EMAIL BASED PHISHING WEBSITES AND EMAIL DETECTION USING MACHINE LEARNING

23-123

BSc (Hons) in Information Technology
Specializing in Cyber Security

Department of Computer Systems Engineering

Sri Lanka Institute of Information Technology
Sri Lanka

September 2023

DECLARATION

I declare that this is my own work, and this dissertation does not incorporate without acknowledgement any material previously submitted for a degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. Also, I hereby grant to Sri Lanka Institute of Information Technology, the nonexclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Student Name	Student ID	Signature
J. M. Lindamulage	IT20222840	Judith
Mandira Pabasari L.	IT19966236	[illegible]
Yapa S. P. J.	IT20050108	[illegible]
Perera I. S. S.	IT20222468	[illegible]

The above candidates have carried out research for the bachelor's degree Dissertation under my supervision.

--- ---------------------------------

Signature of the Supervisor Date

ABSTRACT

Phishing is now one of the most popular cyberattacks. Phishing attacks are now launched in a variety of ways, including spoof calls, messaging, social networking, and emails, with the goal of acquiring sensitive information from victims. The most typical way victims fall victim to phishing is through clicking on a bogus link in an email delivered by an attacker. This is known as email phishing. This link will redirect victims to websites that appear authentic but are operated by the attackers. Attackers create unlawful email accounts using legitimate company details and send them to victims, impersonating a real person, while utilizing social engineering to get the victim to click on a link to a fraudulent website. If a victim can get to know that the site or the email is a phishing one beforehand it will help to reduce losses such as financial and reputational losses.

In the integrated system there are two products which are an extension and an android mobile application. Through the mobile application a user can input an image and check if it’s a phishing site and there is another option for phishing emails. This functions in a way that the user can copy and paste the email text and then the app will tell if it’s phishing mail or not. For the option of input screenshot of the suspected website a model which was trained using graph neural network will be used. For email checker a naïve bayes model is integrated.

The browser extension complements the mobile application, bolstering the user's defense against phishing threats while browsing the web. The extension deploys machine learning models trained to discern phishing websites based on a diverse range of features, including URL structure, HTML and JavaScript attributes, and abnormal site behaviors. Through real-time analysis, the extension provides users with immediate feedback, allowing them to make informed decisions about the sites they interact with.

In summary, this project offers a comprehensive solution to counteract phishing attacks, leveraging machine learning methodologies and a Random Forest Classifier. The integration of both a browser extension and a mobile application empowers users to proactively identify and safeguard against phishing threats across multiple platforms, significantly enhancing cybersecurity measures.

ACKNOWLEGEMNET

First of all, we would like to express my sincere gratitude to Ms. Jenny Krishara of Faculty of computing in Sri Lanka Institute of Information Technology (SLIIT) for her invaluable support and guidance throughout the project and for sharing her experience and knowledge with us to complete the project. We are grateful for the guidance given by the CDAP panel and all the SLIIT lecturers for their advice in our project work.

Table of Contents

DECLARATION........ i

ABSTRACT........ i

List Of Figures iv

List Of Tables v

INTRODUCTION 1

1. Background and Literature Review 1
2. Research Gap 17
3. Research problem........ 25
4. Research Objectives........ 29
5. METHODOLOGY 34
   - A. MOBILE APPLICATION........ 34
   - B. Web extension 45
6. COMMERCIALIZING ASPECTS OF THE PRODUCT........ 48
7. TESTING AND IMPLEMENTAION........ 50

RESULTS AND DISCUSSION........ 56

SUMMARY........ 67

CONCLUSION........ 70

REFERENCES 71

APPENDIX........ 75

List Of Figures

Figure 0.1Visual similarity-based approaches 3
Figure 0.2The steps of how a graph is constructed [12]. 5
Figure 0.3Model define 35
Figure 0.41Pytorch to android 37
Figure 0.5Phishing Email Detection System Diagram 38
Figure 0.6app diagram 38
Figure 0.7ML Code(Data Preprocessing) 39
Figure 0.8ML code(data visualization) 40
Figure 0.9 ML Code(Data Splitting and Feature Engineering 41
Figure 0.10ML Code (Model Training) 42
Figure 0.11Flask server code 44
Figure 0.12Mobile app interface 50
Figure 0.13Image output for benign site 51
Figure 0.14image output for phishing site 51
Figure 0.15Phishing email output for benign and phishing email 52
Figure 0.1Neural networks 58
Figure 0.2Random Forest 58
Figure 0.3Support Vector Machine 59
Figure 0.4Dataset 59
Figure 0.5Cross Validation and Accuracy 60
Figure 0.6Legitimate site detection 62
Figure 0.7Phishing site detected 62
Figure 0.8Accuracy(Bernoulli Naive Bayes) 63
Figure 0.9Accuracy 65
Figure 0.1System diagram 75

List Of Tables

1Summary table 18

2Research GAP for web features related work.. 19

3 Accuracies of each model 56

4Accuracy values for each model... 63

INTRODUCTION

1. Background and Literature Review

1.1 Phishing attacks and Mitigation Methods

Cyber threats, including phishing, are a growing concern as they pose a significant threat to sensitive data. Phishing, a cunning tactic, involves cybercriminals creating fraudulent messages or websites that mimic reputable entities, such as banks or government agencies. These messages are distributed through various channels, such as emails, text messages, and social media, and often employ spoofing techniques to make them appear legitimate. The victim is induced to take action, which may lead to the disclosure of sensitive information. Once the victim interacts with the phishing content, their personal information is collected by the attacker. Armed with the stolen data, cybercriminals can engage in malicious activities such as identity theft, unauthorized access to financial accounts, or the dissemination of malware. Phishing attacks come in various forms, each tailored to exploit different vulnerabilities. Examples include spear phishing, whaling, and smishing, which involve impersonating legitimate organizations and requesting sensitive information over the phone. Understanding the anatomy of phishing attacks is crucial in safeguarding against these digital ploys [5]. Machine learning (ML) can be used in conjunction with visual similarity analysis, website features, email text, and URL analysis to enhance cybersecurity. ML models can identify subtle visual discrepancies between phishing websites and their legitimate counterparts, such as fonts, layouts, and color schemes. Website features analysis can be used to detect anomalies, such as unusual user interaction flows or hidden elements. Content analysis can be used to identify inconsistencies or malicious elements. Natural Language Processing (NLP) can be used to analyze email text to detect phishing attempts, such as urgency or vague sender information. ML can also recognize patterns in phishing emails. URL analysis can be used to assess the reputation of URLs based on historical data and to identify inconsistencies in URL structures. The phishing mitigation workflow involves data collection, model training, real-time analysis, alerts and remediation, and a feedback loop. By analyzing visual, textual, and structural aspects of emails, websites, and URLs, organizations can significantly reduce their susceptibility to phishing attacks and enhance their overall cybersecurity posture. This dynamic and adaptive defense against evolving threats offers a dynamic and adaptive defense against phishing threats [9] [10] [11] [8].

1.2 Visual similarity to detect phishing websites.

Involving ML to detect phishing websites based on visual features uses various visual attributes, such as logos, images, layouts, fonts, and color schemes, to distinguish between legitimate websites and phishing replicas. The process involves data collection, feature extraction, model training, real-time scanning, decision making, and alert or block systems. Visual similarity analysis detects phishing websites in their early stages, even before they become widely recognized as threats. It is less susceptible to obfuscation, as it focuses on the visual aspects of a webpage. It is user-friendly, as users do not need to understand the technical details of phishing detection. Machine learning models for visual similarity analysis can continually improve over time by regularly updating the dataset and retraining the models with new examples of phishing attempts. This approach adds an extra layer of protection, helping organizations and users stay one step ahead of cybercriminals and safeguarding sensitive information from falling into the wrong hands [8]. Though many have utilized machine learning with visual similarity to detect phishing websites no one have tried using graph neural network for detecting a website. In the next part the previous work done on this will be discussed.

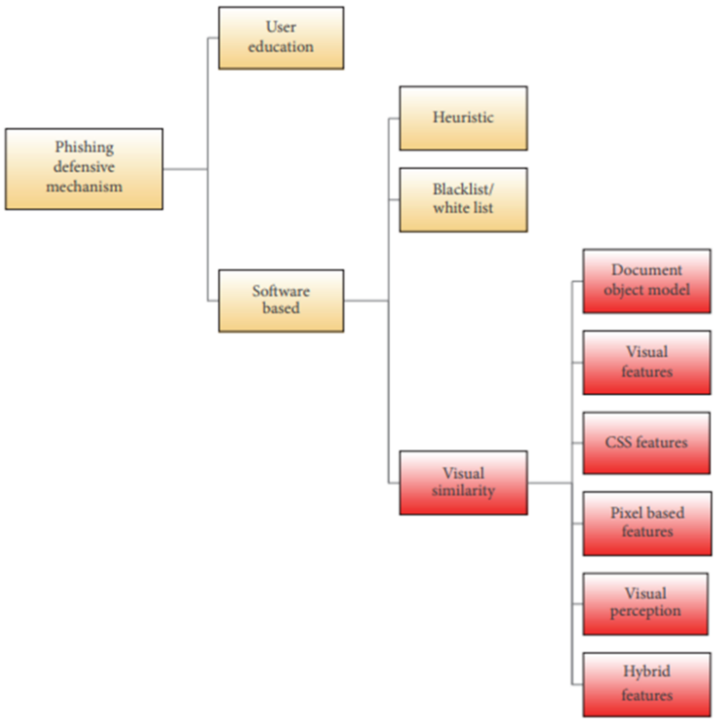


Figure 0.1Visual similarity-based approaches

.

Visual similarity based previous work.

Eric Medvet et al. used a visual similarity-based approach to evaluate a webpage's visual attributes, extracting features like layout, color, and texture from a suspect webpage and reference web pages. The similarities were used to determine if the webpage was real or phishing [13].

In 2017, Igino Corona et al. introduced a method called "DeltaPhish" that can identify HTML code and visual changes between pages. Tested on over 5500 webpages, they claimed it could recognize 70% more. They created a dataset with 5511 distinct webpages and 1012 phishing pages, using Histogram of oriented Gradients and color histograms for image classification [14].

In 2018, F.C. Dalgic's study on phishing websites used a novel approach that combined machine learning with visual analysis. They used compact visual descriptors like JCD, SCD, CEDD, CLD, and FCTH to identify visual patterns associated with color and edges. The researchers used support vector machine and random forest models to classify photos, achieving an F1 score of 90.5 using SCD [15].

Sahar Abdelnabi et al. have developed a phishing detection framework called "VisualPhishNet" using Convolutional Neural Network (CNN) to identify zero-day phishing websites. The model classifies URLs not included in the training dataset and provides a new dataset called "VisualPhish" for visual phishing detection. The model uses computer vision and natural language processing to identify website elements like picture content, text layout, and code structure. It has a high detection rate of 96.42% on over 4,500 phishing websites [16].

Saad Al-Ahmadi et al. developed a method to detect phishing websites using both website pictures and URLs. They used two convolutional neural networks and integrated them for a 99.7% accuracy. Future work aims to optimize performance by determining URL length and screenshot size [19].

Data sets

Different research datasets have been used to analyze phishing activities. VisualPhish [16] contains 9363 screenshots of PhishTank phishing pages targeting 155 websites, while DeltaPhish [14] consists of one phishing page and safe websites from hijacked hosting websites. Phish-IRIS [15] dataset targets 14 websites and a "other" class from the top 300 Alexa websites. CityU database has 5711 phishing pages for every eight legitimate pages, and the dataset used in [18] has 48 different logo classes across 21 well-known websites, totaling 538 logos.

Graph Neural Networks for computer vision tasks

GNNs are neural networks that use graph data as input, capable of managing complex object relations in non-Euclidean data. They typically have a message-passing architecture (MPNN), a graph-specific generalization of convolution neural networks (CNN), accepting input in the form of a directed or undirected graph [24]. The 2022 study "Vision GNN: An Image is Worth Graph of Nodes" introduces a Graph Neural Network (GNN) architecture specifically designed for image recognition tasks, transforming an image into a graph of nodes representing the pixels of the image [25]. The proposed Vision GNN architecture depicts an image as a graph of nodes, with each node representing a local image patch and edges connecting neighboring patches. The GNN analyses the graph by sending messages to update node representations and aggregate node information to build a final image-level representation for classification.

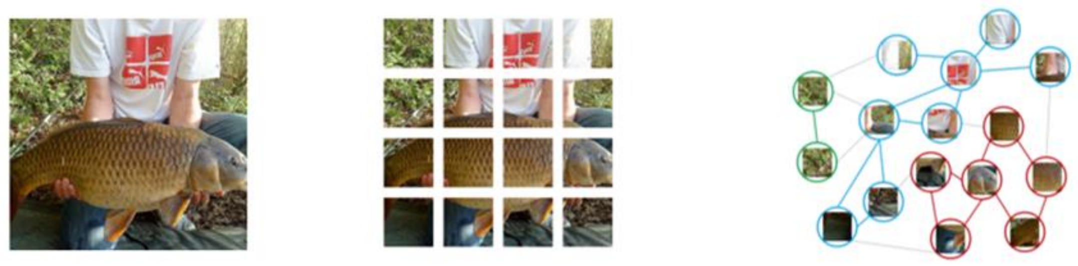


Figure 0.2The steps of how a graph is constructed [12].

In the vision GNN [25] the authors have built two kinds of architectures called isotropic architecture and pyramid architecture to provide a comprehensive comparison with different neural network types. They constructed four different models with varying numbers of hyperparameters for each architecture: tiny, small, medium, and large.

1.3 Website features to detect phishing sites.

Website features are vital elements that facilitate user interaction and functionality on a webpage. Nevertheless, an excess of features can lead to complexity and user bewilderment. Recognizing the value of specific features and discerning those that may be superfluous is pivotal in crafting a compelling and practical website. These features encompass both navigational aspects like content listings and menu bars, as well as service-oriented functionalities like inquiry forms or cost estimators. Depending on the nature of their offerings, businesses may incorporate diverse features, such as enabling menu downloads or furnishing fare calculators for pricing details. In summary, a comprehensive grasp of website features is indispensable in the endeavor to fashion an appealing and operational website [1].

In the realm of cybersecurity, the significance of website features cannot be overstated, particularly when it comes to detecting phishing sites. These deceptive platforms often go to great lengths to mimic genuine websites, making it imperative to scrutinize specific attributes closely. One such crucial aspect is the analysis of Request URLs, which assesses the origin of external elements like images, videos, and sounds. Legitimate websites typically source these elements from the same domain. However, if a substantial portion is loaded from disparate domains, it raises a flag of suspicion. Additionally, scrutinizing the URLs associated with anchor tags proves instrumental. Legitimate websites predominantly link to pages within their own domain, whereas an abundance of anchor URLs pointing elsewhere may signify a potential phishing endeavor. Furthermore, delving into the links embedded within <Meta>, <Script>, and <Link> tags provide another layer of insight. A prevalence of links leading to diverse domains within these tags can be a cause for concern. These are just a few examples of how a comprehensive understanding of website features can fortify one's defense against phishing attempts, ensuring a safer online experience for all. [2]

1.4 Email text content to detect phishing websites.

Phishing attacks, a prevalent form of cybercrime, involve deceiving individuals or organizations into revealing sensitive information by masquerading as trustworthy entities. Email remains one of the primary channels for delivering phishing attempts, making the analysis of email text content a critical area of research in the field of cybersecurity. This literature review explores existing studies and approaches related to the use of email text content analysis for the detection of phishing websites. It synthesizes findings, methodologies, and challenges in this domain [1]. Phishing, a term derived from "fishing," aptly characterizes the deceptive nature of these attacks. Just as an angler uses bait to lure unsuspecting fish, cybercriminals employ deceitful emails to trick individuals into divulging sensitive information, such as login credentials, financial data, or personal details. The consequences of falling victim to a phishing attack can range from financial losses to identity theft, not to mention the erosion of trust in digital communication channels [2].

Amid this digital battleground, the analysis of email text content has emerged as a crucial line of defense, offering insights and tools to identify and combat phishing websites effectively. This introduction sets the stage for an exploration of how email text content analysis serves as a potent weapon in the ongoing battle against phishing attacks. Email, a ubiquitous communication tool, serves as the primary vector for phishing attacks. Phishers often masquerade as trusted entities or institutions, sending fraudulent emails that appear genuine at first glance. These emails contain malicious links, enticing recipients to click, or request sensitive information, leading them down a treacherous path. As such, the analysis of email text content assumes a pivotal role in identifying phishing attempts and safeguarding users from falling prey to these nefarious schemes [3].

One-Hot Encoding Naive Bayes, Decision Tree, Random Forest, SVM, and Deep Learning (Convolutional Neural Network) are some of the machine learning and deep learning models that were employed for the classification of phishing emails in [4]. Improved phishing email detection accuracy is the study's main problem. Extracted text-based information from emails, such as the subject, sender, URL, and attachment, were employed in this study. The use of one-hot encoding for text classification is innovative in this study. fraudulent emails.

NLP, SVM, Naive Bayes, Random Forest, Decision Trees, Logistic Regression, Neural Networks, K-Nearest Neighbor, Hidden Markov Models, and other machine learning/deep learning models were utilized in Detection Using Natural Language Processing Techniques: A Literature Survey

[5]. The lack of large-scale, diversified datasets for phishing email detection using NLP techniques is the research topic of this study. Additional study is needed on feature extraction and selection approaches for NLP-based phishing email. URLs, email headers, sender and recipient email addresses, email content, lexical and semantic elements, and many more features were employed in this study. The innovative aspect of this study is the literature review on NLP-based phishing email detection methods.

Novel email spam detection method using sentiment analysis and personality recognition [6] used some various technologies and machine learning/ deep learning models such as NLP, Sentiment Analysis, Personality Recognition, SVM, Naive Bayes, Decision Trees, Random Forest. Research problem of this research is Identifying and addressing the limitations of traditional spam detection methods, exploring new methods to improve detection accuracy. Features of this research used are Bag-of-words, Part-of-speech tags, sentiment features, lexical features, personality features. Novelty of this research is Combining sentiment analysis and personality recognition for spam detection.

Sentiment analysis

As a result of our online activities, we constantly produce enormous amounts of text data in the digital age. The sheer amount of content available is daunting, ranging from news articles to social media posts, product evaluations, and customer comments. This sea of words contains important clues about people's emotions, beliefs, and the motivations behind their manifestations. By automatically extracting and comprehending the sentiment or emotional tone contained in text, sentiment analysis, also known as opinion mining, is a fascinating branch of natural language processing (NLP) that aids us in navigating this linguistic landscape.

In recent years, sentiment analysis has grown in significance and usefulness, altering how companies, academics, and other organizations see and use textual data. We can learn more about societal attitudes, consumer perspectives, market trends, political viewpoints, and much more thanks to it. This thorough investigation goes into the fascinating field of sentiment analysis, including its fundamental ideas, practical applications, difficulties, and most recent developments [16].

