A PROJECT REPORT
ON
”PREVENTION OF SCAMS USING DEEP LEARNING AND MACHINE LEARNING MODELS”

Submitted to Jawaharlal Nehru Technological University for the partial fulfillment of the requirements for the award of the degree

Bachelor of Technology
In

COMPUTER SCIENCE AND ENGINEERING

By

P. GURU HARITHA	**21L21A0538**
S. IMAMBE	**21L21A0548**
P. SIVANI	**21L21A0539**
S. AYESHA	**21L21A0547**
V. DEEPA	**21L21A0556**

Under the esteemed guidance of

Mr. S. MANIDHEER, M. Tech

Associate Prof, Dept, of CSE VITS, PRODDATUR

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

VAAGDEVI INSTITUTE OF TECHNOLOGY & SCIENCE

(Affiliated to JNTUA, Anantapuramu, College Code: L2)

(Approved by AICTE, NEW DELHI)

Peddasettiplli (V), Proddatur-516 360.
2021-2025.

VAAGDEVI INSTITUTE OF TECHNOLOGY & SCIENCE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project work entitled "PREVENTION OF SCAMS USING DEEP LEARNING AND MACHINE LEARNING MODELS" is a bonafied work of P. GURU HARITHA, S. IMAMBE, P. SIVANI, S. AYESHA, V. DEEPA Submitted to **Vaagdevi Institute of Technology & Science**, Proddatur in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology in COMPUTER SCIENCE & ENGINEERING.** The work reported herein does not from part of any other thesis on which a degree has been awarded earlier.

This is to further certify that they have worked for a period of one semester for preparing their work under our supervision & guidance.

Guide	**Head of The Department**
Mr. S. MANIDHEER, M. Tech	Mr. V. Narasimha Swamy,M.Tech,
Associate Prof, CSE Department.	HOD of CSE Department.
Vaagdevi Institute of Technology & Science	Vaagdevi Institute of Technology & Science
Proddatur- 516360	Proddatur- 516360

External Project viva held on:______________________________________

INTERNAL EXAMINER EXTERNAL EXAMINER

ACKNOWLEDGEMENT

Developing a project is not a single person’s effort but it is a combined achievement of group people. So many contributions from different category of fulfilled. During the development of text mining project, right from the initiation stage to implementation stage, so many people have helped me in studying the system, researching, developing and at last preparing and at last preparing documentation.

I would like to whole heartedly thank and express my sincere gratitude to the correspondent **Sri. G. Hussain Reddy, B.E., VITS** for his constant encouragement in the development of this project. I also sincerely thanks to Principal **Dr. B. Siddeswara Rao, Ph.D.,** for the constant encouragement and stimulating atmosphere provide to me.

I wish to thank **Mr. V. Narasimha Swamy, M.Tech,** and **HOD of CSE** for his valuable advice and support. Most of all, I extend my sincere thanks to my project guide,

Mr. S. Manidheer, M.Tech, Associate prof of CSE for their continuous encouragement, guidance and support throughout the development of this project.

I would like to thanks to the project committee members and faculty, teaching and non-teaching staff of **CSE department, Vaagdevi Institute of Technology and Science, Proddatur.**

Last but far from least, I also thank my parents, family members and my friends for their moral support and constant encouragement. I am very much thankful to one and all those helped me all those helped me for the successful completion of this project.

PROJECT ASSOCIATES

P. GURU HARITHA	**21L21A0538**
S. IMAMBE	**21L21A0548**
P. SIVANI	**21L21A0539**
S. AYESHA	**21L21A0547**
V. DEEPA	**21L21A0556**

INDEX

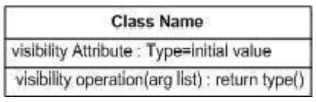
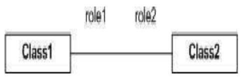
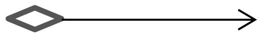
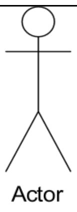
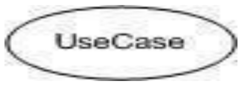
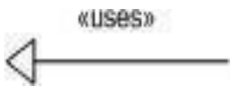
LIST OF FIGURES

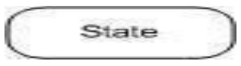
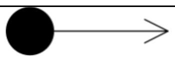
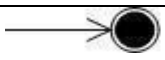
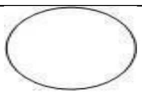
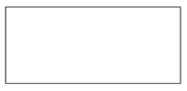
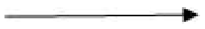
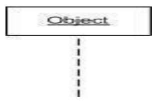
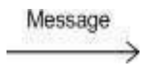
LIST OF SCREENSHOTS

LIST OF ABBREVIATION

UML	**:**	Unified Modelling Language
DNN	**:**	Deep Neural Networks
XGBoost	**:**	Extreme Gradient Boosting
NLP	**:**	Natural Language Processing

LIST OF SYMBOLS

S.NO	SYMBOL NAME	SYMBOL	DESCRIPTION
1.	Class	 Class Name visibility Attribute : Type=initial value visibility operation(arg list) : return type()	Classes represent a collection of similar entities grouped together.
2.	Association		Association represents a static relationship between classes.
3.	Aggregation		Aggregation is a form of association. It aggregates several classes into single class.
4.	Actor		Actors are the users of the system and other external entity that react with the system.
5.	Use case		A use case is a interaction between the system and the external environment.
6.	Relation (Uses)		It is used for additional process communication.
7.	Communication	———	It is the communication between various use cases.

8.	State		It represents the state of a process. Each state goes through various flows.
9.	Initial State		It represents the initial state of the object.
10.	Final State		It represents the final state of the object.
11	Data Process/State		A circle in DFD represents a state or process which has been triggered due to some event or action.
12.	External Entity		It represent any external entity such as keyboard, sensors etc. which are used in the system.
13.	Transition		It represent any communication that occurs between the processes.
14.	Object Lifeline		Object lifelines represents the vertical dimension that objects communicates.
15.	Message		It represents the messages exchanged.

ABSTRACT

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

Scams and fraudulent activities have become more sophisticated, leveraging advanced technology to deceive individuals and organizations. Traditional fraud detection methods often fail to keep up with evolving scam tactics. This paper explores the use of machine learning (ML) and deep learning (DL) models for scam prevention, focusing on their ability to detect patterns, anomalies, and suspicious behaviors in real time. ML techniques, including supervised and unsupervised learning, enable automated fraud detection through pattern recognition and anomaly detection. Deep learning approaches, such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transformers, enhance scam detection by analyzing transaction sequences, detecting fake content, and identifying phishing attempts using natural language processing (NLP). The integration of these AI-driven techniques provides a scalable, adaptive, and efficient solution to prevent scams in financial systems, e-commerce platforms, and cybersecurity. This study highlights the effectiveness of ML and DL models in combating fraudulent activities and the need for continuous advancements to stay ahead of emerging scam tactics.

1. INTRODUCTION

Phishing has become the most serious problem, harming individuals, corporations and even entire countries. The Availability of multiple services such as online banking, entertainment, education, software downloading, and social networking has accelerated the web's evolution in recent years. As a result, a massive amount of data is constantly downloaded and transferred to the Internet. Spoofed emails pretending to be from reputable business and agencies users into giving financial information such as usernames and passwords. Technical tricks involve the installation of malicious software on computers to steal credentials directly, with systems frequently used to intercept users online account usernames and passwords.

Scams and fraudulent activities have become increasingly sophisticated, leveraging advanced technology to deceive individuals and organizations. Traditional fraud detection methods often rely on rule-based systems, which struggle to keep up with evolving scam tactics. Machine learning (ML) and deep learning (DL) have emerged as powerful tools to detect and prevent scams by identifying patterns, anomalies, and suspicious behaviors in real time.

Pattern Recognition: ML algorithms identify common scam patterns, such as unusual transaction frequencies or fake reviews.

Anomaly Detection: DL models can flag deviations from normal behavior, such as sudden spikes in financial transactions.

Natural Language Processing (NLP): NLP techniques help detect phishing emails, scam messages, and fake news.

Real-Time Detection: ML-powered systems continuously learn and adapt, allowing them to detect scams as they occur.

Applications of Deep Learning in Scam Detection

- **Convolutional Neural Networks (CNNs)**: Used to detect fake images and deepfake scams.
- **Recurrent Neural Networks (RNNs) & Long Short-Term Memory (LSTM)**: Analyse sequences of financial transactions to detect fraudulent activities.
- **Transformer Models**: Used in NLP to identify fraudulent text-based scams.

1.1 TYPES OF PHISHING

Deceptive Phishing: This is the most frequent type of phishing assault, in which a Cybercriminal impersonates a well-known institution, domain, or organization to acquire sensitive personal information from the victim, such as login credentials, passwords, bank account information, credit card information, and so on. Because there is no personalization or customization for the people, this form of attack lacks sophistication.

Spear Phishing: Emails containing malicious URLs in this sort of phishing email contain a lot of personalization information about the potential victim. The recipient's name, company name, designation, friends, co-workers, and other social information may be included in the email.

Whale Phishing: To spear phish a "whale," here a top-level executive such as CEO, this sort of phishing targets corporate leaders such as CEOs and top-level management employees.

URL Phishing: To infect the target, the fraudster or cyber-criminal employs a URL link. People are sociable creatures who will eagerly click the link to accept friend invitations and may even be willing to disclose personal information such as email addresses.

This is because the phishers are redirecting users to a false web server. Secure browser connections are also used by attackers to carry out their unlawful actions. Due to a lack of appropriate tools for combating phishing attacks, firms are unable to train their staff in this area, resulting in an increase in phishing attacks. Companies are educating their staff with mock phishing assaults, updating all their systems with the latest security procedures, and encrypting important information as broad counter measures.

1.2 EXISTING SYSTEM

Anti-phishing strategies involve educating netizens and technical defense. In this paper, we mainly review the technical defense methodologies proposed in recent years. Identifying the phishing website is an efficient method in the whole process of deceiving user information. Along with the development of machine learning techniques, various machine learning based methodologies have emerged for recognizing phishing websites to increase the performance of predictions. The primary purpose of this paper is to survey effective methods to prevent phishing attacks in a real-time environment.

While conventional methods focus on pre-defined rules and heuristics.

ML and Deep learning (DL) models enhance detection capabilities by recognizing complex fraud patterns and adapting to evolving scam tactics.

While current fraud detection systems effectively prevent many scams, they have limitations in handling evolving threats. The integration of advanced deep learning techniques, real-time anomaly detection, and adaptive learning models is essential to enhance scam prevention and stay ahead of cybercriminals.

1.3 PROPOSED SYSTEM

The most frequent type of phishing assault, in which a cybercriminal impersonates a well-known institution, domain, or organization to acquire sensitive personal information from the victim, such as login credentials, passwords, bank account information, credit card information, and so on. Emails containing malicious URLs in this sort of phishing email contain a lot of fraudulent activities. To spear phish a "whale," here a top-level executive such as CEO, this sort of phishing targets corporate leaders such as CEOs and top-level management employees.

To infect the target, the fraudster or cyber-criminal employs a URL link. To address the limitations of existing scam prevention systems, the proposed system integrates advanced **machine learning (ML) and deep learning (DL) models** for more accurate, scalable, and adaptive fraud detection. This system leverages real-time anomaly detection, predictive analytics, and intelligent decision-making to minimize fraudulent activities across various domains, such as financial transactions, phishing detection, and e-commerce fraud prevention.

The proposed system enhances scam prevention by integrating **real-time monitoring, deep learning-based anomaly detection, and adaptive fraud analysis**. By leveraging AI-driven automation, the system ensures **faster, more accurate, and continuously evolving** scam detection, significantly reducing financial losses and cyber fraud risks.

1.4 ADVANTAGES

There is no personalization or customization for the people, this form of attack lacks sophistication. Social information may be included in the email. The recipient's name, company name, designation, friends, co-workers may be missing click the link to accept friend invitations and may even have other people information. To infect the target, the fraudster or cyber-criminal employs a URL link. This system leverages real-time anomaly detection, predictive analytics, and intelligent

Integrating **machine learning (ML) and deep learning (DL)** in scam prevention provides numerous benefits over traditional rule-based approaches.

These AI-driven systems enhance fraud detection by offering real-time analysis, adaptability, and accuracy in identifying fraudulent activities.

The integration of machine learning and deep learning in scam prevention provides higher accuracy, real-time detection, adaptability, and scalability.

1.5 MODULES

1.5.1 NORMAL DATASET: A normal dataset is a collection of data used as a reference for analysis. In the case of website rankings, researchers need a reliable dataset to determine which websites are legitimate (benign) and widely trusted. This is where AlexaRank comes into play. AlexaRank gives a list of websites ordered by popularity. It is often used by researchers to check or compare different websites. The dataset contains information like the website's ranking and its domain name, presented in a simple text file format.

1.5.2 PHISHING DATASET: Phishing dataset is a collection of data used to identity and study the fraudulent websites. These websites trick people into revealing sensitive information. To help researchers detect and prevent phishing attacks, a reliable dataset is needed.

2. LITERATURE SURVEY

Many scholars have done some sort of analysis on the statistics of phishing URLs. Our technique incorporates key concepts from past research. We review past work in the detection of phishing sites using URL features, which inspired our current approach. Happy describe phishing as "One of the most dangerous ways for hackers to obtain users' accounts such as usernames, account numbers and passwords, without their awareness." Users are ignorant of this type of trap and will ultimately, they fall into Phishing scam. This could be due to a lack of a combination of financial aid and personal experience, as well as a lack of market awareness or brand trust. In this article, Mehmet et al. suggested a method for phishing detection based on URLs. To compare the results, the researchers utilized eight different algorithms to evaluate the URLs of three separate datasets using various sorts of machine learning methods and hierarchical architectures. The first method evaluates various features of the URL; the second method investigates the website's authenticity by determining where it is hosted and who operates it; and the third method investigates the website's graphic presence. We employ Machine Learning techniques and algorithms to analyze these many properties of URLs and websites. Garera et al. classify phishing URLs using logistic regression over hand-selected variables. The inclusion of red flag keywords in the URL, as well as features based on Google's Web page and Google's Page Rank quality recommendations, are among the features. Without access to the same URLs and features as our approach, it's difficult to conduct a direct comparison.

In this research, Yong et al. created a novel approach for detecting phishing websites that focuses on detecting a URL which has been demonstrated to be an accurate and efficient way of detection. To offer you a better idea, our new capsule-based neural network is divided into several parallel components. One method involves removing shallow characteristics from URLs. The other two, on the other hand, construct accurate feature representations of URLs and use shallow features to evaluate URL legitimacy.

The final output of our system is calculated by adding the outputs of all divisions. Extensive testing on a dataset collected from the Internet indicate that our system can compete with other cutting-edge detection methods. One method involves removing shallow characteristics from URL While consuming a fair amount of time. For phishing detection, Vahid Shahrivar et al. used machine learning approaches. They used the logistic regression classification method, KNN, Adaboost algorithm, SVM, ANN and random forest.

