Phishing Website Detection

By
Aditya Chauhan

A project report presented to the University of Ottawa

In partial fulfilment of the requirements for the degree of

Master of Computer Science

In the program of

Applied Artificial Intelligence

Under the supervision of
Asst. Professor Paula Branco
Professor Guy-Vincent Jourdan

Abstract

The rapid internet expansion, with over a third of the global population now online, has unfortunately fuelled the rise of cybercrime. Traditional detection methods, like blacklists, struggle to keep pace with increasingly sophisticated phishing attacks. This project addresses this critical gap by exploring advanced, adaptable solutions.

By harnessing advancements in technology and computational resources, the project delves into machine learning and deep learning techniques for phishing detection. It also examines existing methods such as URL, image, and web content analysis, in conjunction with machine learning approaches. The project proposes the development of innovative detection methods inspired by recent breakthroughs in the field.

Finally, the integration of machine learning and deep learning models with a micro-service enables real-time classification of websites, providing a comprehensive approach to combating cyber threats.

Acknowledgement

I would like to express my sincere appreciation to my project supervisors, Assistant Professor Paula Branco and Professor Guy-Vincent Jourdan, for their unwavering support, invaluable guidance, and enlightening discussions throughout this project. I am also grateful for the continuous moral and intellectual support offered by my project partner, Sahej Chandok. Julien Cassagne played a pivotal role in the data collection process and the development of the micro-service architecture. Furthermore, I extend my thanks to the University of Ottawa's School of Electrical Engineering and Computer Science for equipping me with the necessary tools and knowledge to execute this project effectively.

Table of Contents

1. Introduction

One of the fastest-growing problems in the digital landscape today is the rise of phishing websites. In such attacks, malicious actors aim to exploit human psychology and internet vulnerabilities to deceive users [1]. These websites are cleverly designed to mimic legitimate platforms, such as financial institutions, e-commerce websites, and corporate organizations. They serve as gateways for identity theft, fraud and launching further cyber attacks. The increasing sophistication of these threats demands correspondingly advanced solutions capable of understanding patterns, anomalies, and nuances in data to identify phishing websites.

In its formative years, phishing was mainly limited to using email as its primary source of deception. Cybercriminals relied on crafting generic emails that attempted to trick recipients into revealing sensitive information, often posing as legitimate organizations or individuals. These emails would typically include a link to a fraudulent website, though often with a relatively basic appearance. However, with increasing awareness among internet users about email-based phishing schemes and the continuous improvement of email filtering systems, cybercriminals have adapted and refined their tactics [2]. This adaptation has resulted in a significant rise in phishing websites—sophisticated online platforms designed to closely mimic legitimate sites such as banking, e-commerce, or social media platforms. These evolved phishing websites have become channels for elaborate attacks, incorporating visually appealing design elements, authentic-looking URLs, and persuasive language to exploit unsuspecting users' trust.

Given the critical and ever-increasing importance of cybersecurity, we need more accurate and adaptable methods for detecting phishing websites. Here is a breakdown of common detection approaches:

- Blacklist and Whitelist Detection: While simple, blacklists (known bad sites) and whitelists (known good sites) fail to catch new phishing websites.
- Uniform Resource Locator Analysis: This method examines the website's address (URL) for suspicious characteristics. Phishing URLs often mimic legitimate sites but might have subtle differences in domain names or IP addresses.
- Web Content Analysis: Here, the focus is on the actual content of the website. Phishing sites often contain elements like suspicious forms, unusual field names, or odd source code that differ from legitimate sites.
- Machine Learning Detection: ML algorithms can learn from large datasets of phishing and legitimate sites, identifying patterns and anomalies. This allows for proactive detection of even brand-new phishing attempts.

The key contributions of this project are outlined as follows:

- The project delves into existing research on phishing detection, assessing the efficacy of various strategies. It provides a comprehensive review of different methodologies explored in the relevant literature.
- It introduces two innovative approaches to phishing detection: a feature-extraction-based method (Section 3.1) and an image classification-based method (Section 3.3).
- It implements two methodologies proposed in the studies of Wei et al. [3] and Rao et al. [4], respectively.

- The project incorporates the integration of machine learning and deep learning models with a micro-service. This facilitates rapid testing of the implemented approaches and the detection of phishing websites.

The report is structured as follows: Section 2 provides a *literature review*, Section 3 delves into the *methodologies* employed, Section 4 outlines the *experimental setup* for each method, and Section 5 presents the *results*. Additionally, Section 6 covers the *deployment setup*, while Sections 7 and 8 explore potential *future works* and *limitations*, respectively. The report is concluded in Section 9.

2. Literature Review

The field of phishing detection has been an area of research for a long time. Over the years, it has become a never-ending game of cat and mouse between the developers trying to identify and stop phishing attempts and the attackers who are constantly evolving their approaches. The blacklist/whitelist-based strategy is a popular technique in the area. It uses a list of known phishing websites, including URLs, IP addresses, and other information. However, because these lists had to be updated frequently, they were unable to handle the massive amount of webpages on the internet. [5][6]

Phishing detection has advanced significantly due to the growing application of machine learning and deep learning in cybersecurity. Several studies [7-25] have been conducted using URLs, website images, website source codes, and phishing kits to extract statistical, lexical, and domain-specific features, image representations, and word embeddings, which are then used to train machine learning and deep learning algorithms. These algorithms have demonstrated

remarkable efficacy in identifying phishing attempts using various datasets. Table 1 (Appendix A) lists and compares a few studies conducted in this area. These are broadly categorized according to the resources they use for feature extraction, representation and embedding generation, or model training purposes. For the purpose of this study, we focus on the use of URLs, images, and source code files in the existing literature and analyze the results of these approaches.

2.1. URL-based methods

Researchers have utilized the findings to support the notion that phishing website URLs exhibit intricate and distinctive characteristics. These features can be harnessed to derive relevant attributes for the classification of phishing websites using machine learning models. Mourtaji et al [7], Sahingoz et al [8], Ucar et al [9], and Afzal et al [10] have conducted extensive work. Their research involves extracting statistics-based heuristics, including URL length, the number of special symbols ('.', '@', '-') in the URL, and the presence of redirection links in the website source. Additionally, studies by Cheng et al [11], Verma et al [12], and Peng et al [13] primarily focus on generating features through expert knowledge and lexical analysis of the URL. These approaches also incorporate third-party features, such as Alexa page rankings and domain information obtained from WHOIS records.

CANTINA+ [14], an advanced anti-phishing solution, presents itself as a 'Feature-rich Machine Learning Framework' because of its comprehensive approach to detecting phishing websites through its elaborate features. This study yielded encouraging results after a thorough review of the various features. Canali et al [15]. introduced Prophiler in their study, which employs a total of 33 features

derived from the analysis of URL and host information and may be broadly classified as syntactical, DNS-based, whois-based, and geoIP-based. Their findings also highlight the relevance of such features for training machine learning systems. Bahnsen et al [16] also extract relevant features from the URLs to train an ML model. However, they introduce an innovative approach to represent the characters in a URL. This involves translating them into 128-dimensional embeddings and subsequently inputting this information into an LSTM network. Al-Ahmadi et al [17] proposed a deep learning based method named *PDGAN* that depends on the website's URL to achieve reliable performance. It leverages the generative adversarial network (GAN) that comprises two components, a generator made of LSTM network to create synthetic phishing URLs and a CNN as a discriminator to decide whether URLs are phishing or legitimate.

2.2. Image-based methods

With the advent of the internet, attackers are able to replicate webpages from legitimate websites with a high degree of visual similarity. This increases users' vulnerability to falling victim to phishing attacks. In response to such techniques, researchers have conducted extensive investigations into the impact of utilizing images for the identification of phishing websites. Haruta et al [18] focused on elements that determine the visual content of websites, such as images, their positions, colours, fonts, and more. Hara et al [19] employed ImgSeek to assess the similarity between image pairs from authentic webpages and their corresponding phishing counterparts. Their approach achieved a detection rate exceeding 80% for phishing images.

In more recent developments, Wei et al [3] and Ouyang et al [20] have employed advanced deep learning techniques in their research endeavours. Wei et al transformed URLs into images through character-level one-hot encoding and the subsequent generation of representations. On the other hand, Ouyang et al utilized graphs derived from the Document Object Model (DOM) tree of HTML source codes as a basis for their analysis.

2.3. Code-based methods

Roopak and Thomas [21] introduced a novel method that relied on HTML source code for creating a mechanism to detect phishing pages. In their proposed solution, they initially conduct a comparison of webpages through attribute matching of HTML tags. Subsequently, they assess the textual content of the pages to determine their cosine similarity. However, their approach may not work under code obfuscation. Li et al [22] employ HTML source code for extracting pertinent features in their proposed stacking model. Their research emphasizes the combined impact of feature extraction from both URLs and HTML. Additionally, they highlight the significance of HTML string embeddings generated through Word2Vec in enhancing the detection of phishing websites.

Zhang et al [23] introduced CrawlPhish, a framework designed for the automated detection and categorization of client-side cloaking techniques employed by phishing websites. Their approach involves leveraging JavaScript files to extract code structure features, along with visual features obtained after exploring every possible execution path through forced execution. However, due to the challenges associated with obtaining JavaScript source code files, researchers resort to utilizing more readily accessible HTML files. More recently,

Rao et al [4] introduced a novel method for detecting phishing sites, employing word embeddings derived from plain text and domain-specific text extracted from the source code. The utilized word embedding algorithms were classified into frequency-based models like TF-IDF and prediction-based models such as Word2Vec and GloVe. While their results showcase a promising potential for this technique, it's essential to note that the evaluation was conducted on a relatively small dataset.

3. Methodology

The project presents four methods for phishing detection that leverage machine learning and deep learning technologies. The following sub-sections provide the system design and methodology followed for each solution.