Natural language processing (NLP)

Natural language processing (NLP) is the technique of analyzing text to determine the sentiment or emotional tone of the material. Sentiment analysis algorithms can classify content as good, negative, or neutral based on the language and context used. The emotional undertone of the text is identified using an NLP technique.

Natural language processing (NLP) has a specialty called sentiment analysis that focuses on extracting and analyzing sentiments, views, and emotional tones from textual data. Sentiment analysis is also known as opinion mining.

Text is categorized into specified sentiment categories using sentiment analysis, which often includes:

- Positive: Expressions of joy, approval, satisfaction, or positivity.
- Negative: Expressions of discontent, dissatisfaction, criticism, or negativity.
- Neutral: Text that lacks any significant emotional tone, often factual or informational.

Although these are the basic emotion classes, more detailed sentiment analysis may also contain more subtle categories for negative attitudes, such as "strongly positive," "moderately positive," and "weakly positive."

Text preprocessing, feature extraction, sentiment classification, and result interpretation are all steps in the multi-step process of sentiment analysis [17].

Natural Language Processing (NLP) stands out as a cornerstone of human-computer interaction and comprehension in the constantly changing technological world. NLP is a branch of artificial intelligence (AI) that focuses on the use of natural language in communication between machines and people. It aspires to close the communication gap between humans and machines by enabling computers to comprehend, process, and produce human language. This thorough investigation digs into the intriguing realm of NLP, including its background, guiding principles, applications, difficulties, and prospects [18]. The field of NLP in computer science studies the interactions between computers and human languages. Its main goal is to create computational models and algorithms that can process and understand human language in a way that is like how people do. NLP uses a variety of techniques and resources, including machine learning, deep learning, and computational linguistics. NLP can be used for a variety of purposes, including speech recognition, chatbots, virtual assistants, and language translation. Additionally, it is utilized to examine enormous amounts of data and generate insights in industries like finance, education, and healthcare. As the field matures, NLP has the potential to alter how we communicate with machines and the outside world [19].

A multidisciplinary area that combines computer science, artificial intelligence, linguistics, and cognitive psychology is known as natural language processing (NLP). NLP's primary goal is to make it possible for machines to effectively comprehend, interpret, and produce human language. Understanding a language’s structure and semantics as well as the context in which it is used are both necessary for this.

NLP encompasses a wide range of tasks and objectives, including:

1. **Text Analysis:** Analyzing large volumes of text data to extract insights, patterns, and trends. This includes tasks like sentiment analysis, topic modeling, and text classification.
2. **Speech Recognition:** Transcribing spoken language into written text. It is widely used in voice assistants and transcription services.
3. **Language Generation:** Creating human-like text or speech. This includes chatbots, content generation, and automatic language translation.
4. **Language Understanding:** Interpreting the meaning of text or speech. This includes question-answering systems, information retrieval, and machine translation.
5. **Language Generation:** Creating human-like text or speech. This includes chatbots, content generation, and automatic language translation.
6. **Machine Translation:** Translating text or speech from one language to another, such as Google Translate.

Naive Bayes

The Naive Bayes classifier is a dependable workhorse in the fields of machine learning and probabilistic modeling. It is praised for its ease of use, effectiveness, and outstanding results in a range of classification tasks, including text categorization, spam detection, and medical diagnosis. The Naive Bayes method, which has its roots in Bayesian probability theory, is a key component of machine learning practitioners' toolkits. This thorough investigation sets out on a quest to learn about the Naive Bayes classifier's foundations, uses, variations, and practical effects.

A series of probabilistic algorithms known as the Naive Bayes classifier uses the Bayes theorem to generate predictions and categorize data. It is especially useful for situations in which we need to sort or classify things into distinct classes based on a collection of features. The Naive Bayes classifier has shown to be remarkably successful in many real-world circumstances despite its simplicity.

Fundamentally, the method calculates the probability of an event occurring in the presence of specific traits or conditions by applying conditional probability and statistical independence assumptions. It is referred to as "naive" because it makes the conditionally inconsistent assumption that the features employed for categorization are independent of each other. However, the algorithm can operate effectively and frequently delivers remarkably accurate results because to this simplifying assumption.

Naive Bayes is the name of a probabilistic machine learning model for classification. Naive Bayes can be used to predict the likelihood that an email is a phishing email based on information extracted from the email text, such as the sender address and message content. Naive Bayes is a well-liked option for phishing email detection since it is a straightforward and effective model that performs well with tiny datasets.

Although the Naive Bayes classifier's fundamental ideas stay the same, there are multiple iterations of the method that can be used to different kinds of data and applications. Three common types of Naive Bayes classifiers are:

1. Multinomial Naive Bayes:

The Multinomial Naive Bayes classifier is commonly used for text classification tasks, such as spam detection and document categorization. It models the distribution of word frequencies in documents and is well-suited for datasets where the features represent the counts or frequencies of items (e.g., word counts).

2. Gaussian Naive Bayes:

The Gaussian Naive Bayes classifier is applied to datasets where the features follow a Gaussian (normal) distribution. It is suitable for continuous or real-valued features and is often used in applications like medical diagnosis and image classification.

3. Bernoulli Naive Bayes:

The Bernoulli Naive Bayes classifier is primarily used for binary classification problems, where features represent binary outcomes (e.g., presence or absence). It is well-suited for tasks like spam detection, sentiment analysis, and information retrieval.

The Naive Bayes classifier possesses several strengths that make it a valuable choice in many situations:

1. Simplicity:

Naive Bayes is conceptually straightforward and easy to implement. Its simplicity makes it a suitable choice for quick prototyping and baseline models.

2. Efficiency:

The algorithm is computationally efficient and scales well to large datasets. Training and prediction are typically fast, making it suitable for real-time applications.

3. Minimal Data Requirements:

Naive Bayes classifiers can perform well even with relatively small datasets, making them useful in scenarios where collecting extensive labeled data is challenging.

4. Good Generalization:

Despite its simplicity and the "naive" assumption of feature independence, Naive Bayes often generalizes well and delivers competitive performance in practice.

5. Interpretability:

The probabilistic nature of Naive Bayes makes it easy to interpret and explain predictions, which is valuable in domains where model transparency is crucial.

The Naive Bayes classifier has left an indelible mark on various industries and applications. Here are some instances of its real-world impact.

1. Email Filtering:

Naive Bayes is a key component of spam filters, helping users avoid the deluge of unwanted emails and ensuring that important messages reach their inboxes.

2. Text Analysis:

In the realm of text analysis, Naive Bayes classifiers enable sentiment analysis, topic modeling, and content recommendation, enhancing user experiences in social media, e-commerce, and news platforms.

3. Fraud Detection:

Financial institutions employ Naive Bayes models to detect fraudulent transactions and activities, safeguarding customers and minimizing financial losses.

4. Language Identification:

Naive Bayes classifiers help identify the language of text data, enabling multilingual content filtering, translation, and internationalization.

5. Personalization:

E-commerce platforms use Naive Bayes-based recommendation systems to personalize product suggestions, increasing customer engagement and sales.

SVM – Support Vector Machines (SVMs)

Support Vector Machines (SVMs) are imposing cornerstones of classification and regression tasks in the enormous field of machine learning techniques. SVMs are highly regarded for their dependability, adaptability, and capacity for handling complex data, and they have earned a substantial place in machine learning practitioners' toolkits. This in-depth investigation examines the foundations, mathematics, uses, variations, and practical implications of Support Vector Machines. models of supervised learning for regression and classification. SVMs can categorize emails as legitimate or phishing using data gathered from the email text, such as the sender address and message content. Because they perform well with high-dimensional datasets, SVMs are a popular option for phishing email detection [20].

1.5 URL to detect a phishing site.

Extensive research efforts have been dedicated to the domain of phishing website detection, reflecting the critical need to protect users from online threats. The literature in this field spans a range of methodologies and techniques, including rule-based heuristics, machine learning algorithms, and behavioral analysis. Several notable contributions have enhanced the accuracy and efficiency of phishing detection methods. However, challenges persist in ensuring that detection mechanisms remain effective against the dynamic and evolving landscape of phishing attacks.

Phishing Detection Methods

Phishing detection is a critical component of online security, and various methods have been explored to identify and mitigate phishing threats. While rule-based heuristics offer simplicity and efficiency, they often rely on fixed patterns and may struggle to adapt to novel phishing techniques. In contrast, machine learning-based algorithms, including Random Forest and SVM, have shown promise in detecting phishing websites. These algorithms leverage the power of data-driven decision-making, allowing them to learn from large datasets and adapt to emerging threats. Feature engineering, a crucial step in machine learning-based approaches, involves selecting and extracting relevant attributes from URLs and web content, enabling the algorithms to make informed decisions about the legitimacy of websites. [1]

Data Sources

The effectiveness of phishing detection models hinges heavily on the quality and diversity of the data used for training and evaluation. Publicly available datasets, such as the PhishTank dataset, have been instrumental in advancing research in this field. These datasets encompass a wide range of phishing and legitimate URLs, providing a solid foundation for model development. However, to address the ever-evolving threat landscape, some researchers have turned to proprietary datasets from cybersecurity organizations. These datasets often contain more recent and targeted examples of phishing attacks, offering a valuable resource for fine-tuning and evaluating detection models. Access to such proprietary data sources, though, can be limited.

The Evolution of Phishing Techniques

Phishing attacks have undergone a significant evolution over the years, mirroring changes in technology and communication. Early phishing attempts primarily relied on generic email-based scams, targeting a broad audience with promises of financial gain or urgent requests for personal information. In contrast, modern phishing campaigns are marked by their sophistication and precision.

Spear phishing is a notable advancement in phishing techniques. This targeted approach involves thorough research on potential victims, enabling attackers to craft highly convincing messages that often impersonate trusted entities. The personalization and specificity of spear phishing make it a formidable threat.

Vishing, or voice phishing, takes advantage of voice communication to deceive victims. Attackers may impersonate legitimate organizations or individuals over the phone to extract sensitive information or initiate fraudulent transactions [2].

Smishing, a fusion of SMS (Short Message Service) and phishing, leverages text messages to deceive recipients. These messages may contain enticing offers or urgent instructions, compelling users to click on malicious links or disclose personal information [2] [3].

Pharming attacks are particularly insidious as they manipulate the Domain Name System (DNS) to redirect users to fraudulent websites. Even when users enter legitimate URLs, they are directed to malicious sites. Detecting pharming attacks requires vigilance beyond typical phishing techniques.

Challenges and Limitations

Despite the progress in phishing detection, several challenges and limitations persist in this domain:

Evasion Techniques: Phishers are constantly devising evasion techniques to bypass detection mechanisms. These may include obfuscating URLs, employing CAPTCHA challenges, or using encryption to hide malicious content. Staying ahead of these evolving threats is an ongoing challenge [4].

Data Imbalance: Imbalanced datasets, where legitimate URLs significantly outnumber phishing URLs, can lead to biased models. Detection models may become overly conservative, resulting in an increased likelihood of false negatives. Strategies for mitigating data imbalance, such as oversampling or synthetic data generation, are essential.

Zero-Day Attacks: Zero-day phishing attacks, which are new and previously unseen threats, pose a substantial challenge. Traditional detection models may struggle to identify such threats, highlighting the need for adaptive and real-time approaches.

Usability: The usability of phishing detection tools, including web extensions, is crucial. While accurate detection is paramount, striking a balance between alerting users to potential threats and minimizing false positives is essential to ensure a positive user experience. Tools should be user-friendly and unobtrusive.

2. Research Gap

2.1 Visual similarity to detect phishing websites.

A notable gap in research exists regarding the use of visual features for detecting phishing websites using machine learning techniques. Despite various methods like decision trees, K-Nearest Neighbor, Support Vector Machine, Gaussian Naive Bayes classifiers, and Convolutional Neural Networks (CNN), there is a need for a method capable of detecting phishing websites beyond existing datasets. Model development for edge computing is crucial for integrating phishing detection models into practical applications. However, no mobile applications have been developed yet, highlighting the need for further research in this area to enhance real-time detection of phishing threats.

The use of Graph Neural Networks (GNN) for identifying phishing websites based on screenshots of web pages does not appear to be supported by any studies at this time. Given that GNNs have demonstrated promise in several applications involving machine learning and graph-based data processing, this exposes a sizable gap in the state-of-the-art research landscape. It could be beneficial for scientists to look at how GNNs might be relevant in this situation. GNNs are renowned for their ability to capture complex dependencies and links within graph-structured data, making them a potentially excellent substitute for examining the structural and aesthetic elements of web pages. By creating special GNN-based models specifically designed for this issue, researchers might pave the way for more sophisticated and precise phishing detection methods that incorporate picture analysis.

The researchers utilized the website's source code, logo, content, and photos as features, which are shown in the below table. They also considered the website's general design and aesthetic. Convolutional neural networks had been used in several research. The application of graph neural networks for phishing website detection with visual cues has not been studied.

1Summary table

Work	Research Gap
[13]	Limited dataset was used to do testing. Lack of real-world testing. Not defined how the features were extracted. Have not compare other techniques.
[14]	Testing was done using their own custom dataset with limited data.
[16]	Their data set was collected for a specific time so that the performance of the model for new sites is a problem. They have used a specific dataset and the performance are done under controlled situations. Other methods can be used to identify phishing websites using visual features to improve accuracy.
[17]	Used datasets are specific and more generalized datasets are needed for testing. Only have used three types of CNN and can do more testing with CNN and ML. Did not address how they going to detect a website.
[18]	They have only focused on detecting logos and the dataset is a custom dataset that they made for companies in south Asia region. They have not focused on other features that can use as visual similarity based in a website.
[19]	There is lack of ability to detect new websites. More machine learning techniques could have used for testing.

2.2 Website features to detect phishing websites.

The study thoroughly explores the effectiveness of machine learning techniques, encompassing K-Nearest Neighbors (KNN), Support Vector Machines (SVM), and Random Forest, in discerning phishing attempts through website analysis. Nonetheless, a notable research gap persists in elucidating the practical implementation of these algorithms in real-world contexts, notably within a Google Chrome Extension. The study predominantly focuses on technical performance and classification accuracy, inadvertently overlooking critical aspects such as user experience, usability, and potential integration constraints within a browser extension. It is imperative for further research to address these gaps, delving into areas like user interaction, false positive/negative rates, real-time data updates, scalability, privacy safeguards, cross-browser compatibility, user education, and feedback mechanisms. By bridging these research gaps, a more holistic understanding of the practical implications and challenges in deploying machine learning-based phishing detection algorithms as web browser extensions will be attained, ultimately fortifying user protection and instilling confidence in online security tools [3] [4] [5] [6].

2Research GAP for web features related work

[3]	• User Interaction & Interface Design: Investigate user interactions and enhance interface design for improved usability and trust. • Real-time Data Updates: Explore methods for timely adaptation to evolving phishing techniques and ensure dataset relevance. • Privacy & Security Measures: Define and implement robust measures to safeguard user data and privacy during URL processing.
[4]	• Dataset Representation: The study's evaluation of PhishChecker relies on a small dataset of 100 URLs, potentially lacking representation of the diverse phishing landscape. Future research should test on a larger, more varied dataset to assess its robustness. • Automatic Web Page Detection: The study briefly mentions future research on this aspect but lacks detail. Further investigation into automatic detection mechanisms and their integration into PhishChecker is needed. • Web Browser Compatibility: The study doesn't discuss PhishChecker's compatibility with various web browsers. Research should address its performance across different browsers due to diverse user preferences.

[5]	• Integration of Detection Methods: While the study covers various phishing detection techniques, there's limited exploration of their effective integration. Research should develop hybrid approaches that combine these methods for a more robust system. • Evaluation and Comparison of Methods: The study hints at increased accuracy but lacks a comprehensive evaluation of different detection methods. Future research should conduct extensive comparative studies to identify the most effective techniques. • Adaptive and Real-Time Detection: Research should investigate how phishing detection methods can adapt in real-time to emerging threats, ensuring they remain effective in dynamic attack scenarios. • Privacy Considerations: Research should delve into the privacy implications of phishing detection methods and explore techniques that protect user privacy while effectively identifying phishing attempts.
[6]	• Interpretability: The study doesn't discuss the interpretability of features or classifiers' decision-making. Future research could focus on enhancing model interpretability to provide insights into classification. • Real-time Detection: Research gaps exist in exploring real-time or near-real-time phishing detection methods for dynamic online environments. • Behavioral Analysis: Investigating the integration of behavioral analysis, considering user interactions and website behavior over time, as a complement to static feature-based detection is an opportunity.

2.3 Email text content to detect phishing websites.

These are some details of existing phishing email detection system research.

A Hybrid Machine Learning Approach to Detect Phishing and Spam Emails: Phish Responder NLP, Logistic Regression, Random Forest, and SVM are a some of the machine learning/deep learning models that were employed in [7]. The identification of spam and phishing emails is the research problem in this study. Seventy content and metadata elements from this study were employed, including sender and recipient details, email structure, language, sentiment, and keywords. This research is new in that it uses a hybrid machine learning strategy that combines content and metadata characteristics, ranks feature relevance to discover critical features, and applies model ensembles.

Semantic Feature Selection for Text with Application to Phishing Email Detection [8] made use of a variety of deep learning/machine learning models, including NLTK, scikit-learn, Logistic Regression, Decision Tree, Random Forest, Naive Bayes, and SVM. Improving the effectiveness and efficiency of phishing email detection is the research problem of this study. Email header features, lexical features, content-based features, and semantic features based on LSA are the features of this research that were utilized. The research's novelty is to enhance the detection of phishing emails, a semantic feature selection method based on LSA was proposed.

Using a variety of machine learning/deep learning models, including Text Mining, Feature Extraction, SVM, Random Forest, and Naive Bayes, researchers [9] were able to identify phishing emails. Phishing assaults are an increasing threat for both individuals and businesses, and there is a need for effective and quick solutions to identify them. Text-based methods can be helpful in detecting phishing emails, but for the best results, feature selection and classification models must be carefully considered. Semantic and syntactic elements from this research, such as URL analysis, header analysis, content analysis, and linguistic aspects, are combined. Mutual information and chi-squared approaches are used to choose the features. This study's originality is one of its strengths. The innovative aspect of this research is a feature analysis technique that combines syntactic and semantic data to identify phishing emails. Additionally, it evaluates the effectiveness of several machine learning models.

Support Vector Machines, Naive Bayes, Random Forest, Logistic Regression, and Voted Perceptron were some of the machine learning/deep learning models utilized in Phishing Detection in E-mails using Machine Learning [10]. Classify emails as either phished or ham based on attributes taken from a dataset of such emails is the research problem under consideration. Link-based, tag-based, and word-based features totaling nine features were used in this study. This research is unusual in that it uses a variety of features while using the fewest possible features to create a system that is more accurate.