They found random forest algorithm provided good accuracy. Dr. G. Ravi Kumar used a variety of machine learning methods to detect phishing assaults. For improved results, they use NLP tools. They were able to achieve high accuracy using a Support Vector Machine and data that had been pre-processed using NLP approaches. Amani Alswailem et al. tried different machine learning model for phishing detection but was able to achieve more accuracy in random forest. Hossein et al. created the “Fresh-Phish” open-source framework. This system can be used to build machine-learning data for phishing websites. They used a smaller feature set and built the query in Python. They create a big, labelled dataset and test several machine-learning classifiers on it. Using machine-learning classifiers, this analysis yields very high accuracy. These studies look at how long it takes to train a model. X. Zhang suggested a phishing detection model based on mining the semantic characteristics of word embedding, semantic feature, and multi-scale statistical features in Chinese web pages to detect phishing performance successfully. To obtain statistical aspects of web pages, eleven features were retrieved and divided into five classes. To obtain statistical aspects of web pages, eleven features were retrieved and divided into five classes. To learn and evaluate the model, AdaBoost, Bagging, Random Forest, and SMO are utilized. The legitimate URLs dataset came from DirectIndustry online guides, and the phishing data came from China's Anti- Phishing Alliance. With novel methodologies, M. Aydin approaches a framework for extracting characteristics that is versatile and straightforward. Phish Tank provides data, and Google provides authentic URLs. C# programming and R programming were utilized to obtain the text attributes. The dataset and third-party service providers yielded a total of 133 features.

3. METHODOLOGY

A phishing website is a social engineering technique that imitates legitimate webpages and uniform resource locators(URLs). The Uniform Resources Locator(URL) is the most common way for phishing assaults to occur. Phisher
has complete control over the URL’s sub-domains. The phisher can alter the URL because item contains file components and directories. This research used the linear-sequential model, often known as the waterfall model. Although the waterfall approach is considered conventional, it works best in instances where there are few requirements. The application was divided into smaller components that were built using frameworks and hand-written code.

3.1 RESEARCH FRAMEWORK

The steps of this research in which some selected publications were read to determine the research gap and, as a result, the research challenge was defined. Feature selection, classification and phishing website detection were all given significant consideration. It's worth noting that most phishing detection researchers rely on datasets they've created. However, because the datasets utilized were not available online for those who use and check their results, it is difficult to assess and compare the performance of a model with other models. As a result, such results cannot be generalized. For the preparation of this dissertation, I used Python as the primary language. Python is a language that is heavily focused on machine learning. It includes several machine learning libraries that may be utilized straight from an import. Python is commonly used by developers all around the world to deal with machine learning because of its extensive library of machine learning libraries. Python has a strong community, and as a result, new features are added with release.

Data Collection: The phishing URLs were gathered using the open source tool Phish Tank. This site provides a set of phishing URLs in a variety of forms, including csv, Json, and others, which are updated hourly. This dataset is used to train machine learning models with 5000 random phishing URLs.

Data Cleaning: Fill in missing numbers, smooth out creaking data, detect and delete outliers, and repair anomalies to clean up the data.

Data Pre-processing: Data preprocessing is a cleaning operation that converts unstructured raw data into a neat, well-structured dataset that may be used for further research. Data preprocessing is a cleaning operation that transforms unstructured raw data into well-structured and neat dataset which can be used for further research.

The data is split into 8000 training samples and 2000 testing samples, before the ML model is trained. It is evident from the dataset that this is a supervised machine learning problem. Classification and regression are the two main types of supervised machine learning issues. Because the input URL is classed as legitimate or phishing, this data set has a classification problem. The following supervised machine learning models were examined for this project's dataset training: Decision Tree , Multilayer Perceptron, Random Forest, Autoencoder Neural Network , XGBoost and Support Vector Machines.

3.2 ADDRESS BASED CHECKING

Below are the categories been extracted from address based

1. **Domain of the URL :**Where domain which is present in the URL been extracted
2. **IP Address in the URL :**The presence of an IP address in the URL is checked. Instead of a domain name, URLs may contain an IP address. If an IP address is used instead of a domain name in a URL, we can be certain that the URL is being used to collect sensitive information.
3. **"@" Symbol in URL :**The presence of the'@' symbol in the URL is checked. When the "@" symbol is used in a URL, the browser ignores anything before the "@" symbol, and the genuine address is commonly found after the "@" symbol.
4. **Length of URL :**Calculates the URL's length. Phishers can disguise the suspicious element of a URL in the address bar by using a lengthy URL. If the length of the URL is larger than or equal to 54 characters, the URL is classed as phishing in this project.
5. **Depth of URL :**Calculates the URL's depth. Based on the'/', this feature determines the number of subpages in the given address.
6. **Redirection "//" in URL :**The existence of"//" in the URL is checked. The presence of the character"//" in the URL route indicates that the user will be redirected to another website. The position of the"//" in the URL is calculated. We discovered that if the URL begins with "HTTP," the "//" should be placed in the sixth position. If the URL uses "HTTPS," however, the "//" should occur in the seventh place.
7. **Http/Https in Domain name :**The existence of "http/https" in the domain part of the URL is checked. To deceive users, phishers may append the "HTTPS" token to the domain section of a URL.
8. **Using URL Shortening Services :**URL shortening is a means of reducing the length of a URL while still directing to the desired webpage on the "World Wide Web." This is performed by using a "HTTP Redirect" on a short domain name that points to a webpage with a long URL.

9. **Prefix or Suffix "-" in Domain :**Checking for the presence of a '-' in the URL's domain part. In genuine URLs, the dash symbol is rarely used. Phishers frequently append prefixes or suffixes to domain names, separated by (-), to give the impression that they are dealing with a legitimate website.

3.3 DOMAIN BASED CHECKING

This category contains a lot of features that can be extracted. This category contains a lot of features that can be extracted. The following were considered for this project out of all of them.

1. **DNS Record :**In the case of phishing websites, the WHOIS database either does not recognize the stated identity or there are no records for the host name .

2. **Web Traffic :**This function determines the number of visitors and the number of pages they visit to determine the popularity of the website. In the worst-case circumstances, legitimate websites placed among the top100,000, according to our data. Furthermore, it is categorized as "Phishing" if the domain has no traffic or is not recognized by the Alexa database.

3. **Age of Domain :**This information can be retrieved from the WHOIS database. Most phishing websites are only active for a short time. For this project, the minimum age of a legal domain is deemed to be 12 months. Age is simply the difference between the time of creation and the time of expiry.

4. **End Period of Domain :**This information can be gleaned from the WHOIS database. The remaining domain time is calculated for this feature by determining the difference between the expiry time and the current time. For this project, the valid domain's end time is regarded to be 6 months or fewer.

3.4 HTML AND JAVA SCRIPT BASED CHECKING

Many elements that fall within this group can be extracted. The following were considered for this project out of all of them.

1. **IFrame Redirection :** IFrame is an HTML tag that allows you to insert another webpage into the one you're now viewing. The “iframe” tag can be used by phishers to make the frame invisible, i.e., without frame borders. Phishers employ the “frame border” attribute in this case, which causes the browser to create a visual boundary.

2. **Status Bar Customization :** Phishers may utilize JavaScript to trick visitors into seeing a false URL in the status bar. To get this feature, we'll need to delve into the webpage source code, specifically the "on Mouseover" event, and see if it alters the status bar.

3. **Disabling Right Click :** Phishers disable the right-click function with JavaScript, preventing users from viewing and saving the webpage source code. This functionality is handled in the same way as "Hiding the Link with on Mouseover.

" Nonetheless, we’ll look for the event’ event. button==2" in the webpage z that it achieved 86.02% accuracy and less than a 1.48% false-positive rate, which indicates a false warning for phishing attacks. The other benefit of this approach is fast access time, which guarantees a real-time environment and products.

Heuristic Strategies: Tan et al. introduced a phishing detection approach named Phish WHO, which consists of three phases. First, it obtains identity keywords by a weighted URL token system and ensembles the N-gram model from the page’s HTML. Secondly, it puts the keywords into mainstream search engines to find the legitimate website and the legal domain.

Next, it compares the legal domain and the target website’s domain to determine if the target website is a phishing website or not. Chiew et al. used a logo image from the website to distinguish if the website was legal. In this paper, the authors extracted a logo from web page images by some machine learning algorithms and then queried the domain via the Google search engine with a logo as a keyword. Therefore, some researchers also called this category search engine-based approach.

Machine Learning-Based Methods: Machine learning-based countermeasures are proposed to address dynamic phishing attacks with higher accuracy performance and lower false positive rates than other methods. Consequently, the machine learning approach consists of six components: data collection, feature extraction, model training, model testing, and predicting. The flowchart of each part. Existing machine learning-based phishing website detection solutions are based on this flowchart to optimize one or more parts to obtain better performance.

Tiny URL Detection: Since tiny URLs do not present the real domain, resource direction, or search parameters, rule-based feature selection techniques might be useless for tiny URLs. Due to tiny URLs generated by different services, it is hard to convert them to original URLs. Furthermore, tiny URLs are short strings that are unfriendly for natural language processing to extract character-level features. If tiny URLs are not specially processed during data cleansing and preprocessing, they are likely to cause false or missed alarms. Internet products are also essential in terms of user experience, and users are also sensitive to false alarms of Internet security products. 6.4. Response Time for Real-Time Systems Rule-based models depend on rule parsing and third-party services from a URL string. Therefore, they demand a relatively long response time in a real-time prediction system that accepts a single URL string as an input in each request from a client. Phishing attacks spread to various communication media and target devices, such as personal computers and other smart devices. It is a big challenge for developers to cover all devices with one solution. Language independence and running environment independence should be taken into consideration to reduce system development complexity and late maintenance costs.

We collected the datasets from the open-source platform called Phishing tank. The dataset that was collected was in csv format. There are 18 columns in the dataset, and we transformed the dataset by applying data pre-processing technique. To see the features in the data we used few of the data frame methods for familiarizing. For visualization, and to see how the data is distributed and how features are related to one another, a few plots and graphs are given. The Domain column has no bearing on the training of a machine learning model. We now have 16 features and a target column. feature extraction file, with no shuffling. We need to shuffle the data to balance out the distribution while breaking it into training and testing sets. This also eliminates the possibility of over fitting during model training.

3.5 DATASET

We collected the datasets from the open-source platform called Phishing tank. The dataset that was collected was in csv format. There are 18 columns in the dataset, and we transformed the dataset by applying data pre-processing technique. To see the features in the data we used few of the data frame methods for familiarizing. For visualization, and to see how the data is distributed and how features are related to one another, a few plots and graphs are given. The Domain column has no bearing on the training of a machine learning model. We now have 16 features and a target column. feature extraction file, with no shuffling. We need to shuffle the data to balance out the distribution while breaking it into training and testing sets. This also eliminates the possibility of over fitting during model training.

3.6 MACHINE LEARNING MODELS

Decision Tree Classifier : For classification and regression applications, decision trees are commonly used models. They basically learn a hierarchy of if/else questions that leads to a choice. Learning a decision tree is memorizing the sequence of if/else questions that leads to the correct answer in the shortest amount of time. The method runs through all potential tests to discover the one that is most informative about the target variable to build a tree.

Random Forest Classifier : Random forests are one of the most extensively used machine learning approaches for regression and classification. A random forest is just a collection of decision trees, each somewhat different from the others. The notion behind random forests is that while each tree may do a decent job of predicting, it will almost certainly overfit on some data. They are incredibly powerful, frequently operate effectively without a lot of parameters adjusting, and don't require data.

3.7 LIBRARIES USED

Pandas: It's a Python-based machine learning library. Pandas is a free and open-source programming language. Pandas is a programming language that is commonly used for dataset loading and data analytics. Pandas is used for machine learning in a variety of domains, including economics, finance, and others. It is extremely user-friendly and can display datasets in a tabular style for easier comprehension.

Numpy: Numpy is a Python-based machine learning package. In Python, Numpy is used to deal with arrays. NumPy is used for all calculations using 1-d or 2-d arrays. Numpy also has routines for working with linear algebra and the Fourier transform.

Matplotlib: Matplotlib is a library for data visualization. It's a Python open-source module for plotting graphs from model results. These diagrams can aid in comprehending the circumstance of the outcomes.

3.8 EVALUATION

In this section, we use different models of machine learning for evaluating the accuracy. It has been explained about the different models in below sections. Where in this project the models are examined, with accuracy as the primary metric. In final stage we have compared the model accuracy. In all circumstances the testing and training datasets are splinted into 20:80 ratio. **Feature Distribution:** Here in below figure shows how the data is distributed and how features are related to one another, a few plots and graphs are given.

Decision Tree Classifier: The method runs through all potential tests to discover the one that is most informative about the target variable to build a tree Where we are predicting the accuracy of the model on the samples collected on both trained and test samples.

On this we found accuracy of test and training datasets are 82.6% and 81%. Below is the execution of Decision tree classifier algorithm. To generate model various parameters are set and the model is fitted in the tree. The samples are divided into X and Y train, X and Y test to check the accuracy of the model.

Random Forest Classifier: We can limit the amount of over fitting by averaging the outcomes of numerous trees that all operate well and over fit in diverse ways. To construct a random forest model, you must first determine the number of trees to construct.

MLP: MLPs can be thought of as generalized linear models that go through numerous phases of processing before deciding. Below is the execution of the MLP algorithm. To generate a model various parameters are set and the model is fitted in the tree. The samples are divided into X and Y train, X and Y test to check the accuracy of the model.

XGBoost: Below is the execution of XGBoost algorithm. To generate a model various parameters are set and the model is fitted in the tree. The samples are divided into X and Y trains, X and Y tests to check the accuracy of the model. Where we are predicting the accuracy of the model on the samples collected on both trained and test samples. On this we found accuracy of test and training datasets are 86.4% and 86.6%.

Auto encoder : The auto-encoder must learn to encode the input to the hidden neurons with fewer neurons. In an auto encoder, the predictors (x) and output(y) are identical. To generate model various parameters are set and the model is fitted in the tree. The samples are divided into X and Y train, X and Y test to check the accuracy of the model.

SVM: An SVM training algorithm creates a model that assigns new examples to one of two categories, making it a non-probabilistic binary linear classifier, given a series of training examples that are individually designated as belonging to one of two categories. To generate model various parameters are set and the model is fitted in the tree.

The samples are divided into X and Y train, X and Y test to check the accuracy of the model. Where we are predicting the accuracy of the model on the samples collected on both trained and test samples. On this we found accuracy of test and training datasets are 81.8% and 79.8%.

Performance Evaluation:

The evaluation of performance was carried out during the testing process. The original dataset would be divided into training data and test data, usually 80% and 20%, respectively. When evaluating the classifier’s behavior on the testing dataset, there were four statistical numbers: the number of correctly identified positive data points (TP), the number of correctly identified negative data points (TN), the number of negative data points labeled by the classifier as positive (FP), and the number of positive data points labeled by the model as negative (FN). There are several broadly used metrics to evaluate performance. The classification accuracy is the ratio of correct predictions to total predictions: accuracy = TP + TN /TP+ TN + FN + FP.

In binary classification cases, it is known that random selection has 50% accuracy. In unbalanced datasets, sometimes high accuracy does not mean that the model is excellent.

The number of false-positive cases (FP) reflects the false warning rate. In real-time phishing detection systems, this directly affects the user experience and trustworthiness:
Precision = TP/TP + FP. The recall is the portion of positive data points labeled as such by the model among all truly positive data points. The number of false-negative cases (FN) represents the number of phishing URLs that has not been detected.

Leak alarms mean that users are likely to receive an attack that could result in the theft of sensitive information. Misleading users can do more harm to users than not detecting them:
Recall = TP /TP + FN.

The F-measure or F-score is the combination of precision and recall. Generally, it is formulated as shown below: Fβ = (β2 + 1)× Precision × Recall/ β2 × Precision + Recall β ∈ (0, ∞).
Here, β quantifies the relative importance of the precision and recall such that β = 1 stands for the precision and recall being equally important, which is also called F1. The F-score does the best job of any single statistic, but all four work together to describe the performance of a classifier: F1 = 2 × Precision × recall /Precision + recall = TP /TP + 1/2 (FP + FN).In addition, many researchers use the N-fold cross-validation technique to measure performance for phishing detection [4,30,31]. The N-fold cross-validation technique is widely used on small datasets for evaluating machine learning models' performance. It is a resampling procedure that divides the original data samples into N pieces after shuffling the dataset randomly. One of the pieces is used in the testing process, and others are applied to the training process.

