Phishing URL Detection using RoBerta for Feature Extraction

Tejasvi Kashyap   
*Department of Computer Science*   
*University of Ottawa*  
Ottawa, Canada  
[tkash023@uottawa.ca](mailto:tkash023@uottawa.ca)Student ID: 300342618

Kaixi Xu  
*Department of Computer Science*   
*University of Ottawa*  
Ottawa, Canada  
[kxu092@uottawa.ca](mailto:kxu092@uottawa.ca)Student ID: 300049787

Rohan Sood

*Department of Computer Science*   
*University of Ottawa*  
Ottawa, Canada  
[rsood044@uottawa.ca](mailto:rsood044@uottawa.ca)

Student ID: 300345439

Supriya Dara  
*Department of Computer Science*   
*University of Ottawa*  
Ottawa, Canada  
[sdara100@uotttawa.ca](mailto:sdara100@uotttawa.ca)Student ID: 300312994

*Abstract*—Phishing, a cyber threat exploiting disguised emails or websites to steal sensitive information, is rapidly evolving, making its detection challenging. Considering the widespread occurrence of these attacks, significant research has been undertaken to address these issues using diverse approaches. This paper presents an innovative machine learning-based methodology for the detection of phishing URLs using RoBERTa, leveraging its advanced feature extraction capabilities. Our approach uniquely analyzes the URLs by decoding not only their lexical features but also their underlying syntactic and semantic characteristics. We further enhance our model's interpretability and reliability by integrating SHAP to interpret the model's predictions, providing insights into the feature attributions that elucidate the decision-making process behind each prediction. The model's effectiveness is validated on a diverse dataset of 550,000 URLs, where it demonstrated a high accuracy of 98.34%. Our findings demonstrate a significant improvement over traditional phishing detection methods, underscoring the effectiveness of our approach.

Keywords—Phishing, Machine Learning, Phishing Detection, RoBERTa, SHAP

# Introduction

In the current digital landscape, where individuals and organizations rely heavily on online interactions and data sharing, the risk of phishing attacks has emerged as a pervasive and formidable challenge. Phishing attacks impersonate trusted sources and aim to manipulate users into disclosing sensitive information, clicking malicious links, or downloading harmful attachments. These nefarious activities threaten the security and privacy of individuals, compromise sensitive data, and pose significant threats to the integrity of both individuals and organizational infrastructures.

In phishing attacks, the most commonly used tool is the malicious Uniform Resource Locator (URL) [22]. These deceptive URLs serve as the gateway for cybercriminals to frequently impersonate reputable entities like financial institutions, credit card companies, or popular e-commerce platforms, employing persuasive tactics to trick users into accessing phishing websites unwittingly. Malicious URLs are one of the primary methods for carrying out cybercrimes, as they serve as hosts for unsolicited content and target unsuspecting users, subjecting them to various forms of scams such as monetary theft, identity theft, and malware installation [9]. For instance, a user might receive an instant message claiming an issue with their bank account and be directed to a website that closely resembles the official bank site.[23]

Furthermore, Figure 1 depicts the cyclical nature of a phishing attack, illustrating the sequence of actions from the perspective of the attacker and the victim. Initially, the attacker sends a crafted phishing URL to the victim (1), which, when clicked, redirects the victim to a fraudulent website designed to mimic a legitimate one. The victim, believing they are accessing a trusted site, may enter sensitive information such as login credentials (2). These credentials are then covertly transmitted to the attacker (3). With this information, the attacker can unauthorizedly access the legitimate website (4), completing the cycle of the phishing attack. This lifecycle not only highlights the attack strategy but also underscores the need for robust detection mechanisms to prevent the initial deceitful step of presenting victims with a phishing URL.



Fig. 1. Website Phishing Lifecycle [21]

Phishing attacks manifest in various forms, including email phishing, spear phishing, whaling, vishing (voice phishing), smishing (SMS phishing), and social engineering attacks, exploiting human psychology and trust [20]. These techniques, while varied in execution, commonly rely on malicious URLs to deceive users. These attacks have become increasingly prevalent in recent years, with staggering statistics highlighting the scale of this threat. For instance, A report by the Anti-Phishing Working Group (APWG) shows that in 2022 there were over 4.7 million attacks, which is an increase of about 150% from the previous year [11].

