset Construction

We take a systematic approach to collect URLs, ensuring the collected data is comprehensive, correctly sample, and up-to-date. PhishTank maintains a real-time database of phishing URLs, regularly updating the URLs collected by security researchers and security solution providers. It allowed our dataset to capture the true state of the latest phishing of URLs and trends. The University of New Brunswick's dataset of URLs provides a collection of legitimate URLs, serving as a base for comparison of the fake links. By combining these trusted sources, we believe we have a solid foundation for our machine learning models.

3.2 Feature Extraction

Once we have separated legitimate and phishing URLs into two distinct files, we can move on to the next step, which is feature extraction. This critical stage involves extracting a comprehensive set of features from each URL within the merged dataset. These features are carefully chosen to encompass various aspects of a URL's structure, content, and potential red flags indicative of phishing attempts. By conducting this process, we ensure that our machine learning classifiers will have all the necessary data to differentiate between legal and fraudulent URLs.

The feature extraction process meticulously extracts 24 distinct features from each URL. These features can be broadly categorized into three groups: URL-based features, Domain based features, Page-based features and Content-based features.

3.2.1 URL-based features

URL-based features focus on the inherent structure of the URL itself, dissecting various elements that can provide clues regarding its legitimacy. Examining elements like the presence of an IP address, the number of dots (.), the length of the hostname, the presence of suspicious characters, and the count of hyphens (-) can all offer valuable insights.

Legitimate websites typically utilize domain names for user-friendliness and easier memorability. Conversely, phishing attempts often resort to IP addresses to obscure their malicious intent and potentially bypass domain blacklisting efforts. A higher number of dots within a URL might suggest a URL with an unusual structure, potentially indicating a phishing attempt. Phishing URLs may adopt abnormally short or long hostnames to deceive users or make them appear more trustworthy by mimicking legitimate websites.

Special characters like '%', '&', '?' and '=' are frequently employed in phishing URLs to obfuscate the true destination or manipulate parameters. The presence of an excessive number of these characters can raise a red flag. An unusually high number of hyphens might signify an attempt to create a misleading or confusing URL resembling a legitimate website.

3.2.2 Domain-based features

Unlike legitimate websites with user-friendly domain names, phishing attempts often hide behind red flags. Domain-based features expose these deceptive tactics. The presence of IP addresses (`use_of_ip`) or unusual characters (`abnormal_url`) can signal a suspicious domain. Additionally, features like the number of dots separating subdomains (`count_dir`) and the existence of embedded domains (`count_embed_domain`) can reveal attempts to create misleading website structures. Shortened URLs (`short_url`) and abnormal domain name lengths (`hostname_length`) further expose potential phishing attempts.

Machine learning models delve even deeper. They can be trained to identify specific patterns or keywords within domain names that are commonly associated with phishing (`sus_url`). By analyzing the Top-Level Domain (Tld) - the suffix like `.com` or `.org` - and its length (`tld_length`), the system can uncover domains with a higher likelihood of being used for phishing. By scrutinizing these domain-based features, machine learning models learn to distinguish legitimate websites from malicious ones, ultimately protecting users from falling victim to phishing scams.

3.2.3 Page-based features

Page-based features focus on the attributes and metadata of the web page itself rather than its URL or domain. These features help in identifying whether a webpage is legitimate or part of a phishing scam. One crucial page-based feature is the Google Index status (`google_index`), which indicates whether the page is indexed by Google. Legitimate websites are typically indexed by Google, while phishing websites often are not, as they aim to remain hidden from search engines to avoid detection.

Machine learning models utilize these features to analyze patterns and identify anomalies. For example, the absence of a page from Google's index can be a strong indicator of potential phishing. Additionally, other metadata such as the age of the webpage, the presence of SSL certificates, and the loading time can provide further insights into the legitimacy of the webpage. By examining these page-based features, machine learning models can better differentiate between trustworthy websites and phishing attempts, enhancing user security.

3.2.4 Content-based features

Content-based features scrutinize the actual content of a webpage to detect signs of phishing. These features focus on the elements within the webpage, such as text, images, links, and scripts, to identify suspicious activities. One significant content-based feature is the Suspicious URL (`sus_url`), which involves analyzing the content of the URL to spot patterns or keywords commonly associated with phishing.

Moreover, content-based features can include the analysis of the webpage's text for common phishing phrases, the examination of images for signs of tampering or misuse, and the inspection of scripts for malicious code. For instance, a high number of form fields asking for sensitive information or the presence of deceptive images mimicking trusted brands can be red flags.

Machine learning models trained on these content-based features can identify and flag webpages that contain suspicious elements, thereby preventing users from interacting with potentially harmful content. By leveraging these detailed content inspections, models provide a robust defense against phishing attempts, ensuring a safer browsing experience.