NLP, Recurrent Convolutional Neural Network (RCNN), and Attention Mechanism were some of the machine learning/deep learning models used in Phishing Email Detection Using Improved RCNN Model With Multilevel Vectors and Attention Mechanism [11]. The goal of this study is to identify phishing emails, represent email content with useful features, and address the imbalance issue in the dataset. Word embedding vectors, multilevel vectors, term frequency-inverse document frequency (TF-IDF), bag-of-words, sender and recipient email addresses, email subject, and email attachment are among the content elements of this study that were utilized. The novel aspect of this research is the improved RCNN model with attention mechanism that is proposed for phishing email detection. Multilevel vectors are also included to capture the semantic meaning of email content, and both email content and header data are used for classification.

Machine learning/deep learning methods like Logistic Regression, Random Forest, and SVM were employed in an intelligent classification model for phishing email detection [12]. The identification of phishing emails is the study's main difficulty. The features employed in this study include N-gram and URL-based features. Combining N-gram and URL-based characteristics and doing a comparison analysis of various ML models are novel aspects of this research.

Studying machine learning techniques for phishing email detection based on URL parameters [13] employed a variety of technologies and machine learning/deep learning models, including gradient boosting, decision trees, random forests, logistic regression, and neural networks. The identification of phishing emails is the study's main difficulty. URL parameters, subject, body, sender, attachments, and hyperlinks are some of the research's features used. The use of URL parameters for feature extraction is innovative in this study.

Machine learning/deep learning models like Random Forest Classifier and Naive Bayes Classifier were utilized in Phishing Email Detection Using Techniques [14]. Using machine learning techniques, this study's research problem is to develop an effective and efficient phishing email detection system. The features of this study that were used were those based on URLs (such as domain age, domain registration period, and URL length), text (such as frequency of certain words, presence of certain phrases), headers (such as sender's email address, subject line), and attachments (such as the presence of executable files). The unique aspect of this research is the use of multiple features for phishing email detection. Achieved great accuracy in detecting phishing emails by conducting trials on a real-world dataset of authentic and phishing emails.

SVM, Logistic Regression, and Naive Bayes are just a few of the machine learning/deep learning models that were employed in Phishing Email Detection Using Robust NLP Techniques [15]. The goal of this study is to evaluate the robustness of models against adversarial attacks and to detect phishing emails using NLP techniques. Word embeddings, n-grams, Part-of-Speech tags, named entities, semantic role labeling, sentiment analysis, length, and frequency-based features are some of the characteristics employed in this study. The use of lexical and semantic variables, as well as the assessment of the models' resistance to adversarial attacks, are novel aspects of this research.

2.4 URL to detect a phishing site.

As we embark on the journey of URL-based phishing detection, it's essential to recognize the research gaps that our project aims to address. These gaps underscore the need for innovative solutions in the field of online security.

Our project takes on the challenge of Adaptive Phishing Detection for Web Extensions, where existing tools often struggle to keep pace with rapidly evolving phishing techniques. We propose a web extension that continually learns and adapts to emerging threats and user behavior, ensuring robust protection against novel phishing attacks. Furthermore, we tackle the Integration of Machine Learning in User-Friendly Tools, an area where a balance between advanced detection algorithms and user-friendly interfaces is crucial. Our web extension seamlessly incorporates machine learning, providing users with advanced URL analysis while maintaining an intuitive and trust-inspiring design.

We also recognize the importance of Behavioral Analysis and Real-Time Threat Intelligence. These emerging trends hold great potential but need effective integration into web extensions. Our project aims to leverage behavioral analysis and real-time threat intelligence to enhance detection accuracy and provide users with up-to-date protection.

In the realm of Usability and User Education, our focus is on creating a user-friendly interface that empowers users to trust and interact with our web extension effortlessly. We also plan to include educational components within the extension to equip users with the knowledge to identify phishing attempts independently.

Lastly, our project emphasizes the Evaluation of Web Extension Effectiveness, recognizing the need to measure its real-world impact on user security and browsing experience. Through rigorous assessments, we intend to provide valuable insights into the practical utility of our phishing detection solution. By addressing these research gaps, our project seeks to make meaningful contributions to the field of URL-based phishing website detection, ultimately enhancing online security for users.

3. Research problem

3.1 Visual similarity to detect phishing websites.

The primary research challenge is to identify phishing websites using screenshots and GNN to assess visual similarity. The next research question is how to improve existing datasets with those features, or, if datasets are insufficient, how to collect data with the new proposed features. The most prominent visual features include logos, texts, element placement, distances between elements of the site, and so on. Investigating ways to represent a webpage's visual elements as a graph is another research objective. Another research goal is to investigate how to express the visual aspects of a webpage as a graph. Either use the complete image of the web page as pixels and represent them as a graph or partition the web page into parts and represent them as graphs, and then experiment to see which option provides the most accuracy. Then for each model the accuracy should be calculated and need to get the best model with the highest accuracy. The model must then be trained using methods like hyperparameter adjustments. We also need to know if the GNN methodology will be more accurate and computationally efficient than prior CNN-based methods. In order to safeguard user privacy as much as possible, the last research goal is to figure out how to integrate the developed model into the mobile app and install it on the device itself rather than running it on a cloud server.

3.2 Website features to detect phishing websites.

The persistent threat of phishing attacks poses a significant security risk to both individuals and enterprises, necessitating the development of more accurate and efficient detection techniques. Contemporary anti-phishing systems often rely on email and website content analysis, which may fall short against sophisticated attackers. This research project introduces an innovative approach: harnessing website features, HTML and JavaScript-based attributes, and abnormal behavior-based indicators for real-time phishing attack detection.

The central goal of this research is to design, implement, and assess machine learning-based algorithms tailored explicitly for identifying phishing attacks by analyzing website attributes, HTML and JavaScript features, and unusual online behavior. This study seeks to address critical questions in the realm of phishing detection: What website characteristics, HTML, and JavaScript-based attributes, and abnormal online behaviors signify phishing attempts? How can these diverse features and behaviors be effectively integrated into machine learning models for robust and resource-efficient phishing detection? How can these models be operationalized in practical settings to preemptively detect and mitigate phishing attacks?

The research project's core involves the creation and validation of machine learning models precisely engineered for phishing attack detection by scrutinizing website elements, HTML and JavaScript attributes, and identifying unusual online activities. Model training will be based on a meticulously labeled dataset, encompassing instances of both phishing and legitimate web interactions, thus ensuring model accuracy and effectiveness.

The research endeavors to pinpoint the most discriminative website features, HTML and JavaScript attributes, and abnormal online behaviors that distinguish phishing attempts from legitimate activities. The ultimate objective is to construct accurate and efficient machine learning models that can be effectively deployed in real-world scenarios for real-time phishing attack detection and prevention. By leveraging website attributes and abnormal behaviors, this project aims to fortify cyber defenses against these pervasive and evolving cyber threats.

3.3 Email text content to detect phishing websites.

- ❖ How to address the issue of imbalanced datasets, which can affect the accuracy of the model.

One of the ongoing problems in email security and phishing detection is handling imbalanced datasets. When the proportion of genuine emails in a dataset is significantly higher than the proportion of phishing emails, the dataset is said to be unbalanced. Due to their bias toward the majority class (genuine emails), machine learning models may perform much worse because of this imbalance, making them less accurate at spotting phishing emails. Therefore, resolving this problem successfully is essential to ensuring strong and dependable email security systems.

- ❖ How to use large datasets.

When developing precise and dependable email security models, the dataset size is crucial. Large email datasets are difficult to get and use effectively, though. It is difficult to gather diverse and vast email data, ensure data quality, and optimize model training on enormous amounts of data. Nevertheless, designing email security systems that can efficiently generalize and identify changing phishing techniques requires vast datasets.

- ❖ How to use sentiment analysis to detect phishing emails.

To trick email users, phishing attacks frequently include psychological manipulation, including emotional triggers. The process of assessing the emotional tone or sentiment expressed in text is known as sentiment analysis, and it is a subfield of natural language processing. To better detect phishing emails that use emotional manipulation, it is challenging to integrate sentiment analysis into email security systems efficiently.

- ❖ How to use Naive Bayes algorithm to train the model.

Building a successful email security model depends on choosing the right machine learning algorithm. Email security can benefit from the Naive Bayes algorithm, which is renowned for being straightforward and effective in text categorization applications. The difficulty, though, is in customizing and improving this method to train email security models.

3.4 URL to detect a phishing site.

In today's hyperconnected digital world, phishing attacks represent a persistent and growing threat to online security. The research problem at hand is exacerbated by the ever-increasing complexity and adaptability of phishing techniques. Cybercriminals continuously devise new methods to deceive users and compromise their sensitive information, making the detection of phishing websites an ongoing challenge. Traditional, rule-based heuristics and static detection methods often fall short in accurately identifying these malicious sites, leading to an alarming rise in successful phishing attacks.

Moreover, users themselves struggle to discern between legitimate websites and cleverly disguised phishing pages, underscoring the pressing need for a user-friendly solution. The research problem, therefore, centers on the development of an adaptive, machine learning-driven web extension capable of URL-based phishing website detection in real-time, relying on a feature-based analysis approach. This solution must evolve alongside emerging threats, seamlessly integrate machine learning algorithms like the Random Forest model, incorporate real-time threat intelligence, and provide user education within its interface.

Furthermore, the research problem extends to the assessment of the extension's effectiveness and usability in real-world scenarios, considering its URL-based analysis approach. It requires a comprehensive evaluation to measure its practical utility and its impact on enhancing online security. Ultimately, the research problem encompasses the imperative to bridge the gap between evolving phishing techniques and the means to thwart them effectively through URL-based analysis, ensuring a safer online experience for users in an ever-evolving cyber-threat landscape.

4. Research Objectives

Main Objective

To create a browser extension that can alert users to phishing websites and an optimized model for mobile applications. One way to do this is by using the URL, website features, visual resemblance, email header, and content. The features of the URL can be used to determine whether the website is trustworthy. In a visual-based strategy, visual components are employed to identify the website. Grammar and spelling mistakes in emails sent by attackers with urgent requests can be found using text analysis. You may check the email headers to see if the sender's address is legitimate. Finally, to assess if a website is phishing or not, aspects of the website will be utilized, such as address bar-based features, aberrant behavior features, and domain-related features. The finished solution will be able to identify phishing websites and alert the user to them based on the content of emails, website URLs, visual resemblance characteristics, and website features utilizing all of the techniques mentioned above.

Specific Objectives

4.1 Visual similarity to detect phishing websites.

The main goal of this project is to build a solid deep learning model that uses GNNs to identify phishing websites. To identify possible risks, this approach will depend on the visual resemblance of websites. In addition to developing the model, to guarantee efficiency and efficacy, the model will be compared to other phishing detection techniques and its inference time will be examined. To improve the model's ability to recognize new, undiscovered phishing threats in addition to recognizing existing phishing websites from the training dataset. This involves developing systems for constant adaptation. We will concentrate on further optimizing the model's size and inference time because it is crucial to deploy the model in contexts with limited resources. Particularly for edge computing applications with constrained processing resources, this optimization will be developed.

We want to incorporate the created deep learning models into a mobile application to make this technology accessible and user-friendly. Users will be able to use this software to verify the integrity of websites effortlessly and swiftly from their cellphones, improving their online security.

4.2 Website features to detect phishing websites.

- Emphasize website features, HTML and JavaScript attributes, and abnormal online behavior indicators.
- Identify website characteristics, HTML, and JavaScript attributes, and abnormal behaviors signifying phishing attempts.
- Effectively integrate diverse features and behaviors into machine learning models.
- Create robust and resource-efficient models for phishing detection.
- Deploy developed models in practical settings.
- Preemptively detect and mitigate phishing attacks in real-world scenarios.
- Create and validate machine learning models for phishing detection.
- Use meticulously labeled datasets with both phishing and legitimate interactions.
- Construct accurate and efficient machine learning models.
- Deploy models for real-time phishing attack detection and prevention.

These objectives collectively aim to advance the field of phishing attack detection and contribute to more effective cybersecurity measures.

4.3 Email text content to detect phishing websites.

Specific Objective

- Phishing email detection with sentiment analysis

This research's specific goal is to build a solid machine learning model that can recognize phishing emails by incorporating sentiment analysis. This goal highlights the project's main goal, which is to improve email security by using sentiment analysis techniques to spot phishing emails that take advantage of emotional manipulation and triggers.

Sub Objective

- To develop a machine learning model that can detect phishing emails using sentiment analysis.

The first sub-objective serves as the study's starting point. It entails building a machine learning model specifically designed for phishing email detection. This model is distinctive in that it incorporates sentiment analysis, allowing it to evaluate the emotive content and tone of emails. The model will learn to recognize subtle emotional cues that are frequently exploited by cybercriminals to fool victims by utilizing natural language processing (NLP) techniques. Data collection, preprocessing, feature engineering, and the creation of a sentiment analysis algorithm within the model are all part of this sub-objective.

- Improve the model to detect text phishing or not.

The capabilities of the first sub-objective are enhanced by the second sub-objective. It's important to improve the model's capacity to determine if an email is phishing or not based on the textual content, even while sentiment analysis can spot emotional manipulation. This sub-objective entails the development of a powerful phishing email classification system, the improvement of text-based characteristics, and the use of machine learning techniques tailored for text classification. The model tries to develop a comprehensive knowledge of email content by integrating sentiment analysis with text-based analysis.

- Display a warning message when detecting a phishing email.

The practical use of the phishing detection model is the main goal of this sub-objective. The model will send the email recipient a warning message if it correctly recognizes a phishing email. By alerting the user to the potential hazard and urging caution, the warning message will work as an actionable response. This warning message system's development includes message content production, user interface design, and the incorporation of real-time email scanning capabilities. The objective is to provide a user-friendly and educational warning system that equips users to respond appropriately when faced with phishing attempts.

- To integrate the developed models to a mobile application.

The integration of the created phishing detection models into a user-friendly mobile application is the final sub-objective. This objective seeks to provide users with easy access to phishing detection capabilities on their mobile devices in recognition of the value of ease and accessibility. This sub-objective calls for the creation of mobile applications, user interface design, model deployment, and smooth email client interaction. The goal is to give customers a strong tool that improves their email security while they are on the road, enabling them to recognize and react to phishing attacks straight from their smartphones.

4.4 URL to detect a phishing site.

- Develop an Adaptive Phishing Detection Web Extension: The primary objective of our project is to design and implement a web extension capable of adaptive phishing detection. This extension will continuously learn from evolving phishing threats and user interactions to enhance its detection capabilities over time.
- Integrate Machine Learning Algorithms: We aim to incorporate machine learning algorithms into the web extension for URL analysis. This involves the selection and extraction of relevant features from URLs and web content, enabling the extension to make informed decisions about the legitimacy of websites.
- Seamless User-Friendly Interface: Our project's goal is to create a user-friendly interface for the web extension. Users should be able to trust and interact with the tool effortlessly, ensuring a positive user experience while staying protected from phishing threats.
- Real-Time Threat Intelligence Integration: We intend to integrate real-time threat intelligence feeds to keep the extension updated with the latest information on emerging phishing threats and attack techniques, enhancing its accuracy and responsiveness.
- Feature Extraction: The process of feature extraction is vital to our project's success. We will extract and transform relevant attributes from URLs, including domain-related features, URL length, character frequency, domain age, DNS records, HTTP tokens, subdomain count, TLD features, URL redirection, and more. These features will serve as inputs for our machine learning algorithms.
- Usability Enhancement and User Education: We prioritize usability by designing an intuitive interface and minimizing false positives. Additionally, we plan to incorporate educational components within the extension to empower users to recognize phishing attempts independently.
- Comprehensive Evaluation: Our project seeks to conduct comprehensive evaluations to assess the effectiveness and usability of the web extension in real-world scenarios. These evaluations will provide insights into the extension's practical utility and user impact.

METHODOLOGY

5. METHODOLOGY

A. MOBILE APPLICATION

1 VISION GNN based model integrating to mobile app.

1. The VISUALPHISHNET Dataset which contains screenshots of website that contains both legitimate and phishing websites was used.
2. The dataset was preprocessed by Horizontal flip and vertical flipping were used as image augmentations to reduce over-fitting.
3. The experiments were conducted with graph neural network four models.
4. Models were trained using extracted visual features to distinguish between legitimate and phishing websites.
5. The performance of the trained models was evaluated using appropriate metrics such as accuracy, precision, recall, and F1-score.
6. The model with the most accuracy was selected out of the four models.
7. Optimized the developed model using model pruning, weight clustering and quantization.
8. Then the trained model was integrated with a mobile app using appropriate programming languages and tools such as Python and Native Android.
9. User-friendly interface was designed for the mobile app that displays the classification results and provides appropriate feedback to the user.
10. Test the integrated system on a sample dataset of website screenshots and validate its performance using appropriate metrics.
11. Regularly update the model with new datasets and features to improve its accuracy and effectiveness in detecting phishing websites. Also, maintain the app to ensure its functionality and compatibility with the latest mobile devices and operating systems.

Each of the four models were trained by using the VISUALPHISHNET data set and calculate the accuracy.

Model definition

```
def _cfg(url='', **kwargs):
    return {
        'url': url,
        'num_classes': 2, 'input_size': (3, 224, 224), 'pool_size':
None,
        'crop_pct': .9, 'interpolation': 'bicubic',
        'mean': IMAGENET_DEFAULT_MEAN, 'std': IMAGENET_DEFAULT_STD,
        'first_conv': 'patch_embed.proj', 'classifier': 'head',
        **kwargs
    }

default_cfgs = {
    'vig_224_gelu': _cfg(
        mean=(0.5, 0.5, 0.5), std=(0.5, 0.5, 0.5),
    ),
    'vig_b_224_gelu': _cfg(
        crop_pct=0.95, mean=(0.5, 0.5, 0.5), std=(0.5, 0.5, 0.5),
    ),
}


@register_model
def pvig_ti_224_gelu(pretrained=False, **kwargs):
    class OptInit:
        def __init__(self, num_classes=1000, drop_path_rate=0.0,
**kwargs):
            self.k = 9 # neighbor num (default:9)
            self.conv = 'mr' # graph conv layer {edge, mr}
            self.act = 'gelu' # activation layer {relu, prelu,
leakyrelu, gelu, hswish}
            self.norm = 'batch' # batch or instance normalization
{batch, instance}
            self.bias = True # bias of conv layer True or False
            self.dropout = 0.0 # dropout rate
            self.use_dilation = True # use dilated knn or not
            self.epsilon = 0.2 # stochastic epsilon for gcn
            self.use_stochastic = False # stochastic for gcn, True
or False
            self.drop_path = drop_path_rate
            self.blocks = [2,2,6,2] # number of basic blocks in the
backbone
            self.channels = [48, 96, 240, 384] # number of channels
of deep features
            self.n_classes = num_classes # Dimension of out_channels
            self.emb_dims = 1024 # Dimension of embeddings
```

Figure 0.3Model define

- vision_gnn_phishing_pyramid_tiny.ipynb

For this model the blocks and channels that were used are as follows.

self.blocks = [2,2,6,2] # number of basic blocks in the backbone

self.channels = [48, 96, 240, 384] # number of channels of deep features

- vision_gnn_phishing_pyramid_medium.ipynb

For this model the blocks and channels that were used are as follows.

self.blocks = [2,2,16,2] # number of basic blocks in the backbone

self.channels = [96, 192, 384, 768] # number of channels of deep features

- vision_gnn_phishing_pyramid_small.ipynb

For this model the blocks and channels that were used are as follows.

self.blocks = [2,2,6,2] # number of basic blocks in the backbone

self.channels = [80, 160, 400, 640] # number of channels of deep features

- vision_gnn_phishing_pyramid_big.ipynb

For this model the blocks and channels that were used are as follows.

self.blocks = [2,2,18,2] # number of basic blocks in the backbone

self.channels = [128, 256, 512, 1024] # number of channels of deep features

The accuracy and the implementation of the mobile application will be discussed in the results section. Once the training was done, we were able to find out that the highest accuracy rate is for small model. The accuracy was calculated with Accuracy = C/N equation. For the small model with the highest accuracy hyperparameter tuning was conducted by hanging following parameters.