The evolving nature of phishing attacks, particularly through URLs that serve as gateways to fraudulent activities, underscores the pressing need for more advanced detection mechanisms. The problem of detecting these phishing URLs is both critical and complex. As the digital world expands, so do the techniques employed by cybercriminals, making conventional detection methods often insufficient. In recent years, substantial strides have been made in the field of phishing detection, as evidenced by various academic studies. Azeez, Nureni Ayofe, et al.,[12] in their paper explore a novel strategy that focuses on an automated whitelist approach. This method enhances the reliability and accuracy of phishing attack detection by distinguishing benign URLs from malicious ones more effectively. Further Aljofey, Ali, et al.[1] introduced an innovative model which utilizes a character-level convolutional neural network to analyze URLs, demonstrating a significant improvement in phishing detection by analyzing minute details in the URL structure. More recently, Jishnu, K. S., and B. Arthi [10] presented the advanced capabilities of RoBERTa for extracting features from URLs and LSTM for classification, offering a cutting-edge approach that aligns with the latest advancements in natural language processing and machine learning. While significant progress has been made in the field of phishing detection over the past decade, cybercriminals continue to devise new and sophisticated phishing tactics, making the task of detection increasingly complex [24]. Phishing attacks can deceive even the most vigilant users to less refined attempts using tactics like replacing URL characters with similar-looking Unicode characters or employing IP addresses instead of domain names [2].

This paper focuses on the problem: How can we more effectively identify and detect phishing URLs? The importance of solving this issue is undeniable, given the immense potential harm phishing attacks can cause to users globally. Furthermore, the paper aims to provide a comprehensive review of various phishing detection methods, with a primary focus on machine learning and deep learning approaches that rely solely on URL analysis. We delve into the intricacies of these techniques, discussing their advantages and disadvantages. In particular, our proposed model harnesses the power of RoBERTa, a state-of-the-art transformer-based model, to extract and understand the linguistic features of URLs without the need for manual feature engineering [25]. The model's training leverages a substantial Kaggle dataset comprising 550,000 unique URLs, offering a novel perspective on phishing detection that aligns with the latest advancements in machine learning and natural language processing. A key aspect of our approach is the incorporation of SHapley Additive exPlanations (SHAP) values, which enhance the model's interpretability. By providing clear insights into the contribution of each feature to the model's predictions, SHAP values help Unravel the decision-making process, fostering trust and reliability in the system's outputs [24].

In the subsequent sections of this paper, we will delve into the intricacies of our approach, elucidate the experimental setup, and present the implications of our findings. Additionally, we will critically assess the limitations of our model and suggest potential avenues for further research, thereby contributing to the ongoing efforts to strengthen cybersecurity in the face of evolving digital threats.

# Literature review

## Different Techniques for Phishing Detection

Since phishing is an ever-evolving issue there is no decisive solution for it. Aljofey et al in their paper [1], divide the protection methods into two categories: expanding the user knowledge and software-based detection. These different detection techniques as per [1] are given in Fig. 1. Each of these is discussed in detail as follows.

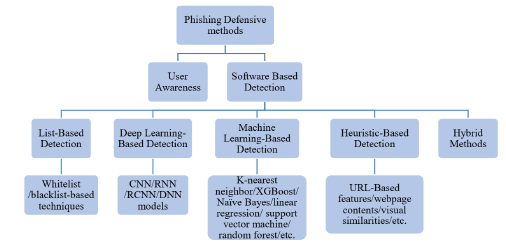


Fig. 2. Overview of phishing detection techniques [1]

* *List-Based Detection:-* Detection methods based on lists can be categorized into two main types: *whitelist-based* and *blacklist-based techniques*. The whitelist comprises a catalog of trusted URLs and IP addresses that are used to verify the legitimacy of a suspicious URL. An example of a whitelist-based approach is [12] which uses an automated whitelist using similarity to known and trusted websites. On the other hand, blacklist-based methods are tools that compare URLs against a continually updated list of known phishing websites and alert users when a URL is flagged as a phishing threat. The method proposed by Felegyhazi et al. [16] is an example of domain blacklisting. Although these methods are efficient, involving a straightforward database lookup, and are anticipated to exhibit low rates of false positives, a significant drawback is their inability to provide comprehensive coverage, especially when dealing with newly created URLs. This limitation is particularly critical since new URLs emerge on a daily basis [9].
* *Heuristic-Based Detection:-* These detection methods rely on the creation of phishing site characteristics based on many hand-crafted features which range from URL-based features, webpage content, website visual similarity, etc. An example of this is Cantina+ which was proposed by Xiang et al. [14]. It takes 15 heuristic features as the input to train and was able to achieve an F1-score of *96.32%.*
* *Machine Learning-Based Detection:-* These use machine learning techniques (e.g., K-nearest neighbor, XGBoost, Naïve Bayes, linear regression, support vector machine, and random forest) and treat phishing as a binary classification problem. Extracted features from the URL are used to train the model which then classifies the suspected URL as malicious or benign. The process of using machine learning for the detection of malicious URLs generally involves two main stages: the initial step is to acquire a suitable feature representation from the URL, followed by the second step, which entails utilizing this URL representation to train prediction models based on machine learning [9]. The main disadvantages of these methods are (i) They are unable to capture semantic or sequential data from the URLs. (ii) They require manual feature engineering which is time-consuming and human input heavy and (iii) Due to a fixed set of features they are unable to handle unseen features. An example of a machine learning-based detection system is CatchPhish by Rao et al [15] which uses handcrafted and Term Frequency-Inverse Document Frequency features to train a random forest classifier achieving an accuracy of 95.67%. We will go through more studies on these methods in section D.
* *Deep Learning-Based Detection:-* Owing to the notable achievements in Natural Language Processing (NLP) through deep learning methods, some of these techniques have recently found application in the field of phishing detection. This includes models like Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Recurrent Convolutional Neural Networks (RCNN), and Deep Neural Networks (DNN). Although previously unused due to high training times, a number of recent studies have started using deep learning methods due to the fact that they are able to extract features automatically.
* *Hybrid Method-Based Detection:-* These methods are a combination of two or more of the above techniques to achieve good performance. The proposed model falls under this category since it is a combination of deep learning and machine learning techniques.