After merging two files, we extracted 24 features from the URL. These features are explained in Table 3.2. We have uploaded our dataset on Kaggle and link for dataset is:

<https://www.kaggle.com/datasets/pankajsharma78/phishing-website-detection>

Category	Feature	Description	Data Type
URL-based	abnormal_url	Detection of abnormal URL patterns	Integer
	count(.)	Number of dots (.) in the URL	Integer
	count-(www)	Number of 'www' in the URL	Integer
	count-(@)	Number of '@' symbols in the URL	Integer
	count-(?)	Number of '?' symbols in the URL	Integer
	count(-)	Number of '-' symbols in the URL	Integer
	count-(=)	Number of '=' symbols in the URL	Integer
	url_length	Length of the entire URL	Integer
	count-(dir)	Number of elements associated with the directory path	Integer
	count_embed_domain	Number of embedded domains within the URL	Integer
	short_url	Presence of a shortened URL	Integer

	count-https	Number of 'https' in the URL	Integer
	count-http	Number of 'http' in the URL (likely always 1 or 0)	Integer
	count-(%)	Number of '%' symbols in the URL	Integer
	count-digits	Number of digits (0-9) in the hostname	Integer
	count-letters	Number of letters (a-z, A-Z) in the hostname	Integer
	fd_length	Length of the first directory in the URL	Integer
Page-based	google_index	Google index status of the URL	Integer
Domain-based	use_of_ip	Presence of an IP address in the URL	Integer
	hostname_length	Length of the hostname portion of the URL	Integer
	Tld	Top-level domain (e.g., .com, .org) of the URL	Integer
	tld_length	Length of the top-level domain	Integer

Content-baesd	sus_url	Suspicious URL indicator	Integer
---------------	---------	--------------------------	---------

Table 3.2: Description of features

3.4 IMPLEMENTATION

The report outlines a method for classifying URLs as authentic or phishing, which involves creating a training set comprising URLs labelled with their corresponding classifications. This training set is then used to train a classifier or machine learning model, enabling it to discern patterns and features indicative of authenticity or phishing attempts. Fig. 3.4 likely illustrates this process diagrammatically. Through this approach, the classifier learns from the provided examples and can subsequently make predictions on new URLs, aiding in the identification of potential phishing threats.

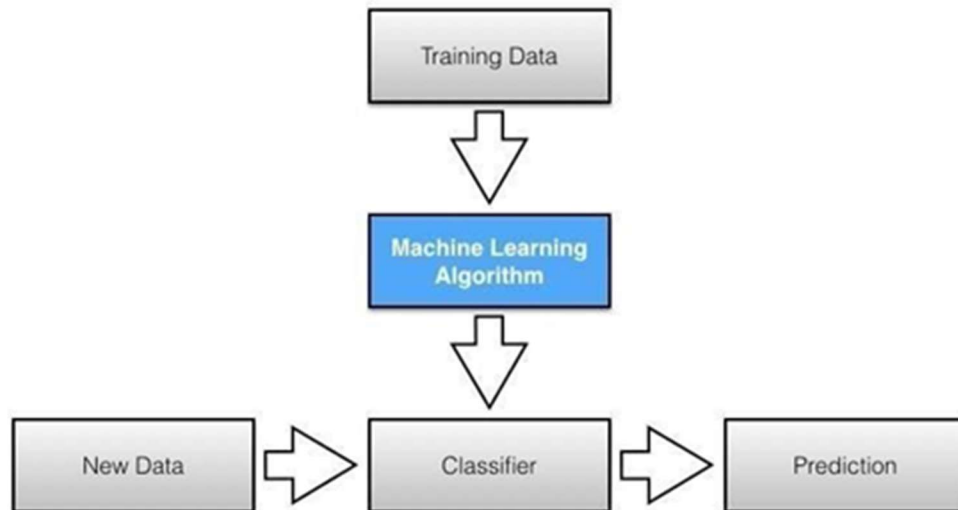


Figure 3.4: Working of ML Technique

We first created our own dataset. In which we extracted 24 features shown in table 3.2

3.4.1 TOOLS

- a. Python Language
- b. Google Colab
- c. Machine learning Library
- d. URL from PhishTank
- e. URL from University of New Brunswick

3.4.2 TECHNIQUES

We used various machine learning algorithm to train our dataset such as Decision Tree, Naïve Bayes, Random Forest Tree, SVM, Logistic Regression, ENSEMBLE Technique(Stacking, Gradient boosting)