The samples are divided into X and Y train, X and Y test to check the accuracy of the model. Where we are predicting the accuracy of the model on the samples collected on both trained and test samples. On this we found accuracy of test and training datasets are 81.8% and 79.8%.

3.9 ARCHITECTURE

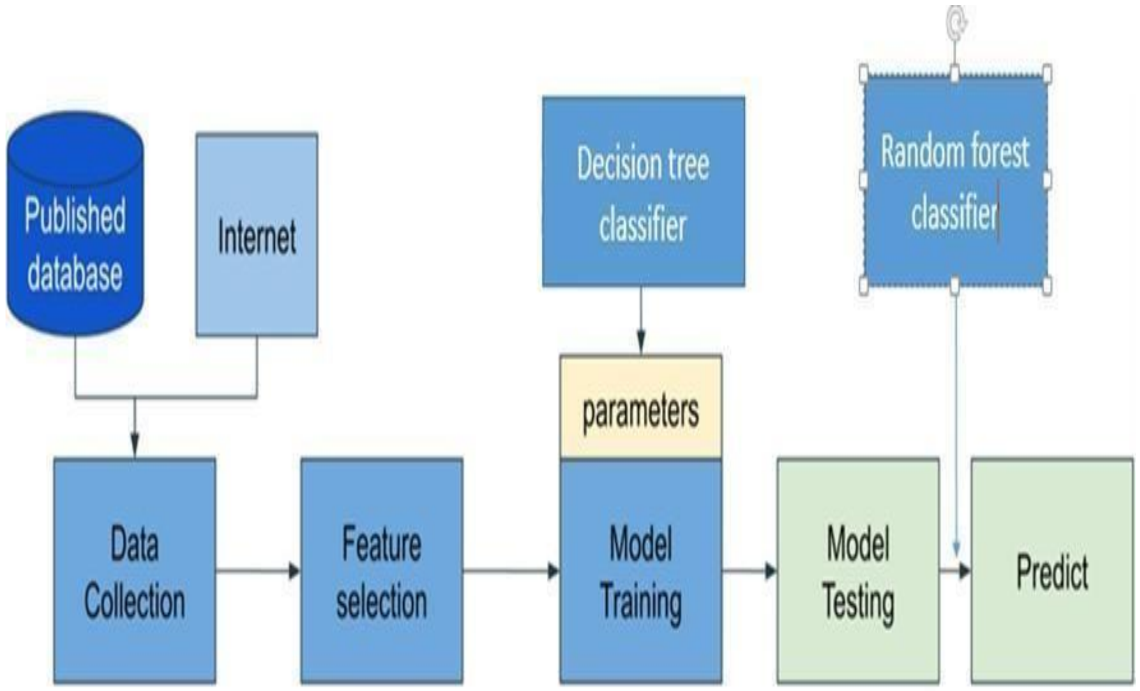


Fig: 3.9 Architecture

3.10 FLOW DIAGRAM

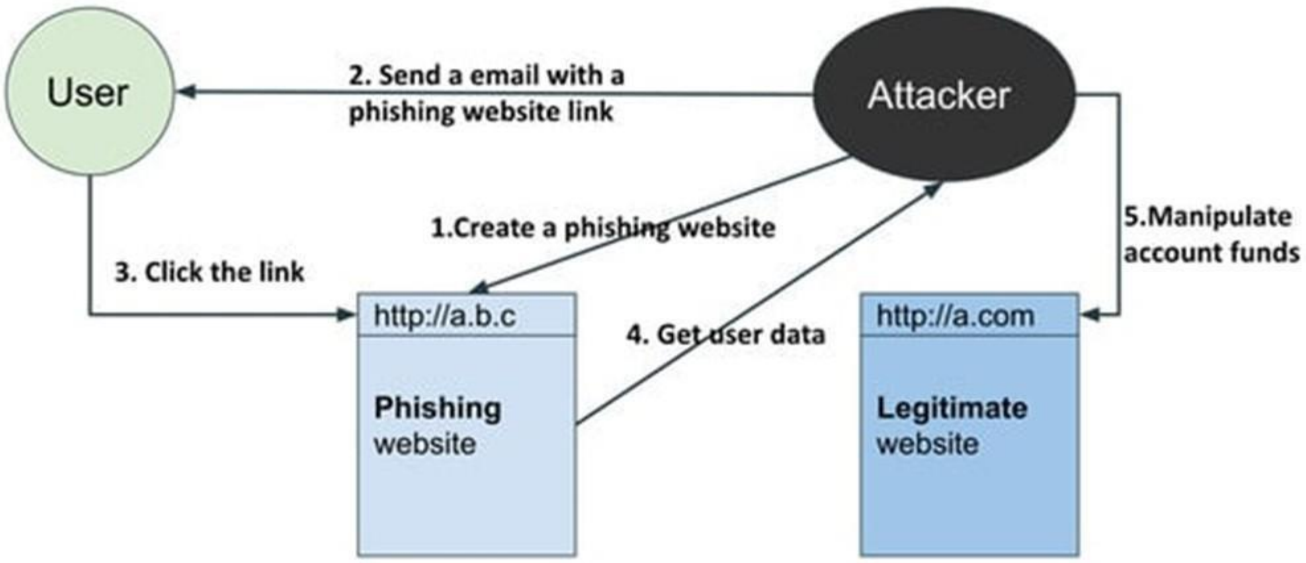


Fig :3.10 Flow diagram

3.11 DECISION TREE ALGORITHM

There’s not much mathematics involved here. Since it is very easy to use and interpret it is one of the most widely used and practical methods used in Machine Learning. It is a tool that has applications spanning several different areas. Decision trees can be used for classification as well as regression problems. The name itself suggests that it uses a flowchart like a tree structure to show the predictions that result from a series of feature-based splits.

It starts with a root node and ends with a decision made by leaves. Root Nodes – It is the node present at the beginning of a decision tree from this node the population starts dividing according to various features. Decision Nodes – the nodes we get after splitting the root nodes are called Decision Node. Leaf Nodes – the nodes where further splitting is not possible are called leaf nodes or terminal nodes. Sub-tree – just like a small portion of a graph is called sub-graph similarly a subsection of this decision tree is called sub-tree. Pruning – is nothing but cutting down some nodes to stop overfitting.

Entropy: Entropy is nothing but the uncertainty in our dataset or measure of disorder. Let me try to explain this with the help of an example. Suppose you have a group of friends who decides which movie they can watch together on Sunday. There are 2 choices for movies, one is “Lucy” and the second is “Titanic” and now everyone has to tell their choice.

After everyone gives their answer, we see that “Lucy” gets 4 votes and “Titanic” gets 5 votes. Which movie do we watch now? Isn’t it hard to choose 1 movie now because the votes for both the movies are somewhat equal. This is exactly what we call disorderness, there is an equal number of votes for both the movies, and we can’t really decide which movie we should watch. It would have been much easier if the votes for “Lucy” were 8 and for “Titanic” it was 2. Here we could easily say that the majority of votes are for “Lucy” hence everyone will be watching this movie.

Information Gain: Information gain measures the reduction of uncertainty given some feature and it is also a deciding factor for which attribute should be selected as a decision node or root node. It is just entropy of the full dataset – entropy of the dataset given some feature. To understand this better let’s consider an example: Suppose our entire population has a total of 30 instances. The dataset is to predict whether the person will go to the gym or not. Let’s say 16 people go to the gym and 14 people don’t. Now we have two features to predict whether he/she will go to the gym or not. Feature 1 is “Energy” which takes two values “high” and “low” Feature 2 is “Motivation” which takes 3 values “No motivation”, “Neutral” and “Highly motivated”.

Let’s see how our decision tree will be made using these 2 features. We’ll use information gain to decide which feature should be the root node and which feature should be placed after the split.

Let’s calculate the entropy: To see the weighted average of entropy of each node we will do as follows: Now we have the value of E(Parent) and E(Parent| Energy), information gain will be: Our parent entropy was near 0.99 and after looking at this value of information gain, we can say that the entropy of the dataset will decrease by 0.37 if we make “Energy” as our root node. Similarly, we will do this with the other feature “Motivation” and calculate its information gain. In this example “Energy” will be our root node and we’ll do the same for sub-nodes. Here we can see that when the energy is “high” the entropy is low and hence we can say a person will definitely go to the gym if he has high energy, but what if the energy is low? We will again split the node based on the new feature which is “Motivation”.

You must be asking this question to yourself that when do we stop growing our tree? Usually, real-world datasets have a large number of features, which will result in a large number of splits, which in turn gives a huge tree. Such trees take time to build and can lead to overfitting. That means the tree will give very good accuracy on the training dataset but will give bad accuracy in test data. There are many ways to tackle this problem through hyperparameter tuning. We can set the maximum depth of our decision tree using the max_depth parameter. The more the value of max_depth, the more complex your tree will be. The training error will off-course decrease if we increase the max_depth value but when our test data comes into the picture.

We will get a very bad accuracy. Hence you need a value that will not overfit as well as underfit our data and for this, you can use GridSearchCV.

Another way is to set the minimum number of samples for each spilt. It is denoted by min_samples_split. Here we specify the minimum number of samples required to do a split. For example, we can use a minimum of 10 samples to reach a decision. That means if a node has less than 10 samples then using this parameter, we can stop the further splitting of this node and make it a leaf node.

Pruning: It is another method that can help us avoid overfitting. It helps in improving the performance of the tree by cutting the nodes or sub-nodes which are not significant. It removes the branches which have very low importance.

There are mainly 2 ways for pruning:

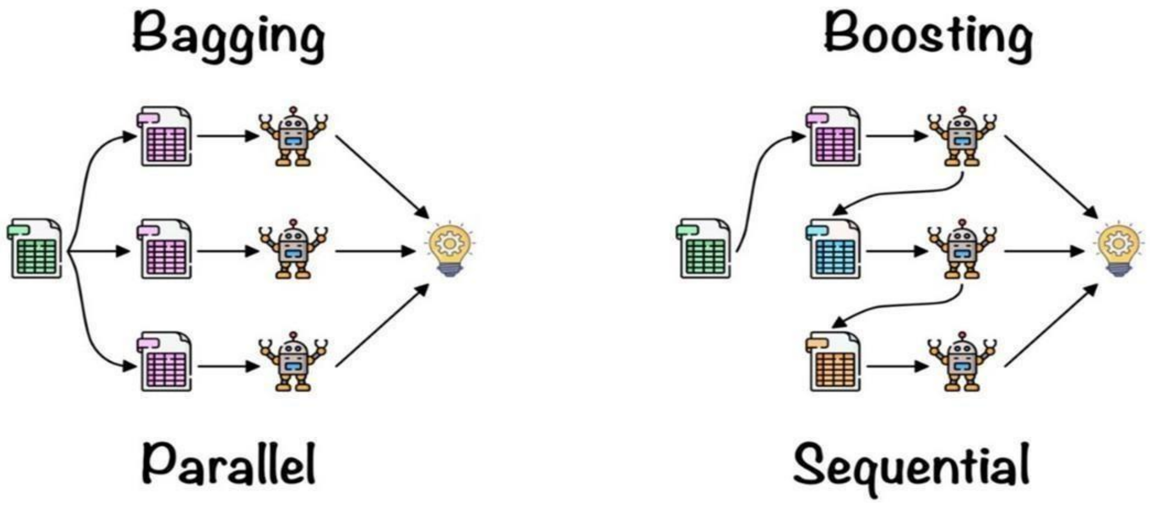
Pre-pruning – we can stop growing the tree earlier, which means we can prune/remove/cut a node if it has low importance while growing the tree.

Post-pruning – once our tree is built to its depth, we can start pruning the nodes based on their significance.

3.12 RANDOM FOREST ALGORITHM

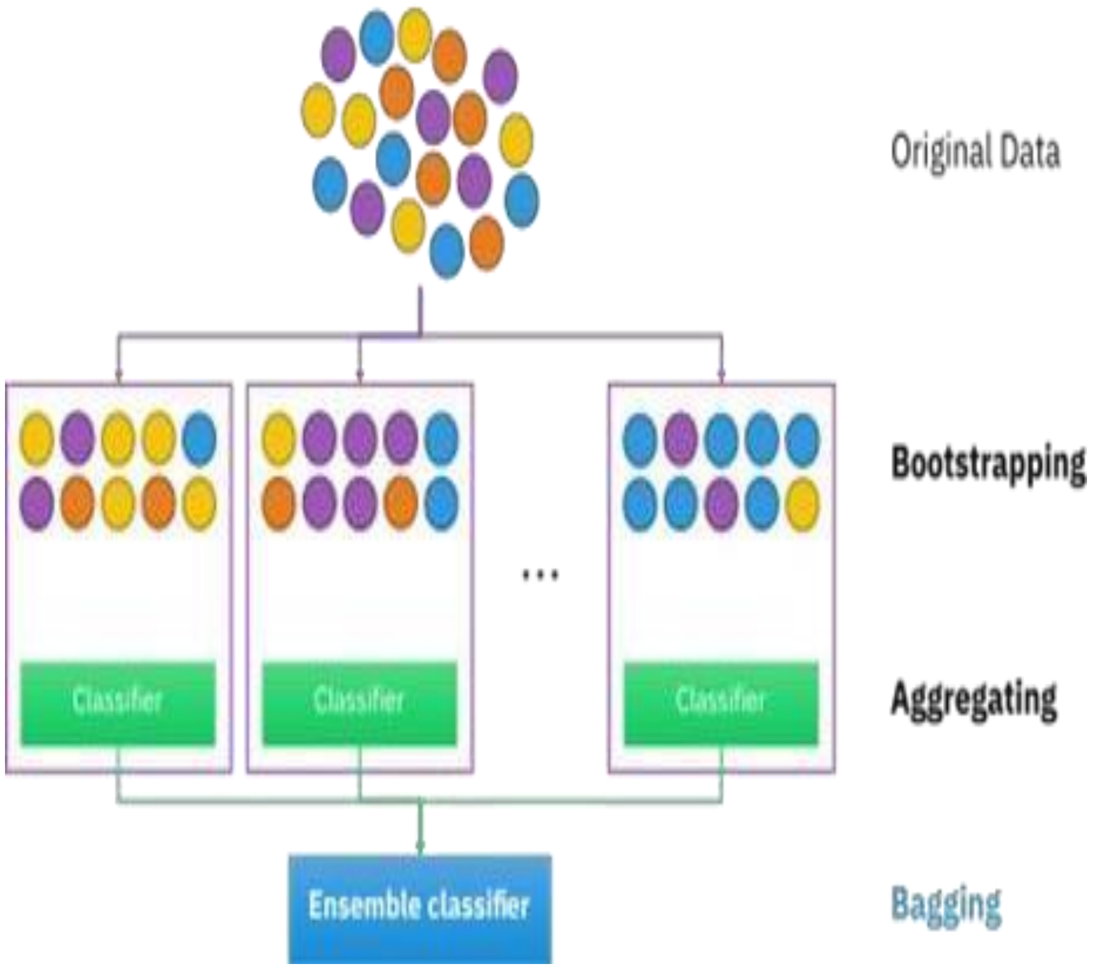
Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression. One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables as in the case of regression and categorical variables as in the case of classification. It performs better results for classification problems. Let's dive into a real-life analogy to understand this concept further. A student named X wants to choose a course after his 10+2, and he is confused about the choice of course based on his skill set. So, he decides to consult various people like his cousins, teachers, parents, degree students, and working people. He asks them varied questions like why he should choose, job opportunities with that course, course fee, etc. Finally, after consulting various people about the course he decides to take the course suggested by most of the people.Ensemble uses two types of methods:

1. **Bagging–** It creates a different training subset from sample training data with replacement & the final output is based on majority voting. For example, Random Forest.
2. **Boosting–** It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy.