## URL Structure

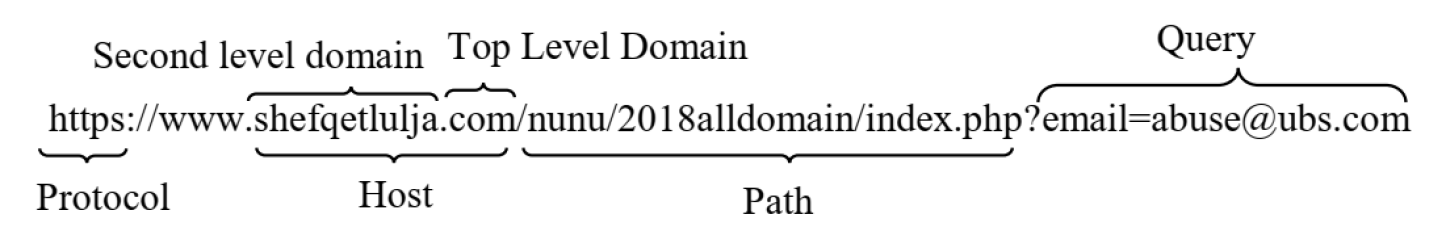


Fig. 3. URL Components [1]

The term "Uniform Resource Locator," or URL, refers to a web address that identifies the location of a resource on the internet. It is a collection of characters that refers to the location of the resource and the protocol that was used to access it. The structure of a typical URL is given in Fig. 2. Following are the elements of a URL[13]:

* *Protocol or Scheme:-* This component specifies the protocol to be employed for accessing the resource on the Internet. Frequently used protocols include HyperText Transfer Protocol (HTTP) and its secure counterpart, Secure HyperText Transfer Protocol (HTTPS).
* *Domain or Host:-* It specifies the location of the resource on the internet. It can be an IP address or a human-readable domain name
* *Port:-* This part defines the network port number to use for the connection. It is often omitted, with default port numbers used for common protocols
* *Path:*- The path specifies the location of a specific resource on the server, like a web page or a file. It follows the domain and is separated by slashes.
* *Query:*- The query component is used to send data to the resource as parameters, typically in a key-value format. It follows a question mark "?".

## URL Based features

The structure of URLs (Uniform Resource Locators) provides a wealth of information that can be leveraged for phishing URL detection [13]. Features extracted from URLs include:

* *Host-Based Features:* These features focus on the characteristics of the host or server where the URL is hosted. For example, examining the IP address, server response times, and SSL certificates can reveal valuable insights into the legitimacy of a website.
* *Domain-Based Features*: Domain analysis involves assessing the URL's domain name, subdomains, and top-level domains (TLDs). Unusual or suspicious domain structures are often indicative of phishing attempts.
* *Lexical Features*: Lexical analysis involves scrutinizing the words and characters within the URL. Common phishing tactics involve misspelled domain names, subdomains resembling trusted entities, or the presence of certain keywords related to phishing. These are the most commonly used features in research. Lexical features suffer from a few disadvantages: (i) They are unable to interpret new words in test URLs. (ii) Inability to gain insight from rare words and (iii) Lack of information related to semantics and neighbors [9].
* *Behavioral Features*: These features monitor the behavior of URLs and their response to user interactions. For instance, tracking redirects, the use of JavaScript, or collecting user data are all behavioral cues that can indicate a phishing attempt.
* *Content-Based Features*: Analysing the content of web pages linked by the URL can reveal phishing attempts, especially when comparing the content with known legitimate websites.