3.1. URL-based feature extraction

A critical step in machine learning for website security involves extracting informative characteristics, or features, from raw website data. This process is crucial for classifying websites as malicious (phishing) or legitimate by analyzing their Uniform Resource Locator (URL). Numerous studies have been conducted to identify these informative features for machine learning algorithms. In this specific experiment, we focus on exploring the effectiveness of existing features, identified through a review of background literature presented in Section 2.1. The primary objective is to assess if these established features remain effective as phishing techniques become more sophisticated.

Table 2 provides a breakdown of the 12 features utilized for this method. It details the source research articles from which each feature was identified, along with a brief explanation and an example to further illustrate its purpose.

Table 2. Features extracted from URLs.

Source Article	Feature Name	Description	Example
Xiang et al. **[CANTINA+: A Feature rich Machine Learning Framework for Detecting Phishing Web Sites]**	Is domain an IP address	A binary value is used to indicate whether the domain name of a URL is an IP address or not.	http://145.145.1.0/signup Value: **1**
	Contains embedded Domain	Examines the presence of domain names in the path part of the URL.	https://www.domain.com/login?dest=h6f78bg-www.ebay.com/ Value: **1**
	Number of Dots	Counts the number of dots in the URL. Phishing URLs tend to contain more dots than legitimate sites.	https://ietresearch**.**onlinelibrary**.**wiley**.**com/doi/full/10**.**1049/iet-ifs**.**2013**.**0202 Value: **6**
	Out of position Top-level Domains	Checks if a TLD appears in an unusual position in the URL.	http://cgi.ebay.com.ebaymotors.732issapidll.private99dll.qqmotorsqq.ebmdata**.com** Value: **1**
	Number of Sensitive Words	Counts the number of the eight sensitive words proposed by Garera et al. ["secure", "account", "webscr", "login", "ebayisapi", "signin", "banking", "confirm"]	https://www.example.com/**login** Value: **1**
Li et al. **[A Stacking Model Using URL and HTML Features for Phishing Webpage Detection]**	Number of Special Characters	Counts the number of characters that have a reserved special action in URL, including '@', '–' and '~'. For example, all strings at the right of the '@' symbol are ignored when parsed.	https://www.example.com/summer**-**clothing/filter?color_profile=dark_grey Value: **3**
	HTTPS present	A binary value is used to indicate whether the protocol of a URL is "https" or not.	**https:**//perishablepress.com/stop-using-unsafe-characters-in-urls/ Value: **1**
	Length of URL	The length of the URL is a significant factor in detecting phishing URLs. Phishing URLs tend to be longer than legitimate URLs.	https://www.quora.com/What-are-some-good-examples-of-URLs Value: **57**

Source Article	Feature Name	Description	Example
Anand et al. **[A Stacking Model Using URL and HTML Features for Phishing Webpage Detection]**	Number of Digits	Counts the number of digits in an URL. Presence of digits strongly indicate to the URL being phishy.	http://cgi.ebay.com.ebaymotors.**732**issapidll.private**99**dll.qqmotorsqq.ebmdata.com Value: **5**
	Number of Queries	Counts the number of queries present in the URL. Counted as the number of '&' symbol in the URL.	https://www.example.com/products/products.asp?N=200063**&**Ne=500955**&**ref=foo%2Cbar**&**Cn=Accessories. Value: **3**
	Shannon Entropy	Legitimate URLs always have meaningful names causing their entropies to be lower than phishy URLs which tend to have gibberish text.	https://www.quora.com/What-are-some-good-examples-of-URLs Value: **4.393**
	Number of Subdirectories	Counts the number of subdirectories within the URL's path.	https://www.example.com/index.shtml/discuss/category/school/061121/html/interview/category/health/070223/html/category/business/070302/html/category/community/070413/html/FAQ.htm Value: **19**

3.1.1. Ensemble Model

While traditional machine learning models can leverage these features for phishing detection, this project proposes a more robust approach. It aims to construct an ensemble model, combining the strengths of multiple machine learning algorithms to accurately distinguish between phishing and legitimate websites.

This work leverages a novel ensemble model constructed using a Soft-Voting based Classifier. This approach combines the strengths of various tree-based

models including Random Forest, Extra Trees, Gradient Boosting, and XGBoost Random Forest Classifiers. Additionally, Bagging and AdaBoost techniques are incorporated, utilizing SVC and Logistic Regression as their respective base estimators. The Soft-Voting Classifier employed in the ensemble model functions as follows: each individual base estimator predicts the probability distribution across all classes. The final prediction is made by averaging these probabilities, ultimately selecting the class with the highest average probability. Such a classifier can be useful for a set of equally well performing models in order to balance out their individual weaknesses.

3.2. URL-based character level encoding

This method also utilizes the URL address of a webpage to extract relevant information for classification of phishing and legitimate websites. We employ a methodology inspired by the experiments proposed by Wei et al. [3], in which they present a one-hot character-level encoding technique and convolutional neural networks (CNN) for the analysis of URLs.

3.2.1. Character-level Encoding

This work adopts a pre-processing approach similar to Zhang et al. [25]. They employ one-hot encoding to convert each character within the URL into a numerical representation. This maximizes the available space for encoding characters deemed more informative than the protocol identifiers (http:// or https://). Specifically, these protocol designations are removed from URLs to allow for the encoding of 7-8 additional characters.

Following one-hot encoding, a set of additional pre-processing techniques are used to generate embeddings from the encoded URLs. These embeddings serve as the input for the Convolutional Neural Network (CNN) proposed by the authors. Inspired by [25], the encoding utilizes the following set of 70 characters.

a b c d e f g h i j k l m n o p q r s t u v w x y z 0 1 2 3 4 5 6 7 8 9 - , ; . ! ? : ' ' ' / \ | _ @ # $

To optimize the network architecture, a maximum URL length of 256 characters is established for analysis. The one-hot encoded URLs are then transformed into embeddings with a size of 128x16 for CNN training. Details regarding the dataset statistics are presented in Table 3.

Table 3. Statistics of URL dataset used in the experiments.

Number of all URLs	2839
Number of phishing URLs	1410 (49%)
Number of legitimate URLs	1429 (51%)
Number of characters in the CNN input vector	256
The longest URL	600 characters
The shortest URL	16 characters
Average URL size	65

This project replicates the pre-processing steps described above. However, it deviates from the authors in two key ways.

I. Label Encoding vs. One-Hot Encoding: Instead of character-level one-hot encoding, this project utilizes a character-level label encoding technique based on the provided dictionary. This choice aims to explore

the effectiveness of label encoding compared to one-hot encoding for this task.

II. Larger Embedding Size: The project employs a larger embedding size of 256 x 16 that deviates from the authors' 128 x 16 size. The objective is to extract the maximum amount of contextual information from the encoded URLs, potentially leading to improved model performance.

3.2.2. Convolutional Neural Network

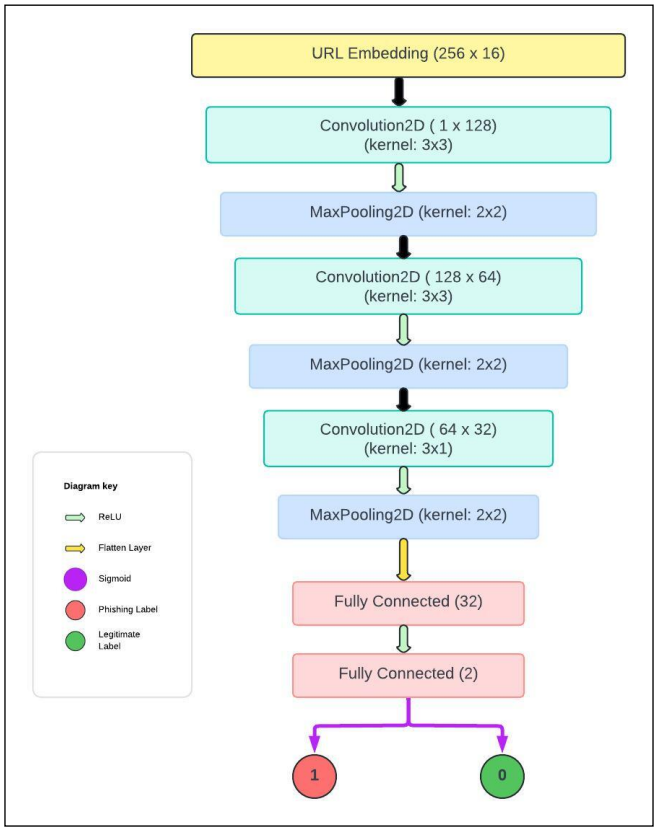
This work leverages a Convolutional Neural Network (CNN), a deep learning architecture adept at learning patterns directly from data. CNNs excel at identifying patterns in various data formats, including images, speech, and audio signals. They achieve this through three primary types of layers:

- Convolutional Layer: Extracts features from the input data using learnable filters.
- Pooling Layer: Downsamples the data, reducing complexity and computational cost.
- Fully Connected (FC) Layer: Performs high-level reasoning and classification based on the extracted features.

Taking reference of the CNN architecture proposed by the authors, this method employs a similar convolutional network (Fig 1). The inputs are the embeddings of shape 256 × 16. These embeddings are then processed through standard CNN layers. The final layer is a sigmoid layer, preceded by a

fully-connected layer with two outputs corresponding to the classification (legitimate or phishing URL).

Fig 1. CNN architecture used for Method 2.