- **Decision Tree:** An algorithm with just conditional control statements is provided using a decision tree, which is a hierarchical decision support model that uses a tree-like representation of decisions and probability outcomes, such as resource costs, benefits, and contingency outcomes. [17]
- **Naïve Bayes:** Naive Bayes classifiers are a class of algorithms for classification that rely on Bayes' Theorem. Rather than being a single technique, it is essentially a family of algorithms founded on the idea that every pair of features that is being classed stands alone. [18]
- **Random Forest Tree:** The Random Forest classifier applies multiple decision trees to different dataset subsets, and averages them to increase the prediction accuracy of the dataset. Among supervised learning methods, a well-known machine learning technique is Random Forest. This can be applied to ML contexts of classification and regression. It is based on the concept of ensemble learning.[19]
- **Support Vector Machine (SVM):** A machine learning algorithm called the support vector machine (SVM) 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.[20]
- **Logistic Regression:** Logistic regression is one of the supervised machine learning techniques used primarily in classification problems to determine whether a

particular pattern belongs to a particular class A statistical technique called logistic regression can be used to determine how two data items are highly correlated. [21]

- **Stacking:** One technique for grouping several classifications or regression models is stacking. Stacking, also known as Stacked Generalisation, 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 utilise them to construct an intermediate prediction, one for every model that has been learnt.[22]
- **Gradient boosting:** A robust boosting approach called gradient boosting turns several weak learners into strong learners. Gradient descent is used to train each new model to minimise the loss function, which could be anything from the mean squared error to the cross-entropy of the prior model.[23]

3.4.3 CODE SNIPPETS: FEATURE EXTRACTION & MODEL TRAINING

```
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
```

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

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

```
(22798, 2)
```

	url	type
0	br-icloud.com.br	0.0
1	signin.eby.de.zukruygxctzmmqi.civpro.co.za	0.0
2	http://www.marketingbyinternet.com/mo/e56508df...	0.0
3	https://docs.google.com/spreadsheets/viewform?f...	0.0
4	retajconsultancy.com	0.0

Figure 3.5. Importing Required files

In Figure 3.5, we are importing all the required libraries for the extraction of the features. And 'malicious_phish.csv' contains all the merged phishing and legitimate URLs, which are shown in the above image.

In Figure 3.6 ,we have imported 're', which is used to evaluate the regular expression. Through this we will find the ip address of the url by the '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.

```
[ ] 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])\\.'
        '([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))
```

Figure 3.6 Importing 're'

```

from googlesearch import search

def google_index(url):
    site = search(url, 5)
    return 1 if site else 0
df['google_index'] = df['url'].apply(lambda i: google_index(i))

def count_dot(url):
    count_dot = url.count('.')
    return count_dot

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

def count_www(url):
    url.count('www')
    return url.count('www')

df['count-www'] = df['url'].apply(lambda i: count_www(i))

def count_atrate(url):
    return url.count('@')

df['count@'] = df['url'].apply(lambda i: count_atrate(i))

def no_of_dir(url):
    urldir = urlparse(url).path
    return urldir.count('/')

```

Figure 3.7. Defining the functions for feature Extraction

In figure 3.7 we have written some function for feature extraction, and generated file 'output_with_features.csv' which contain the features extracted using all the function.

```

[ ] 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 Code to generate the output file

In figure 3.8, we are using a 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.
- **Train-Test Split:** An 80-20 split is used to divide the dataset into training and testing sets, and the split is done using a random seed for reproducibility.
- **Decision Tree Model Initialization:** A given random seed is used to generate an instance of the Decision Tree classifier.
- **Model Training:** The training set of data is fitted to the Decision Tree model.
- **Model Prediction:** The test set is used to generate predictions.
- **Model Evaluation:** The accuracy of the model is calculated and printed.

- **Additional Metrics and Confusion Matrix:** To give more evaluation metrics, the categorization report and confusion matrix are printed.
- **Visualising Confusion Matrix:** Using matplotlib and seaborn, a heatmap of the confusion matrix is plotted.

This code offers a thorough illustration of how to create, train, and assess a Decision Tree model for a binary or multiclass classification issue. The model's performance on the test set is visually represented by the confusion matrix heatmap.

```
[18] 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}")

      # 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 3.9 Training of Decision Tree

```

import pandas as pd
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}")

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

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

```

Fig 3.10. Train Random Forest

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

- **Train-Test Split:** An 80-20 split is used to divide the dataset into training and testing sets, and the split is done using a random seed for reproducibility.
- **Random Forest Model Initialization:** A given random seed is used to generate an instance of the Random Forest classifier.
- **Model Training:** The Random Forest model is fitted to the training data.
- **Model Prediction:** The test set is used to generate predictions.
- **Model Evaluation:** The accuracy of the model is calculated and printed.
- **Additional Metrics and Confusion Matrix:** To give more evaluation metrics, the categorization report and confusion matrix are printed.
- **Visualising Confusion Matrix:** Using matplotlib and seaborn, a heatmap of the confusion matrix is plotted.

While the format of this code is similar to that of the preceding example, a Random Forest classifier is used in place of a Decision Tree. The Random Forest technique is a group

approach that constructs several decision trees and combines their forecasts to get outcomes that are more reliable and precise. The model's performance on the test set is visually represented by the confusion matrix heatmap.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
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 SVM model
svm_model = SVC(kernel='linear', random_state=12)

# 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 Training of SVM

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

- **SVM Model Initialization:** Using a linear kernel and a predetermined random seed, an instance of the SVM classifier is generated.
- **Model Training:** The training set of data is fitted to the SVM model.
- **Model Prediction:** The test set is used to generate predictions.
- **Model Evaluation:** The accuracy of the model is calculated and printed.