## Related Work

A number of studies have been conducted on phishing detection using URL detection here a few noteworthy ones are discussed. Safi et al. [6] state that the most commonly researched phishing detection techniques are based on URLs. The reason for the same is the speed of detection of these techniques is much higher leading to more practical applications for the same. The most popular datasets used across studies are DBOZ (now deprecated), Alexa.com, Yahoo URL generator (deprecated) for legitimate URLs, and PhishTank, APWG, and OpenPhish for phishing URLs. Das et al. [13] found that sources like Alexa.com list domain names only, which leads to artificially inflated results in the studies using this dataset. Zouina et al. [2] developed a lightweight SVM model which had very fast detection times and could be easily used on mobile devices. The authors trained their model on only 6 features one of which is a custom feature based on similarity index. They use a small dataset of only 2,000 records with 1,000 phishing URLs. The features used were extracted only from the domain and sub-domain of the URL. Zouina et al. used RBF (radial basis function) as their kernel. Fivefold-cross validation was used for testing. The authors were able to achieve a recognition rate of 95.8%.

In [7], Dobolyi et al. introduce PhishMonger, an API for URL verification that operates in conjunction with the PhishTank platform. This API is responsible for staying synchronized with PhishTank and retrieving files from suspicious web pages down to the deepest level. However, it's worth noting that this approach can lead to a substantial computational burden, particularly on more complex websites or when dealing with significant variations in file formats. Also, it relies on external services to function and is vulnerable to zero-day attacks.

Almeida et al. [3] developed a system to check emails and URLs for standard features in phishing scams based on social engineering and malicious code infection attempts. The authors developed an API, based on the principles of web crawling and scraping, with the purpose of using string search engines to look for the characteristics and features mentioned above. The authors were able to detect phishing URLs with accuracy values between 73.3% and 97.66% based on the complexity of the page with an average time of 30 seconds. Although these low detection times enable it to be a real-time solution it suffers from the disadvantage of relying on external services as well as the need for a bigger ever-expanding database.

In [18] Rao et al. authors recommended an ensemble model that fused Extra-Tree, Random Forest, and XGBoost to assess the importance of both heuristic-based and blacklist-based filters. The model uses a visual similarity-based blacklist to filter out similar phishing websites as a first-level filter. Heuristic filtering is used as a second-level filter on the URLs that were able to successfully pass through the first filter. Heuristic features were used to detect non-blacklist phishing websites. The solution was able to achieve an accuracy of 98.72%. The model tries to overcome the disadvantages of a list-based detection system by using a hybrid approach.

In [8] Blum et al., implement a real-time phishing detection system using online learning. The model is largely a lexical model trained on features extracted from a content-inspection-based approach. A confidence-weighted learning algorithm is used for the online model. The model was seen to perform well even on previously unseen URLs. For feature extraction, the URL was broken down into three parts: protocol, domain, and path. A bag of words approach was then used and tokens were obtained from these subgroups.

El Aassal et al. [5] made a benchmarking tool PhishBench for all phishing detection solutions. It was the first of its kind in the domain and filled the need for a uniform comparison between all the different techniques available. The paper delves into phishing detection based on URLs, website content as well as email, we will mainly focus on the findings of the URL-based classification study. The study found that host-based features perform the best (94%) while lexical features have the worst performance (84%). The authors performed an in-depth study on feature importance and best feature selection metrics. The study found that although chi-squared overcomes the limitation of choosing features that take a large range of values which is a well-known issue of Information gain, the models trained on top features from chi-squared perform the worst when compared to Information gain and Gain ratio. It was also found that models trained on the top features were 20 times faster with minimal performance impact. Table 13 [5] can be referred to view the top 20 features obtained through the different selection methods.

Most of the solutions discussed rely on a fixed feature set for training which makes them vulnerable to zero-day attacks since any changes would require re-training of the model and hence a need for real-time NLP-based models with automatic feature extraction and selection was felt.

Baykan et al. [4] studied URL topic classification using machine learning. Although the aim of the paper was not necessarily phishing detection similar techniques can be used to advance the field of phishing detection. The authors converted URL topic classification to a binary classification problem by considering each topic category to be a flag. NLP techniques such as n-gram are used to feed neighbor information to the model. The authors experimented with multiple different NLP techniques and Machine Learning Techniques, and the best results were obtained with a combination of all-grams (a combination of 4-, 5-, 6-, 7-, and 8-grams) directly on the URL gave the best results.

Dhamija et al. [9] propose an end-to-end deep learning framework which they call URLNet. Deep learning is used to learn nonlinear URL embeddings for Malicious URLs directly from the URL without any need for manual feature extraction. Convolutional Neural Network (CNN) is applied both at a character level and a word level. For the word level CNN advanced word embedding was learned using character-level data of the word which helps identify sub-word level information.