3.3 Bagging and Boosting

Bagging: Bagging, also known as Bootstrap Aggregation is the ensemble technique used by random forest. Bagging chooses a random sample from the data set. Hence each model is generated from the samples (Bootstrap Samples) provided by the Original Data with replacement known as row sampling. This step of row sampling with replacement is called bootstrap. Now each model is trained independently which generates results. The final output is based on majority voting after combining the results of all models. This step which involves combining all the results and generating output based on majority voting is known as aggregation.



3.4 Bagging

Steps involved in random forest algorithm:

Step 1: In Random forest n number of random records are taken from the data set having k number of records.

Step 2: Individual decision trees are constructed for each sample.

Step 3: Each decision tree will generate an output.

Step 4: Final output is considered based on Majority Voting or Averaging for Classification and regression respectively.

For example: consider the fruit basket as the data as shown in the figure below. Now a number of samples are taken from the fruit basket and an individual decision tree is constructed for each sample. Each decision tree will generate an output as shown in the figure. The final output is considered based on majority voting. In the below figure you can see that the majority decision tree gives output as an apple when compared to a banana, so the final output is taken as an apple.

Important Features of Random Forest

Diversity- Not all attributes/variables/features are considered while making an individual tree, each tree is different. Immune to the curse of dimensionality- Since each tree does not consider all the features, the feature space is reduced. Parallelization-Each tree is created independently out of different data and attributes. This means that we can make full use of the CPU to build random forests. Train-Test split- In a random forest we don't have to segregate the data for train and test as there will always be 30% of the data which is not seen by the decision tree. Stability- Stability arises because the result is based on majority voting/ averaging.

4. FEASIBILITY STUDY

The feasibility of the project is analysed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

4.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization.

The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

4.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

4.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

5. SYSTEM REQUIREMENTS

5.1 SOFTWARE REQUIREMENTS

- **Development Tools:** Python, Jupyter Notebook, VS code.
- **AI Models** **:** DNN, XGBoost.
- **Libraries** **:** NumPy, Pandas, TensorFlow, NLTK, Scikit-learn.
- **Data Sources** **:** Real-time Transaction Scam Dataset.

5.2 HARDWARE REQUIREMENTS

- **System** **:** Intel i3 Processor .
- **Hard Disk** : 250 GB.
- **Monitor** : 14’ Colour Monitor.

6. SYSTEM DESIGN

6.1 UML DIAGRAM

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML. The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts. Be independent of particular programming languages and development process.
3. Provide a formal basis for understanding the modeling language.
4. Encourage the growth of OO tools market.
5. Support higher level development concepts such as collaborations, frameworks, patterns and components.
6. Integrate best practices.

Collaboration diagram:

A collaboration diagram groups together the interactions between different objects. The interactions are listed as numbered interactions that help to trace the sequence of the interactions.

The collaboration diagram helps to identify all the possible interactions that each object has with other object.

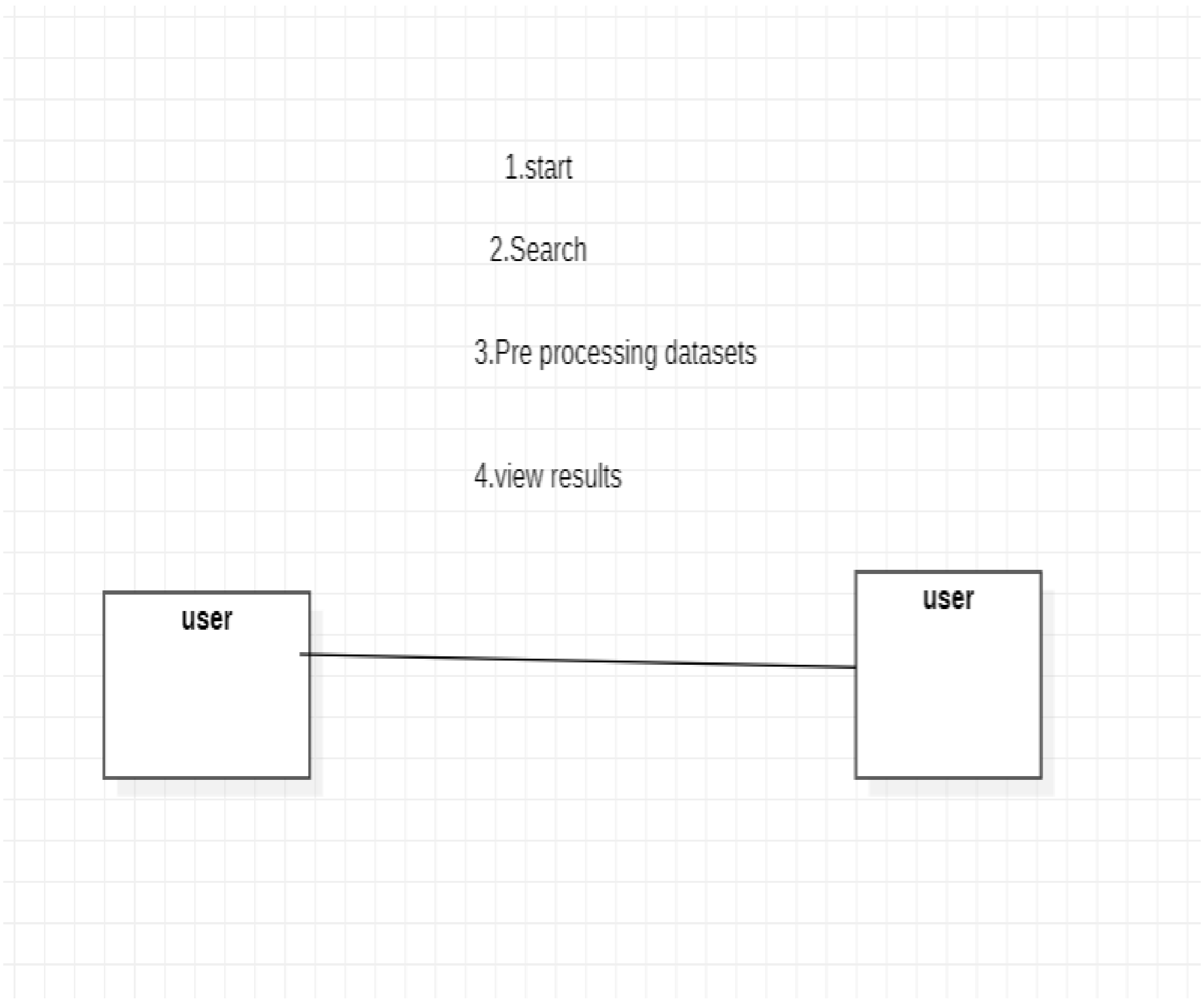


Fig:6.1 Collaboration Diagram

State diagram:

A state diagram, as the name suggests, represents the different states that objects in the system undergo during their life cycle. Objects in the system change states in response to events. In addition to this, a state diagram also captures the transition of the object's state from an initial state to a final state in response to events affecting the system.

UML Diagram:

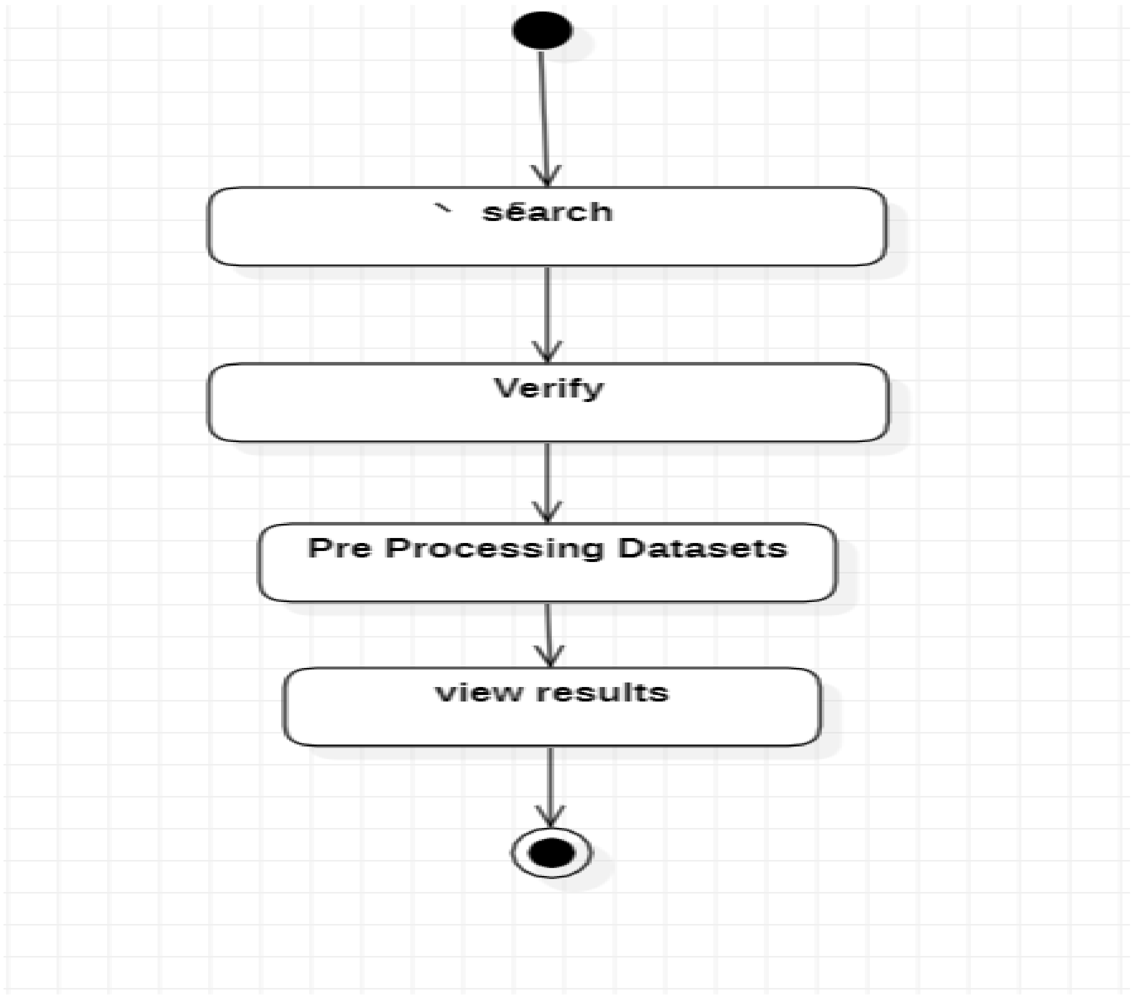


Fig: 6.2 UML Diagram

The provided diagram appears to represent a workflow for scam prevention using machine learning (ML) and deep learning (DL) models. Below is an explanation of each step in the context of fraud detection:

Explanation of the Diagram:

The process depicted follows a structured scam prevention pipeline, starting from data collection to the final detection of fraudulent activities.

1. **Start (Black Circle)**
   - This represents the initiation of the fraud detection process.

2. **Search**
    - The process depicted follows a structured scam prevention pipeline, starting from data collection to the final detection of fraudulent activities.
    - The system gathers relevant data from various sources, such as transaction logs, user interactions, financial records, or emails.
    - Web scraping and API calls might be used to fetch scam-related datasets.
3. **Verify**
    - The process depicted follows a structured scam prevention pipeline, starting from data collection to the final detection of fraudulent activities.
    - The collected data is verified for accuracy, completeness, and consistency.
    - This step may include removing duplicates, checking for missing values, and ensuring the integrity of the dataset.
4. **Pre-Processing Datasets**
    - Data preprocessing is a crucial step in ML/DL based scam detection.
    - This may involve:
        - **Cleaning Data**: Removing noise, irrelevant features, and inconsistencies.
        - **Feature Engineering**: Extracting important attributes like transaction patterns, user behaviour, or text analysis for phishing detection.
        - **Normalization & Encoding**: Transforming categorical data into numerical formats for model processing**.**
5. **View Results**
    - The system applies machine learning or deep learning models to the preprocessed data.
    - The final results could include:
        - Fraud risk scores
        - Anomalous transaction detections
        - Identified scam messages or phishing attempts
        - A detailed fraud report

6. **End (Black Circle)**

   The provided diagram appears to represent a workflow for scam prevention using machine learning (ML) and deep learning (DL) models. Below is an explanation of each step in the context of fraud detection:

   The process depicted follows a structured scam prevention pipeline, starting from data collection to the final detection of fraudulent activities.

   - This marks the completion of the fraud detection process, where the system either flags fraudulent activities or updates its model for future learning.

6.2 USECASE DIAGRAM

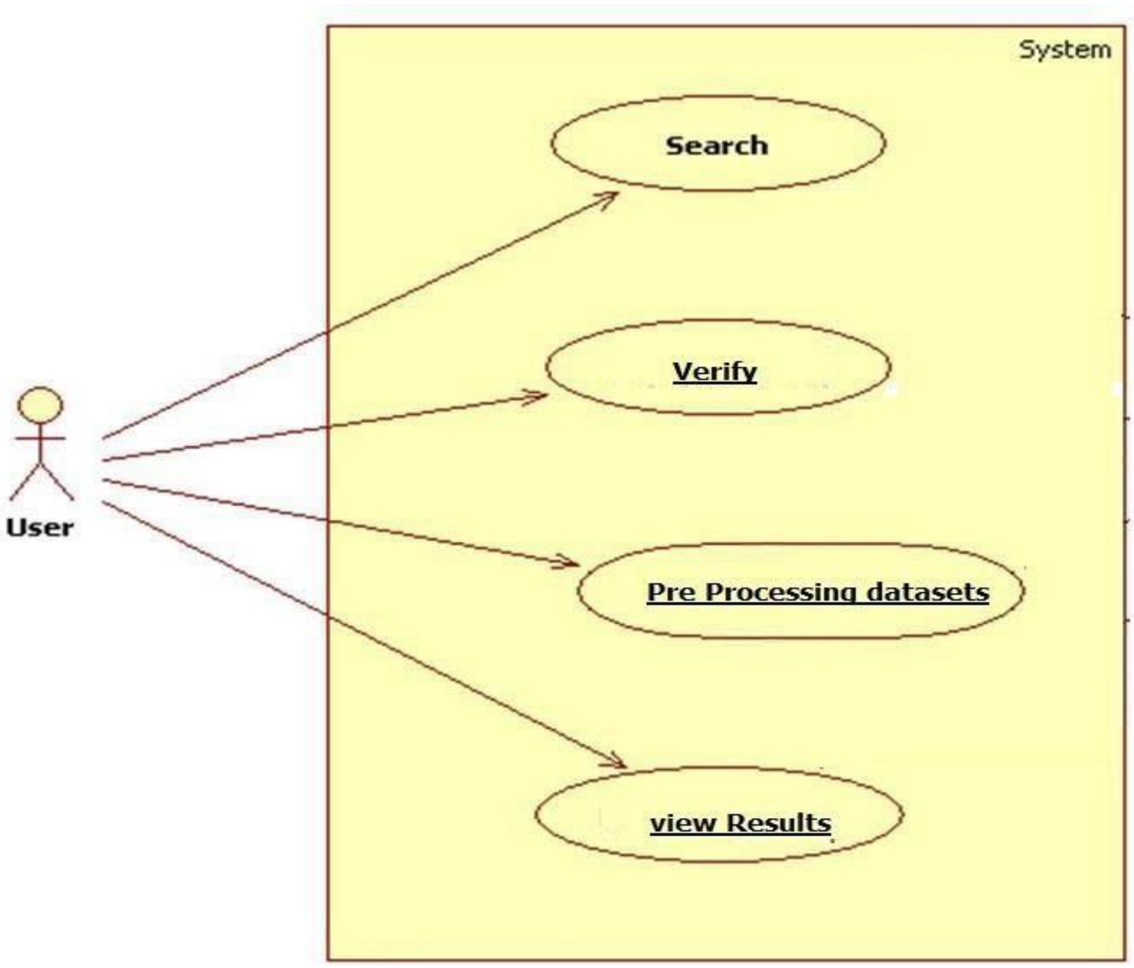


Fig:6.3Usecase Diagram

Explanation of the Diagram:

The provided diagram appears to represent a workflow for scam prevention using machine learning (ML) and deep learning (DL) models. Below is an explanation of each step in the context of fraud detection:

- **Actors:**
  - The User (depicted as a stick figure) interacts with the System to detect and prevent scams.
- **System:**
  - The yellow box represents the Scam Prevention System, which contains multiple functions.
- **Use Cases:**
  - The ovals inside the system represent different functionalities that the user can access.
  - The arrows indicate the user's interaction with these functionalities.