3.3. Webpage-based image classification

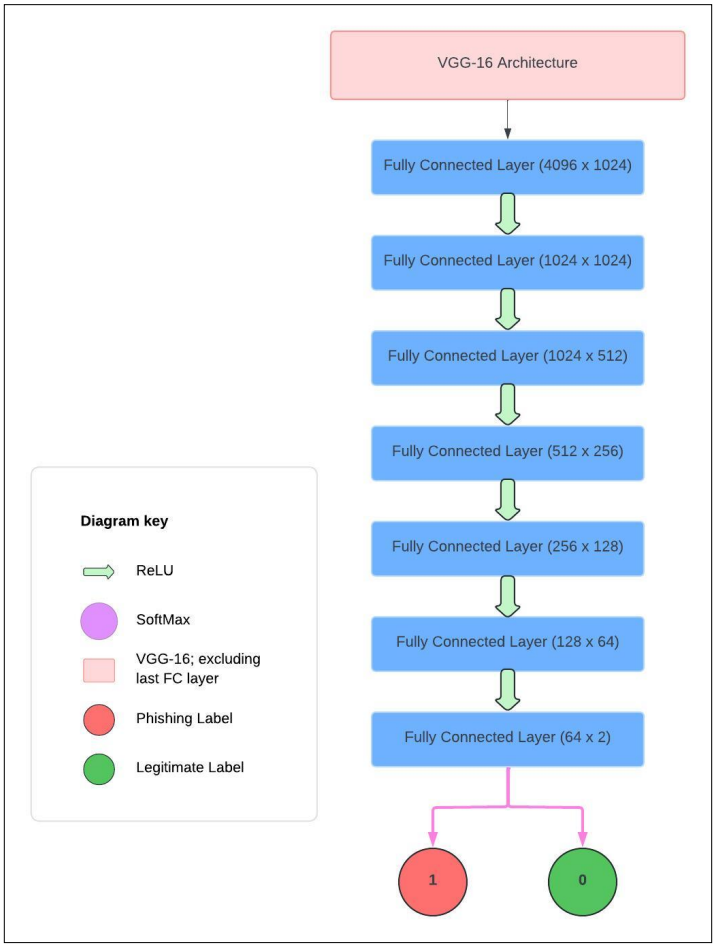
This method proposes a novel approach for detecting phishing websites. It leverages deep learning-based image classification techniques, specifically focusing on Convolutional Neural Networks. It delves into the progression of CNN architectures, spanning from foundational models such as LeNet and AlexNet to more intricate designs like VGG-16, ResNet, and EfficientNet. Notably, this method aims to utilize the well-established VGG-16 architecture for website

image classification. Introduced by Simonyan & Zisserman in 2014, VGG-16 is renowned for its simplicity and effectiveness in image recognition tasks.

Furthermore, this section delves into the preprocessing techniques used to prepare website image data for deep learning models. Additionally, it explores transfer learning strategies, which involve fine-tuning pre-trained models like VGG-16 on specific datasets to expedite convergence and enhance performance.

3.3.1.VGG-16 Architecture

Fig 2. VGG-16 network used for Image classification



VGG-16 is distinguished by its depth, comprising 16 layers, including 13 convolutional layers and 3 fully connected layers, totalling 138 million parameters. The pre-trained model is trained on images spanning over 200 categories, yielding 1000 outputs. Therefore, for this project's scope, we exclude the final fully connected layer and append a custom fully connected block. Figure 2 depicts the architecture of the added classification block within the VGG-16 framework.

Figure 3 showcases sample images utilized to train the proposed CNN model. These website images undergo resizing, transformation into tensors, and normalization to ensure uniformity in input size and format.

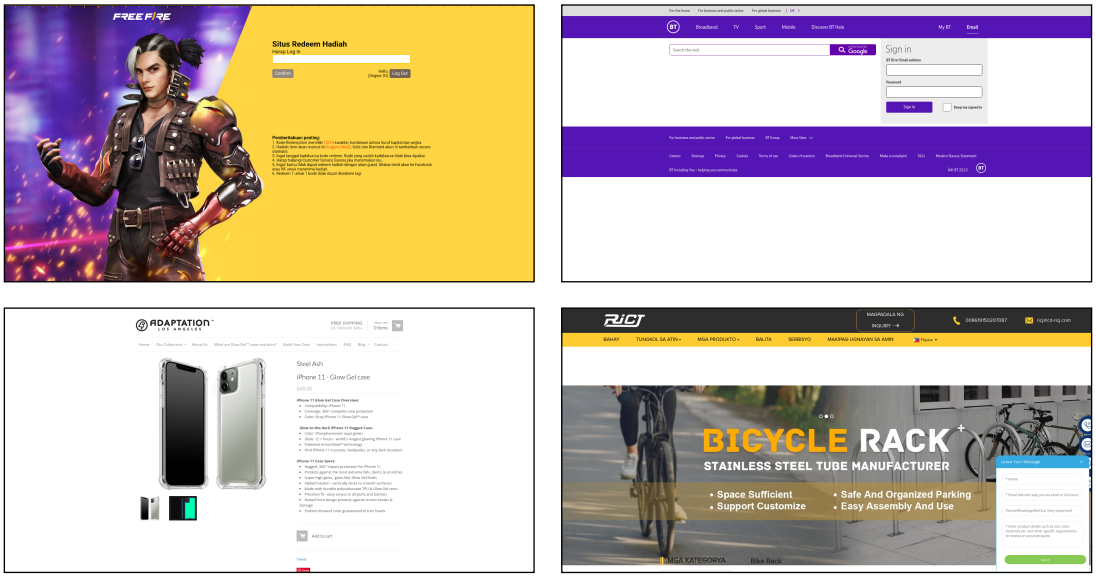


Fig 3. Sample images used as input for CNN model. (Top row: Phishing Website examples, Bottom row: Benign Website examples)

3.4.HTML source-code text-based classification

This method integrates techniques introduced by Rao et al. [4] to detect phishing websites by utilizing word embeddings derived from both plain text and domain-specific text extracted from source code. The proposed approach employs two distinct input types: PlainText (PT), which covers all text within a webpage, and domain-specific text (DST), extracted from source code locations such as copyright notices, titles, and headers. The rationale behind selecting DST lies in its higher probability of containing domain information about the designed website, which can also provide information for phishing sites. Feature vectors are generated for these inputs and then utilized as input for machine learning algorithms for classification.

Additionally, the authors introduce an ensemble model and a multimodal model in their study. The multimodal model combines all vectors generated using the word embedding algorithms for a website. To assess these models, a range of machine learning algorithms including Random Forest, SVM, Logistic Regression, Decision Tree, and XGBoost were employed.

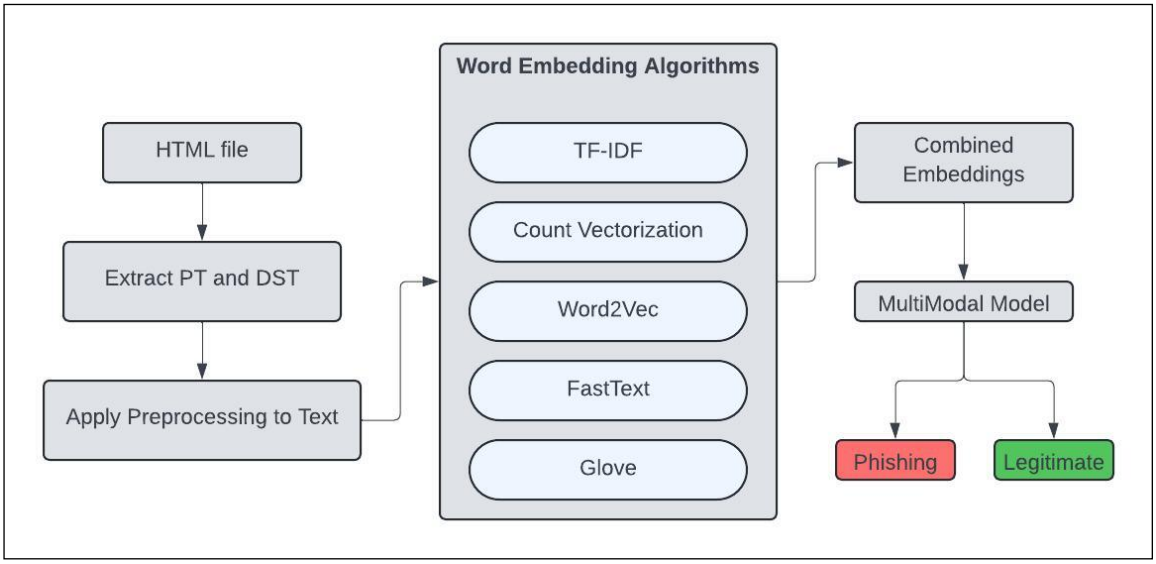
3.4.1.Word Embeddings

In this approach, the goal is to utilize insights derived from the authors' multimodal model and implement it for both PT and DST extracted from HTML files from the collected source code files. The authors conduct extensive experiments to evaluate the performance of various word embedding algorithms. These encompass frequency-based methods such as TF-IDF and Count Vectorization, as well as prediction-based techniques like Word2Vec,

FastText, and GLoVe. Additionally, Word2Vec and FastText offer two algorithms each—Continuous Bag of Words (CBOW) and Skip-gram model. The former aims to predict the word given the context, which can be a single word or a group of words. In contrast, the latter seeks to predict the context of the given word.

The architecture used in the project is depicted in Fig 4. The combined embeddings are generated by concatenating the embeddings output by all algorithms, with the aim of providing as much contextual information to the machine learning algorithms as possible. To evaluate the performance of the multimodal input, they are trained and tested on the same machine learning models as used by the authors.

Fig 4. Architecture of proposed work



4. Experiment Setup

This section outlines the application of each method within the project. As different methods utilize varied data types for analysis, their setups, parameters, and training strategies differ accordingly.

4.1. URL-based feature extraction

The ensemble model is trained and tested using the dataset generated from extracting the feature set from URLs, as detailed in Table 2. The models utilized to construct the final ensemble model are sourced from the scikit-learn library in Python. Random Forest, Extra Trees, eXtreme Gradient Boosting, and Gradient Boosting classifiers all employ Decision Trees as their base estimators with default hyper-parameter settings. Additionally, the Bagging classifier is applied with a Support Vector Machine using an RBF kernel as its base estimator. The AdaBoost classifier utilizes Logistic Regression as its base estimator. Finally, the scikit-learn implementation of Voting Classifier is employed with 'soft' voting to assemble the final ensemble model.