Aljofey et al. [1] propose a deep-learning-based solution to overcome the challenges and the cumbersome work of manual feature extraction. The model uses CNN to learn sequential information from the URL. The output of CNN then goes through a non-linear activation function. After this it goes through a pooling layer then finally the output is fed to a fully connected layer for classification. It uses a character-level Convolutional Neural Network (CNN). The solution achieved an accuracy ranging from 95.02% to 98.58% on different datasets.

Sahingoz et al. [17] propose an anti-phishing system that operates in real-time. This system leverages various classification algorithms and resources and uses natural language processing (NLP). The advantages of such a system are language independence, real-time execution, and independence from third-party services. The construction of classifiers and resources in this system utilizes the Random Forest algorithm, which achieves an impressive 97.98% accuracy in the identification of phishing URLs. The primary disadvantage of these methods is the fact that they cannot deal with unseen URLs well.

With the advancements in Transformers, a few studies have emerged that leverage them for phishing UR L detection. A couple of them are discussed in this paper. Jishnu et al. [10] use RoBERTa for feature extraction and integrate it with LSTM. The URL is first tokenized and then fed into the architecture of RoBERTa and LSTM. The model is then trained on Optimisation and loss functions. The LSTM layer is used to extract the sequential dependencies in the features extracted by RoBERTa. This information is then fed into a fully connected layer for classification. The authors are able to achieve an accuracy of 97.14%.

Maneriker et al. [19] introduce URLTran a deep-learning-based system that uses transformers to improve the performance of phishing URL detection. The study compares two settings: 9 (i) An existing transformer is pre-trained on only URL data. BERT is used for this. (ii) A Fine-tuned publicly available pre-trained transformer is used. For this BERT and RoBERTa are fine-tuned on the URL classification task. The model is able to improve the True Positive Rate from 71.2% of URLNet to 86.2% at a False Positive Rate of 0.01%. URLTran using Bert performs the best achieving an F1 score of 99.71%. on the dataset.

# Methodology

Our approach utilizes a deep learning model, specifically a fine-tuned RoBERTa model, to classify URLs as either 'phishing' or 'legitimate'. The project commences with a comprehensive dataset that is first subject to tokenization which dissects the dataset into smaller, more manageable pieces — tokens, which are then uniformly padded to ensure each data entry is of the same length. Tensors are then generated from these tokens, which serve as the inputs to the model during training and testing, with batches set to 32 for training and 64 for testing to maintain computational efficiency.

We enhance the integrity of our model's performance through a 5-fold cross-validation method to ascertain the model's robustness and its performance consistency across different subsets of the data. Following the cross-validation, the RoBERTa sequential classifier takes the stage, utilizing the structured input to discern patterns and make predictions. The outcomes of the classification are then meticulously compiled into a comprehensive result. Finally, these results are visualized through various graphical representations, such as charts and plots, to provide an intuitive understanding of the model's performance metrics, class balance, and the nature of the data processed. The flow of the proposed method is presented in Figure 4.