This code explains the steps involved in applying a linear support vector machine (SVM) to a classification problem: data preparation, model training, prediction, and

evaluation. The model's performance on the test set is visually represented by the confusion matrix heatmap. For data that can be separated linearly, the SVM's linear kernel is appropriate.

```
[ ] 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)

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

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

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

Figure 3.12. Training of Stacking

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

- **Define Base Models:** Two base models, a Random Forest Classifier and a Gradient Boosting Classifier, are defined as a list of tuples.
- **Define Meta-Model:** A Logistic Regression model is defined as the meta-model, which takes predictions from base models as input.
- **Create Stacking Model:** The Stacking Classifier is created with the defined base models and meta-model.
- **Fit Stacking Model:** The Stacking model is trained on the training data

- **Make Predictions and Evaluate:** On the test set, predictions are produced, and the stacking model's accuracy is computed and reported.

This code showcases a simple ensemble approach where predictions from multiple base models (Random Forest and Gradient Boosting) are combined using a Logistic Regression meta-model. Stacking allows the model to take advantage of the advantages of many algorithms and potentially improve overall performance. The accuracy metric is used to evaluate the stacking model on the test set.

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 non-phishing datasets:** One of the first challenges is to combine datasets containing phishing and non-phishing 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 standardised 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 non-phishing 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 analyzed 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. We make sure that we use the same amount of phishing and legitimate URL for the dataset.

The challenges encountered highlight the importance of robust data sets in the success of phishing detection systems and highlight the need for ongoing flexibility in the development and preconstruction of data systems emphasise

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

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

Case	01
Name	Importing modules
Input	Import module statements
Output	The module was imported successfully and usable.
Remark	Success

Table 4.1: Importing python module

4.2.2 Importing of the data

Case	02
Name	Importing dataset
Input	Import dataset statements
Output	The dataset was imported successfully and usable.
Remark	Success

Table 4.2: Importing data

4.2.3 Unit Testing

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

Table 4.3: Testing Case 1

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

Table 4.4: Testing Case 2

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

Table 4.5: 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 4.6: Testing Case 4

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

Table 4.7: Testing Case 5

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

Table 4.8: 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. The confusion matrix [24] is represented in abstract form in fig. 5.1. :

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Figure 5.1: Confusion matrix

True positives (TP), True negatives (TN), False positives (FP), and False negatives (FN) are represented in the above figure.

Accuracy is the percentage of correctly corrected samples in the entire sample set is known as precision.

Recall is the portion of data samples, out of all samples for a class, that a machine learning model correctly classifies as being part of the class of interest (the "positive class").

$$Recall = \frac{TP}{TP + FN}$$

The F1 score [25] is a metric that provides a fair assessment of a model's performance by combining recall and precision. The harmonic mean of recall and precision is used to calculate it.

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

One measure of a machine learning model's effectiveness is precision, or how well the model makes positive predictions. Precision can be calculated by dividing the total number of positive predictions by the number of genuine positives.

$$Precision = \frac{TP}{TP + FP}$$

5.2 EXPERIMENTAL RESULTS

Using our dataset, we ran five machine learning algorithms. Table 5.1 displays the accuracy, f1 score, recall and precision outcomes for all the algorithms.

S. No.	Model Name	Accuracy	f1 Score	Recall	Precision
1	Decision Tree	98.43%	0.98	0.97	0.98
2	Naïve Bayes	51.52%	0.52	0.54	0.73
3	Random Forest Tree	99.02%	0.98	0.96	0.99
4	SVM	97.04%	0.87	0.83	0.95
5	Logistic Regression	95.11%	0.88	0.82	0.95
6	Gradient boosting	99.41%	0.98	0.97	0.99
7	Stacking (RandomForest, Gradient boosting)	99.75%	0.98	0.97	0.99
8	Stacking (Decision Tree, SVM) (Logistic Regression)	99.48%	0.95	0.93	0.96
9	Stacking (Decision Tree, SVM) (RandomForest)	99.26%	0.93	0.91	0.94

Table 5.1 Comparison of various models