The dataset consists of 2,839 URLs, comprising 49% phishing instances and 51% benign instances. A training set size equivalent to 60% of the data was utilized, with the remaining portion allocated for testing the ensemble model. The outcomes of the experiment are detailed in Section 5.1. Subsequently, the final ensemble model is saved as a pickle file for future deployment.

4.2. URL-based character level encoding

This approach employs the same dataset utilized in the previous method. The training set constitutes 60% of the data. The CNN network, as described in Section 3.2.3, is implemented using the PyTorch library. The model undergoes training with Stochastic Gradient Descent, featuring a mini-batch size of 45, momentum of 0.5, and a learning coefficient of 0.1. Training spans 15 epochs, utilizing Cross-Entropy loss as its loss function.

The state dictionary of the CNN model maps the learnable parameters to each layer of the network. This is saved to be used later for deployment.

4.3. Webpage-based image classification

This approach utilizes screenshots of websites extracted from URLs within the dataset established for the previous method. These screenshots are used for training the CNN model proposed in Section 3.3. The analysis employed a total of 2,535 PNG image files, which were divided into three sets for training, testing, and validation.

- Training Set: Comprises 65% of the images and is used to train the model.
- Testing Set: Represents 30% of the images and is used to evaluate the model's performance on unseen data.
- Validation Set: The remaining 150 images form the validation set. This set provides an unbiased assessment of how well the model generalizes to unseen data after training on the training set.

For both training and testing data loaders, a batch size of 8 images is used. The validation data loader utilizes a slightly larger batch size of 10 images.

To expedite development, the pre-trained VGG-16 model with its default weights was obtained from the PyTorch library. The custom classification block added on top of VGG-16 was also implemented using PyTorch.

The model employs Cross Entropy loss as its loss function, aiming to minimize the difference between the predicted and actual class distributions. For optimization, the Adam algorithm is used with a learning rate of 0.0001. The training process continues for a total of 15 epochs. Finally, the trained model's state dictionary, containing all its learned parameters, is saved for future deployment. The detailed results achieved by the classifier are presented in Section 5.3.

4.4.HTML source-code text-based classification

This method utilizes Plain Text and Domain Specific Text extracted from HTML code files to generate word embeddings. To accomplish this, we parse 2,535 HTML files, generating distinct datasets for PT and DST (Table 4). We utilize the *BeautifulSoup* library to parse HTML content and extract pertinent text elements.

Additionally, we employ the *NLTK* library to conduct fundamental text cleaning procedures. This process involves the following tasks on the text data:

- Removes special characters and punctuation except for alphabets, digits, spaces, periods, commas, and question marks.

- Replaces hyphens with spaces.
- Converts text to lowercase and removes extra spaces.
- Removes common stopwords like "the," "a," "an," “the,” etc. (Uses *NLTK's* stopwords list)
- Applies lemmatization to convert words to their base form (e.g., "running" becomes "run") using *NLTK's WordNetLemmatizer*.

Table 4. Statistics of datasets used in this experiment

Dataset	Train-Test (%)	Split	Instances
Plain Text	**60% - 40%**	Total instances	1,889
		Phishing	889 (47%)
		Benign	1,000 (53%)
Domain Specific Text		Total instances	2,327
		Phishing	1,288 (56%)
		Benign	1,039 (44%)

The processed text is subsequently utilized to generate embeddings employing different word embedding algorithms. The parameters employed for the final models are outlined in Table 5.

The resultant embeddings are combined to create input embeddings of size 950. These embeddings are subsequently utilized as input for machine learning

Table 5. Parameters used from the experiments presented by Rao et al. [4]

Library	Algorithm	Parameters
Scikit-learn	TF-IDF Vectorizer	max_features = 150
	Count Vectorizer	max_features = 150
Gensim	Word2Vec - *CBOW*	min_count=5, workers=10, vector_size=150, window=10, epochs=50
	Fast Text - *CBOW*	min_count=5, workers=4, vector_size=100, window=10
	Word2Vec - *Skipgram*	sg=1, min_count=5, workers=10, vector_size=150, window=10, epochs=50
	Fast Text - *Skipgram*	sg=1, min_count=5, workers=4, vector_size=100, window=10
Mittens	Glove	n=150, learning_rate=0.05, max_iter=30

algorithms to classify between phishing and legitimate instances. The optimized parameters for the machine learning models are detailed as follows: RandomForest (n_estimators = *100*), Decision Tree (criterion = *'gini'*, splitter=*'best'*), SVM (gamma = *'auto'*, C = *1*, kernel = *'rbf'*, degree = *3*), XGBClassifier (), Logistic Regression (solver=*'lbfgs'*, max_iter = *1000*). The most effective models for both PlainText and Domain Specific Text are ultimately stored as pickle files for later use in deployment. The outcomes of the word embedding algorithms and multimodal embeddings are detailed in Section 5.4.

5. Results

5.1. Ensemble Model

Various metrics and visualization tools from the scikit-learn library are employed for analyzing the extracted features and evaluating the ensemble model. The significance of each extracted feature is depicted in Fig 5.

Model	Accuracy	F1 Score	False Predictions		
			FP	FN	Total
Random Forest	91.02	90.79	32	70	102
Extra Trees	**91.28**	91.05	**30**	69	**99**
Gradient Boosting	90.22	90.15	46	65	111
XG Boost	90.22	90.15	46	65	111
Ada Boost	63.99	53.68	93	316	409
Bagging Classifier	83.45	84.41	124	64	188
Ensemble Model	91.19	**91.08**	38	**62**	100

Table 6. Accuracy, F1 scores, and, distribution of false predictions.
(Best results are highlighted in **bold**)

The performance of the proposed ensemble model is assessed using metrics such as accuracy and F1 score (Table 6). Additionally, Fig 6 illustrates the confusion matrix derived from the testing data.

The results indicate promising outcomes from the proposed model. The ensemble model achieves the highest F1 score of 91.08%. Notably, the Extra Trees (ET) classifier attains the highest accuracy of 91.28% with minimal false predictions.

Furthermore, the proposed model performs closely to the ET algorithm, achieving an accuracy of 91.19%. Enhancing the performance of the ensemble model could involve utilizing superior models in its composition. Experimenting with variations of AdaBoost and Bagging Classifier could lead to identifying the best-performing model.

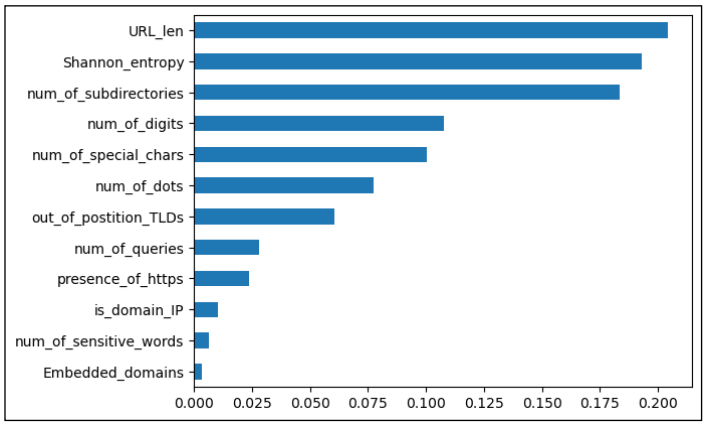


Fig 5. Features based on importance weights.

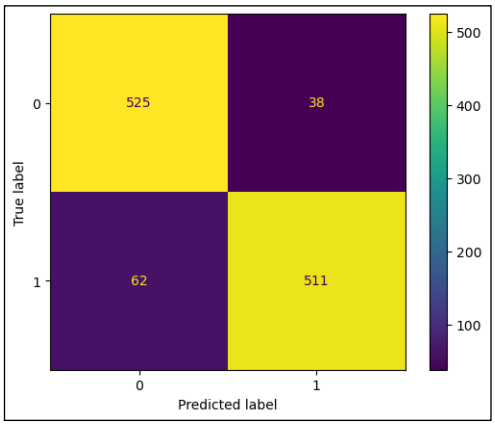
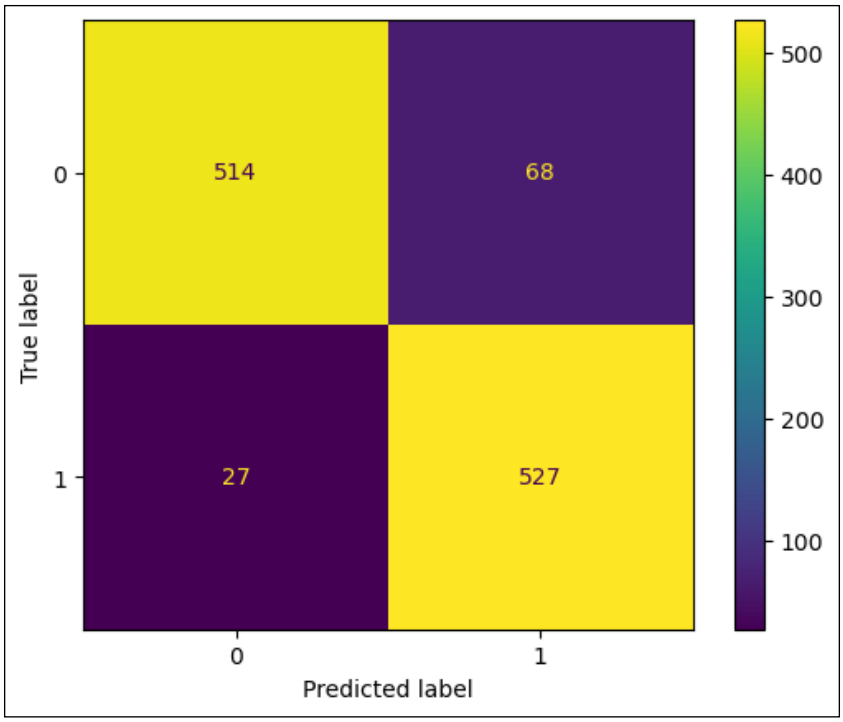


Fig 6. Confusion Matrix of Ensemble Model

5.2. URL-Encoding CNN Model

The CNN model, inspired by [3], undergoes evaluation using the same metrics as the approach described above, along with precision and recall scores. The model attains an impressive F1 score of 91.7% and an accuracy of 91.6%. The confusion matrix depicting the model's performance is showcased in Fig 7. Additionally, the model demonstrates a high recall of 95.1% and a precision of 88.6%.