Functionalities of the Scam Prevention System

1. **Search**
   - The ovals inside the system represent different functionalities that the user can access.
   - The system collects and fetches data related to potential scams.
   - It can search through financial transactions, messages, emails, or online activities for scam indicators.
2. **Verify**
   - This step validates and authenticates the collected data to distinguish between genuine and fraudulent activities.
   - It may use supervised ML models (e.g., Random Forest, SVM) to classify transactions as legitimate or fraudulent.
3. **Pre-Processing Datasets**
   - Raw data is cleaned, formatted, and transformed for ML/DL models.
   - Tasks include:

- Removing duplicates
- Handling missing values
- Feature extraction for better scam detection

This step validates and authenticates the collected data to distinguish between genuine and fraudulent activities.

- Deep Learning models such as LSTMs (for fraud pattern detection) and NLP (for phishing scam detection) rely on well-pre-processed data.

4. View Results

a. The user can see scam detection results, such as:

   i. Whether a transaction is fraudulent

   ii. The scam probability score

   iii. The flagged messages/emails for phishing

b. The system uses AI-driven visual reports to present the findings effectively.

6.3 SEQUENCE DIAGRAM

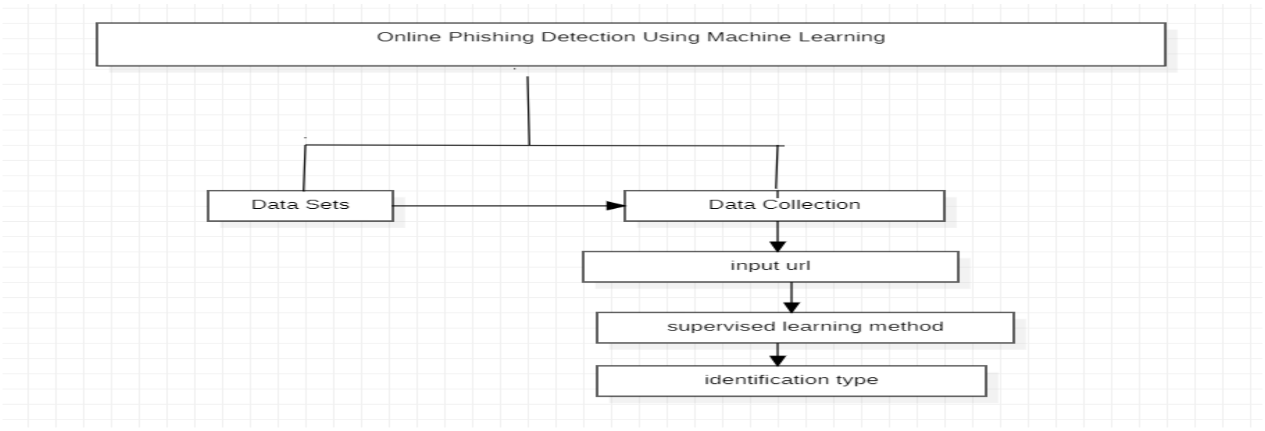

Online Phishing Detection Using Machine Learning
Data Sets
Data Collection
input url
supervised learning method
identification type

Fig: 6.4 Sequence Diagram

Explanation of the Diagram

This sequence diagram consists of five key components:
This step validates and authenticates the collected data to distinguish between genuine and fraudulent activities.

1. **User** – The person initiating a scam detection request.
2. **Search** – The module that processes the user's query.
3. **Verify** – The system component that verifies and validates the search query.
4. **DL SqlHelper** – A helper module that interacts with the database using deep learning algorithms**.**
5. **Database** – The data storage system containing scam-related datasets.

The diagram illustrates the workflow for Online Phishing Detection Using Machine Learning, outlining the step-by-step process involved in identifying phishing websites. It begins with the Data Sets, which consist of legitimate and phishing URLs collected from various sources. These datasets are then passed through the Data Collection phase, where relevant information is gathered for training the machine learning model. Once the data is collected, an input URL is provided by the user or system, which needs to be analyzed to determine whether it is legitimate or a phishing attempt.

To classify the input URL, the system employs a supervised learning method, where a trained machine learning model, such as Decision Trees, Random Forest, or Neural Networks, processes the URL by extracting key features that differentiate phishing sites from genuine ones. Finally, the system arrives at the identification type, where it determines whether the input URL is phishing or legitimate based on the learned patterns from the training data. This structured approach ensures an efficient and accurate method for detecting phishing attempts using machine learning.

6.4 ACTIVITY DIAGRAM

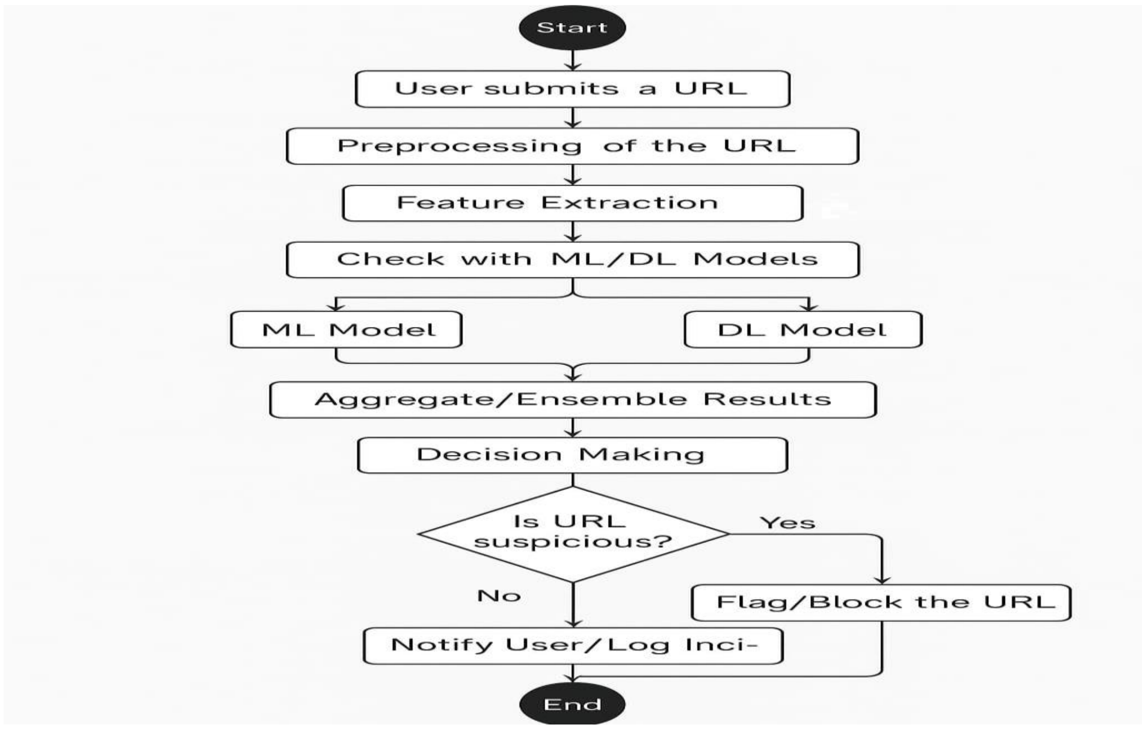


Fig: 6.5 Activity Diagram

6.5 CLASS DIAGRAM

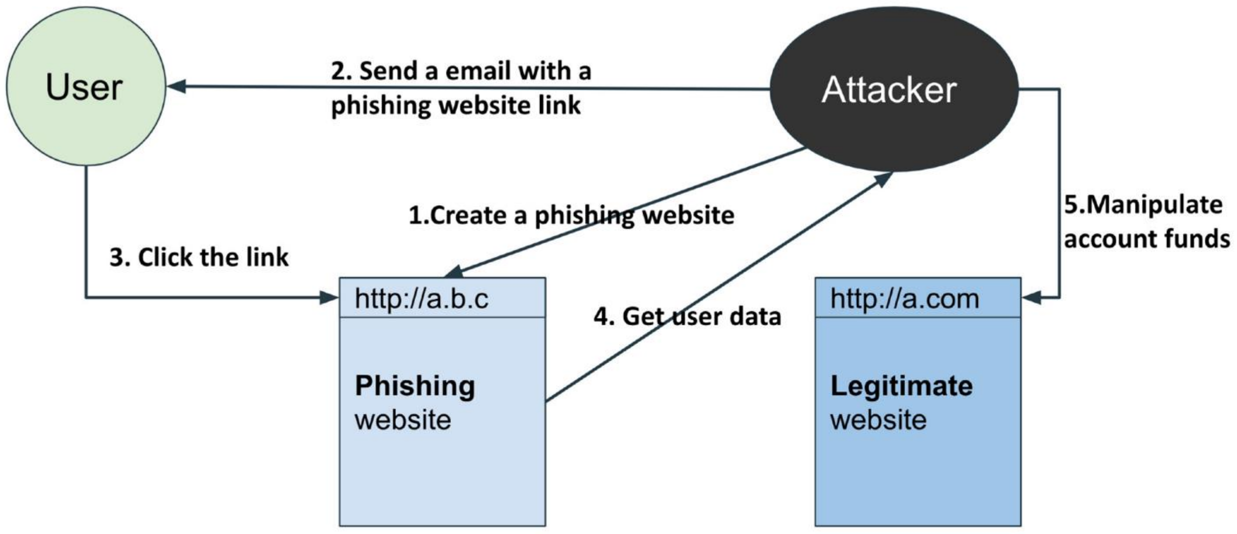


Fig: 6.6 Class Diagram

7. IMPLEMENTATION

7.1 PYTHON PROGRAMMING LANGUAGE

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted:** Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive:** You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language:** Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include:

- **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read:** Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain:** Python's source code is fairly easy-to-maintain.
- **A broad standard library:** Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

- **Interactive Mode:** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases:** Python provides interfaces to all major commercial databases.
- **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable:** Python provides a better structure and support for large programs than shell scripting.

Python has a big list of good features

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Operators in Python

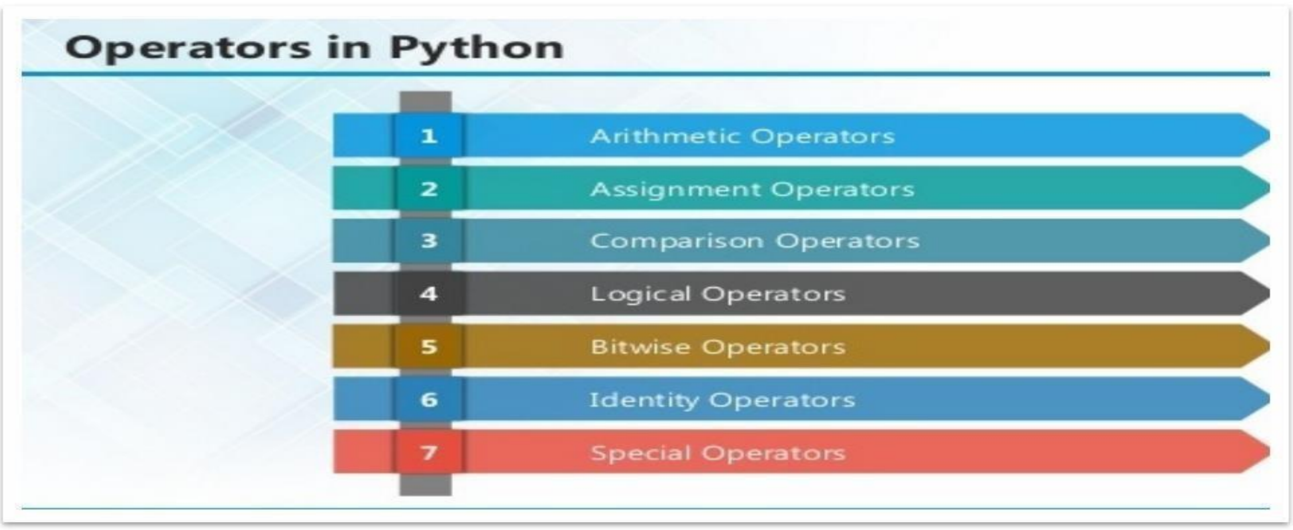


Fig: 7.1 Operators in Python

ARITHMETIC OPERATOR

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	a + b = 30
- Subtraction	Subtracts right hand operand from left hand operand.	a – b = -10
* Multiplication	Multiplies values on either side of the operator	a * b = 200
/ Division	Divides left hand operand by right hand operand	b / a = 2
% Modulus	Divides left hand operand by right hand operand and returns remainder	b % a = 0
** Exponent	Performs exponential (power) calculation on operators	a**b =10 to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity):	9//2 = 4 and 9.0//2.0 = 4.0, -11//3 = -4, -11.0//3 = -4.0

ASSIGNMENT OPERATOR

Operator	**Description**	**Example**
=	Assigns values from right side operands to left side operand	c = a + b assigns value of a + b into c
+= Add AND	It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a
-= Subtract AND	It subtracts right operand from the left operand and assign the result to left operand	c -= a is equivalent to c = c - a
*= Multiply AND	It multiplies right operand with the left operand and assign the result to left operand	c *= a is equivalent to c = c * a
/= Divide AND	It divides left operand with the right operand and assign the result to left operand	c /= a is equivalent to c = c / ac /= a is equivalent to c = c / a

%= Modulus AND	It takes modulus using two operands and assign the result to left operand	c %= a is equivalent to c = c % a
**= Exponent AND	Performs exponential (power) calculation on operators and assign value to the left operand	c **= a is equivalent to c = c ** a
//= Floor Division	It performs floor division on operators and assign value to the left operand	c //= a is equivalent to c = c // a

IDENTITY OPERATOR

Operator	**Description**	**Example**
Is	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	x is y, here is results in 1 if id(x) equals id(y).
is not	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.	x is not y, here is not results in 1 if id(x) is not equal to id(y

COMPARISON OPERATOR

Operator	**Description**	**Example**
& Binary AND	Operator copies a bit to the result if it exists in both operands	(a & b) (means 0000 1100)
| Binary OR	It copies a bit if it exists in either operand.	(a | b) = 61 (means 0011 1101)
^ Binary XOR	It copies the bit if it is set in one operand but not both.	(a ^ b) = 49 (means 0011 0001)
~ Binary Ones Complement	It is unary and has the effect of 'flipping' bits.	(~a) = -61 (means 1100 0011 in 2's complement form due to a signed binary number.
<< Binary Left Shift	The left operands value is moved left by the number of bits specified by the right operand.	a << 2 = 240 (means 1111 0000)

>> Binary Right Shift	The left operands value is moved right by the number of bits specified by the right operand.	a >> 2 = 15 (means 0000 1111)

LOGICAL OPERATOR

Operator	Description	Example
and Logical AND	If both the operands are true then condition becomes true.	(a and b) are true.
or Logical OR	If any of the two operands are non-zero then condition becomes true.	(a or b) is true.
not Logical NOT	Used to reverse the logical state of its operand.	Not(a and b) is false.