In this study, we evaluate and compare the effectiveness of different machine learning models, including Decision Trees, Naïve Bayes, Random Forests, SVM, Logistic Regression, and ensemble techniques like stacking, for the task of phishing detection based on their respective accuracy, f1 Score, recall, and precision metrics.

- **Decision Trees:** Decision Trees are a popular supervised learning algorithm used for classification tasks. This model achieves an impressive accuracy of 98.43%, indicating its ability to correctly classify instances. The f1 Score of 0.98 reflects a good balance between precision (0.98) and recall (0.97), demonstrating high effectiveness in identifying phishing instances while minimising false positives and false negatives.
- **Naïve Bayes:** Naïve Bayes is a probabilistic classifier based on Bayes' theorem with strong independence assumptions between features. The lower accuracy of 51.52% suggests limitations in handling the complexity of phishing detection datasets. The f1 Score of 0.52, recall of 0.54, and precision of 0.73 indicate challenges in correctly identifying phishing instances, possibly due to the oversimplified feature assumptions.
- **Random Forest:** Random Forest is an ensemble learning method that constructs multiple decision trees and aggregates their predictions. This model achieves high accuracy (99.02%) and strong f1 Score (0.98), recall (0.96), and precision (0.99). 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 97.04% with an f1 Score of 0.87, recall of 0.83, and precision of 0.95. While SVM performs well in accuracy and precision, its recall rate suggests some difficulty in capturing all phishing instances effectively.
- **Logistic Regression:** Logistic Regression is a linear model for binary classification that estimates probabilities using a logistic function. With an accuracy of 95.11%, f1 Score of 0.88, recall of 0.82, and precision of 0.95, Logistic Regression offers a balanced approach for phishing detection. It demonstrates good accuracy and precision while maintaining a reasonable recall rate.
- **Stacking (RandomForest, Gradient Boosting) with Meta classifier of LogisticRegression:** Stacking is an ensemble learning technique that combines

base models such as RandomForest and Gradient Boosting [18], leveraging their strengths to achieve exceptional predictive performance. In this case, the stacking ensemble with LogisticRegression as the meta classifier achieves outstanding accuracy (99.75%), along with a strong f1 Score (0.98), recall (0.97), and precision (0.99), demonstrating superior performance in phishing detection.

- **Stacking (Decision Tree, SVM) with Meta classifier of LogisticRegression:** This stacking ensemble combines base models including Decision Tree and SVM, leveraging their diverse capabilities. With LogisticRegression as the meta classifier, the ensemble achieves a notable accuracy of 99.48% and impressive f1 Score of 0.95, recall of 0.93, and precision of 0.96. This stacking approach synergistically enhances overall performance, effectively detecting phishing attempts with high accuracy and reliability.
- **Stacking (Decision Tree, SVM) with a Meta classifier of RandomForest:** This stacking ensemble combines Decision Tree and SVM base models, complementing their strengths. Utilising RandomForest as the meta classifier, the ensemble achieves high accuracy (99.26%) with a strong f1 Score of 0.93, recall of 0.91, and precision of 0.94. This stacking strategy effectively capitalises on the diverse capabilities of individual models, demonstrating robust performance in phishing detection tasks.

Among the stacking configurations evaluated, the ensemble model "Stacking (RandomForest, Gradient Boosting) with a meta classifier of LogisticRegression" stands out with the highest accuracy of 99.75%. This indicates superior performance in identifying phishing attempts compared to other stacking configurations and individual models. Additionally, this ensemble achieves strong f1 Score, recall, and precision metrics, highlighting its effectiveness in leveraging the combined strengths of RandomForest and Gradient Boosting as base classifiers, guided by the LogisticRegression meta classifier. The exceptional accuracy underscores the potential of this ensemble approach for robust phishing detection in real-world applications.

A comparison map comparing the precision performance of the algorithms SVM, Naïve Bayes, Decision Tree, and Random Forest classifier as well as stacking and logistic regression is shown in Figure 5.2

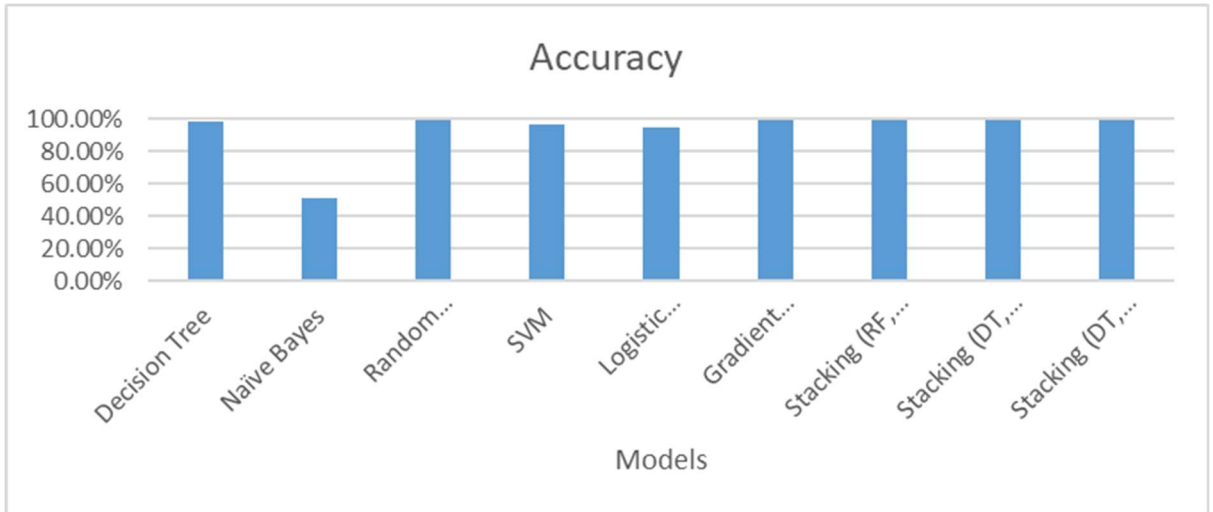


Fig. 5.2 Comparative plot of accuracy scores

Figure 5.3 presents a comparative plot depicting the precision performance of the algorithms: SVM, Decision Tree, Naïve Bayes, Random Forest classifier, Logistic Regression and Stacking.

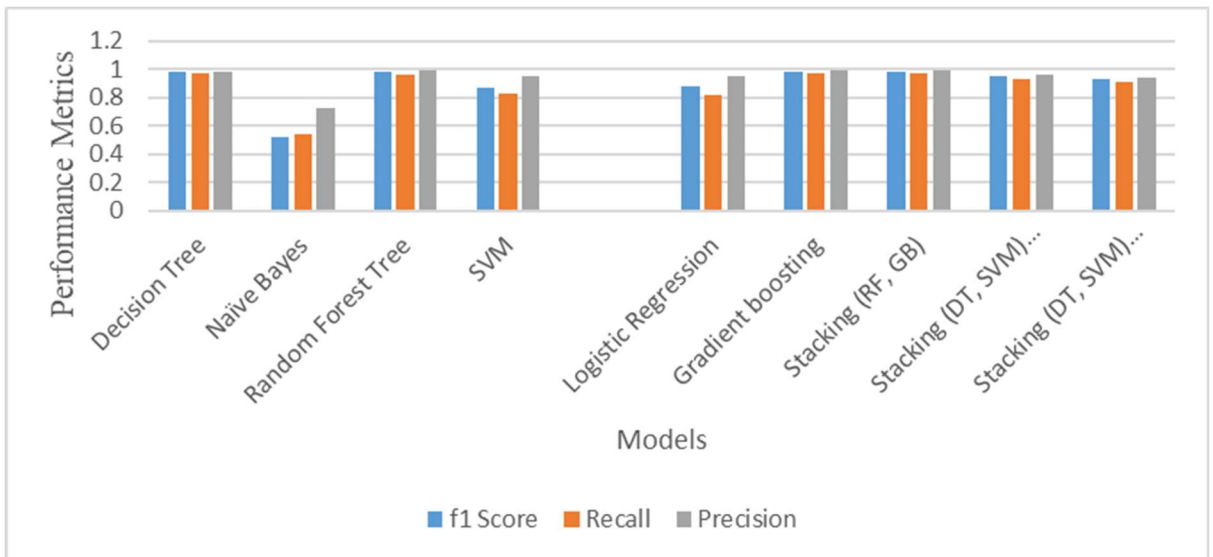


Fig 5.3: Comparison between different models and their f1 score, recall and precision

CHAPTER 6: CONCLUSIONS AND FUTURE SCOPE

6.1 CONCLUSIONS

In today's rapidly evolving technological landscape, phishing represents a growing and increasingly sophisticated threat. As societies worldwide embrace cashless transactions, e-commerce, and digital ticketing, phishing attacks pose a significant obstacle to progress, eroding trust in online platforms.