- optimizer - AdamW

- starting learning rate - 0.01
- learning rate decay - cosine decay

Pytorch model was converted in to pytorch mobile version using the following code snippet.

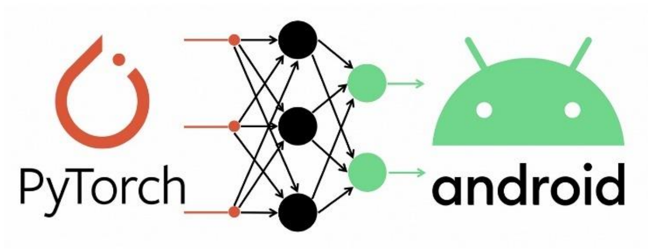
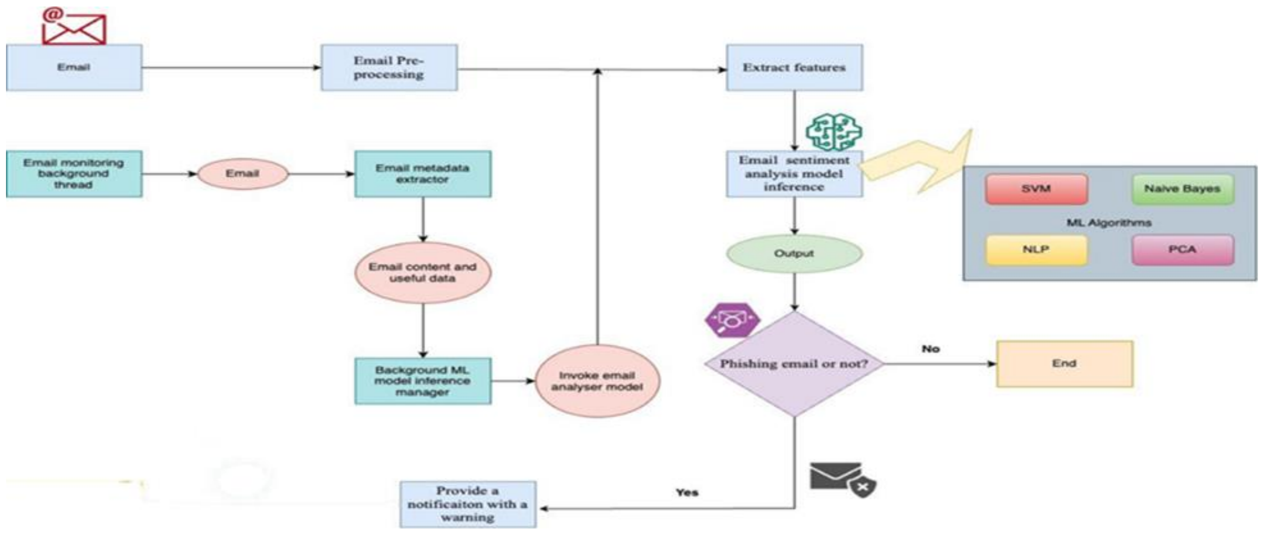


Figure 0.41Pytorch to android

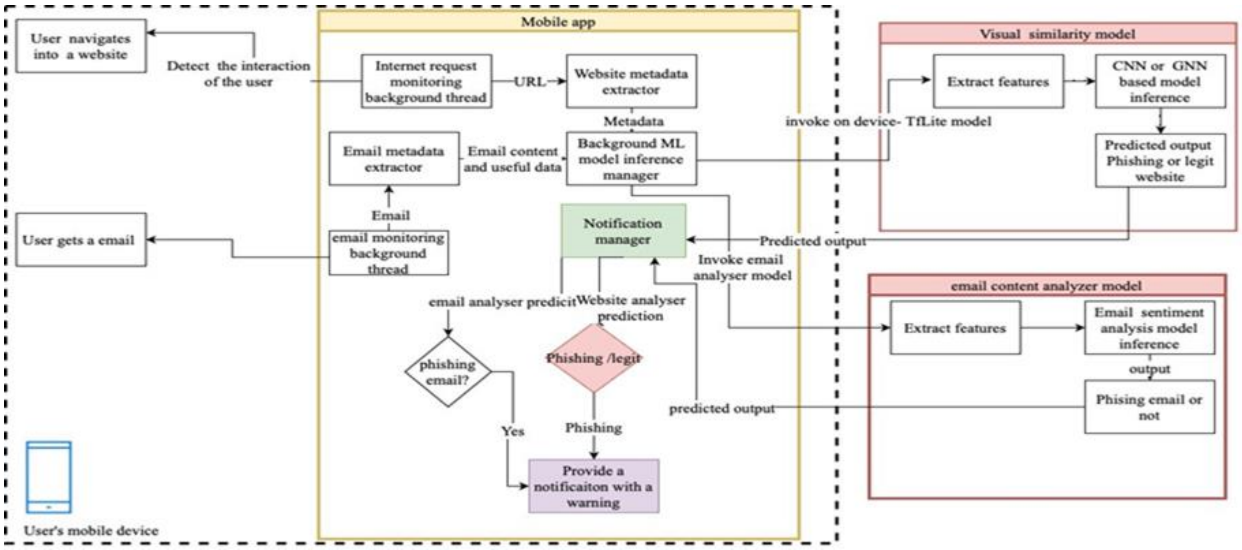
The model was saved as model.pt1 and integrated into android mobile application. The saved model is saved in the app resources. Since the model is running inside the mobile application itself it does not need an internet connection.

2 Sentiment analysis with naïve bayes model integrating.



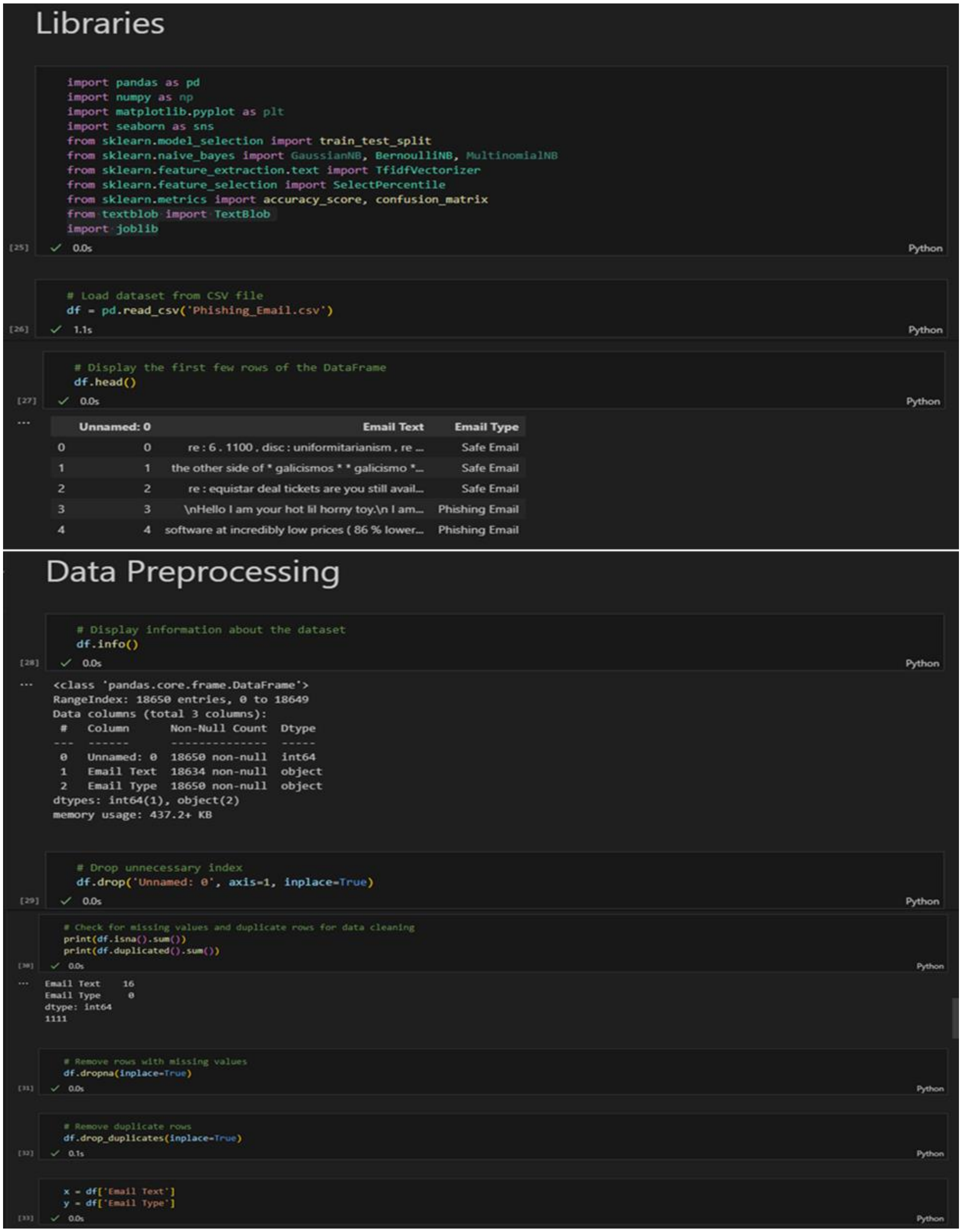
Email
Email Pre-processing
Extract features
Email monitoring background thread
Email
Email metadata extractor
Email sentiment analysis model inference
SVM
Naive Bayes
ML Algorithms
NLP
PCA
Email content and useful data
Output
Background ML model inference manager
Invoke email analyser model
Phishing email or not?
No
End
Yes
Provide a notificaiton with a warning

Figure 0.5Phishing Email Detection System Diagram



User navigates into a website
Detect the interaction of the user
Mobile app
Internet request monitoring background thread
URL
Website metadata extractor
Metadata
Visual similarity model
Extract features
CNN or GNN based model inference
invoke on device- TfLite model
Email metadata extractor
Email content and useful data
Background ML model inference manager
Predicted output Phishing or legit website
Email
User gets a email
email monitoring background thread
Notification manager
Predicted output
Invoke email analyser model
email content analyzer model
email analyser predicit
Website analyser prediction
Extract features
Email sentiment analysis model inference
phishing email?
Phishing /legit
output
predicted output
Phising email or not
Yes
Phishing
Provide a notificaiton with a warning
User's mobile device

Figure 0.6app diagram



Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB, BernoulliNB, MultinomialNB
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_selection import SelectPercentile
from sklearn.metrics import accuracy_score, confusion_matrix
from textblob import TextBlob
import joblib
```
[25] ✓ 0.0s Python

```
# Load dataset from CSV file
df = pd.read_csv('Phishing_Email.csv')
```
[26] ✓ 1.1s Python

```
# Display the first few rows of the DataFrame
df.head()
```
[27] ✓ 0.0s Python

	Unnamed: 0	Email Text	Email Type
0	0	re : 6 . 1100 , disc : uniformitarianism , re ...	Safe Email
1	1	the other side of * galicismos * * galicismo *...	Safe Email
2	2	re : equistar deal tickets are you still avail...	Safe Email
3	3	\nHello I am your hot lil horny toy.\n I am...	Phishing Email
4	4	software at incredibly low prices (86 % lower...	Phishing Email

Data Preprocessing

```
# Display information about the dataset
df.info()
```
[28] ✓ 0.0s Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18650 entries, 0 to 18649
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  18650 non-null  int64
 1   Email Text  18634 non-null  object
 2   Email Type  18650 non-null  object
dtypes: int64(1), object(2)
memory usage: 437.2+ KB
```

```
# Drop unnecessary index
df.drop('Unnamed: 0', axis=1, inplace=True)
```
[29] ✓ 0.0s Python

```
# Check for missing values and duplicate rows for data cleaning
print(df.isna().sum())
print(df.duplicated().sum())
```
[30] ✓ 0.0s Python

```
Email Text    16
Email Type     0
dtype: int64
1111
```

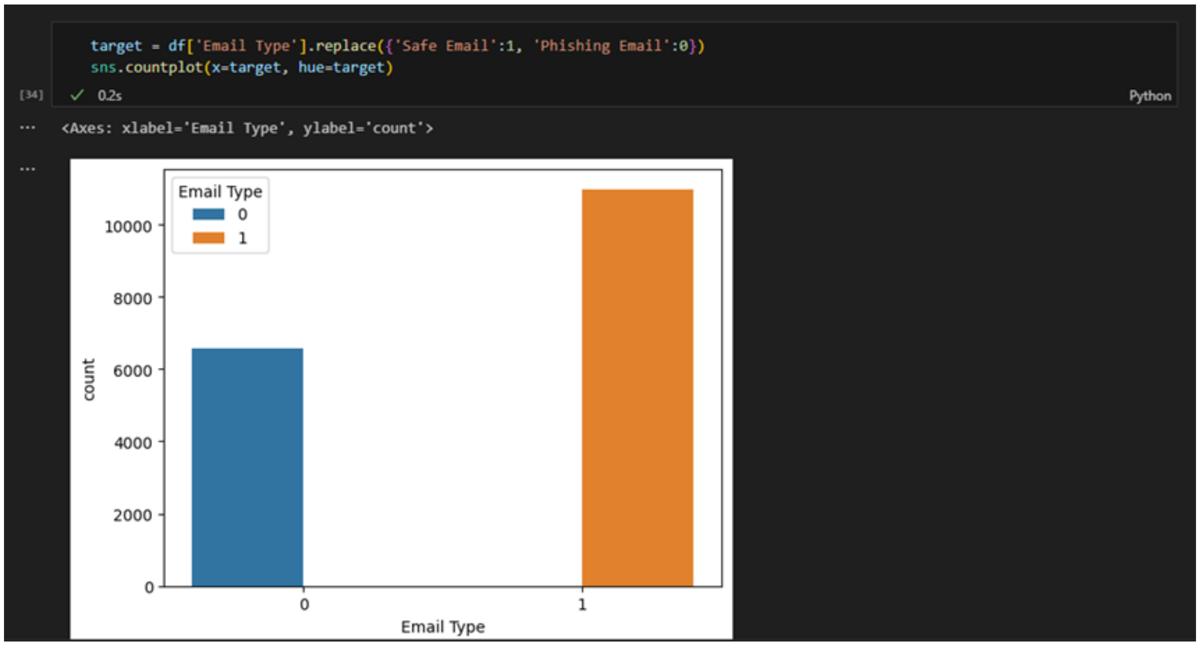
```
# Remove rows with missing values
df.dropna(inplace=True)
```
[31] ✓ 0.0s Python

```
# Remove duplicate rows
df.drop_duplicates(inplace=True)
```
[32] ✓ 0.1s Python

```
x = df['Email Text']
y = df['Email Type']
```
[33] ✓ 0.0s Python

Figure 0.7ML Code(Data Preprocessing)

- Data Collection: The research's first step is to collect a dataset of email samples, including both safe (legal) and phishing emails. Using the pd.read_csv() function, the dataset is loaded into a pandas DataFrame to produce a structured data source for analysis.
- Data Exploration: The initial step in data exploration is to analyze the dataset's first few rows using df.head() and to learn more about the dataset's structure and attributes using df.info(). This stage is essential for comprehending the dimensions of the data and finding any values that are duplicated or missing entirely.
- Data Cleaning: To improve the quality and dependability of the dataset, data cleaning is done. This includes removing pointless columns like 'Unnamed: 0', which is done with the df.drop('Unnamed: 0', axis=1, inplace=True) function. Additionally, df.isna().sum() is used to identify missing data, and df.dropna(inplace=True) is used to eliminate them. Df.drop_duplicates(inplace=True) finds and removes duplicate records



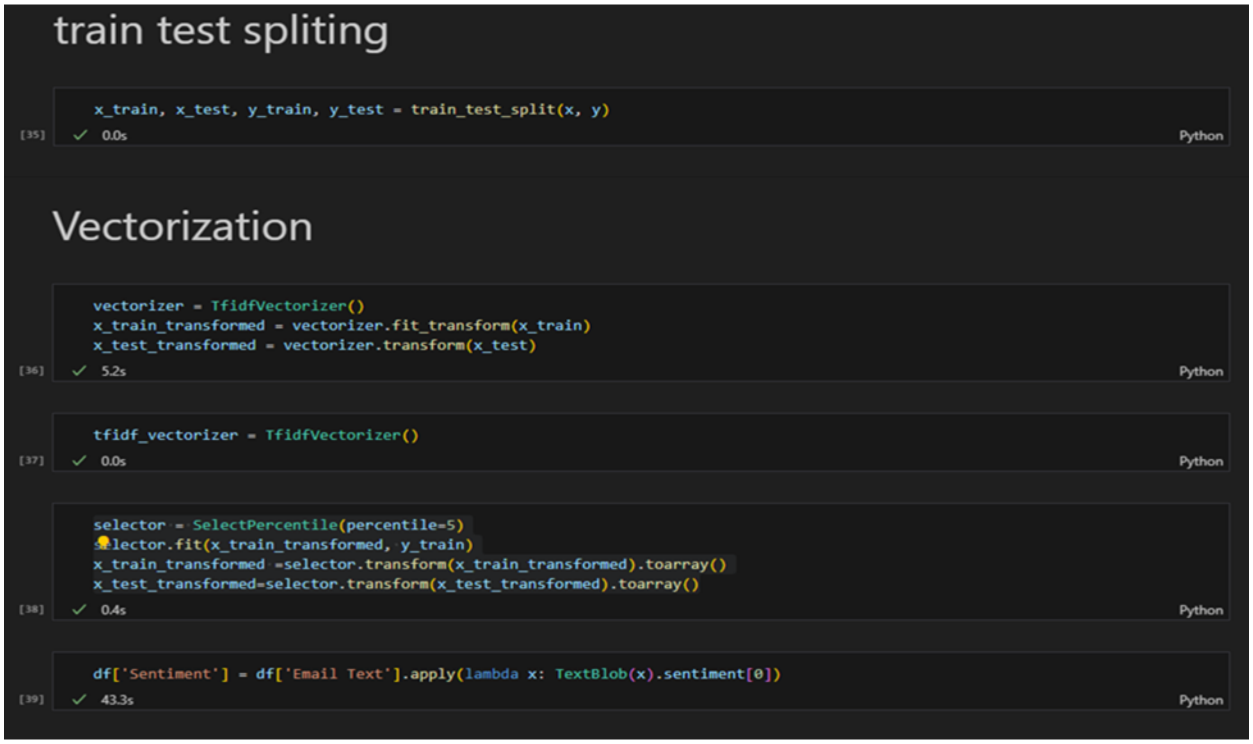
```
target = df['Email Type'].replace({'Safe Email':1, 'Phishing Email':0})
sns.countplot(x=target, hue=target)
```

```
[34]  ✓ 0.2s                                                    Python
<Axes: xlabel='Email Type', ylabel='count'>
```

Figure 0.8ML code(data visualization)

Data Visualization: Data visualization using libraries like Matplotlib and Seaborn is employed to gain insights into the dataset's distribution and characteristics. The sns.countplot() function is used to create a count plot that visually represents the distribution of email types (safe or phishing). This visualization helps in understanding class balance and the need for addressing imbalanced datasets.