MEMBERSHIP OPERATOR

Operator	Description	Example
In	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here in results in a 1 if x is a member of sequence y.
not in	Evaluates to true if it does not find a variable in the specified sequence and false otherwise.	x not in y, here not

		in results in a 1 if x is not

PYTHON OPERATOR’S PRECENDENCE

Operator	Description
**	Exponentiation (raise to the power)
~ + -	Complement, unary plus and minus (method names for the last two are +@ and -@)
* / % //	Multiply, divide, modulo and floor division
+ -	Addition and subtraction
>> <<	Right and left bitwise shift
&	Bitwise 'AND'
^ |	Bitwise exclusive `OR' and regular `OR'
<= < > >=	Comparison operators
<> == !=	Equality operators
= %= /= //= -= += *= **=	Assignment operators
is not	Identity operators
in not in	Membership operators

7.2 PYTHON LIBRARIES

It can be used as a scripting language or can be compiled to byte-code for building large applications.

7.2.1 Streamlit: Streamlit is an open-source Python library that makes it easy to create interactive web applications for data science, machine learning and analytics without needing deep web development skills. It is particularly useful for quickly turning Python scripts into interactive dashboards.

7.2.2 NumPy: NumPy is a powerful open-source Python library used for numerical computations. It provides support for multi-dimensional arrays, mathematical functions, and linear algebra operations, making it essential for data science, machine learning, and scientific computing.

7.2.3 Pandas: Pandas is powerful open-source Python library used for data manipulation, analysis, and cleaning. It provides data structure like Series and Data frame, making it easy to handle structured data. It is easy-to-use data structures.

7.2.4 Requests: The requests library is a popular Python module used for making HTTP requests in a simple and user-friendly way. It allows you to send GET, POST, PUT, DELETE, and other types of HTTP requests to interact with web services and APIs.

7.2.5 Matplotlib: Matplotlib is a powerful Python library used for creating static, animated, and interactive visualizations. It is widely used in data science, machine learning, and scientific computing to plot graphs, charts and images.

8. CODING

8.1 CODE DESCRIPTION

```
import streamlit as st

import numpy as np

import pickle

from urllib.parse import urlparse

import requests

from urllib.parse import urlparse

from datetime import datetime

import re

from requests.exceptions import
SSLError, Timeout # Add this import for
SSLError and Timeout

def get_domain(url):

    domain = urlparse(url).netloc

    if re.match(r"^www.",domain):

        domain =
domain.replace("www.","")

    return domain

def having_ip(url):
    try:

        ipaddress.ip_address(url)

        ip = 1

    except:

        ip = 0

    return ip

def have_at_sign(url):

    if "@" in url:

        at = 1

    else:

        at = 0

    return at

def get_length(url):

   if len(url) < 54:

       length = 0

else:

        length = 1

    return length

def get_depth(url):

    s = urlparse(url).path.split('/')

    depth = 0
    for j in range(len(s)):
        if len(s[j]) != 0:
            depth = depth+1
    return depth
def redirection(url):
    pos = url.rfind('//')
    if pos > 6:
        if pos > 7:
            return 1
        else:
            return 0
else:
        return 0
def http_domain(url):
domain = urlparse(url).netloc
    if 'https' in domain:
        return 1
    else:
        return 0
def tiny_url(url):
```

```
r"yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|" \
r"short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|" \
r"doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|db\.tt|" \
r"qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|q\.gs|is\.gd|" \
r"po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|x\.co|" \
r"prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.me|v\.gd|"  \
r"tr\.im|link\.zip\.net"
match=re.search(shortening_services,url)
    if match:
       return 1
    else:
        return 0
def prefix_suffix(url):
    if '-' in urlparse(url).netloc:
        return 1
    else:
return 0
def web_traffic(url):
    try:
```

```
querystring = {"domain": url}

        headers = {

            "X-RapidAPI-Key": "cd4733fedbmsh6f2cfc21cf195f2p1d088djsn84e6c824c74e",

            "X-RapidAPI-Host": "similar-web.p.rapidapi.com"

        }

        response = requests.get("https://similar-web.p.rapidapi.com/get-analysis",
headers=headers, params=querystring)

        data = response.json()

rank = data['GlobalRank']['Rank']

        rank = int(rank)

    except (requests.exceptions.RequestException, ValueError, KeyError):

        rank = 1

 if rank < 100000:

        return 1

else:

        return 0

def iframe(response):

    if response == "":

        return 1

    else:
```

```
 if re.findall(r"[<iframe>|<frameBorder>]", response.text):
            return 0
        else:
            return 1
def mouse_over(response):
    if response == "" :
        return 1
    else:
        if re.findall("<script>.+onmouseover.+</script>", response.text):
            return 1
else:
            return 0
def right_click(response):
    if response == "":
        return 1
    else:
        if re.findall(r"event.button ?== ?2", response.text):
            return 0
        else:
            return 1
```

```
def forwarding(response):
    if response == "":
        return 1
    else:
        return 0
If len(response.history) <= 2:
          return 0
 else:
            return 1
def get_http_response(url):
    try:
        response = requests.get(url, timeout=5)  # Set a timeout of 5 seconds
        return response
    except requests.exceptions.RequestException as e:
        st.error(f"Error: {e}")
        return None
def extract_features(url):
    features = []
      # Address bar-based features
    features.append(having_ip(url))

    features.append(have_at_sign(url))

    features.append(get_length(url))

features.append(get_depth(url))

     features.append(redirection(url))

     features.append(http_domain(url))

     features.append(tiny_url(url))

     features.append(prefix_suffix(url))

# Domain based features

    dns = 0

    dns_age = 0

    dns_end = 0

    features.append(dns)

    features.append(dns_age)

    features.append(dns_end)

    features.append(web_traffic(url))

    response = get_http_response(url)

# HTML & Javascript based features

    if response is not None:

        features.append(iframe(response))
        features.append(mouse_over(response))

        features.append(right_click(response))

        features.append(forwarding(response))

else:

        # If response is None, set these features to 0 or None

        features.extend([0, 0, 0, 0])

return features

def predict_phishing(features):

    # Load the model

    with open('mlp_model.pkl', 'rb') as file:

        loaded_model = pickle.load(file)

  # Make predictions

    new_data = np.array([features])

    prediction = loaded_model.predict(new_data)

    print("prediction:", prediction)

return prediction

    # Try:

     # Make sure you provide the correct file path to your model

           #     with open("model_file.pkl", "rb") as file:
```

```
    #     return prediction

    # except Exception as e:

  #     print(f"Error occurred while loading model or making prediction: {e}")

  #     return None

def main():

    st.title('Prevention of Scams Using Deep Learning and Machine Learning Models')

    st.write("Enter a URL to check if it's phishing or not.")

     # Input URL

    url = st.text_input("Enter URL:")

    if st.button("Check"):

        # Extract features

        st.write("Extracting features...")

        features = extract_features(url)

# Make prediction

        st.write("Predicting...")

        prediction = predict_phishing(features)

# Display prediction

        if prediction[0] == 0:

            st.write("Prediction made:")

            st.error("Spam Alert! This URL is classified as phishing.")
```

```
else:

          st.write("Predicting made:")

           st.success("No Scams Detected. This URL seems safe.")

 if _name___== '_main_':

    main( )
```

9. OUTPUT SCREENSHOTS

9.1 OPEN ACER

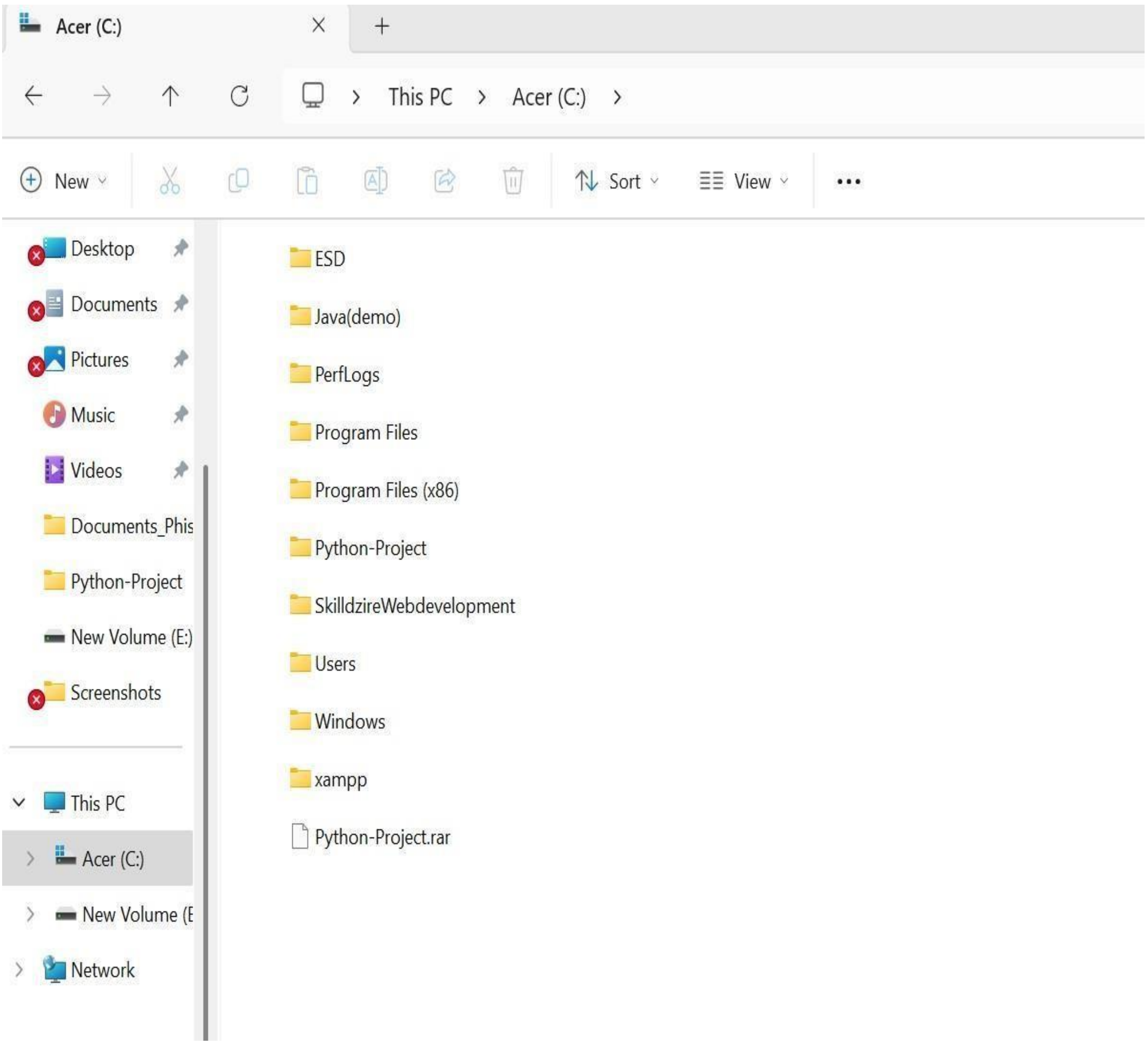
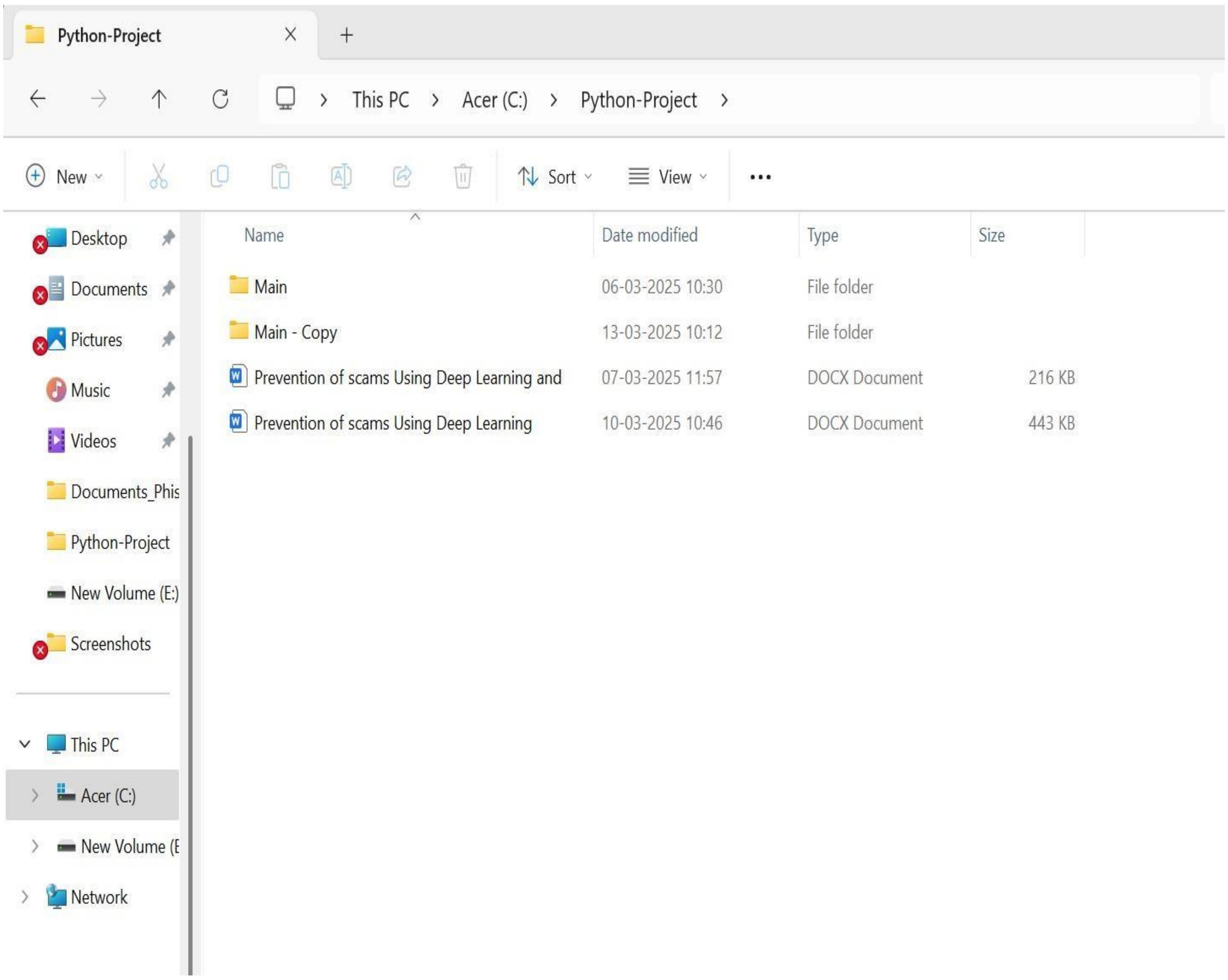