This research contributes by exploring the application of machine learning for phishing website detection. The objective was to develop an accurate, efficient, and cost-effective phishing detection system using machine learning tools and methodologies.

The proposed approach involved evaluating five machine learning classifiers and comparing their performance across six algorithms. Notably, the study achieved a high level of accuracy. The classifiers employed included Support Vector Machine, Naïve Bayes, Decision Tree, Random Forest Classifier, Stacking, and Logistic Regression.

Among these methods, Stacking demonstrated superior performance, achieving an impressive accuracy rate of 99.75%. This underscores the effectiveness of ensemble techniques in enhancing phishing detection capabilities and highlights the potential of machine learning in combating cyber threats in today's digital age.

6.2 FUTURE SCOPE

Combinations of models can be used to further improve the model and achieve higher accuracy scores. Machine learning approach, using ensemble methods, combines multiple basic models in order to create an improved 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.

To finish this system, the system might additionally incorporate additional phishing variations like smishing, vishing, etc. It is necessary to assess the methodology's potential for handling collection increases even further in the future. Since the collections should ideally expand gradually over time, a method for applying a classifier to the additional data gradually must be found. Additionally, the classifier may be subject to feedback that could eventually cause it to change.

Further we can implement a real time detection, to improve the accessibility of the project, by deploying the system on the website.

REFERENCES:

1. Phishing Statistic <https://www.techopedia.com/phishing-statistics#:~:text=Phishing%20attacks%20accounted%20for%2036,reached%205%20million%20in%202023.>
2. 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>
3. 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
4. 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 (2020): 747-766.
5. P. Yang, G. Zhao, and P. Zeng, "Phishing website detection based on multidimensional features driven by deep learning," IEEE Access, 7:15196–15209, 2019
6. T. Nathezhtha, D. Sangeetha, and V. Vaidehi, "WC-PAD: Web crawling based phishing attack detection," In 2019 International Carnahan Conference on Security Technology (ICCST), pages 1–6, 2019.
7. Y. Huang, Q. Yang, J. Qin, and W. Wen, "Phishing url detection via CNN and attention-based hierarchical RNN," In 2019 18th IEEE International Conference On 55 Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), pages 112–119, 2019.
8. M. M. Yadollahi, F. Shoeleh, E. Serkani, A. Madani, and H. Gharaee, "An adaptive machine learning based approach for phishing detection using hybrid features," In 2019 5th International Conference on Web Research (ICWR), pages 281–286, 2019.
9. 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.

10. S. Parekh, D. Parikh, S. Kotak and S. Sankhe, "A New Method for Detection of Phishing Websites: URL Detection," *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, Coimbatore, India, 2018, pp. 949-952, doi: 10.1109/ICICCT.2018.8473085.
11. V. Gupta, A. Mehta, A. Goel, U. Dixit and A. C. Pandey, "Spam Detection Using Ensemble Learning," *2018 Advances in Intelligent Systems and Computing*, vol 741. Springer, Singapore. https://doi.org/10.1007/978-981-13-0761-4_63
12. J. Li and S. Wang, "PhishBox: An approach for phishing validation and detection," In *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, pages 557–564, 2017.
13. Python Modules. <https://docs.python.org/3/library/ipaddress.html>
14. Data Flow Diagram (DFD). <https://www.lucidchart.com/pages/data-flow-diagram>
15. PhishTank <https://phishtank.org/>
16. University of New Brunswick "URL dataset (ISCX-URL)" <https://www.unb.ca/cic/datasets/url-2016.html>
17. Decision Trees. [https://scikit-learn.org/stable/modules/tree.html#:~:text=Decision%20Trees%20\(DTs\)%20are%20a,as%20a%20piecewise%20constant%20approximation](https://scikit-learn.org/stable/modules/tree.html#:~:text=Decision%20Trees%20(DTs)%20are%20a,as%20a%20piecewise%20constant%20approximation).
18. Naive Bayes Classifiers <https://www.geeksforgeeks.org/naive-bayes-classifiers/>
19. Random Forest. <https://www.ibm.com/topics/random-forest>
20. Support Vector Machines (SVM). <https://scikit-learn.org/stable/modules/svm.html>
21. Logistic Regression <https://www.geeksforgeeks.org/understanding-logistic-regression/>
22. Stacking <https://www.javatpoint.com/stacking-in-machine-learning>
23. Gradient boosting <https://www.geeksforgeeks.org/ml-gradient-boosting/>
24. Confusion Matrix <https://www.geeksforgeeks.org/confusion-matrix-machine-learning/>
25. F1 score <https://www.geeksforgeeks.org/f1-score-in-machine-learning/>

APPENDIX

CODE SNIPPETS

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

	url	type	use_of_ip	abnormal_url	google_index	count.	
0	br-icloud.com.br		0.0	0	0	1	2
1	signin.eby.de.zukruygcxtzmmqi.civpro.co.za		0.0	0	0	1	6
2	http://www.marketingbyinternet.com/mo/e56508df...		0.0	0	1	1	2
3	https://docs.google.com/spreadsheet/viewform?f...		0.0	0	1	1	2
4	retajconsultancy.com		0.0	0	0	1	1

5 rows x 25 columns

Figure 7.1 Loading of data