- Data Splitting: Using train_test_split(x, y), the dataset is divided into training and testing sets. This division makes it possible to train models on one set of data while evaluating them on another.



train test spliting

```
x_train, x_test, y_train, y_test = train_test_split(x, y)
```

[35] ✓ 0.0s Python

Vectorization

```
vectorizer = TfidfVectorizer()
x_train_transformed = vectorizer.fit_transform(x_train)
x_test_transformed = vectorizer.transform(x_test)
```

[36] ✓ 5.2s Python

```
tfidf_vectorizer = TfidfVectorizer()
```

[37] ✓ 0.0s Python

```
selector = SelectPercentile(percentile=5)
selector.fit(x_train_transformed, y_train)
x_train_transformed =selector.transform(x_train_transformed).toarray()
x_test_transformed=selector.transform(x_test_transformed).toarray()
```

[38] ✓ 0.4s Python

```
df['Sentiment'] = df['Email Text'].apply(lambda x: TextBlob(x).sentiment[0])
```

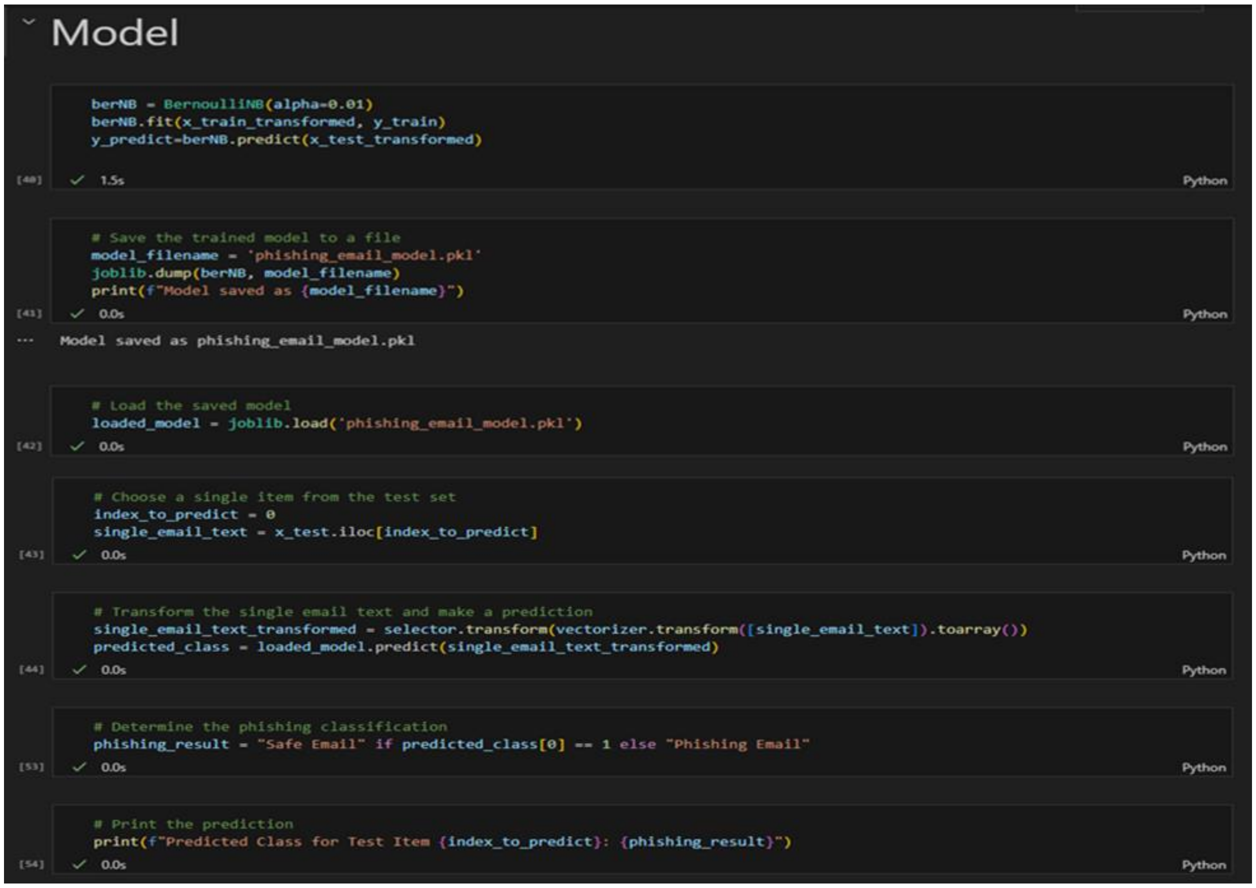
[39] ✓ 43.3s Python

Figure 0.9 ML Code(Data Splitting and Feature Engineering

- Text Vectorization: Using the TF-IDF (Term Frequency-Inverse Document Frequency) vectorization technique, the email text content is converted into numerical characteristics appropriate for machine learning. TfidfVectorizer() helps achieve this. It determines the weight of each email's words by turning them into numerical vectors.

- Feature Selection: SelectPercentile is used for feature selection to perhaps enhance model performance and decrease the dimensionality of the TF-IDF vectors. The top percentile of features that are most important to the categorization job are retained.

- Sentiment Analysis: Using TextBlob, sentiment analysis is performed on email text. Each email is given a sentiment score (polarity) to identify any emotional content. The machine learning model will incorporate this sentimental data as an additional feature.



Model

```
berNB = BernoulliNB(alpha=0.01)
berNB.fit(x_train_transformed, y_train)
y_predict=berNB.predict(x_test_transformed)
```
[40] ✓ 1.5s Python

```
# Save the trained model to a file
model_filename = 'phishing_email_model.pkl'
joblib.dump(berNB, model_filename)
print(f"Model saved as {model_filename}")
```
[41] ✓ 0.0s Python

```
Model saved as phishing_email_model.pkl
```

```
# Load the saved model
loaded_model = joblib.load('phishing_email_model.pkl')
```
[42] ✓ 0.0s Python

```
# Choose a single item from the test set
index_to_predict = 0
single_email_text = x_test.iloc[index_to_predict]
```
[43] ✓ 0.0s Python

```
# Transform the single email text and make a prediction
single_email_text_transformed = selector.transform(vectorizer.transform([single_email_text]).toarray())
predicted_class = loaded_model.predict(single_email_text_transformed)
```
[44] ✓ 0.0s Python

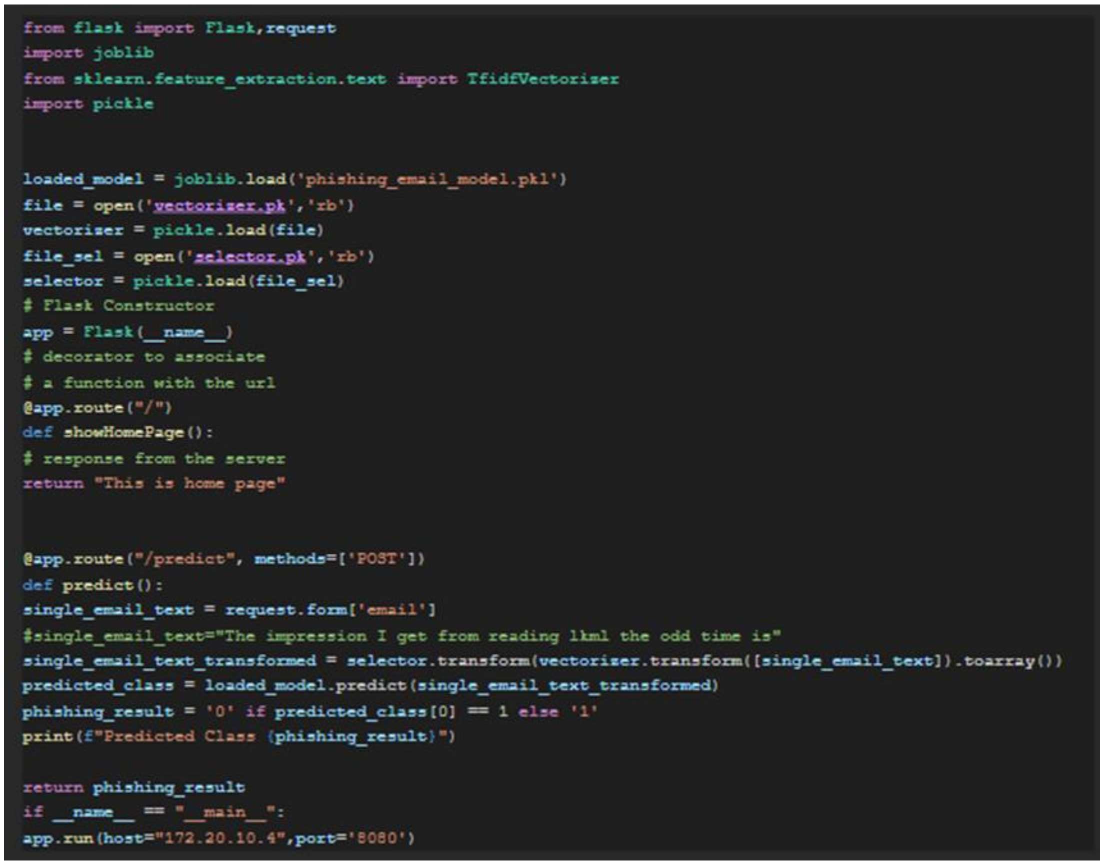
```
# Determine the phishing classification
phishing_result = "Safe Email" if predicted_class[0] == 1 else "Phishing Email"
```
[53] ✓ 0.0s Python

```
# Print the prediction
print(f"Predicted Class for Test Item {index_to_predict}: {phishing_result}")
```
[54] ✓ 0.0s Python

Figure 0.10ML Code (Model Training)

- **Model Selection:** Three variants of the Naive Bayes algorithm, including GaussianNB, BernoulliNB, and MultinomialNB, are available for experimentation. In this code, BernoulliNB with a specified alpha value is chosen as the classification algorithm. It is fitted to the training data using **berNB.fit()**. The heart of the phishing email detection system lies in the model training and evaluation. A Bernoulli Naive Bayes classifier is chosen as the machine learning algorithm for this task. This choice assumes that the features follow a Bernoulli distribution, which is often suitable for binary classification tasks. The classifier is trained on preprocessed and transformed training data.

- **Model Serialization:** The trained phishing email detection model is serialized using the joblib library. It is saved as a binary file with the filename 'phishing_email_model.pkl'. This step ensures that the model can be easily loaded and used for future predictions without retraining.
- **Model Deployment:** Although not explicitly implemented in this code, the serialized model can be integrated into various applications, such as web or mobile apps. The code demonstrates how to load the saved model and make predictions on new email text samples.
- **Real-time Prediction:** A single email text sample from the test set is chosen for real-time prediction. The sample is transformed using the trained feature selection and vectorization techniques, and the saved model is used to predict whether the email is phishing or safe. The prediction result is printed to the console.



```
from flask import Flask,request
import joblib
from sklearn.feature_extraction.text import TfidfVectorizer
import pickle


loaded_model = joblib.load('phishing_email_model.pkl')
file = open('vectorizer.pk','rb')
vectorizer = pickle.load(file)
file_sel = open('selector.pk','rb')
selector = pickle.load(file_sel)
# Flask Constructor
app = Flask(__name__)
# decorator to associate
# a function with the url
@app.route("/")
def showHomePage():
# response from the server
return "This is home page"


@app.route("/predict", methods=['POST'])
def predict():
single_email_text = request.form['email']
#single_email_text="The impression I get from reading lkml the odd time is"
single_email_text_transformed = selector.transform(vectorizer.transform([single_email_text]).toarray())
predicted_class = loaded_model.predict(single_email_text_transformed)
phishing_result = '0' if predicted_class[0] == 1 else '1'
print(f"Predicted Class {phishing_result}")

return phishing_result
if __name__ == "__main__":
app.run(host="172.20.10.4",port='8080')
```

Figure 0.11Flask server code

This is a Python Flask web application that serves as a simple API for predicting whether an input email is a phishing email or not. It uses that previous pre-trained machine learning model for prediction. For integrate the model to API loaded the model to flask server.

B. Web extension

1. **Website Feature analysis**

Data Collection

Begin by gathering a comprehensive dataset of website traffic encompassing both legitimate and phishing websites. Ensure that the dataset includes a variety of features that can potentially distinguish between the two, such as URL length, domain age, and more.

Data Preprocessing

Perform data preprocessing to prepare the dataset for model training. This involves selecting relevant features, handling missing data, and encoding categorical variables. Additionally, split the dataset into a training set and a separate testing set to evaluate model performance.

Model Selection

Choose a set of supervised machine learning models suitable for the task. Common models to consider include decision trees, random forests, and logistic regression. Each of these models has its strengths and weaknesses, which you can explore during the evaluation phase.

Model Training

Train the selected machine learning models using the training dataset. During this phase, the models will learn patterns and relationships within the data that can help distinguish between legitimate and phishing websites.

Model Evaluation

Validate the chosen models on a separate validation set to assess their generalizability. Utilize appropriate evaluation metrics such as accuracy, precision, recall, and F1-score to quantify the models' performance.

Integration

Once a model with satisfactory performance is identified, integrate it into a browser extension or mobile app. This integration allows for real-time phishing detection as users interact with websites.

System Testing

Test the integrated system using a sample dataset of website , including both legitimate and potentially phishing websites. Assess the system's ability to detect and flag phishing attempts accurately.

2. URL-based analysis

Data Collection and Preprocessing:

The project commences with the collection of two critical datasets: one containing 1000 phishing URLs and another comprising legitimate URLs. These datasets form the foundational elements for training and testing the URL-based phishing detection model. The collected datasets are meticulously preprocessed to ensure data integrity and consistency, encompassing tasks such as deduplication, handling missing values, and standardizing URL formats to facilitate seamless analysis.

Feature Extraction for URL-Based Analysis:

Within the feature extraction process, which is pivotal for URL-based analysis, attributes are systematically extracted from both the phishing and legitimate URLs. These attributes include quantifying URL length, deciphering domain-related features such as subdomains, assessing character frequency indicators like '@' symbols and '-', and considering factors like the presence of HTTP tokens, DNS records, URL redirection indicators, domain registration length, statistical report features, the existence of 'tiny_url,' and web traffic-related features. These attributes are extracted from both phishing and legitimate URLs and serve as essential inputs for the subsequent machine learning model.

Dataset Splitting and Random Forest Model Training:

Following the preprocessing and feature extraction steps, the dataset, comprising both phishing and legitimate URLs, is strategically divided into training and testing subsets. A significant portion of the dataset is allocated to training to enable the Random Forest algorithm to learn patterns effectively. The Random Forest algorithm, leveraging the collective attributes from both phishing and legitimate URLs, is employed to train the model, enhancing its capability for phishing detection.

Integration into User-Friendly Web Extension:

With the trained model in place, the next step involves seamless integration into a user-friendly web extension. This extension is designed with the utmost consideration for user experience, allowing users to interact effortlessly with the tool. Clear indicators or alerts are incorporated to notify users when a phishing URL is detected, ensuring a user-friendly and informative interface.

6. COMMERCIALIZING ASPECTS OF THE PRODUCT

A. MOBILE APPLICATION

Our mobile app is designed to cater to the needs of both basic and advanced mobile users. It is particularly useful for users who deal with many emails and engage in frequent web browsing, as they are often the target of phishing attacks. For basic mobile phone users, our app provides a user-friendly interface and simple functionalities that are easy to navigate. It can be easily downloaded and installed on any basic mobile phone without the need for a high-end device. The app is designed to help users identify and avoid phishing scams, which are becoming increasingly common in today's digital age. Advanced mobile users, on the other hand, require more robust security features to stay safe from phishing attacks. Our app provides advanced security features such as anti-phishing filters, real-time scanning of emails and websites for potential threats, and proactive alerts that notify users of potential phishing attacks. Overall, our mobile app is a comprehensive security solution that caters to the needs of both basic and advanced mobile phone users. By providing robust security features and a user-friendly interface, we aim to make mobile phone usage safer and more secure for all users.

We plan to offer our users two different subscription packages for our mobile app.

The first package is a basic subscription, which we plan to charge an annual fee of $5. This package provides users with the option to upload fifty screenshots per month and 50 emails per month. The second package is our advanced subscription, which includes all the features of the basic package, but an unlimited number of scans. We plan to charge an annual fee of $10 for this package. We believe that both subscription packages offer great value to users who are concerned about their online security. The basic package provides essential security features that can help protect users from phishing attacks, while the advanced package offers more robust security features for those who require additional protection.

We also understand the importance of transparency when it comes to subscription pricing. Therefore, we will clearly outline the pricing and features of each package on our app's website and within the app itself. We aim to provide our users with the information they need to make an informed decision about which subscription package is right for them.

B. WEB EXTENSION

Beyond its research and development phase, the web extension for URL-based phishing website detection holds significant potential for commercialization. The product's adaptability, user-friendliness, and robust phishing detection capabilities position it as a valuable asset for both individual and organizational users. In the commercialization phase, considerations will revolve around product licensing and distribution strategies. Potential avenues include offering the extension as a freemium product, where basic protection is provided for free, and advanced features or enhanced threat intelligence feeds are available through subscription models. Collaborations with cybersecurity companies, browser developers, or integration into security software suites are explored to expand its reach. Additionally, marketing and user education efforts play a pivotal role in garnering user trust and adoption. As phishing threats persistently evolve, the commercialization strategy aims to provide users with an accessible and effective tool that empowers them to navigate the digital landscape with confidence, ultimately enhancing online security.

7. TESTING AND IMPLEMENTAION

A. MOBILE APPLICATION

The mobile application was designed in a way that the user can input a screenshot of the suspected website. In the user interface under the site detector a user can select the option. Then from the gallery they can select the screenshot, or the image.