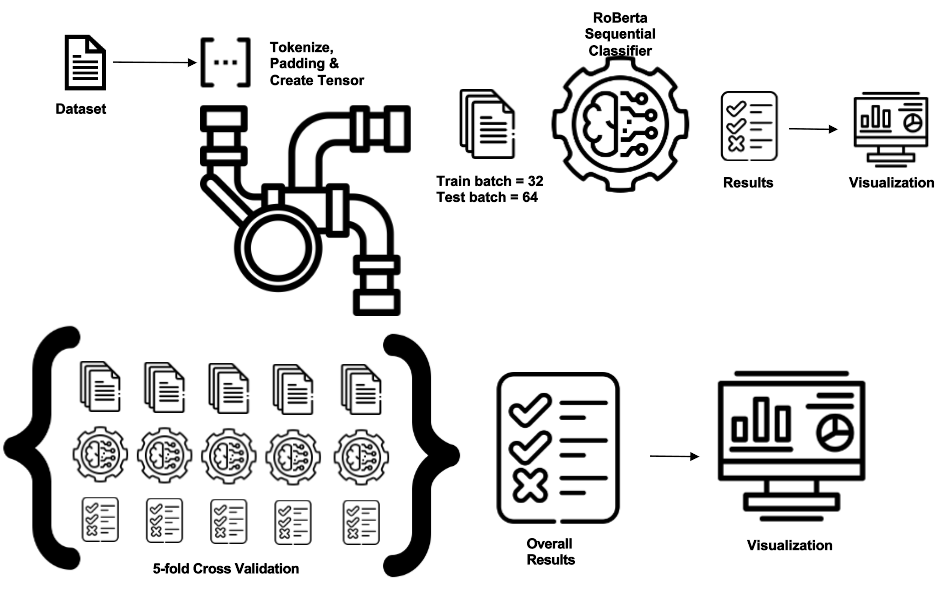


Fig. 4. Flow of the proposed method

## Data Preprocessing:

Data preprocessing involves cleaning and transforming URL data to make it suitable for analysis. First, the data is cleaned to remove any irrelevant or redundant information. This step might include removing duplicate URLs, filtering out irrelevant data points, and handling missing values. Then it involves tokenizing, where URLs are tokenized into smaller units, known as tokens. This process breaks down complex URL structures into manageable pieces, making them easier to analyze. Each token represents a component of the URL, such as the protocol, domain name, or path [1]. Then we transform tokens into numerical representations using RoBERTa's tokenizer.

## Model Architecture - RoBERTa:

In our phishing detection system, we employed the RoBERTa (Robustly Optimized BERT Approach) model, renowned for its state-of-the-art performance in text classification tasks. RoBERTa builds upon the BERT model, which revolutionized natural language processing by introducing a transformer-based architecture. Transformers use attention mechanisms to process words in relation to all other words in a sentence, rather than one at a time [26]. This allows for a deeper understanding of context. It works by processing words in context to their surroundings in a sentence, which is ideal for understanding the structure and semantics of URLs. Figure 4 depicts the model architecture employed in the study.

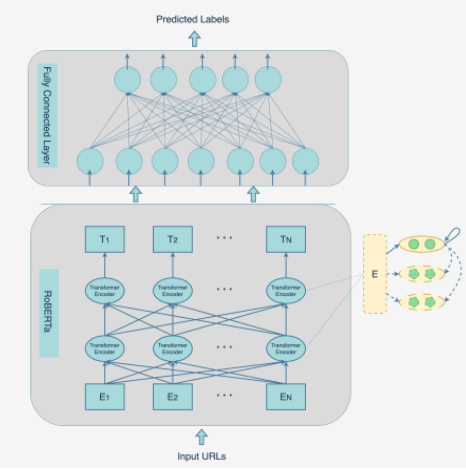


Fig. 5. Model Architecture

The RoBERTa model involves several internal steps for processing URL dataset:

#### Input Representation

* *Tokenization*: The RoBERTa tokenizer converts input URLs into tokens. This includes splitting the text into words or subwords and converting them into numerical IDs.
* *Special Tokens*: Special tokens like [CLS] (used at the beginning of each example) and [SEP] (used to separate segments) are added. The [CLS] token's final hidden state is used as the aggregate representation for classification tasks.
* *Positional Encodings*: Since transformers do not use recurrence or convolutions, positional encodings are added to give the model information about the position of each token in the sequence.

#### Feature Extraction

* *Input Encoding*: Each token is encoded into a high-dimensional vector representation. This encoding captures both the token itself and its positional information, allowing the model to understand the order of tokens in the sequence
* *Embedding Layer*: The tokenized input is passed through an embedding layer. This layer converts each token into a vector, which encapsulates its semantic meaning. The embedding layer is pre-trained on a large corpus of text data, enabling it to understand a wide range of linguistic nuances.
* *Transformer Layers*: RoBERTa uses multiple layers of transformers. Each transformer layer consists of multi-head self-attention mechanisms and feedforward neural networks. This architecture enables the model to capture complex word relationships and dependencies.
* *Bidirectional Context:* The model processes each token by considering its context from both the left and the right side of the sequence, allowing for a deeper understanding of the text.

#### Internal Processing

* *Self-Attention Mechanism:* RoBERTa employs a self-attention mechanism in its layers. This mechanism allows the model to focus on different parts of the input sequence when processing a particular token. It essentially weighs the importance of each token relative to others in the sequence, enabling the model to capture dependencies and relationships between different tokens.
* *Transformer Blocks*: The self-attention output is passed through several transformer blocks. Each transformer block contains a multi-head self-attention layer and a feed-forward neural network. These blocks apply non-linear transformations to the data, allowing the model to learn complex patterns and relationships in the text.
* *Layer Normalization*: Each transformer block is followed by layer normalization. This step is crucial for stabilizing the training process and ensuring that the values across each layer are normalized. It helps the model to train more effectively and avoid issues like vanishing or exploding gradients.
* *Pooling Operations*: Finally, the outputs from the transformer blocks are subjected to pooling operations. This could be pooling, max pooling, or other techniques to summarize the information into a fixed-length representation. In some cases, the hidden state corresponding to the first token (often a special token like [CLS] in BERT) is used as a representation of the entire sequence.