Fig:9.1 Open Acer

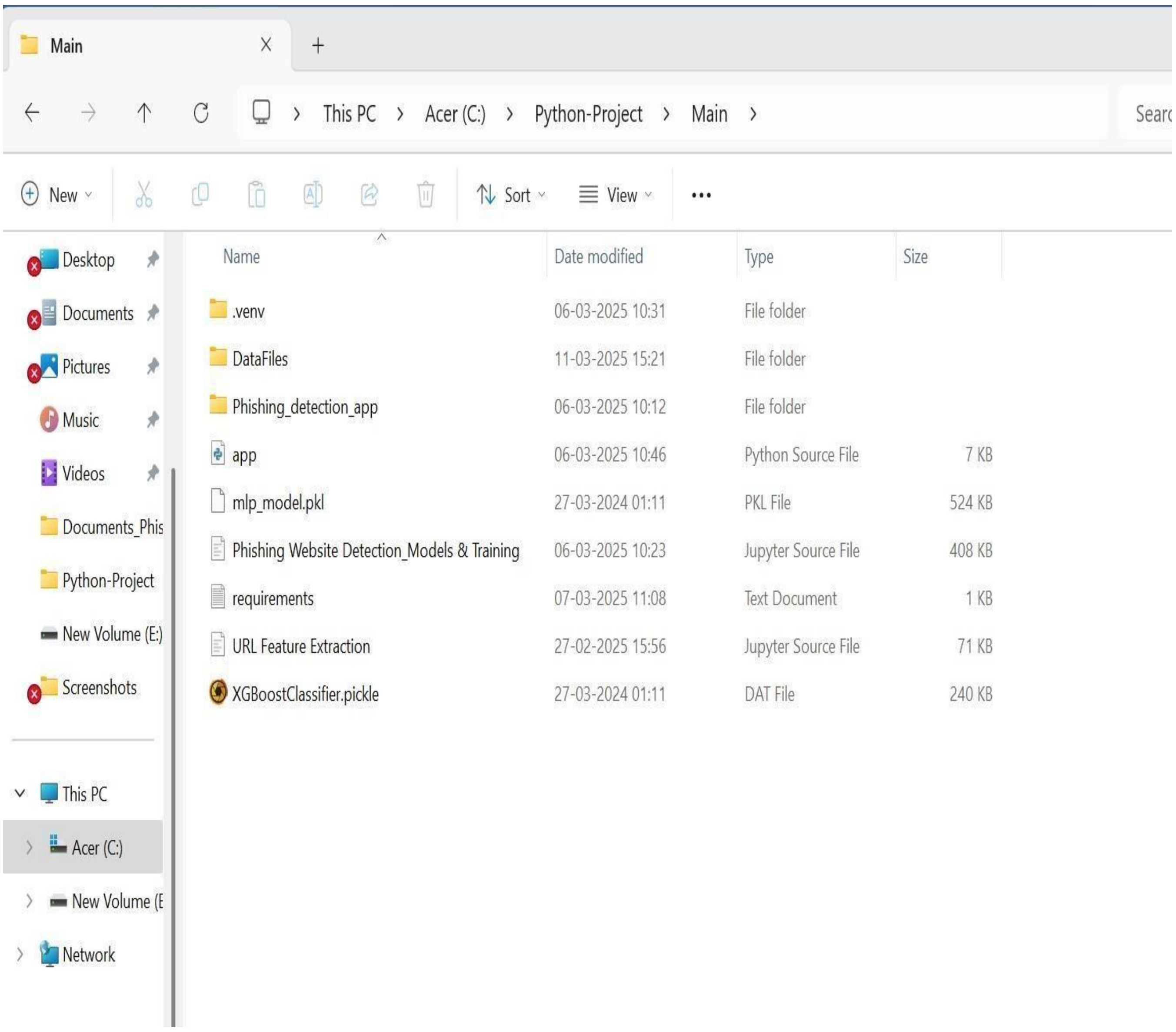
9.2 OPEN PYTHON PROJECT FILE



Name	Date modified	Type	Size
Main	06-03-2025 10:30	File folder	
Main - Copy	13-03-2025 10:12	File folder	
Prevention of scams Using Deep Learning and	07-03-2025 11:57	DOCX Document	216 KB
Prevention of scams Using Deep Learning	10-03-2025 10:46	DOCX Document	443 KB

Fig:9.2 Python Project file

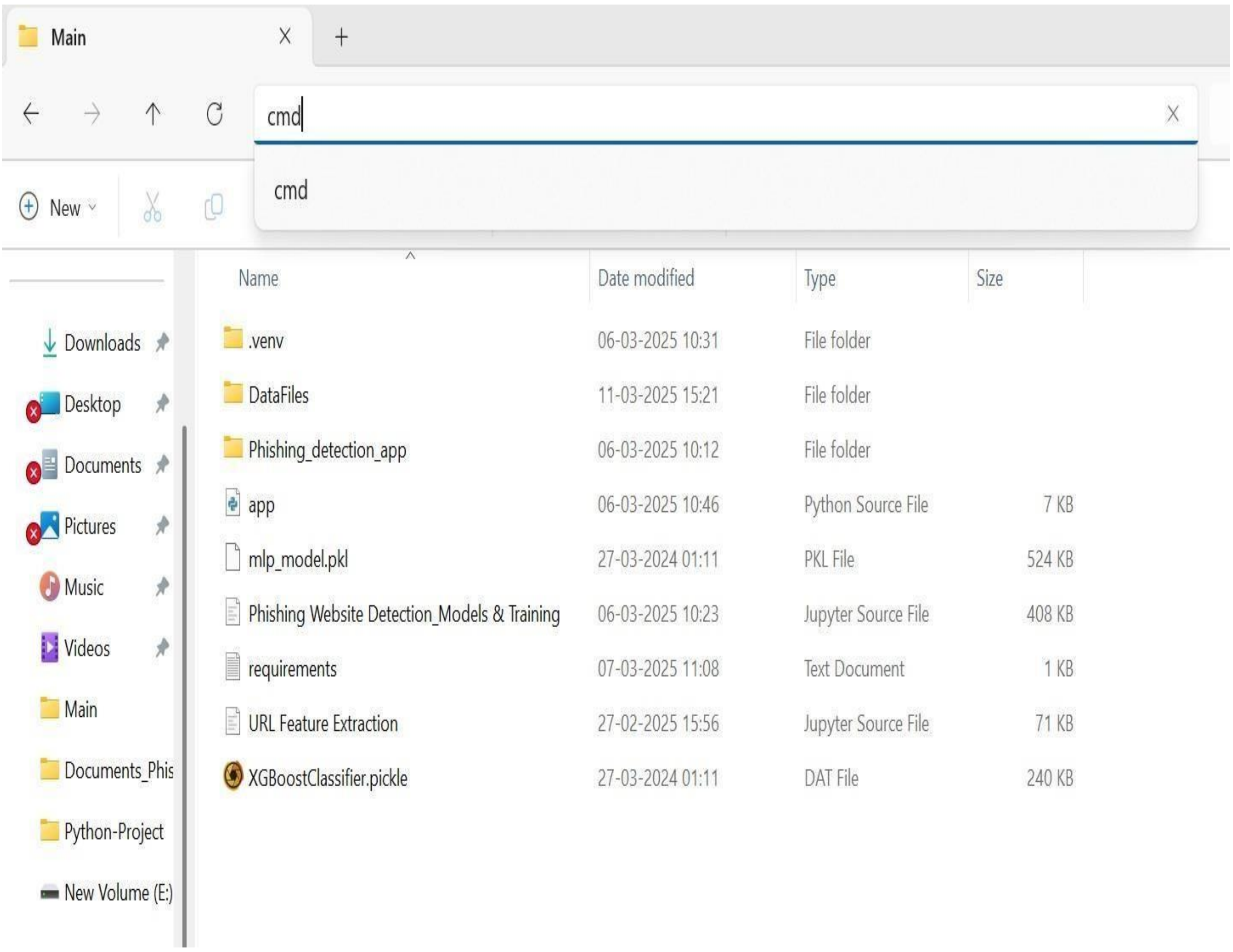
9.3 OPEN MAIN FILE



Name	Date modified	Type	Size
.venv	06-03-2025 10:31	File folder	
DataFiles	11-03-2025 15:21	File folder	
Phishing_detection_app	06-03-2025 10:12	File folder	
app	06-03-2025 10:46	Python Source File	7 KB
mlp_model.pkl	27-03-2024 01:11	PKL File	524 KB
Phishing Website Detection_Models & Training	06-03-2025 10:23	Jupyter Source File	408 KB
requirements	07-03-2025 11:08	Text Document	1 KB
URL Feature Extraction	27-02-2025 15:56	Jupyter Source File	71 KB
XGBoostClassifier.pickle	27-03-2024 01:11	DAT File	240 KB

Fig: 9.3 Open Main File

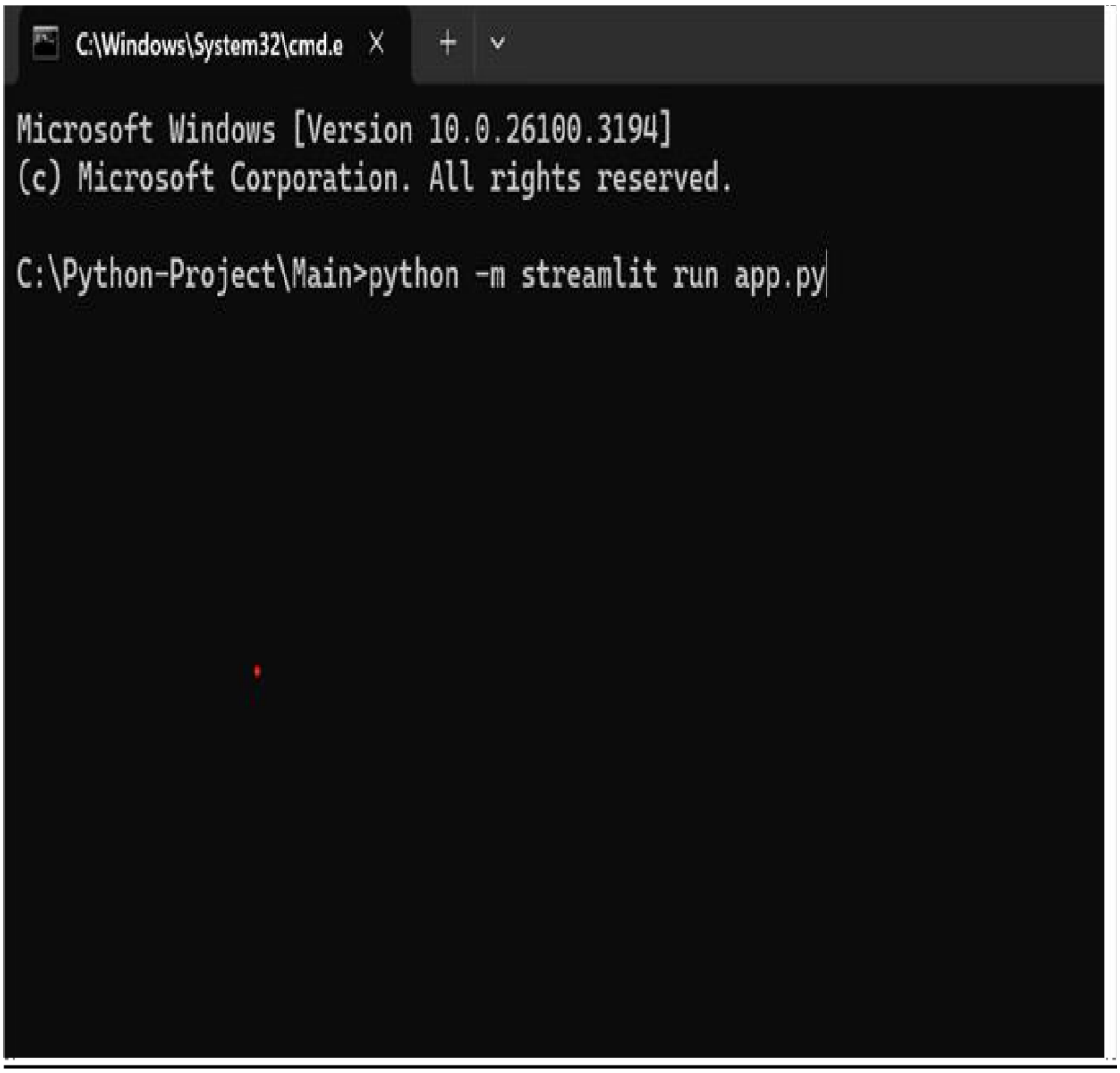
9.4 MAIN FILE



Name	Date modified	Type	Size
.venv	06-03-2025 10:31	File folder	
DataFiles	11-03-2025 15:21	File folder	
Phishing_detection_app	06-03-2025 10:12	File folder	
app	06-03-2025 10:46	Python Source File	7 KB
mlp_model.pkl	27-03-2024 01:11	PKL File	524 KB
Phishing Website Detection_Models & Training	06-03-2025 10:23	Jupyter Source File	408 KB
requirements	07-03-2025 11:08	Text Document	1 KB
URL Feature Extraction	27-02-2025 15:56	Jupyter Source File	71 KB
XGBoostClassifier.pickle	27-03-2024 01:11	DAT File	240 KB

Fig:9.4 Main File

9.5 COMMAND FOR EXECUTION

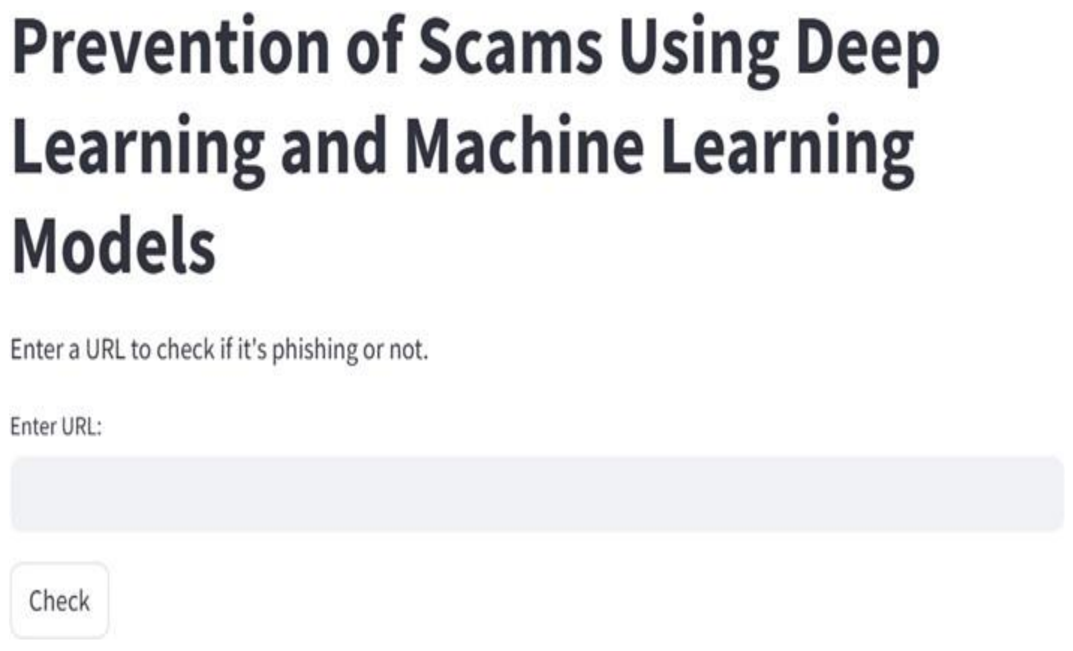


```
Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

C:\Python-Project\Main>python -m streamlit run app.py
```