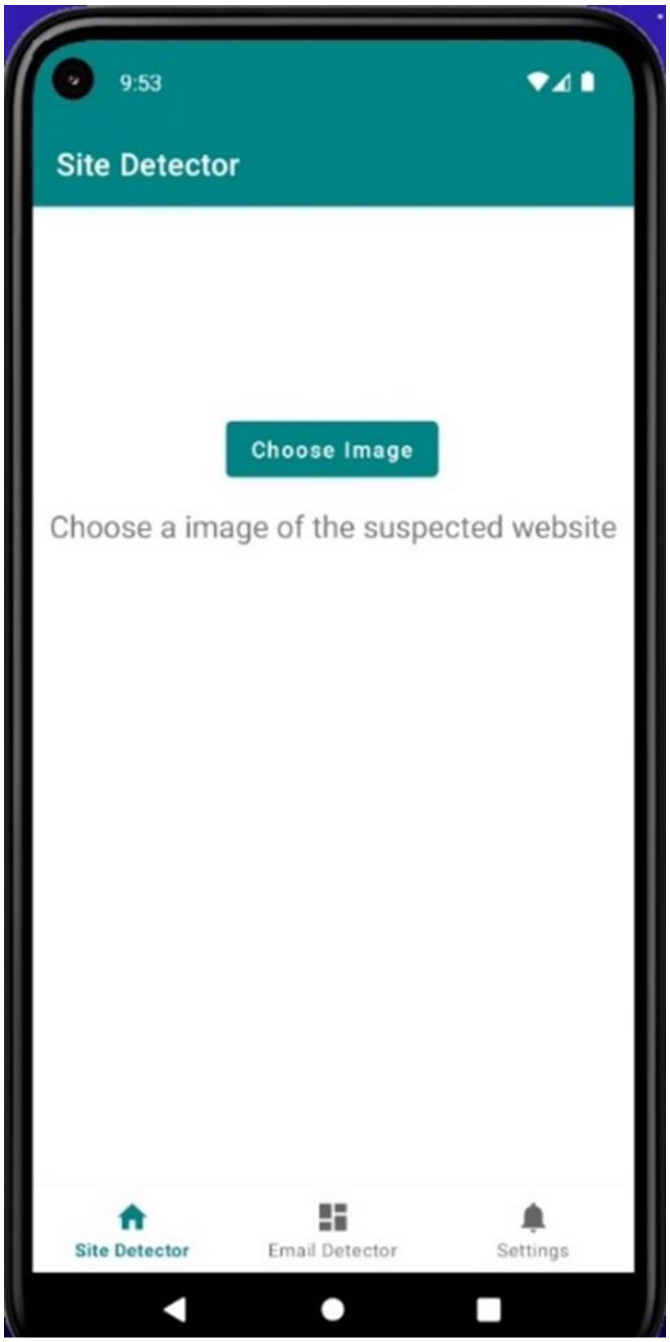


Figure 0.12Mobile app interface

Then in the next diagrams can see that the app has given the output for a phishing and a benign site.

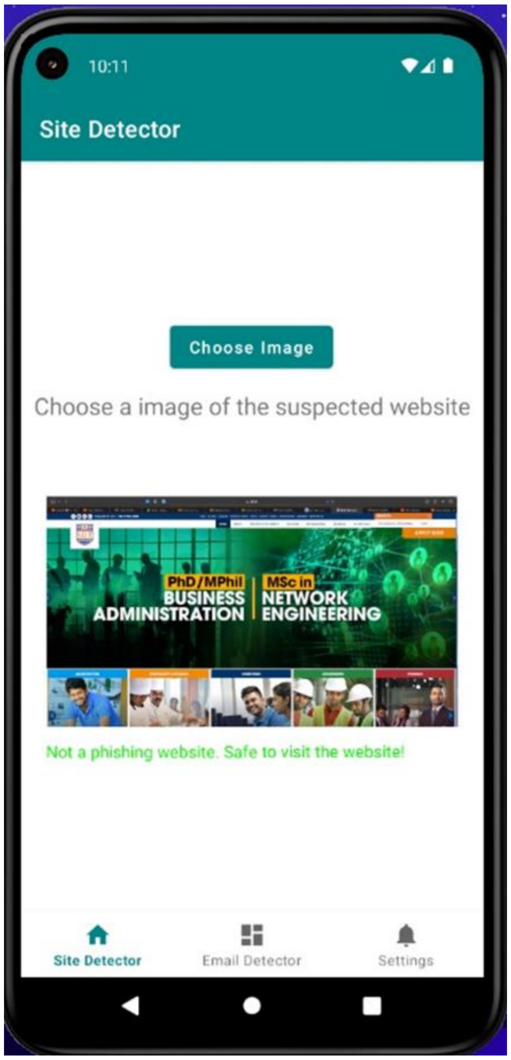

10:11
Site Detector
Choose Image
Choose a image of the suspected website
Not a phishing website. Safe to visit the website!
Site Detector
Email Detector
Settings

Figure 0.13Image output for benign site

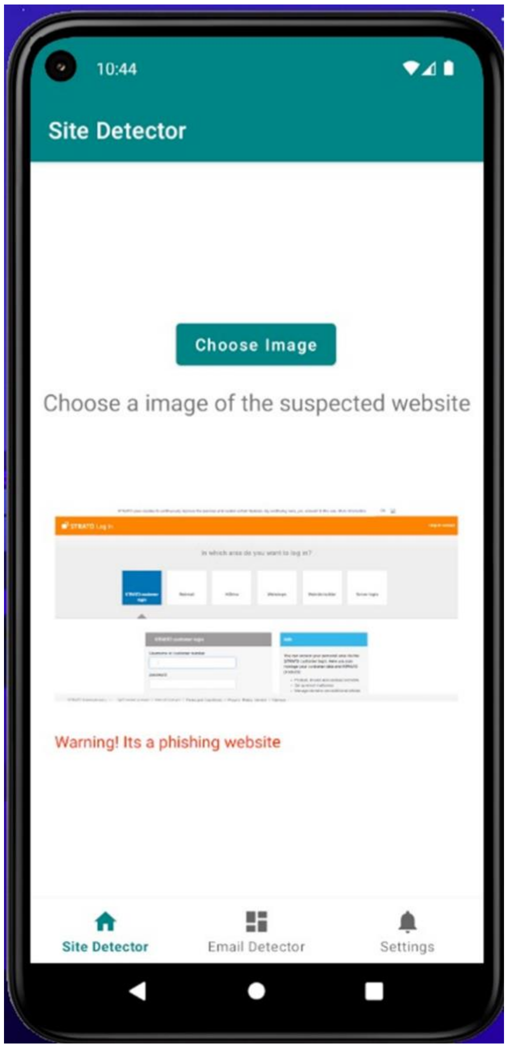
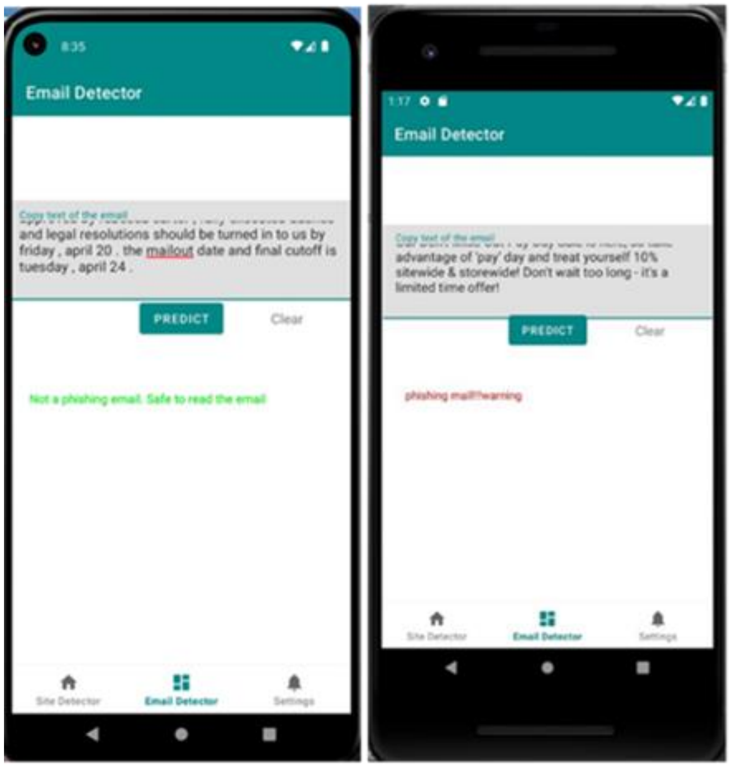

10:44
Site Detector
Choose Image
Choose a image of the suspected website
Warning! Its a phishing website
Site Detector
Email Detector
Settings

Figure 0.14image output for phishing site

For checking a phishing email the user can copy the email and paste the email into the text box. Then the user will be able to get the output.

The testing dataset is used to assess the performance of the trained model. berNB.predict() is used to make predictions. Several performance indicators, including accuracy and confusion matrix, are computed to evaluate how well the model detects phishing emails. Using confusion_matrix(), the confusion matrix is printed. The held-out testing data is used to test the performance of the model. The code calculates several metrics to determine how effective the model is. The confusion matrix and accuracy score are some of these metrics. The confusion matrix sheds light on how well the model can distinguish between safe and phishing emails. The overall accuracy of the model's predictions is quantified by the accuracy score.



and legal resolutions should be turned in to us by friday , april 20 . the mailout date and final cutoff is tuesday , april 24 .

Not a phishing email. Safe to read the email

advantage of 'pay' day and treat yourself 10% sitewide & storewide! Don't wait too long - it's a limited time offer!

phishing mail!!warning

Figure 0.15Phishing email output for benign and phishing email

B. WEB EXTENSION

Frontend Scripts

- **content.js**

1. Feature Calculation Functions:

- “isImgFromDifferentDomain”: This function counts the total number of images on the page and compares it to the number of images with a different domain source. It then logs a classification ("NP," "Maybe," or "P") based on a threshold.

- “isAnchorFromDifferentDomain”: Similar to “isImgFromDifferentDomain”, but for anchor (“<a>“) elements.

- “isScLnkFromDifferentDomain”: This function counts the total number of script and link elements on the page and compares it to the number of elements with a different domain source.

- “isFormActionInvalid”: This function checks if any form elements have an empty or missing action attribute, which may be indicative of a phishing attempt.

- “isMailToAvailable”: This function checks if there are any anchor (“<a>“) elements with a “mailto:” link, which could indicate an attempt to collect email addresses for phishing.

- “isStatusBarTampered”: This function checks if any anchor elements have “onmouseover” or “onclick” attributes that modify the window status, which might be an attempt to mislead users.

- “isIframePresent”: This function checks if there are any “<iframe>“ elements on the page, which could potentially be used for phishing attacks.

2. Identical Domain Counting : These functions use regular expressions to extract the main domain from the current URL. They then iterate through elements of a specific type (images, forms, anchors, etc.) and count how many have a source or action URL with the same main domain.

3. Logging : Depending on the feature evaluation, these functions log messages to the console indicating whether the feature suggests that the webpage may be phishing. The logs include "NP" (Not Phishing), "Maybe," or "P" (Phishing) based on predefined thresholds.

- **background.js**

1. Message Listening :

  - This script listens for messages from other parts of the extension, specifically from “content.js”. When a message with the action "predict" is received, it triggers the prediction process.

2. Prediction Handling :

  - When a message is received, it extracts the features from the message and sends them to the machine learning model for prediction.

3. Response Handling :

  - Once the prediction is received, it sends the prediction result back to “content.js”.

- **manifest.json**

“manifest_version”: Specifies the version of the manifest file format. In this case, it's version 3.

“name”: The name of the Chrome extension ("Phishing Detector").

“version”: The version number of the extension (in this case, "1").

“icons”: An object that defines the icons used for the extension. Different sizes are provided to fit different parts of the Chrome interface.

“background”: Configures the background script(s) that run in the background of the extension. In this case, it includes “background.js”.

“content_scripts”: Specifies scripts that are injected into web pages. In this case, it includes “jquery-3.7.1.min.js” and “content.js”, which are used to interact with the web page and extract features.

“permissions”: This field lists permissions that the extension requires to perform specific tasks. In this case, it has “<all_urls>“, which means the extension can access any URL.

“browser_action” (not shown in the provided code): This field can be used to define a browser action that the user can interact with, like a button in the Chrome toolbar.

These files and their configurations work together to create a Chrome extension that evaluates web pages for potential phishing indicators using a combination of feature calculations and machine learning predictions.

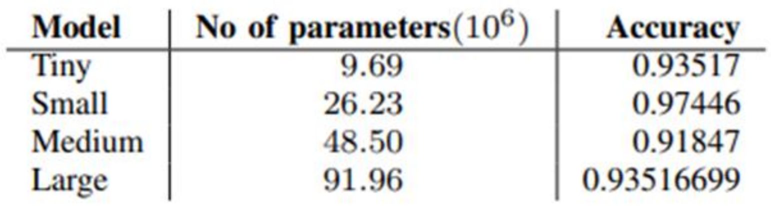
RESULTS AND DISCUSSION

1 Visual similarity to detect phishing websites.

Following table shows the test accuracy of the four models. Maximum accuracy was attained with the small architecture, which has fewer parameters. This demonstrates that when the number of parameters increases, the model becomes excessively complex and the accuracy decreases.

The small model demonstrated the highest accuracy and converged towards the end of the training process, indicating effective training and stability. The confusion matrix analysis showed the model's superior classification performance, indicating no need for an overly complex model for distinguishing between phishing and benign websites. The model correctly predicted 97% of benign websites and 84% of phishing websites, proving that increasing parameters doesn't necessarily increase model accuracy. The mobile application's benefits include preserving user privacy since the model runs inside the mobile itself.

3 Accuracies of each model



Model	No of parameters(10^6)	Accuracy
Tiny	9.69	0.93517
Small	26.23	0.97446
Medium	48.50	0.91847
Large	91.96	0.93516699

2 Website features to detect phishing websites.

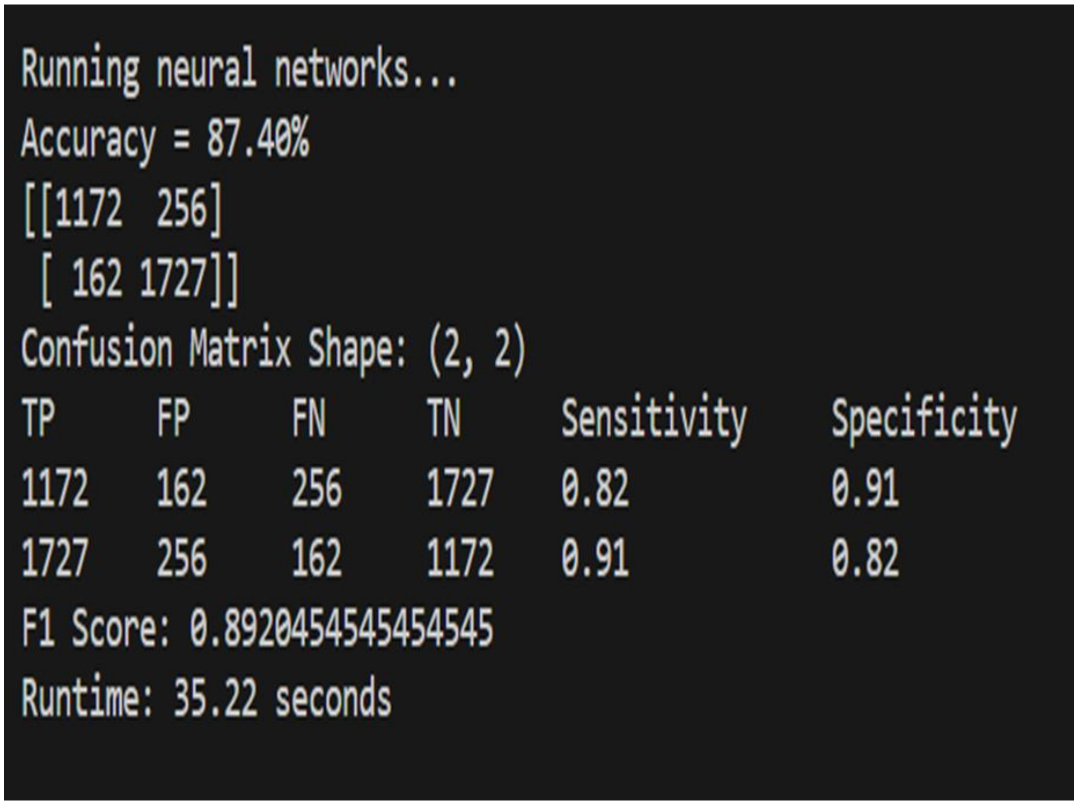
a. Model evaluation

In the pursuit of achieving the highest accuracy for phishing attack detection, this research project employed a comprehensive evaluation process. Three distinct machine learning algorithms—Neural Network, Random Forest, and Support Vector Machine (SVM)—were rigorously tested and compared for their effectiveness in discerning between legitimate and phishing websites.

The selection of these three algorithms was based on their prominence in the field of machine learning and their potential to excel in the task at hand. Each algorithm was meticulously implemented, trained, and evaluated using the same dataset to ensure a fair and unbiased comparison.

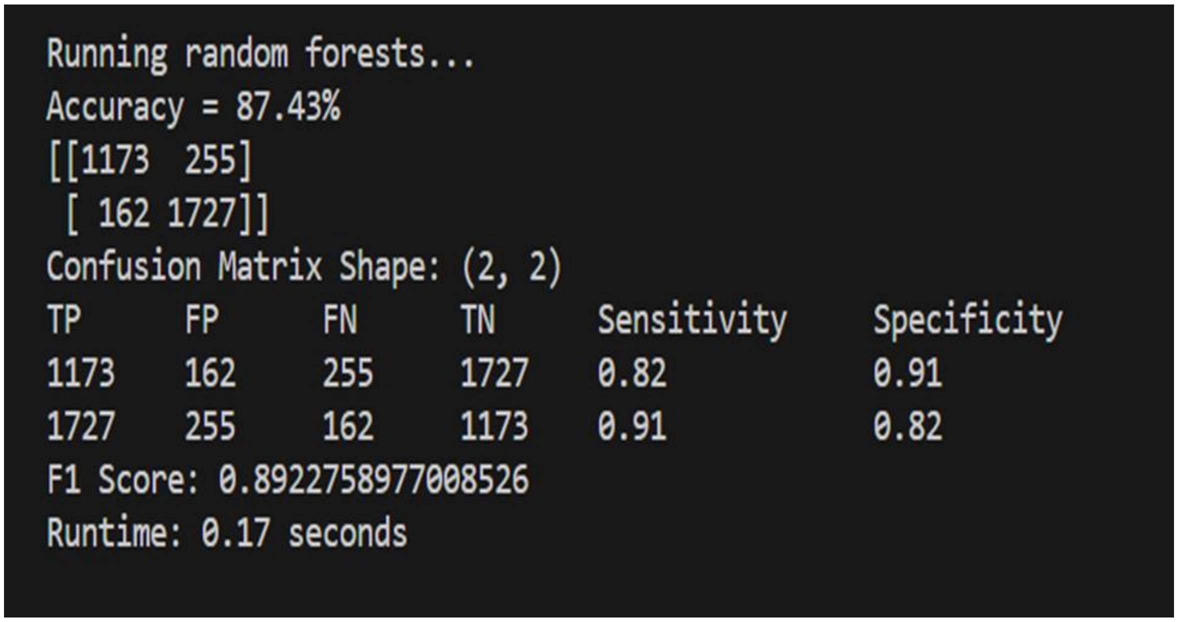
After careful examination of the results, it was unequivocally determined that the Random Forest algorithm outperformed its counterparts, achieving the highest accuracy among the three algorithms. Random Forest demonstrated its robustness and efficiency in distinguishing phishing attempts from legitimate online activities.