#### Classification Layer

On top of the RoBERTa base model, a fully connected neural network layer is added for the classification task. This layer takes the output from the [CLS] token and outputs a vector of size equal to the number of classes (in this case, 'phishing' and 'legitimate'). The softmax activation function is applied to this vector to obtain a probability distribution over the classes.

#### Training and Fine-Tuning.

The model was trained using a batch size of 32. A smaller batch size was chosen to ensure a good balance between the model's learning efficiency and memory constraints. The learning rate was set at 2e-5, a relatively low value to allow for gradual, more precise adjustments to the model's weights during training. Training was conducted for 4 epochs. This number of epochs was selected to allow the model sufficient exposure to the training data while preventing overfitting. AdamW optimizer was used to update the model's weights effectively during training.

The entire model, including the RoBERTa base and the classification layer, is fine-tuned for the phishing detection task. This involves training the model on a labeled dataset of phishing and legitimate URLs, allowing the model to adapt its weights specifically to this task. The cross-entropy loss function is used during training, which is typical for classification tasks. This function measures the difference between the predicted probability distribution and the true distribution (the actual labels).

#### Output

For each input URL, the model outputs a probability distribution across the two classes [27]. The class with the higher probability is chosen as the model's prediction, categorizing the URLs as either 'phishing' or 'legitimate'.

## Validation and Evaluation

The dataset was divided into five folds, with the model being trained on four folds and validated on the fifth fold. This process was repeated five times, each time with a different fold used for validation. We monitored accuracy, F1 score, precision, recall, and ROC-AUC score during training and validation.

## Integration of SHAP with RoBERTa

The methodology we employ goes beyond detection; it also integrates the SHAP framework to provide interpretable outcomes. SHAP values offer a method for explaining the output of machine learning models [28]. They work by attributing the prediction of a model to its input features, which in the case of our text-based model, are individual tokens (words or subwords) in the input text. The key idea is to quantify the contribution of each token to the model's prediction, helping us understand how much each token sways the model towards classifying a text as 'phishing' or 'legitimate'.

We utilized SHAP's text explainer, which works by creating perturbations of the input URL (i.e., modifying, adding, or removing tokens) and observing how these changes affect the model's output. By repeatedly altering the input text and feeding these variations into the RoBERTa model, the explainer can calculate the impact of each token on the model's classification decision. For each input text, the SHAP explainer assigns a SHAP value to each token. A positive SHAP value indicates that the presence of the token increases the likelihood of the text being classified as phishing, while a negative value suggests the opposite.

The analysis of SHAP values allows us to identify which tokens (words or phrases) are strong indicators of phishing [28]. These tokens typically include specific keywords or phrases commonly used in phishing attempts. By highlighting these key tokens, SHAP values provide deeper insights into the tactics used in phishing attacks [29]. This comprehensive approach not only identifies phishing attempts with high accuracy but also sheds light on the 'why' behind the predictions, making it a step forward in the domain of cybersecurity.

# Experimental Settings

## Dataset

Our study utilized a comprehensive dataset specifically designed for phishing detection. Comprising a comprehensive collection of 550,000 distinct URLs, it was categorized into two classes: Class 0, which consisted of URLs identified as legitimate and safe for users, and Class 1, which comprised URLs flagged as phishing attempts designed to deceive users. As shown in Fig. 5 the class distribution of the dataset was visibly imbalanced, with a greater number of phishing URLs present.

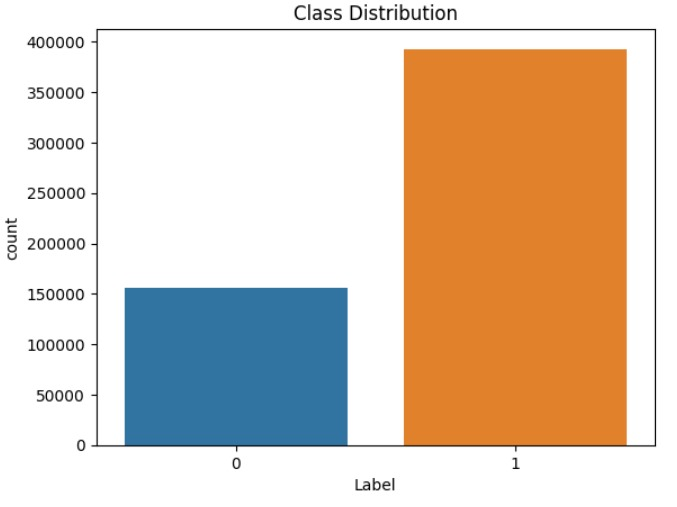


Fig. 6. Class Distribution of the Dataset

## Preprocessing

Utilizing the RoBERTa tokenizer, URLs were converted into standardized token sequence format. This process involves breaking down each URL into discrete elements, or 'tokens', which serve as the input for the subsequent machine learning models. To maintain consistency across the dataset, each sequence was subjected to padding and truncation strategies. We handled missing values and anomalies to maintain data integrity. No data augmentation techniques were applied to preserve the original dataset characteristics.