```
#Listing the features of the dataset
data.columns

Index(['url', 'type', 'use_of_ip', 'abnormal_url', 'google_index', 'count.',
      'count-www', 'count@', 'count_dir', 'count_embed_domian', 'short_url',
      'count-https', 'count-http', 'count%', 'count?', 'count-', 'count=',
      'url_length', 'hostname_length', 'sus_url', 'count-digits',
      'count-letters', 'fd_length', 'tld', 'tld_length'],
      dtype='object')
```

Figure 7.2 Listing of Features

```
# shuffling the rows in the dataset so that when splitting the train and test set are equally distributed
data = data.sample(frac=1).reset_index(drop=True)
data.head()
```

	url	type	use_of_ip	abnormal_url	google_index	count.	count-www	count@	
0	halkbank-parafbireysel.com		0.0	0	0	1	1	0	0
1	elimitless.org		0.0	0	0	1	1	0	0
2	en.wikipedia.org/wiki/Blessed_by_a_Broken_Heart		1.0	0	0	1	2	0	0
3	http://viajeskymbo.com/kst/images/Ser-ici/inde...		0.0	0	1	1	2	0	0
4	http://www.piano09.com/board/data/care/CodingN...		0.0	0	1	1	3	1	0

Figure 7.3 Shuffling of rows