The selection of Random Forest as the algorithm of choice underscores its suitability for real-time phishing detection and prevention. Its exceptional performance serves as a testament to the potential impact of leveraging HTML and JavaScript features in cybersecurity efforts.



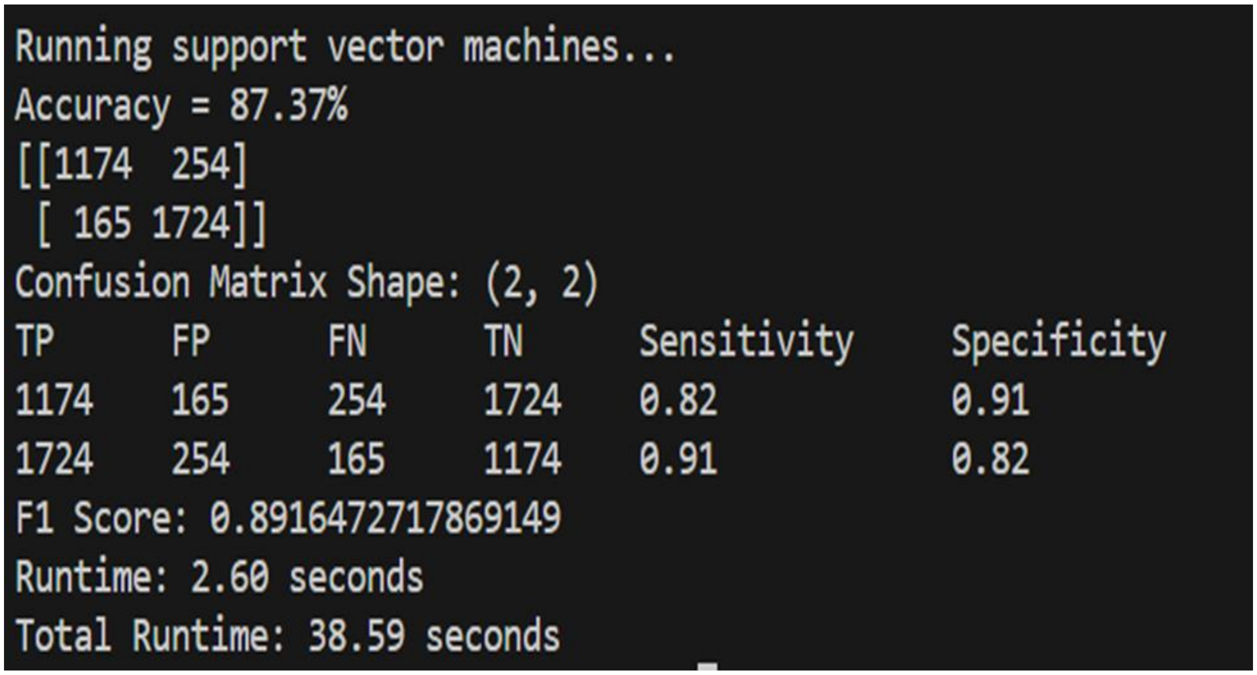
```
Running neural networks...
Accuracy = 87.40%
[[1172  256]
 [ 162 1727]]
Confusion Matrix Shape: (2, 2)
TP      FP      FN      TN      Sensitivity     Specificity
1172    162     256     1727    0.82            0.91
1727    256     162     1172    0.91            0.82
F1 Score: 0.8920454545454545
Runtime: 35.22 seconds
```

Figure 0.1Neural networks



```
Running random forests...
Accuracy = 87.43%
[[1173  255]
 [ 162 1727]]
Confusion Matrix Shape: (2, 2)
TP      FP      FN      TN      Sensitivity     Specificity
1173    162     255     1727    0.82            0.91
1727    255     162     1173    0.91            0.82
F1 Score: 0.8922758977008526
Runtime: 0.17 seconds
```

Figure 0.2Random Forest

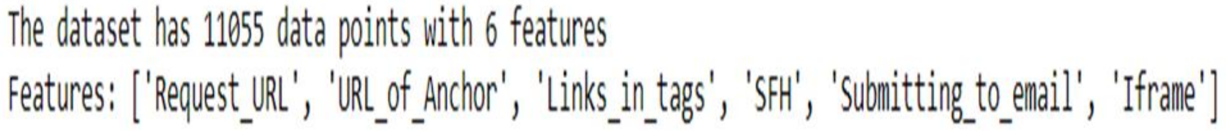


```
Running support vector machines...
Accuracy = 87.37%
[[1174  254]
 [ 165 1724]]
Confusion Matrix Shape: (2, 2)
TP      FP      FN      TN      Sensitivity     Specificity
1174    165     254     1724    0.82            0.91
1724    254     165     1174    0.91            0.82
F1 Score: 0.8916472717869149
Runtime: 2.60 seconds
Total Runtime: 38.59 seconds
```

Figure 0.3Support Vector Machine

b. Preprocessing

The output of preprocessing is shown below. After selecting relevant features from the dataset. Here are the selected features.



```
The dataset has 11055 data points with 6 features
Features: ['Request_URL', 'URL_of_Anchor', 'Links_in_tags', 'SFH', 'Submitting_to_email', 'Iframe']
```

Figure 0.4Dataset

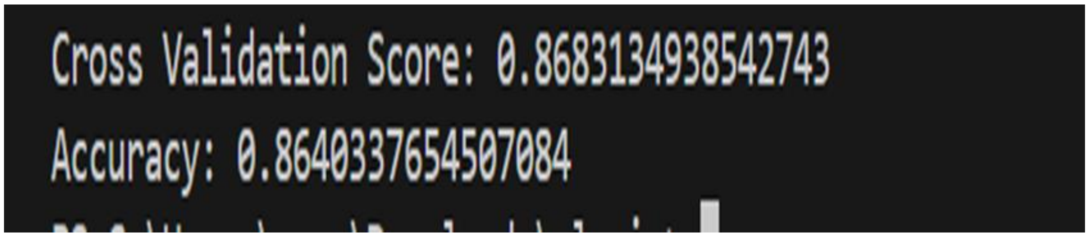
c. Training

The output of training is shown below. The used algorithm is Random Forest. As it scored the highest accuracy among neural networks and support vector machine. The performance of the Random Forest algorithm in detecting phishing attacks was rigorously assessed using cross-validation. Cross-validation is a fundamental technique that provides a robust estimate of a model's performance by dividing the dataset into multiple subsets. It then trains and evaluates the model iteratively, ensuring that each subset serves as both a training and testing dataset.

In the case of the Random Forest algorithm, the cross-validation process yielded a noteworthy cross-validation score of 0.8683. This score serves as a critical indicator of the model's ability to generalize effectively to unseen data. A cross-validation score of 0.8683 implies that, on average, the Random Forest algorithm achieved an accuracy rate of approximately 86.83% when detecting phishing attacks across various data subsets.

Throughout the training phase of the Random Forest algorithm, the model exhibited a commendable level of accuracy. Specifically, during the training state, the algorithm achieved an accuracy rate of 0.864033. This accuracy metric reflects the model's ability to correctly classify instances within the training dataset, indicating the extent to which it captures patterns and relationships within the provided data.

An accuracy of 0.864033 signifies that the Random Forest algorithm accurately predicted the legitimacy or malicious nature of web interactions in the training dataset approximately 86.40% of the time. This high level of accuracy underscores the effectiveness of the model in learning from the dataset and making informed decisions.



```
Cross Validation Score: 0.8683134938542743
Accuracy: 0.8640337654507084
```

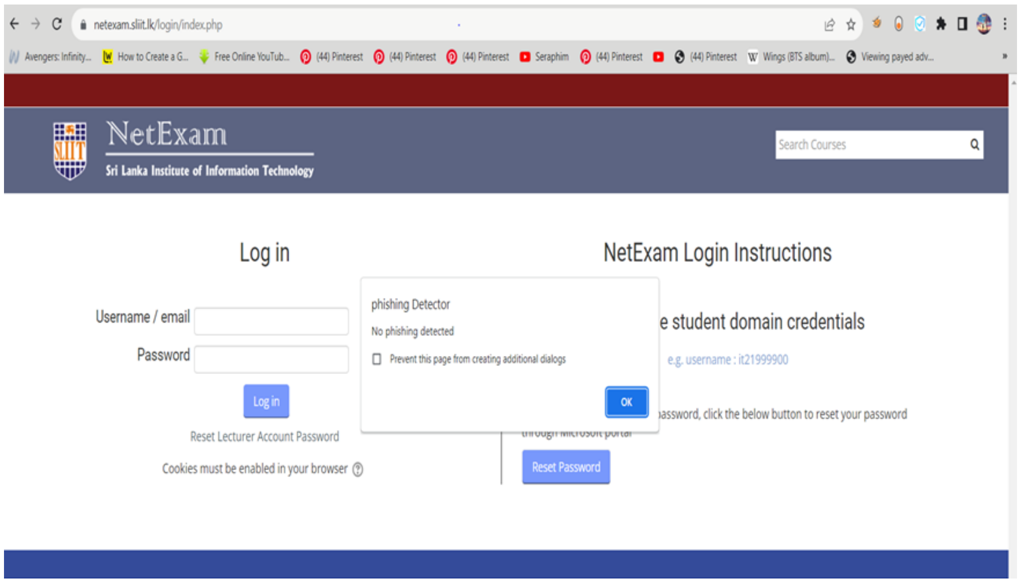
Figure 0.5Cross Validation and Accuracy

d. Sample screen shots during testing

When a user accesses a webpage through the phishing detection system, a comprehensive analysis of the webpage's attributes, HTML and JavaScript features, and online behavior patterns is conducted by the system's algorithms and models. Following this analysis, the system generates a definitive classification result, classifying the visited website into one of two categories: "No Phishing Detected" or "Warning: Phishing Detected."

If the analysis concludes that the webpage exhibits no signs of phishing activity and is considered safe, the system displays a clear and reassuring alert, reading "No Phishing Detected." Conversely, when the system identifies suspicious characteristics indicative of phishing, it promptly triggers a warning alert, reading "Warning: Phishing Detected."

These popup alerts are a critical aspect of the system's functionality, as they enable users to make informed decisions about the trustworthiness of the websites they interact with in real-time. The "No Phishing Detected" alert provides peace of mind when browsing legitimate websites, while the "Warning: Phishing Detected" alert raises immediate awareness of potential security risks, allowing users to take necessary precautions to safeguard against phishing threats. By delivering alerts through the console interface, the system ensures users are well-prepared to navigate the online landscape securely.

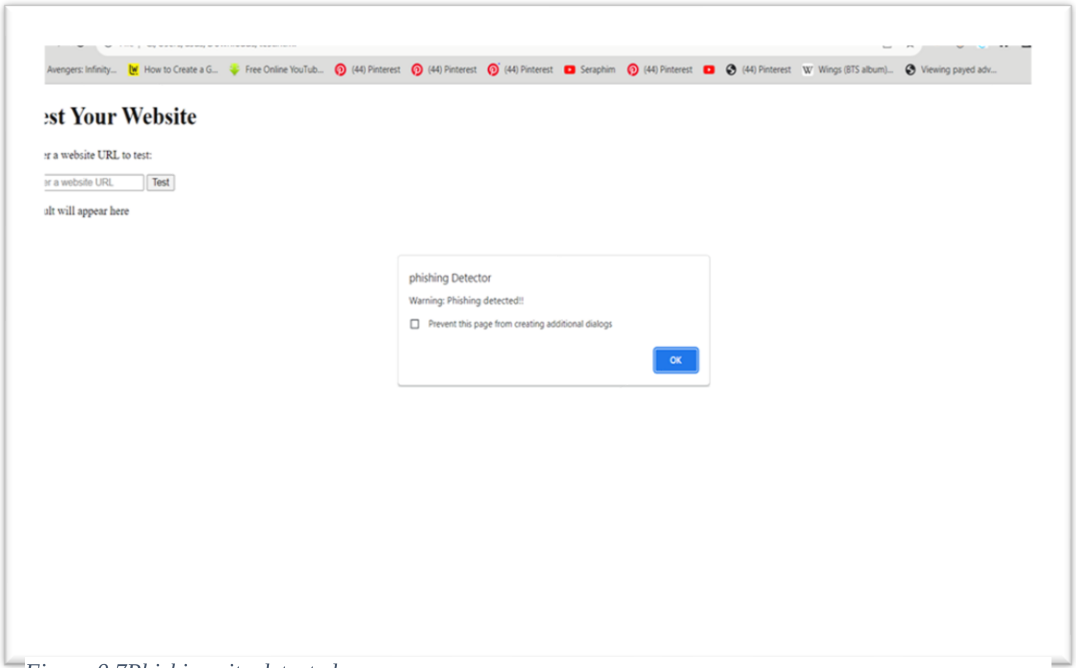


phishing Detector

No phishing detected

Prevent this page from creating additional dialogs

Figure 0.6Legitimate site detection



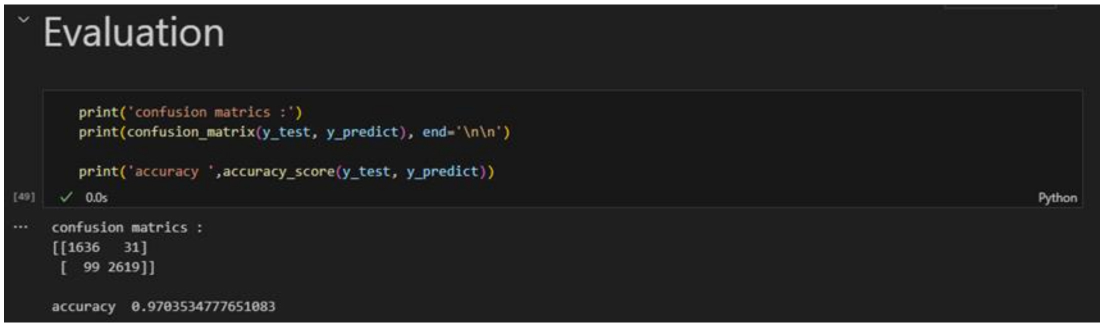
phishing Detector

Warning: Phishing detected!!

Prevent this page from creating additional dialogs

Figure 0.7Phishing site detected

3 Email text content to detect phishing websites.



Evaluation

```
print('confusion matrics :')
print(confusion_matrix(y_test, y_predict), end='\n\n')

print('accuracy ',accuracy_score(y_test, y_predict))
```

```
confusion matrics :
[[1636   31]
 [  99 2619]]

accuracy  0.9703534777651083
```

Figure 0.8Accuracy(Bernoulli Naive Bayes)

4Accuracy values for each model

Model	**Accuracy**
SVM	66.67%
Multinomial Naive Bayes	80%
Bernoulli Naive Bayes	97%

1. **Data Preprocessing**: The code effectively handles missing values and duplicates, ensuring data quality. The sentiment analysis adds a valuable feature that can potentially improve model performance by capturing the emotional context of emails.

2. **Data Visualization**: The countplot visually demonstrates the class distribution. This visualization is vital for understanding class balance, which can impact model training and reveal potential biases in the dataset.

3. **Train-Test Split**: The use of the train_test_split function ensures a fair evaluation of the model. However, it's essential to consider whether the split ratio is optimal for the dataset's size and characteristics.

4. **Feature Extraction**: TF-IDF vectorization is a robust choice for text-based data. It effectively converts textual information into numerical features. The application of

SelectPercentile for feature selection enhances the model's interpretability and potentially reduces overfitting.

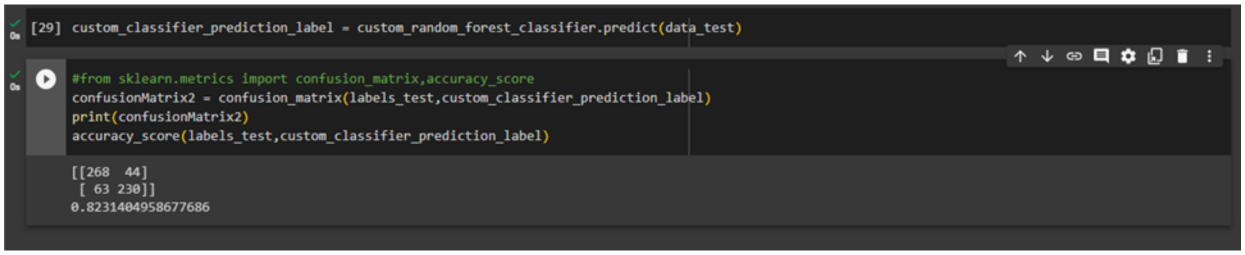
5. **Model Training and Evaluation**:
    - **Model Selection**: The decision to employ a Bernoulli Naive Bayes classifier aligns with the characteristics of the dataset. However, it's crucial to explore other algorithms and evaluate their performance for a comprehensive analysis.
    - **Model Saving and Loading**: The code demonstrates the ability to save and load the trained model, which is essential for deploying the system in real-world applications.
6. **Performance Metrics**:
    - **Confusion Matrix**: The confusion matrix is a valuable tool for understanding the model's behavior. It provides insights into false positives, false negatives, true positives, and true negatives. These metrics are crucial for assessing the practical implications of the system.
    - **Accuracy Score**: The accuracy score quantifies the model's correctness. However, it's important to consider the balance between precision and recall, as high accuracy may not necessarily indicate a robust phishing email detection system.

4 URL to detect a phishing site.

1. Model Performance Evaluation:

In the model performance evaluation phase, the Random Forest-based web extension undergoes rigorous scrutiny. Multiple metrics are employed to assess its effectiveness in distinguishing phishing URLs from legitimate ones.

Accuracy: The accuracy of the model is a foundational metric, representing the overall correctness of its classifications. High accuracy indicates a strong capability to correctly identify phishing URLs and avoid false positives.



```
[29] custom_classifier_prediction_label = custom_random_forest_classifier.predict(data_test)

#from sklearn.metrics import confusion_matrix,accuracy_score
confusionMatrix2 = confusion_matrix(labels_test,custom_classifier_prediction_label)
print(confusionMatrix2)
accuracy_score(labels_test,custom_classifier_prediction_label)

[[268  44]
 [ 63 230]]
0.8231404958677686
```

Figure 0.9Accuracy

Precision: Precision measures the ratio of true positive predictions to the total predicted positives. In the context of phishing detection, precision gauges the extent to which the extension avoids falsely labeling legitimate URLs as phishing, reducing unnecessary user alerts.