Fig 7. Confusion Matrix of CNN model.



5.3. Image Classification CNN Model

This approach introduces a VGG-16 based CNN model designed for classifying website images. The training and testing processes were conducted on a

machine equipped with a GeForce GTX 1050Ti GPU with 4 GB of RAM, housed in an Intel i5-7300HQ 2.50GHz 16 GB RAM system. Due to the limitation of computational power, only the classifier part of the VGG-16 architecture is unfrozen, which allows the weights of the parameters in these layers to be updated during training.

The CNN network was trained for a total of 15 epochs using the configuration outlined in Section 4.3. Testing of this trained CNN model involved over 700 images, and its performance is depicted in the confusion matrix shown in Fig 8. Remarkably, the model achieves an accuracy of 92.3% and an impressive F1 score of 93.06%.

Furthermore, the performance can be assessed by analyzing the slope of the loss curve generated during the training phase across all epochs, as illustrated in Fig 9.

The low count of false predictions reflects a promising outlook for the custom VGG-16 based model and the application of transfer learning techniques. The custom fully-connected block demonstrably contributes to the model's ability to learn relevant features from website images. This success opens doors for several future experiments aimed at further improvement. These experiments could involve manipulating the number of layers within the model or adjusting their sizes. Additionally, with access to more powerful computational resources, training the entire model, not just the classifier block, directly on the website image dataset could be explored.

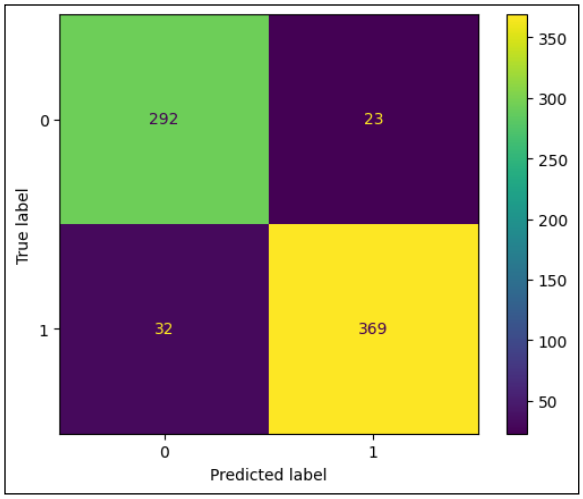


Fig 8. Confusion Matrix of Image-classification CNN

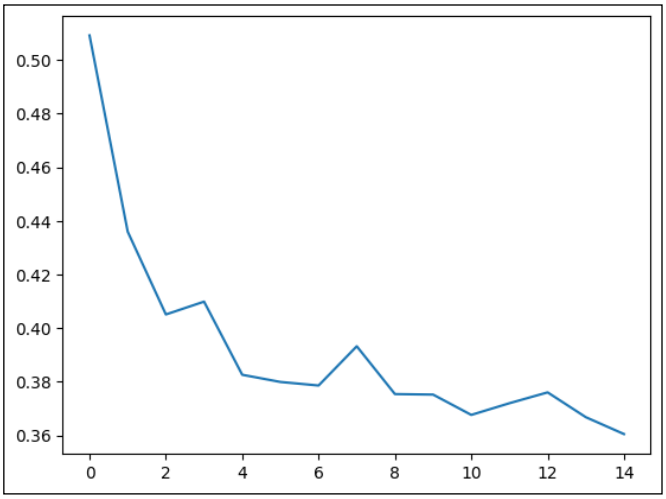


Fig 9. Loss Curve while training CNN classifier

5.4. Word Embedding Models

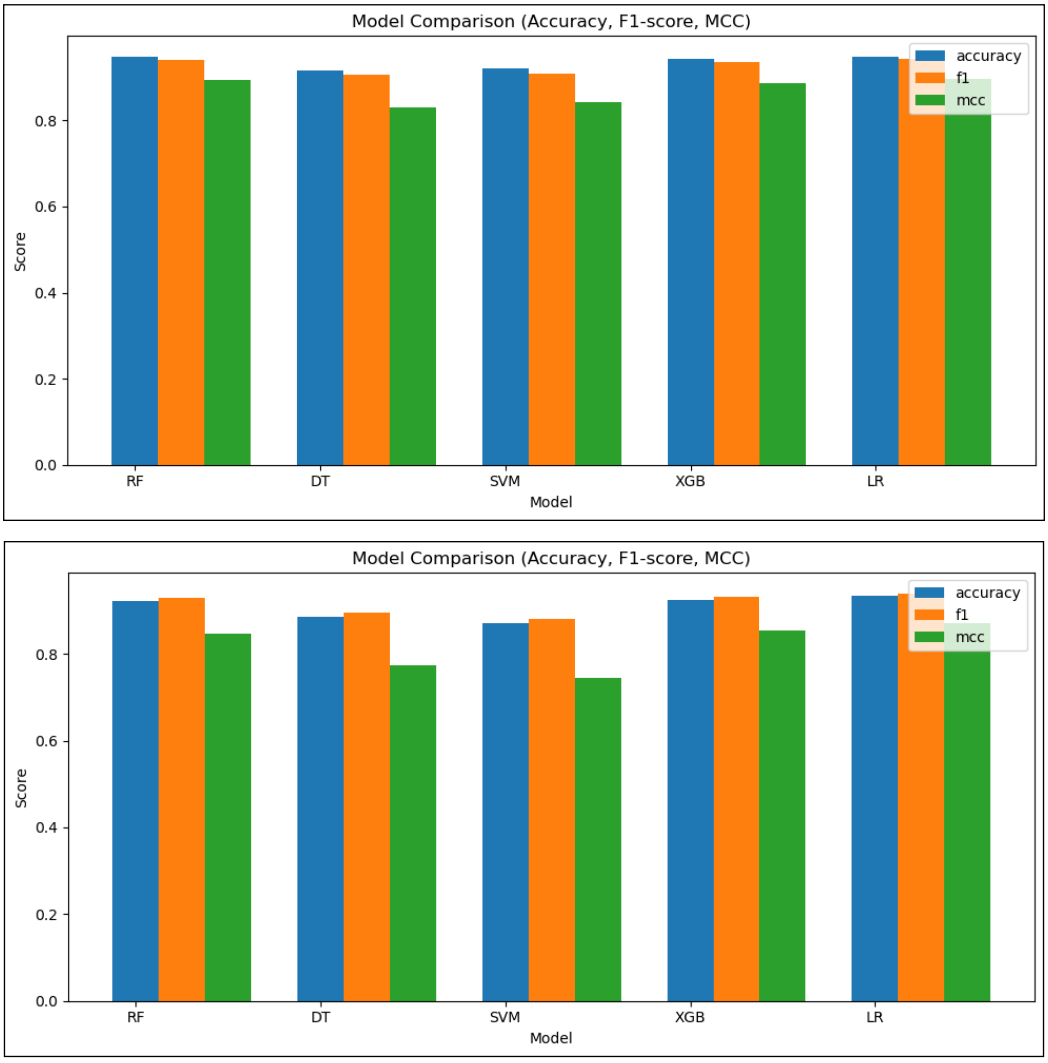
This method utilizes various word embedding algorithms discussed in Section 4.4, drawing inspiration from experiments outlined in [4]. In addition to the F1 score and accuracy, the authors incorporate another metric known as the Matthews Correlation Coefficient (MCC) for evaluation. MCC considers both true and false positives and negatives, providing a balanced measure suitable even

Table 9. Result of multimodal model with PlainText and Domain Specific Text

Input Text	Classifiers	F1-Score	Accuracy	MCC
Plain Text	RF	94.00	94.70	89.51
	SVM	90.87	92.06	84.28
	LR	**94.41**	**94.88**	**89.69**
	DT	90.66	91.53	82.94
	XGBoost	93.67	94.35	88.69
Domain Specific Text	RF	92.89	92.27	84.77
	SVM	88.06	87.12	74.55
	LR	**93.97**	**93.41**	**86.98**
	DT	89.44	88.55	77.32
	XGBoost	93.17	92.56	85.30

for imbalanced class sizes. The results of word embedding algorithms applied to PlainText and Domain Specific Text are presented in Table 7 and Table 8, respectively (Appendix A). The findings reveal that for PlainText, the highest scores are attained by an XGBoost model trained with embeddings generated from the FastText skip-gram algorithm. This model achieved an F1 score of *94.3%* and an MCC score of *89.73%*. Conversely, for Domain Specific Text, the top scores are also achieved by an XGBoost model, but utilizing embeddings

Fig 10. Comparison of models trained using multimodal.
(Top: PlainText, Bottom: Domain Specific Text)



generated from the Word2Vec model with the CBOW setting. This model achieved an MCC score of *87.78%* and an F1 score of *94.49%*.

Multimodal embeddings are also evaluated using different ML algorithms, and the outcomes are illustrated in Table 9. The findings reveal that Logistic Regression (LR) models perform admirably with both PlainText (PT) and Domain Specific Text (DST). Specifically, the LR model achieves an F1 score of *94.41%* and an MCC of *89.69%* with PT, while with DST, it attains an F1 score of *93.37%* and an MCC of *86.98%*. A comparison of these different models is depicted in Fig 10.