Fig:9.5 Command for Execution

9.6 BEFORE ENTERING THE URL FOR EXECUTION PAGE



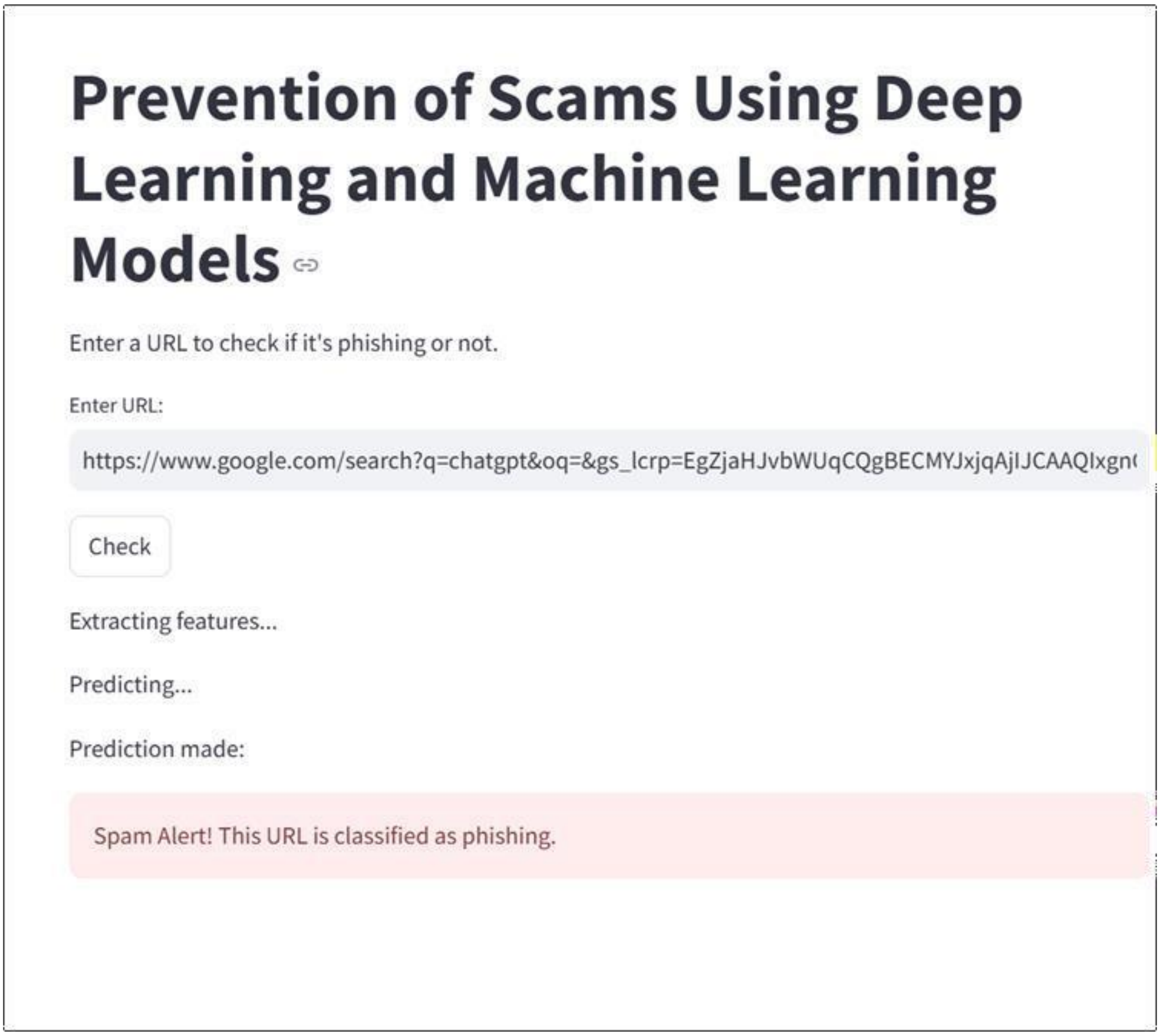
Prevention of Scams Using Deep Learning and Machine Learning Models

Enter a URL to check if it's phishing or not.

Enter URL:

Fig: 9.6 Before entering the URL for execution page

9.7 AFTER ENTERING THE URL



Prevention of Scams Using Deep Learning and Machine Learning Models

Enter a URL to check if it's phishing or not.

Enter URL:

https://www.google.com/search?q=chatgpt&oq=&gs_lcrp=EgZjaHJvbWUqCQgBECMYJxjqAjIJCAAQIxgn

Extracting features...

Predicting...

Prediction made:

Spam Alert! This URL is classified as phishing.

Fig:9.7 After entering URL

10. TESTING

The following are the Testing Methodologies:

- Unit Testing.
- Integration Testing.
- User Acceptance Testing.
- Output Testing.
- Validation Testing.

10.1 UNIT TESTING

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module's control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing. During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All-important processing path are tested for the expected results. All error handling paths are also tested.

10.2 INTEGRATION TESTING

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

The following are the types of Integration Testing:

10.2.1 Top-Down Integration

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

10.2.2 Bottom-up Integration

In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom-up integration strategy may be implemented with the following steps:

- The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.
- A driver (i.e.) the control program for testing is written to coordinate test case input and output.
- The cluster is tested.
- Drivers are removed and clusters are combined moving upward in the program structure

The bottom-up approaches test each module individually and then each module is module is integrated with a main module and tested for functionality.

10.3 USER ACCEPTANCE TESTING

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required.

The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

10.4 OUTPUT TESTING

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

10.5 VALIDATION TESTING

Validation checks are performed on the following fields.

Text Field:

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

The bottom-up approaches test each module individually and then each module is module is integrated with a main module and tested for functionality.

Numeric Field:

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error message. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data.

The individually tested modules are integrated into a single system. Testing involves executing the real data information is used in the program the existence of any program defect is inferred from the output.

The testing should be planned so that all the requirements are individually tested. A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

Preparation of Test Data

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

Using Live Test Data:

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves. It is difficult to obtain live data in sufficient amounts to conduct extensive testing.

And, although it is realistic data that will show how the system will perform for the typical processing requirement, assuming that the live data entered are in fact typical, such data generally will not test all combinations or formats that can enter the system. This bias toward typical values then does not provide a true systems test and in fact ignores the cases most likely to cause system failure.

Using Artificial Test Data:

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values.

In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.

The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications.

The package “Virtual Private Network” has satisfied all the requirements specified as per software requirement specification and was accepted.

USER TRAINING

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed. For this purpose, the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is very easy.

MAINTAINENCE

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, we have more accurately defined the user’s requirements during the process of system development. Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent.

With development in technology, it may be possible to add many more features based on the requirements in future.

The coding and designing are simple and easy to understand which will make maintenance easier.

TESTING STRATEGY :

A strategy for system testing integrates system test cases and design techniques into a well-planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation .A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements. Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing represents an interesting anomaly for the software.

SYSTEM TESTING:

Testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing represents an interesting anomaly for the software.

Software once validated must be combined with other system elements (e.g. Hardware, people, database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

UNIT TESTING:

In unit testing different are modules are tested against the specifications produced during the design for the modules. Unit testing is essential for verification of the code produced during the coding phase, and hence the goals to test the internal logic of the modules. Using the detailed design description as a guide, important Conrail paths are tested to uncover errors within the boundary of the modules.

This testing is carried out during the programming stage itself. In this type of testing step, each module was found to be working satisfactorily as regards to the expected output from the module.

In Due Course, latest technology advancements will be taken into consideration. As part of technical build-up many components of the networking system will be generic in nature so that future projects can either use or interact with this. The future holds a lot to offer to the development and refinement of this project.

11. CONCLUSION & FUTURE ENHANCEMENT

11.1 CONCLUSION

Phishing detection mechanism aims to improve current blacklist methods, protecting users from malicious login forms. Our work provides an updated dataset PILU-90K for researchers to train and test their approaches. This dataset includes legitimate login URLs which are the most representative scenario for real-world phishing detection.

We explored several URL-based detection models using deep learning and machine learning solutions trained with phishing and legitimate home URLs. The main advantage of our approach is the low false-positive rate when classifying this type of URL. Among the different evaluated models, TFIDF combined with N-gram and LR algorithm obtained the best results with a 96:50% accuracy.

11.2 FTURE ENHANCEMENT

Enhancing scam prevention using machine learning (ML) and deep learning (DL) involves several key advancements to improve detection accuracy, adaptability, and response time. One significant enhancement is the use of anomaly detection models such as Autoencoders, Isolation Forests, and One-Class SVMs, which can identify unusual patterns in user behavior, transactions, or interactions, helping to flag potential fraud. Additionally, Graph Neural Networks (GNNs) can analyze relationships between entities, such as users, devices, and transactions, making it easier to detect organized scam networks and fraudulent collaborations. Deep learning techniques, particularly Natural Language Processing (NLP) models like BERT and GPT, can also be used to detect scam patterns in emails, messages, and phone calls, while speech recognition models such as WaveNet and Deep Speech can identify fraudulent robocalls and voice scams.

Additionally, AI can play a crucial role in securing digital transactions through blockchain-powered fraud detection, smart contracts with ML-based risk analysis, and AI-enhanced CAPTCHAs to counter scam bots.

.

12. REFERENCES

[1] Statista. (2020). Adoption Rate of Emerging Technologies in Organizations Worldwide as of 2020. Accessed: Sep. 12, 2021. [Online]. Available:https://www.statista.com/statistics/661164/worldwide-cio-surveyoperati%onal-priorities/

[2] R. De', N. Pandey, and A. Pal, ``Impact of digital surge during COVID-19 pandemic: A viewpoint on research and practice,'' Int. J. Inf. Manage.,

vol. 55, Dec. 2020, Art. no. 102171.

[3] P. Patel, D. M. Sarno, J. E. Lewis, M. Shoss, M. B. Neider, and C. J. Bohil,``Perceptual representation of spam and phishing emails,'' Appl. Cognit.

Psychol., vol. 33, no. 6, pp. 1296_1304, Nov. 2019.

[4] J. A. Chaudhry, S. A. Chaudhry, and R. G. Rittenhouse, ``Phishing attacks and defenses,'' Int. J. Secur. Appl., vol. 10, no. 1, pp. 247_256, 2016.

[5] M. Hijji and G. Alam, ``A multivocal literature review on growing social engineering based cyber-attacks/threats during the COVID-19

pandemic: Challenges and prospective solutions,'' IEEE Access, vol. 9, pp. 7152_7169, 2021.

[6] A. Alzahrani, ``Coronavirus social engineering attacks: Issues and recommendations,'' Int. J. Adv. Comput. Sci. Appl., vol. 11, no. 5, pp. 154_161, 2020.

[7] Phishing Activity Trends Report 3Q, Anti-Phishing Working Group, International, 2017. Accessed: Sep. 12, 2021.

[8] Phishing Activity Trends Report 1Q, Anti-Phishing Working Group, International, 2021. Accessed: Sep. 14, 2021.

[9] R. Chen, J. Gaia, and H. R. Rao, ``An examination of the effect of recent phishing encounters on phishing susceptibility,'' Decis. Support Syst., vol. 133, Jun. 2020, Art. no. 113287.

[10] Phishing Activity Trends Report 4Q, Anti-Phishing Working Group, International, 2020. Accessed: Sep. 12, 2021.

[11] S. Bell and P. Komisarczuk, ``An analysis of phishing blacklists: Google safe browsing, OpenPhish, and PhishTank," in Proc. Australas. Comput. Sci. Week Multiconf., Feb. 2020, pp. 1_11.

[12] A. Oest, Y. Safaei, P. Zhang, B. Wardman, K. Tyers, Y. Shoshitaishvili, A. Doupé, and G.-J. Ahn, ``Phishtime: Continuous longitudinal measurement of the effectiveness of anti-phishing blacklists," in Proc. 29th USENIX Secur. Symp., 2020, pp. 379_396.

[13] L. Li, E. Berki, M. Helenius, and S. Ovaska, ``Towards a contingency approach with whitelist- and blacklist-based anti-phishing applications:

What do usability tests indicate?" Behaviour Inf. Technol., vol. 33, no. 11,

pp. 1136_1147, Nov. 2014.

[14] N. Samarasinghe and M. Mannan, ``On cloacking behaviours of malicious websites," Comput. Secur., vol. 101, pp. 102_114, Feb. 2021.

[15] L. Halgas, I. Agra_otis, and J. R. C. Nurse, ``Catching the phish: Detecting phishing attacks using recurrent neural networks (RNNs)," in Information Security Applications (Lecture Notes in Computer Science), vol. 11897. Cham, Switzerland: Springer, 2020, pp. 219_233.

[16] R. S. Rao and A. R. Pais, ``Jail-phish: An improved search engine-based phishing detection system," Comput. Secur., vol. 83, pp. 246_267, Jun. 2019.

[17] Z. Dou, I. Khalil, A. Khreishah, A. Al-Fuqaha, and M. Guizani, ``Systematization of knowledge (SoK): A systematic review of softwarebased web phishing detection," IEEE Commun. Surveys Tuts., vol. 19, no. 4, pp. 2797_2819, 4th Quart., 2017.

[18] K. L. Chiew, E. H. Chang, C. Lin Tan, J. Abdullah, and K. S. C. Yong, ``Building standard of_ine anti-phishing dataset for benchmarking," Int.J. Eng. Technol., vol. 7, no. 4, pp. 7_14, 2018.

[19] H. Yuan, Z. Yang, X. Chen, Y. Li, and W. Liu, ``URL2 Vec: URL modeling with character embeddings for fast and accurate phishing website detection," in Proc. IEEE Int. Conf. Parallel Dis-trib. Process.

(ISPA/IUCC/BDCloud/SocialCom/SustainCom), Dec. 2018, pp. 265_272.

[20] M. Sánchez-Paniagua, E. Fidalgo, V. González-Castro, and E. Alegre,``Impact of current phishing strategies in machine learning models for phishing detection," in Computational Intelligence in Security for Information Systems Conference, vol. 12676. Cham, Switzerland: Springer, 2021,pp. 87_96.

[21] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, ``Machine learning based phishing detection from URLs," Expert Syst. Appl., vol. 117, pp. 345_357, Mar. 2019.

[22] Y. Cao, W. Han, and Y. Le, ``Anti-phishing based on automated individual white-list," in Proc. 4th ACM Workshop Digit. Identity Manage. (DIM),2008, pp. 51_59.

[23] P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta, ``PhishNet: Predictive blacklisting to detect phishing attacks," in Proc. IEEEINFOCOM, Mar. 2010, pp. 1_5.

[24] A. K. Jain and B. B. Gupta, ``A novel approach to protect against phishing attacks at client side using auto-updated white-list," EURASIP J. Inf.Secur., vol. 2016, no. 1, pp. 1_11, Dec. 2016.

[25] E. Buber, B. Diri, and O. K. Sahingoz, ``NLP based phishing attack detection from URLs," in Proc. Int. Conf. Intell. Syst. Design Appl., vol. 736, 2018, pp. 608_618.