```
[ ] import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# 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)
```

Figure 7.4 Accuracy of models

Open notebook

Examples >

Recent >

Google Drive >

GitHub >

Upload >

Search notebooks















Title	Last opened	First opened	
 Training and testing.ipynb	9:17 PM	November 28	 
 Welcome To Colaboratory	9:16 PM	Aug 17, 2021	
 Untitled1.ipynb	November 29	November 29	 
 Feature Extract.ipynb	November 29	November 1	 
 Major.ipynb	November 27	October 3	 

Figure 7.5 Google Colab

```
# Test the function with a URL
user_url = input("Enter a URL to check its legitimacy: ")
result = check_url_legitimacy(user_url)
print("The URL is:", result)
```

Enter a URL to check its legitimacy: www.facebook.com
The URL is: Legitimate

Figure 7.6 Test Result of url 1

```
+ Code + Text
✓ 20s ▶ user_url = input("Enter a URL to check its legitimacy: ")
        result = check_url_legitimacy(user_url)
        print("The URL is:", result)

↔ Enter a URL to check its legitimacy: www.google.com
   The URL is: Legitimate
```

Figure 7.7 Test Result of url 2

```
+ Code + Text
✓ 30s ▶ result = check_url_legitimacy(user_url)
        print("The URL is:", result)

↔ Enter a URL to check its legitimacy: https://www.juit.ac.in/
   The URL is: Legitimate
```

Figure 7.8 Test Result of url 3

```
+ Code + Text
✓ 14s ▶ user_url = input("Enter a URL to check its legitimacy: ")
        result = check_url_legitimacy(user_url)
        print("The URL is:", result)

↔ Enter a URL to check its legitimacy: infonews.co.nz/news-index.cfm?l=7&t=0
   The URL is: Phishing
```

Figure 7.9 Test Result of url 4

ORIGINALITY REPORT

17%

SIMILARITY INDEX

11%

INTERNET SOURCES

7%

PUBLICATIONS

7%

STUDENT PAPERS

PRIMARY SOURCES

1	bmsit.ac.in Internet Source	3%
2	doctorpenguin.com Internet Source	1%
3	www.mdpi.com Internet Source	1%
4	N. Varsha, D.Praveen Kumar, Bashetty Akhil, Kuduteeri Mouli. "Phishing sites detection using Machine Learning", 2023 International Conference on Integrated Intelligence and Communication Systems (ICIICS), 2023 Publication	1%
5	www.ijraset.com Internet Source	1%
6	"Innovations in Electrical and Electronic Engineering", Springer Science and Business Media LLC, 2024 Publication	1%
7	www2.mdpi.com Internet Source	1%

8	Submitted to UNITEC Institute of Technology Student Paper	<1 %
9	d197for5662m48.cloudfront.net Internet Source	<1 %
10	dokumen.pub Internet Source	<1 %
11	Submitted to Queen Mary and Westfield College Student Paper	<1 %
12	Moruf Akin Adebowale, Khin T. Lwin, M. A. Hossain. "Intelligent phishing detection scheme using deep learning algorithms", Journal of Enterprise Information Management, 2020 Publication	<1 %
13	Submitted to University of Illinois at Urbana-Champaign Student Paper	<1 %
14	Submitted to Liverpool John Moores University Student Paper	<1 %
15	www.techscience.com Internet Source	<1 %
16	Submitted to University of Salford Student Paper	<1 %

17 Submitted to University of Northumbria at Newcastle <1 %
Student Paper

18 diglib.tugraz.at <1 %
Internet Source

19 Aysar Weshahi, Feras Dwaik, Mohammad Khouli, Huthaifa I. Ashqar, Amani Shatnawi, Mahmoud ElKhodr. "Chapter 45 IoT-Enhanced Malicious URL Detection Using Machine Learning", Springer Science and Business Media LLC, 2024 <1 %
Publication

20 Submitted to De Montfort University <1 %
Student Paper

21 Submitted to Signal Mountain Middle High School <1 %
Student Paper

22 Liadira Kusuma Widya, Fateemah Rezaie, Jungsub Lee, Jongchun Lee, Juhee Yoo, Woojin Lee, Saro Lee. "Mapping Indoor Radon Concentrations in Chungcheongbuk-do, South Korea: A Geospatial Analysis using Machine Learning Models", Research Square Platform LLC, 2024 <1 %
Publication

23 Submitted to Middlesex University <1 %
Student Paper

Exclude quotes Off

Exclude matches Off

Exclude bibliography Off