6. Deployment Setup

A significant contribution of this project was the integration of machine learning and deep learning models with a micro-service. This integration allows for real-time analysis of websites, determining whether they are malicious or benign. It forms an integral part of a machine learning service responsible for handling predictions through a RESTful API. Utilizing FastAPI, a modern Python web framework, the micro-service is designed to build efficient and scalable APIs. The key functionalities of the micro-service include:

- **API Endpoint Definition:** The service defines API endpoints under the *'/api/v1'* route to perform various prediction-related tasks, such as listing available models, creating predictions for reports, and saving/retrieving predictions to/from the database.

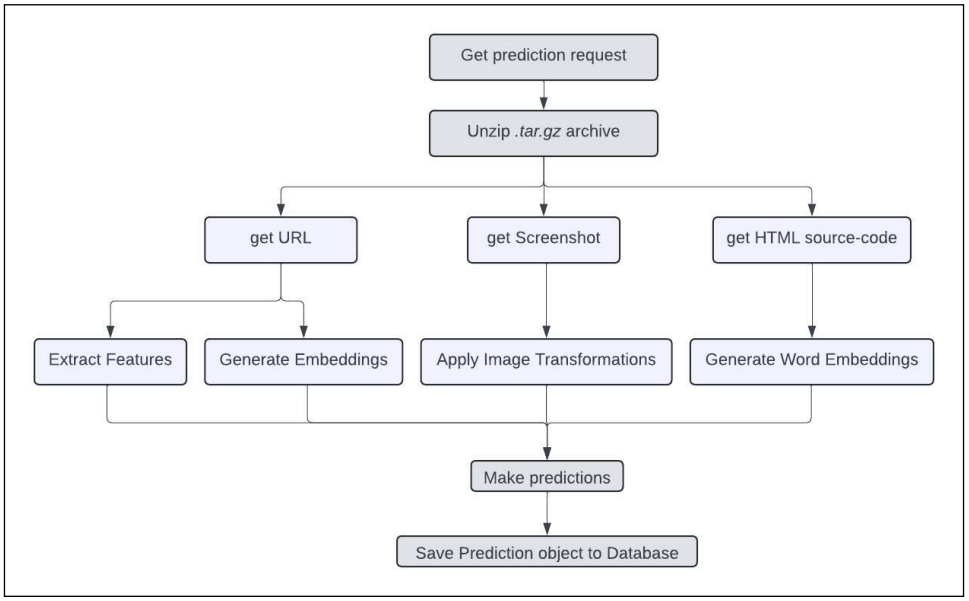
- **Model Handling:** It encapsulates the machine learning models within the *'ML_MODELS'* list, allowing for easy management and utilization of different models for prediction tasks.

- **Prediction Processing:**

- When a prediction request is received for a report, the code fetches the necessary data from external sources, such as report details (e.g., URLs) and associated files (e.g., screenshots, HTML files).
- It then iterates over the saved machine learning models to make predictions based on the type of model and the available data.
- Predictions generated by each model are saved in a database, providing a persistent record of the prediction results.

- **Data Retrieval and Filtering:** Provides an endpoint that allows users to retrieve a list of existing predictions. It allows filtering by report ID for focused retrieval and offers pagination options (limit and skip) to control the number

Fig 11. Basic workflow of the micro-service.



of returned results. This enables clients to fetch specific predictions based on report identifiers and control the number of results returned.

- **Error Handling:** Exception handling mechanisms are implemented to gracefully handle errors that may occur during prediction processing, such as when the requested report is not available or when an API request fails.

Figure 11 illustrates the sequence of key steps undertaken when the micro-service receives a prediction request based on a report ID. Additionally, Table 10 showcases an example demonstrating the storage of prediction objects for a phishing kit within the connected MongoDB database.

Table 10. Predictions saved in MongoDB for the report ID 660ede825ce051b809851c7d

ID	created_at	updated_at	Report ID	Model	Version	Output	Confidence
66311067b845b1be3590917c	2024-04-30T15:38:15.526000Z	2024-04-30T15:38:15.526000Z	660ede825ce051b809851c7d	EnsembleModel	1.0	1	0.733565952602358
66311067b845b1be3590917d	2024-04-30T15:38:15.548000Z	2024-04-30T15:38:15.548000Z	660ede825ce051b809851c7d	URL_CNN_Model	1.0	0	0.725121796131134
66311067b845b1be3590917e	2024-04-30T15:38:21.002000Z	2024-04-30T15:38:21.002000Z	660ede825ce051b809851c7d	Image_CNN_Model	1.0	0	0.731058597564697
66311067b845b1be3590917f	2024-04-30T15:38:27.924000Z	2024-04-30T15:38:27.924000Z	660ede825ce051b809851c7d	MultiModal_Text_Model	1.0	1	0.989350107590744

7. Future Works

This section provides an analysis of the developed methodologies, focusing on the potential for future enhancements to improve their performance. The main areas for future exploration are outlined as follows:

- Ensuring coherence in the data sources from which phishing/non-phishing samples are collected is crucial. Enhanced data quality and quantity typically translates to better learning models, serving as the foundational step towards generating relevant datasets for phishing detection. These datasets can encompass diverse forms of data, including URLs, website images, or source code files.

- Regarding the ensemble-based feature extraction method described in Section 3.1, there is potential to incorporate additional relevant features from the referenced studies to facilitate more comprehensive experimentation. Moreover, implementing hyper-parameter tuning for the individual models can significantly enhance the performance of the resulting voting-based ensemble model.

- The VGG-16 based image classification method introduced in Section 3.3 provides a solid foundation for future research endeavours. Subsequent experiments could focus on refining image transformation steps tailored specifically to website images. Additionally, exploring the potential of the custom classification block with diverse possibilities relevant to image data could be interesting.

- The Word-embedding based approach outlined in Section 3.4 is inspired by the experiments conducted by [4]. An area for significant improvement in the future lies in implementing more effective text extraction and preprocessing steps prior to generating word embeddings. As evident from the results, combining various word embeddings offers a promising avenue for exploration. Furthermore, leveraging deep learning algorithms on this ensemble of embeddings could yield more robust classification models.

8. Limitations

- Limited computational resources pose a challenge, particularly when training and testing large CNNs for tasks like image classification, resulting in prolonged and inefficient processes.

- The proposed models utilizing word embeddings rely solely on plain text and domain-specific text. Consequently, these models fail when confronted with image data in place of textual content.

- The word embedding based models face difficulties in scenarios where HTML source code files lack textual content, such as those containing embedded JavaScript code or encoded values.

9. Conclusion

This project investigated four innovative machine and deep learning methods to detect phishing websites. Each method tackled a different aspect of website data.

Some extracted features from URLs, while others classified website images or analyzed text within the website's source code. All the approaches were successful, achieving F1 scores above *91%*. Notably, the method that analyzed HTML source code text with Logistic Regression and multimodal embeddings reached the highest F1 score of *94.41%*.

Furthermore, the project created a micro-service architecture that integrates these models. This allows for real-time analysis of phishing websites, paving the way for a more complete phishing detection system.

Although these solutions are effective, the report acknowledges limitations in data quality, computational resources, and model flexibility. Future work should focus on addressing these limitations and refining the machine and deep learning models for even stronger phishing detection capabilities.

References

[1] Cecilia Hu, Fang Liu, Shehroze Farooqi, Stella Zhu, Daiping Liu, Jodie Ma, Jingwei Fan and Tao Yan (2023). "Recent Trends in Internet Threats: Common Industries Impersonated in Phishing Attacks, Web Skimmer Analysis and More".

[2] Phil Muncaster, 2023. "Volume of HTTPS Phishing Sites Surges 56% Annually".

[3] Wei Wei, Qiao Ke, Jakub Nowak, Marcin Korytkowski, Rafał Scherer, Marcin Woźniak, Accurate and fast URL phishing detector: A convolutional neural network approach, Computer Networks, Volume 178, 2020, 107275, ISSN 1389-1286. https://doi.org/10.1016/j.comnet.2020.107275

[4] Rao, R.S., Umarekar, A. & Pais, A.R. Application of word embedding and machine learning in detecting phishing websites. Telecommun Syst 79, 33–45 (2022). https://doi.org/10.1007/s11235-021-00850-6

[5] Cao, Y., Han, W., & Le, Y. (2008). "Anti-phishing based on automated individual white-list". In Proceedings of the 4th ACM workshop on digital identity.

[6] A. Oest, Y. Safaei, A. Doupé, G. -J. Ahn, B. Wardman and K. Tyers, "PhishFarm: A Scalable Framework for Measuring the Effectiveness of Evasion Techniques against Browser Phishing Blacklists," 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 2019, pp. 1344-1361, doi: 10.1109/SP.2019.00049.

[7] Youness Mourtaji, Mohammed Bouhorma, Daniyal Alghazzawi, Ghadah Aldabbagh, Abdullah Alghamdi, "Hybrid Rule-Based Solution for Phishing URL Detection Using Convolutional Neural Network", Wireless Communications and Mobile Computing, vol. 2021, Article ID 8241104, 24 pages, 2021. https://doi.org/10.1155/2021/8241104.

[8] Ozgur Koray Sahingoz, Ebubekir Buber, Onder Demir, Banu Diri, "Machine learning based phishing detection from URLs", Expert Systems with Applications, Volume 117, 2019, Pages 345-357, ISSN 0957-4174. https://doi.org/10.1016/j.eswa.2018.09.029.

[9] E. Ucar, M. Ucar, and M. O. Incetas¸, “A deep learning approach for detection of malicious urls,” in 6th International Management Information Systems Conference, 2019, pp. 12–20.