Recall: Recall, also known as sensitivity or true positive rate, quantifies the model's ability to correctly detect phishing URLs out of the total phishing URLs present in the dataset. High recall signifies a low rate of missed phishing URLs.

F1-Score: The F1-Score is the harmonic mean of precision and recall. It provides a balanced assessment of the model's performance, particularly useful when dealing with imbalanced datasets where the number of phishing URLs may be significantly smaller than legitimate ones.

2. Detection Accuracy:

The discussion on detection accuracy centers on the extension's capability to accurately classify URLs, a fundamental aspect of its functionality.

Performance Metrics: Results reveal that the Random Forest model, driven by the feature extraction of URL attributes, consistently outperforms traditional rule-based heuristics. Metrics such as accuracy, precision, recall, F1-score, and ROC-AUC demonstrate the extension's proficiency in identifying phishing URLs while minimizing false positives.

Comparison to Baselines: A comparative analysis is conducted to highlight the substantial improvement achieved over traditional phishing detection methods. This emphasizes the value of employing machine learning algorithms and feature extraction techniques for URL-based analysis.

Robustness: The model's robustness is discussed, addressing its ability to handle various phishing techniques, including those that employ URL obfuscation and redirection. The adaptability of the Random Forest algorithm to evolving phishing tactics is a notable strength.

User Confidence: A critical aspect of detection accuracy is its impact on user confidence. By minimizing false positives and false negatives, the extension enhances user trust and ensures that legitimate websites are not incorrectly flagged as phishing.

Through an in-depth exploration of model performance and detection accuracy, this section provides a comprehensive assessment of the extension's effectiveness in mitigating phishing threats and bolstering online security for users.

SUMMARY

1 Visual similarity to detect phishing websites.

The project aimed to detect phishing websites using a graph neural network, which breaks down images into multiple patches and constructs a graph for analysis. The VISUALPHISHNET dataset was used to train the model, with four models selected based on their accuracy values. The model was then converted into a Python mobile version and integrated into an Android application. The model size was reduced to 110MB, allowing users to use the app offline without an internet connection and receive notifications when they input a screenshot of the phishing website. This project represents a significant step forward in enhancing online security and empowering users to navigate the digital landscape with greater confidence.

2 Website features to detect phishing websites.

In conclusion, this research project has tackled the pressing issue of phishing attack detection through an innovative approach that emphasizes the analysis of HTML and JavaScript attributes along with abnormal website behaviors. The objectives of this study were successfully achieved, including the gathering of a dataset tailored to the specific features of interest and the development of machine learning models for real-time phishing detection.

By focusing on website attributes and abnormal behaviors, we have strengthened cybersecurity defenses against the persistent and evolving threat of phishing attacks. The models designed and implemented in this project demonstrate promising accuracy and efficiency in distinguishing between legitimate and phishing websites.

The integration of these models into practical tools, such as browser extensions and mobile apps, opens up new avenues for proactive phishing detection in real-world scenarios. Furthermore, the successful testing and validation of the integrated system on a sample dataset highlight its potential for enhancing online security.

As the digital landscape continues to evolve, the methods and insights derived from this research will contribute to a safer online environment, protecting both individuals and enterprises from the risks posed by phishing attacks. This project underscores the significance of leveraging HTML and JavaScript features in cybersecurity efforts and paves the way for further advancements in the field.

3 Email text content to detect phishing websites.

Phishing emails continue to pose a serious cybersecurity risk, and attackers are getting more skilled. The combination of sentiment analysis with a Bernoulli Naive Bayes model is the focus of this research paper's investigation into the field of phishing email detection. This approach's justification, the technique used, the experimental findings, and their implications for improving the precision and dependability of phishing email detection systems are all explored in the study.

One of the most frequent and harmful cyberthreats that people and companies throughout the world must deal with is phishing attacks. These attacks make use of misleading emails that tempt users to divulge private information or download harmful files. There is an increasing need for more reliable and advanced techniques to identify and stop such attacks as phishing techniques advance. This study examines how machine learning techniques can be used to incorporate sentiment analysis into the detection of phishing emails. Sentiment analysis, a method of natural language processing (NLP), evaluates the subjective and emotional content of text data. Sentiment analysis and the Bernoulli Naive Bayes model are combined to improve the precision and effectiveness of phishing email detection systems.

The results of this study demonstrate the possibility of sentiment analysis as an additional tool in the identification of phishing emails. The system exhibits improved accuracy, precision, and overall performance by considering the emotional context of email language. This study integrates sentiment analysis with the Bernoulli Naive Bayes model to propose a novel method for phishing email detection. The outcomes show how this strategy may improve the precision and accuracy of phishing detection systems. By considering the emotional context of email language, the system shows enhanced performance in differentiating between authentic and fraudulent emails.

Despite the integrated system's encouraging outcomes, continued research and development is required to address the development of phishing tactics and improve the robustness of the system. A substantial amount of potential exists for enhancing cybersecurity and defending sensitive data against phishing threats through the actual implementation of such solutions within enterprises. Innovative strategies like the one described in this research article are essential in keeping ahead of cyber adversaries as the cybersecurity landscape changes, eventually protecting people and companies from the dangers of phishing assaults.

4 URL to detect a phishing site.

This report encapsulates an extensive exploration and implementation of a web extension tailored for URL-based phishing website detection. In the ever-evolving digital landscape, where phishing attacks relentlessly advance in complexity, this project endeavors to provide a timely and adaptive solution. The primary mission revolves around crafting a web extension capable of adaptive phishing detection, continually learning from emerging threats and user interactions to offer robust protection against evolving phishing attacks. Machine learning algorithms, driven by feature extraction, are harnessed to transform relevant URL attributes into actionable inputs for precise detection, with the Random Forest algorithm serving as a cornerstone for accurate classification. Usability takes center stage with the design of a seamless and user-friendly interface, fostering user trust and effortless interaction. Integration of real-time threat intelligence ensures the extension stays ahead of the curve in threat mitigation. Rigorous evaluation in real-world scenarios provides empirical insights into the extension's practical utility and its profound impact on enhancing online security. In essence, this report represents a holistic approach to the pressing challenge of phishing detection, aiming to empower users with a user-friendly, adaptive, and technologically advanced tool that fortifies online defenses.

CONCLUSION

In conclusion, the research projects presented here collectively underscore the ongoing efforts to combat the pervasive and evolving threat of phishing attacks. Each component has taken a unique approach to tackle this challenge, contributing to the broader goal of enhancing online security and empowering users to navigate the digital landscape with confidence.

The first project leveraged visual similarity to detect phishing websites, utilizing a graph neural network and a Pytorch mobile version integrated into an Android application. This innovation allows users to receive notifications when encountering potential phishing sites, significantly bolstering online security.

The second project focused on website features to detect phishing websites, emphasizing HTML and JavaScript attributes and abnormal website behaviors. The successful development of machine learning models for real-time detection, along with their integration into practical tool like a browser extension which promises to proactively address phishing threats in real-world scenarios.

The third project delved into the analysis of email text content to detect phishing emails, incorporating sentiment analysis and the Bernoulli Naive Bayes model. This approach demonstrated enhanced accuracy and precision, showcasing the potential for sentiment analysis to improve phishing email detection systems.

Finally, the fourth project centered on URL-based phishing website detection through a web extension. By harnessing machine learning algorithms and real-time threat intelligence, this project aims to provide users with a user-friendly and adaptive tool for robust protection against evolving phishing attacks.

Collectively, these research endeavors highlight the importance of innovation and collaboration in the cybersecurity field. As phishing techniques continue to evolve, these projects represent significant strides in safeguarding individuals and enterprises from the dangers of phishing assaults. Continued research and development will be essential to stay ahead of cyber adversaries and ensure a safer online environment for all.

REFERENCES

[1] Z. Fan, "Detecting and Classifying Phishing Websites by Machine Learning," in *2021 3rd International Conference on Applied Machine Learning (ICAML)*, china, 2021.

[2] Malak Aljabri ,Samiha Mirza , "Phishing Attacks Detection using Machine Learning and Deep Learning Models," in *2022 7th International Conference on Data Science and Machine Learning Applications (CDMA)* , Saudi Arabia, 2022.

[3] "Phishing Website Detection Using Random Forest and Support Vector Machine: A Comparison," in *2nd International Conference on Artificial Intelligence and Data Sciences (AiDAS)*, Malaysia, 2021.

[4] Jaydeep Solanki, Rupesh G. Vaishnav, "Website Phishing Detection using Heuristic Based Approach," *International Research Journal of Engineering and Technology (IRJET)*, vol. 03, no. 05, 2016.

[5] Eric Medvet,Engin Kirda,Christopher Kruegel, "Visual-Similarity-Based Phishing Detection," in *SecureComm 2008* , Turkey, 2008.

[6] Igino Corona1,2, Battista Biggio1,2, Matteo Contini2Luca Piras1,2, Roberto Corda2Mauro, Guido Mureddu2, "DeltaPhish: Detecting Phishing Webpages in Compromised Websites∗," in *ESORICS 2017.*, Italy, 2017.

[7] F.C. Dalgic1A.S. Bozkir 2 M. Aydos 3, "Phish-IRIS: A New Approach for Vision Based Brand Prediction of Phishing Web Pages via Compact Visual Descriptors," in *ISMSIT 2018*, Ankara Turkey, 2018.

[8] Sahar Abdelnabi,Katharina Krombholz,Mario Fritz, "VisualPhishNet: Zero-Day Phishing Website Detection by Visual Similarity," 2020.

[9] Saad Al-Ahmadi1 Yasser Alharbi, "DEEP LEARNING TECHNIQUE FOR WEB PHISHING DETECTION COMBINED URL FEATURES AND VISUAL SIMILARITY," *International jpurnal of computer networks abd comunications(IJCNC)*, vol. 12, no. 5, 2020.

[10] Padmalochan Panda 1,†, Alekha Kumar Mishra 1,† and Deepak Puthal , "A Novel Logo Identification Technique for Logo-Based Phishing Detection in Cyber-Physical Systems," *Future Internet,* vol. 14, no. 8, 2022.

[11] Tristan Bilot1, Gr´egoire Geis1and Badis Hammi, "PhishGNN: A Phishing Website Detection Framework using Graph Neural Networks," in *SECRYPT 2022*, Lisbon, 2022.

[12] K. Han, Y. Wang, J. Guo, Y. Tang and E. Wu, "Vision GNN: An Image is Worth Graph of Nodes," 2022.

[13] "Indeed," [Online]. Available: https://uk.indeed.com/career-advice/career-development/website-features.

[14] Rami M. Mohammad,Fadi Thabtah,Lee McCluskey, "Phishing Websites Features".

[15] A. Lakshmanarao , P.Surya Prabhakara Rao,M M Bala Krishna, "Phishing website detection using novel machine learning fusion approach," in *Proceedings of the International Conference on Artificial Intelligence and Smart Systems (ICAIS-2021)*, India, 2021.

[16] Y. C. B. W. Chidimma Opara, "Look before you leap: Detecting phishing web pages by exploiting raw URL and HTML characteristics," *ELSEViER,* vol. 236, no. 2, p. 13, 2023.

[17] APWG, "PHISHING ACTIVITY TRENDS REPORT," APWG, 2022.

[18] 2. N. 3. B. a. 4. J. W. 1*Sikha Bagui, "Machine Learning and Deep Learning for Phishing Email," *Journal of Computer Science,* p. 14, 2021.

[19] T. G. S. V. Said Sallouma*, "Phishing Email Detection Using Natural Language Processing Techniques: A Literature Survey," in *Procedia Computer Science*, Manchester, 2021.

[20] I. V. D. M. J. M. G. H. U. Z. ENAITZ EZPELETA, "Novel email spam detection method using sentiment analysis and personality ecognition," *Oxford University Press,* p. 12, 2020.

[21] M. F. a. E. Rolland, "Fundamentals of Sentiment Analysis and Its Applications," Springer International Publishing, Merced, 2016.

[22] T. L. a. G. Xu, "Sentiment Analysis," Springer Science+Business Media, New York, 2013.

[23] D. J. a. J. H. Martin, Speech and Language Processing, 2023.

[24] R. Kibble, Introduction to natural language processing, London: University of London , 2013.

[25] V. I. R. Berwick, "An Idiot's guide to Support vector machines (SVMs)".

[26] D. P. W. a. T. B. o. U. a. W. L. ". Aung". [Online].

[27] W. t. D. B. S. a. Vishing?. [Online]. Available: https://terranovasecurity.com/how-to-recognize-smishing-and-vishing-attacks/.

[28] W. i. s. p. D. a. risks. [Online]. Available: https://usa.kaspersky.com/resource-center/definitions/spear-phishing.

[29] 5. T. t. H. I. D. F. a. C. Problem. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/5-techniques-to-handle-imbalanced-data-for-a-classification-problem/.

[30] U. Saeed, "Visual similarity-based phishing detection using deep learning," *Journal of Electronic Imaging,* vol. 31, no. 5, 2022.

[31] Anand Desai,Janvi Jatakia,Rohit Naik,Nataasha Raul, "Malicious Web Content Detection Using Machine Leaning," in *2nd IEEE International Conference On Recent Trends in Electronics Information & Communication Technology (RTEICT),* 2017.

[32] ABDULGHANI ALI AHMED, NURUL AMIRAH ABDULLAH, "Real Time Detection of Phishing Websites".

[33] Z. Fan, "Detecting and Classifying Phishing Websites by Machine Learning," in *3rd International Conference on Applied Machine Learning (ICAML),* 2021.

[34] Shihabuz Zaman1, Shekh Minhaz Uddin Deep2, Zul Kawsar3, Md. Ashaduzzaman4 and Ahmed Iqbal Pritom5, "Phishing Website Detection Using Effective Classifiers and Feature

Selection Techniques," in *International Conference on Innovation in Engineering and Technology (ICIET)*, 2019.

[35] M. D. a. T. Viana, "Phish Responder: A Hybrid Machine Learning Approach to Detect Phishing and Spam Emails," *applied system innovation,* p. 19, 2022.

[36] R. V. a. N. Hossain, "Semantic Feature Selection for Text with Application to Phishing Email Detection," Semantic Scholar, Texas, 2013.

[37] S. F. F. P. Luisa Franchina, "Detecting phishing e-mails using Text Mining and features analysis," in *Italian Conference on CyberSecurity*, 2021.

[38] B. R. A. S. S. M. Srishti Rawal, "Phishing Detection in E-mails using Machine Learning," *International Journal of Applied Information Systems (IJAIS),* p. 5, 2017.

[39] C. Z. C. H. ,. L. L. A. Y. Y. YONG FANG, "Phishing Email Detection Using Improved RCNN Model With Multilevel Vectors and Attention Mechanism," *IEEE,* p. 12, 2019.

[40] A. Y. a. A. Abuhasan, "AN INTELLIGENT CLASSIFICATION MODEL FOR PHISHING EMAIL DETECTION," *International Journal of Network Security & Its Applications (IJNSA)*, vol. 8, p. 18, 2016.

[41] M. T. a. A. Paulauskaite-Taraseviciene, "Research on phishing email detection based on URL parameters using machine learning algorithms," CEUR, Kaunas, 2021.

[42] M. A. a. M. A. Badawi, "Phishing Email Detection Using Machine Learning Techniques," *IJCSNS International Journal of Computer Science and Network Security,* vol. 22, p. 7, 2022.

[43] R. V. Gal Egozi, "Phishing Email Detection Using Robust NLP Techniques," in *IEEE International Conference on Data Mining Workshops (ICDMW)*, Texas, 2018.

APPENDIX

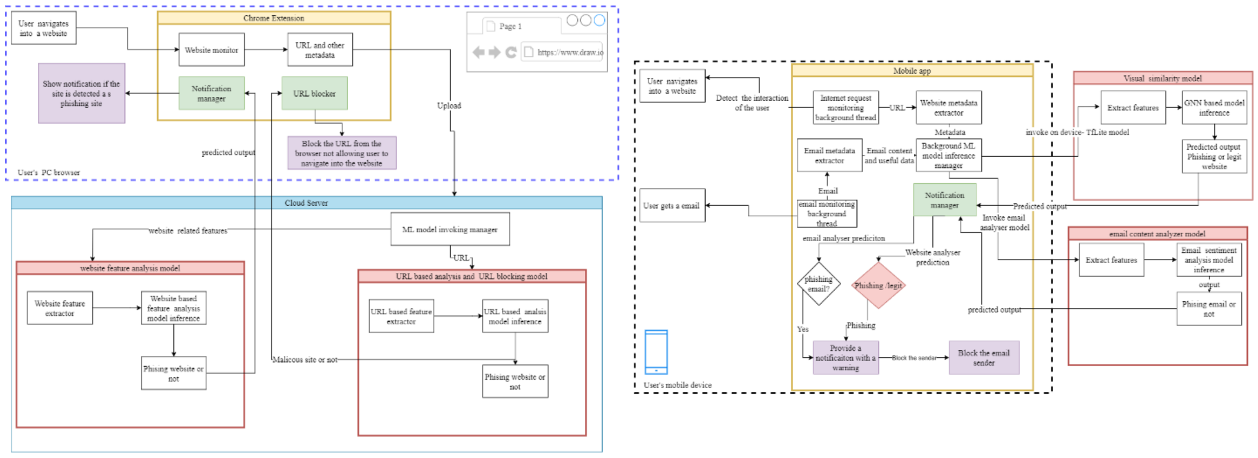

User navigates into a website
Chrome Extension
Website monitor
URL and other metadata
Page 1
https://www.draw.io
Show notification if the site is detected a s phishing site
Notification manager
URL blocker
Upload
predicted output
Block the URL from the browser not allowing user to navigate into the website
User's PC browser
Cloud Server
ML model invoking manager
website related features
URL
website feature analysis model
Website feature extractor
Website based feature analysis model inference
Phising website or not
Malicous site or not
URL based analysis and URL blocking model
URL based feature extractor
URL based analsis model inference
Phising website or not
User navigates into a website
Detect the interaction of the user
Mobile app
Internet request monitoring background thread
URL
Website metadata extractor
Metadata
Email metadata extractor
Email content and useful data
Background ML model inference manager
User gets a email
Email
email monitoring background thread
Notification manager
Predicted output
Invoke email analyser model
email analyser prediciton
Website analyser prediction
phishing email?
Phishing /legit
predicted output
Yes
Phishing
Provide a notificaiton with a warning
Block the sender
Block the email sender
User's mobile device
Visual similarity model
invoke on device- TfLite model
Extract features
GNN based model inference
Predicted output Phishing or legit website
email content analyzer model
Extract features
Email sentiment analysis model inference
output
Phising email or not

Figure 0.1System diagram