[10] Afzal, S., Asim, M., Javed, A. R., Beg, M. O., & Baker, T. (2021). “URLdeepDetect: A Deep Learning Approach for Detecting Malicious URLs Using Semantic Vector Models”. Journal of Network and Systems Management, 29(3). doi: 10.1007/s10922-021-09587-8

[11] Yanan Cheng, Tingting Chai, Zhaoxin Zhang, Keyu Lu, Yuejin Du, “ Detecting Malicious Domain Names with Abnormal WHOIS Records Using Feature-Based Rules”. The Computer Journal, Volume 65, Issue 9, September 2022, Pages 2262–2275. https://doi.org/10.1093/comjnl/bxab062

[12] R. Verma and K. Dyer, “On the Character of Phishing URLs: Accurate and Robust Statistical Learning Classifiers,” in ACM Conference on Data and Application Security and Privacy, 2015, pp. 111–121.

[13] Y. Peng, S. Tian, L. Yu, Y. Lv, R. Wang, "MALICIOUS URL RECOGNITION AND DETECTION USING ATTENTION-BASED CNN-LSTM," KSII Transactions on Internet and Information Systems, vol. 13, no. 11, pp. 5580-5593, 2019. DOI: 10.3837/tiis.2019.11.017.

[14] Guang Xiang, Jason Hong, Carolyn P. Rose, and Lorrie Cranor. 2011. CANTINA+: A Feature-Rich Machine Learning Framework for Detecting Phishing Web Sites. ACM Trans. Inf. Syst. Secur. 14, 2, Article 21 (September 2011), 28 pages. https://doi.org/10.1145/2019599.2019606

[15] Davide Canali, Marco Cova, Giovanni Vigna, and Christopher Kruegel. 2011. Prophiler: a fast filter for the large-scale detection of malicious web pages. In Proceedings of the 20th international conference on World wide web (WWW '11). Association for Computing Machinery, New York, NY, USA, 197–206. https://doi.org/10.1145/1963405.1963436

[16] A. C. Bahnsen, E. C. Bohorquez, S. Villegas, J. Vargas and F. A. González, "Classifying phishing URLs using recurrent neural networks," 2017 APWG Symposium on Electronic Crime Research (eCrime), Scottsdale, AZ, USA, 2017, pp. 1-8, doi: 10.1109/ECRIME.2017.7945048.

[17] S. Al-Ahmadi, A. Alotaibi and O. Alsaleh, "PDGAN: Phishing Detection With Generative Adversarial Networks," in IEEE Access, vol. 10, pp. 42459-42468, 2022, doi: 10.1109/ACCESS.2022.3168235.

[18] S. Haruta, H. Asahina and I. Sasase, "Visual Similarity-Based Phishing Detection Scheme Using Image and CSS with Target Website Finder," GLOBECOM 2017 - 2017 IEEE Global Communications Conference, Singapore, 2017, pp. 1-6, doi: 10.1109/GLOCOM.2017.8254506.

[19] M. Hara, A. Yamada and Y. Miyake, "Visual similarity-based phishing detection without victim site information," 2009 IEEE Symposium on Computational Intelligence in Cyber Security, Nashville, TN, USA, 2009, pp. 30-36, doi: 10.1109/CICYBS.2009.4925087.

[20] L. Ouyang and Y. Zhang, "Phishing Web Page Detection with HTML-Level Graph Neural Network," 2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Shenyang, China, 2021, pp. 952-958, doi: 10.1109/TrustCom53373.2021.00133.

[21] S. Roopak and T. Thomas, "A Novel Phishing Page Detection Mechanism Using HTML Source Code Comparison and Cosine Similarity," 2014 Fourth International Conference on Advances in Computing and Communications, Cochin, India, 2014, pp. 167-170, doi: 10.1109/ICACC.2014.47.

[22] Yukun Li, Zhenguo Yang, Xu Chen, Huaping Yuan, Wenyin Liu, “A stacking model using URL and HTML features for phishing webpage detection”, Future Generation Computer Systems,nVolume 94, 2019, Pages 27-39, ISSN 0167-739X. https://doi.org/10.1016/j.future.2018.11.004

[23] P. Zhang et al., "CrawlPhish: Large-scale Analysis of Client-side Cloaking Techniques in Phishing," 2021 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 2021, pp. 1109-1124, doi: 10.1109/SP40001.2021.00021

[24] Ankit Kumar Jain, B. B. Gupta, "Phishing Detection: Analysis of Visual Similarity Based Approaches", Security and Communication Networks, vol. 2017, Article ID 5421046, 20 pages, 2017. https://doi.org/10.1155/2017/5421046

[25] X. Zhang, J. Zhao, Y. LeCun, Character-level convolutional networks for text classification, in Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, NIPS'15, MIT Press, Cambridge, MA, USA, 2015, pp.649–657.

Appendix A

Table 1. Comparative table of studies done in the subdomain of Phishing Detection.

Data	Authors & Year	Main Idea	Findings & Results	Dataset used	Limitations
URL	Al-Ahmadi et al, 2022	This paper proposes a model (PDGAN) to detect phishing websites using Generative Adversarial Networks (GANs). They used LSTM to generate synthetic phishing URLs, then employed CNN for classification.	The PDGAN demonstrated superior performance compared to other methods, boasting a low False Positive (FP) Rate of 2.1% and a F1 score of 97.64%.	The authors obtain an extensive dataset comprising almost 2 million URLs from DomCop and PhishTank websites.	The generator should produce synthetic phishing URLs that are similar to the real URLs and is based on the assumption that the discriminator can distinguish between legitimate and phishing URLs.
	Peng et al, 2019	In this paper, the authors introduce an approach merging an Attention mechanism with a CNN and LSTM to identify malicious URLs. They use WHOIS check method to extract and filter features, and subsequently input them to the constructed CNN convolution layer to extract local features.	The key highlight of this paper is the introduction of an attention mechanism in the expression of the malicious URL feature that allows for the highlighting of key features to detect malicious URLs.	The dataset is collected from PhishTank and contains 16,055 malicious URLs.	The paper does not provide any report or analysis on the dataset or its collection and is missing the benign instances.
	Bahnsen et al, 2017	This study introduces a novel method utilizing RNN to classify phishing URLs. This approach eliminates the need for manual feature extraction by directly acquiring a representation from the sequence of characters within the URL. More precisely, the model employs LSTM units, treating the URL as a character sequence, where each character is transformed into a 128-dimensional embedding as input.	The LSTM model achieved an accuracy rate of 98.7%, outperforming the RF model, which achieved an accuracy rate of 93.5%. On average, the LSTM network has an F1 score 5% higher than the feature engineer model with RF. This paper also highlights that phishing URLs tend to be longer than benign URLs. It also highlights the benefits of utilizing representations of URL character sequences to classify websites.	For their experiments, a dataset of real and phishing URLs was created consisting of 2 million instances. The dataset had a balanced distribution. The legitimate URLs came from Common Crawl whereas the phishing URLs came from Phishtank, a website used as a phishing URL deposit.	The core concept of the paper involves translating each input character into a 128-dimensional embedding, which is then inputted into LSTM units. This hypothesis is derived from the assumption that characters close to each other in a URL are likely to be correlated.

Data	Authors & Year	Main Idea	Findings & Results	Dataset used	Limitations
	Xiang et al, 2011	The paper proposes a novel anti-phishing solution named CANTINA+ that aims to address the weaknesses of both blacklists and feature-based methods in a unified framework. It contains three major modules. First leverages the high similarity among phishing webpages. The second detects the presence of login forms that request sensitive information. The third and core module introduces 15 highly expressive features with ML algorithms to classify webpages.	This study presents eight innovative features that improve the detection of phishing attacks by utilizing diverse resources. It tackles the challenge of high FPs by using a layered structure with login form filtering. CANTINA+ attained high TP rates of 92.54% and 93.47%, accompanied by low FP rates of 0.407% and 0.608% with and without login form filtering, respectively. However, the authors note that the filtering step notably decreases the TP rate.	The webpage collection consists of phishing cases from PhishTank, and legitimate webpages from five sources.	The disadvantage of this solution is the use of a search engine to find out whether the address certainly matches the desired page causing additional network load. Moreover, the attacker could promote the phishing website in the search engine to make it seem legitimate.
Image	Ouyang and Zhang, 2021	The authors introduce an innovative approach for detecting phishing web pages, utilizing a graph neural network framework. They utilize the HTML's inherent DOM tree structure, representing it as a graph to extract local features from each node through RNN and distributed representations. This solution merges the strengths of RNN for extracting local features with GNN's ability to capture broader semantic context.	The authors introduce their approach as the first study into modelling HTML codes as graphs and classifying web pages using GNNs. Their proposed solution surpasses current anti-phishing methods, achieving an outstanding accuracy of 95.5%. The model demonstrates its optimal performance when utilizing information extracted from the HTML DOM tree graphs.	Phishing webpages were obtained from PhishTank and OpenPhish. To obtain the HTML content of these pages, they developed a custom crawler. For benign web pages, they utilize the TrancoTop1M which contains top 1 million domain names by traffic. The authors use this as a starting point and crawl web pages by following the links on them. Their final dataset consisted of 26,578 phishing and 121,983 benign instances.	The experiments conducted by the authors primarily aim at generating the DOM structure of HTMLs which may not be applicable for all types of web pages. The computational resources required for the proposed method are not discussed, limiting its applicability to resource constrained environments.

Data	Authors & Year	Main Idea	Findings & Results	Dataset used	Limitations
	Wei et al, 2020	In this paper, the authors present a method to detect malicious URL addresses with almost 100% accuracy using CNN. Their method includes encoding the URLs using one-hot character-level representation of URLs to create input images for the neural network. The proposed method aims to provide nearly 100% accurate security, zero-day defence, and fast detection, making it suitable for real-time URL checking and mobile device usage.	Key findings include using an embedding layer to improve the accuracy of the proposed method. Employing CNNs with character-level URL encoding resulted in nearly perfect accuracy, reaching close to 100% in identifying phishing URLs, surpassing other existing methods relying on engineered features or content analysis.	The experiments are based on a publicly available PhishTank phishing sites database and benign URLs from CommonCrawl. The authors collected 10,604 random unique URLs of each category.	The authors made decisions, such as imposing a character limit of 256, without offering any supporting explanations. Furthermore, the authors did not consider the presence of special symbols such as '@' or '%' in URLs, which could indicate the use of encoding and obfuscation techniques.
	Jain & Gupta, 2017 *[24]*	This article's primary aim is to offer an overview of visual similarity techniques used in identifying phishing websites. It utilizes classifiers like Earth Mover's Distance (EMD) based image classifier. Image properties serve as a component of the feature set to compare against legitimate websites. Additionally, the phishing detection system suggested here employs a signature generated from the webpage's text, images, and its overall visual presentation.	The authors integrate hybrid features, including text properties extracted from textual content and image properties such as colour histograms for creating signatures to compare webpages. They classify the visual similarity based approaches into HTML document object model (DOM) tree, visual features, Cascading Style Sheet (CSS) similarity, pixel based, visual perception, and hybrid approaches.	The paper does not conduct any experiments explicitly and hence do not use any dataset. Although, the authors present a comprehensive analysis of phishing attacks and some of the recent visual similarity based approaches for phishing detection.	The authors do not present any quantifiable results or experiments that can help evaluate the proposed methods. A comparison with the previous works mentioned could have provided more insight into the various methods and feature sets used.

Data	Authors & Year	Main Idea	Findings & Results	Dataset used	Limitations
Code	Canali et al, 2011	Prophiler was developed as a filter for large scale detection of malicious webpages. It inspects two main sources of information for features, the content of the page (HTML and JavaScript code) as well as the associated URL (lexical and host-based characteristics). The authors developed Prophiler as a filter that can reduce the number of web pages that need to be analyzed dynamically to identify malicious web pages by tools like Wepawet.	The outcomes obtained by Prophiler align with the authors' objectives. They conducted a large-scale assessment by deploying Prophiler across the dataset used. Prophiler identified 14.3% of these pages as malicious, leading to an 85.7% reduction in the workload for the back-end analyzer. Additionally, their solution demonstrated superior performance compared to previous systems, showcasing lower rates of false negatives and false positives.	This experimentation used two datasets: an evaluation dataset and a validation dataset. The evaluation dataset consisted of 18,939,908 pages, all of which were unlabeled. The validation dataset contained 153,115 pages, with 139,321 benign and 13,794 malicious.	The study focuses on the design and implementation of a fast filter for detecting malicious web pages, but does not provide a comprehensive analysis of the effectiveness of the filter. The study does not address the detection of zero-day exploits or new types of malicious code that may not be captured by the filter.
	Rao, Umarekar and Pais, 2021	This paper introduces a new approach for identifying phishing websites, employing word embeddings derived from both plain and domain specific text extracted from HTML source code. They utilize word embedding techniques like Word2Vec and Glove and evaluate their model using an ensemble by combining different word embedding algorithms with RF and LR classifiers for its purposes. Additionally, they utilize various traditional classifiers to compare the effectiveness of their approach.	The word embeddings are generated using both, the plain texts and domain specific texts. The results obtained reinforce the usage of domain specific texts as all classifiers perform better when using it rather than plain text. It is also observed that the proposed multimodal model outperformed existing works with a significant accuracy of 99.34% and Matthews Correlation Coefficient (MCC) of 98.68%.	The approach involves utilizing a dataset comprising HTML source code extracted from various websites. The dataset is composed of 5076 instances classified as benign and 5438 instances classified as phishing. The source codes undergo parsing, and word embedding algorithms are applied to process the text. The division between training and testing sets is performed with an 80% - 20% split.	The model relies on textual data, both general and domain-specific, and encounters challenges when presented with images instead of text. However, details regarding the dataset's availability on the internet and the specific technique employed for its collection are not provided.

Data	Authors & Year	Main Idea	Findings & Results	Dataset used	Limitations
	Zhang et al, 2021	The authors of this paper introduce CrawlPhish, a framework for large scale detection and categorization of client-side cloaking techniques used by known phishing websites. They consider both the visual and code structure components of phishing websites. The code structure includes features like web API calls, web event listeners, and ASTs. CrawlPhish also examines the visual similarity between force-executed screenshots and an unmodified WebKitGTK+ screenshot.	A major contribution of this paper was a proposed taxonomy of eight types of evasion techniques in three categories (User Interaction,Bot Behaviour, Fingerprinting). The CrawlPhish framework exhibited low false-positive and false-negative rates of 1.45% and 1.75% respectively. They also identified 1,128 distinct implementations of cloaking techniques.	A dataset of 112,005 phishing websites was collected and analyzed over a 14-month period. Of these phishing websites, 35,067 (31.3%) were found to use client-side cloaking techniques. The use of client-side cloaking by attackers increased from 23.32% in 2018 to 33.70% in 2019.	Some limitations of this study is that it does not provide insights into the prevalence and impact of advanced client-side cloaking techniques. Evaluating the effectiveness of cloaking techniques is also limited to browser-based phishing detection and does not consider other anti-phishing defences.

Data	Authors & Year	Main Idea	Findings & Results	Dataset used	Limitations
	Li et al, 2019	This paper introduces a stacking model designed for the classification of phishing webpages, incorporating both URL and HTML features. The authors extract 12 features from the HTML source code, specifically advocating for the use of the Word2Vec model to generate string embeddings from HTML. These embeddings and features are then fed into a stacking model, employing Gradient Boosting Decision Tree (GBDT), XGBoost, and LightGBM for the ensemble model.	The presented approach was compared against various baselines, including CANTINA, and demonstrated the lowest false alarm rate at 3.7%. To showcase the efficacy of their stacking model, a comparison was conducted against individual machine learning models using the results from the 50K-PD dataset. The proposed model surpasses all individual models, achieving an accuracy of 97.30%. Their analysis highlights the effectiveness of employing HTML string embeddings generated by Word2Vec, requiring no domain knowledge of phishing.	The authors employed three datasets to assess their methodology. Initially, they utilized the 2K Phishing Detection Dataset (2K-PD), a well-balanced collection consisting of 1000 legitimate websites and 1000 phishing websites, each accompanied by their respective HTML code. Legitimate websites were sourced from Alexa rankings, while phishing instances were gathered from PhishTank. Additionally, the authors incorporated the 50K Phishing Detection Dataset (50K-PD) and the 50K Image Phishing Detection Dataset (50K-IPD) into their evaluation.	Phishing instances are sourced exclusively from PhishTank for a brief period, specifically over a span of 2 days. This approach may cause data homogeneity and does not encompass the diverse array of sources for phishing attacks.

Table 7. Results of various word embedding algorithms with PlainText

Word Embedding	Classifiers	Metrics		
		F1-Score	Accuracy	MCC
TF-IDF	RF	87.19	89.06	78.40
	SVM	57.92	72.83	52.10
	LR	80.51	83.95	68.42
	DT	84.88	87.12	74.44
	XGBoost	86.25	88.35	77.07
Count Vectorization	RF	88.01	89.77	79.84
	SVM	82.59	82.01	66.23
	LR	84.42	84.83	70.18
	DT	83.71	86.06	72.22
	XGBoost	88.52	88.88	78.22
Word2Vec (CBOW)	RF	93.46	94.17	88.35
	SVM	89.58	91.18	82.89
	LR	91.73	92.59	85.10
	DT	91.39	92.06	84.03
	XGBoost	93.62	94.35	88.75
FastText (CBOW)	RF	93.01	93.82	87.70
	SVM	81.43	85.36	72.46
	LR	81.91	84.65	69.42
	DT	88.00	88.88	77.66
	XGBoost	93.04	93.82	87.66
Glove	RF	77.93	78.13	57.11
	SVM	0	54.14	0
	LR	0	54.14	0
	DT	67.41	69.13	38.22
	XGBoost	72.07	73.89	47.61
Word2Vec (Skipgram)	RF	93.38	94.17	88.45
	SVM	86.14	88.71	78.62
	LR	91.56	92.59	85.24
	DT	90.21	91.35	82.68
	XGBoost	93.25	94.00	85.24

FastText (Skipgram)	RF	93.60	94.35	88.79
	SVM	79.82	83.42	67.36
	LR	86.12	88.00	76.07
	DT	91.08	92.06	84.06
	XGBoost	**94.3**	**94.88**	**89.73**

Table 8. Results of various word embedding algorithms with DST

Word Embedding	**Classifiers**	**Metrics**		
		F1-Score	**Accuracy**	**MCC**
TF-IDF	RF	86.16	83.69	66.72
	SVM	78.26	70.67	41.60
	LR	84.12	81.25	61.68
	DT	84.40	81.54	62.29
	XGBoost	86.06	83.69	66.69
Count Vectorization	RF	86.67	84.12	67.70
	SVM	82.09	76.53	54.04
	LR	85.78	82.97	65.37
	DT	84.82	81.83	62.97
	XGBoost	85.74	82.97	65.34
Word2Vec (CBOW)	RF	92.83	92.27	84.94
	SVM	88.97	88.41	77.86
	LR	88.47	87.69	75.98
	DT	91.54	90.70	81.38
	XGBoost	**94.39**	**93.84**	**87.78**
FastText (CBOW)	RF	91.12	90.27	80.57
	SVM	75.60	75.53	53.56
	LR	78.59	77.39	55.51
	DT	88.02	86.83	73.59
	XGBoost	91.76	90.98	82.04

Glove	RF	74.72	70.67	39.93
	SVM	72.21	56.50	0
	LR	72.21	56.50	0
	DT	70.12	67.09	33.61
	XGBoost	74.53	70.67	40.03
Word2Vec (Skipgram)	RF	92.10	91.55	83.74
	SVM	87.53	86.83	74.56
	LR	91.07	90.27	80.67
	DT	89.26	88.26	76.58
	XGBoost	94.30	93.70	87.39
FastText (Skipgram)	RF	92.45	91.84	84.05
	SVM	78.90	7897	60.78
	LR	85.10	83.97	68.27
	DT	91.00	89.98	79.76
	XGBoost	92.58	91.84	83.69