## Model Training

We selected RoBERTa as our model architecture, using the 'roberta-base' version. The optimizer of choice was AdamW, with a learning rate of 1e-4 and weight decay set to 1e-3, providing a balance between fast convergence and preventing overfitting on the training data. Our loss function was Cross-Entropy Loss, with class weights adjusted for imbalance, to ensure that the model does not become biased towards more frequent labels. For the training process, we implemented a k-fold cross-validation approach with our dataset, comprising 40,000 entries, to enhance the model's generalizability

## Hyperparameters

The model was trained for a total of 6 epochs.Training and testing were efficiently executed with batch sizes of 32 and 64, respectively.

## Evaluation Metrics

The Proposed model performance was assessed using various metrics: accuracy, F1 score, precision, recall, and ROC-AUC. We utilized SHAP's text explainer, a tool specifically designed for interpreting natural language processing models. This explainer is integrated with the RoBERTa tokenizer.

## Runtime Environment

The experiments were conducted on the Kaggle IPython Notebook environment, chosen for its ease of use and integration with various data science tools and libraries. To facilitate the model's training and to handle the substantial computational requirements, we utilized 2 T4 GPUs. These GPUs allowed us to train our model efficiently and explore different hyperparameters within a reasonable timeframe. Moreover, the model training was parallelized to fully utilize computational resources, reducing the training time and increasing the efficiency of our experiments.

# Results and Discussion

## A.Model Performance and Metrics

The study involved a large dataset of 550,000 URLs, classified into two categories: legitimate and phishing. The model achieved a 98.32% accuracy rate, a significant improvement over the existing methods. This high accuracy indicates the model's strong capability in distinguishing between phishing and legitimate URLs. The success is attributed to RoBERTa's advanced feature extraction, capturing complex patterns and relationships in URL data [10]. A subset of 40,000 URLs was subjected to 5-fold cross-validation. The model achieved an accuracy of 97.45%, reinforcing its reliability across different subsets of data. The slightly lower accuracy in cross-validation compared to the larger dataset might be due to variations in data subsets, showcasing the model's adaptability.

Metrics such as accuracy, F1 score, precision, recall, and AUROC (Area Under the Receiver Operating Characteristics) are used to evaluate the model's performance. Table I presents the results of the phishing URL detection model.

1. Performance Metrics

| Metrics | Dataset Size | Accuracy | F1 Score | Precision | Recall | AUROC |
| --- | --- | --- | --- | --- | --- | --- |
| Method |
| Train/Test Split | 550,000 | 0.9832 | 0.9832 | 0.98 | 0.98 | 0.983 |
| 5-fold Cross Validation | 40,000 | 0.9745 | 0.9743 | 0.97 | 0.97 | 0.975 |

Precision-recall curves, ROC curves, and confusion matrices were used. These visual representations provided deeper insights into the model's performance, particularly in terms of false positives and negatives. Figures 7 present the confusion matrix for the Train/Test Split with class\_weights method.

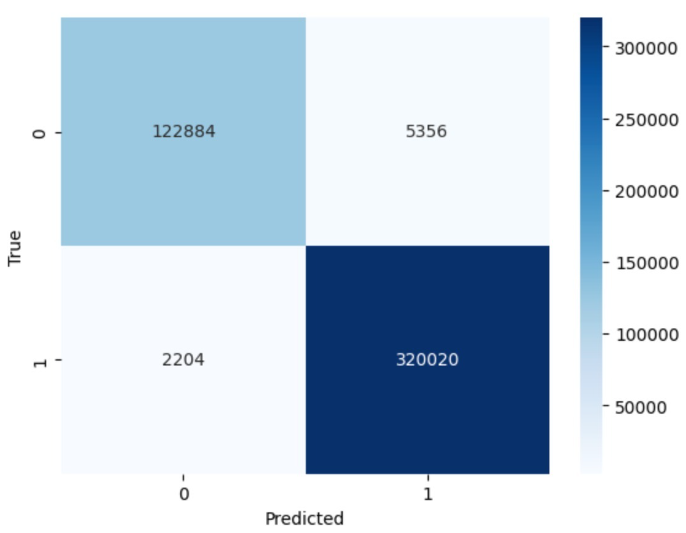


Fig. 7. Confusion Matrix for Train/Test Split with class\_weights method

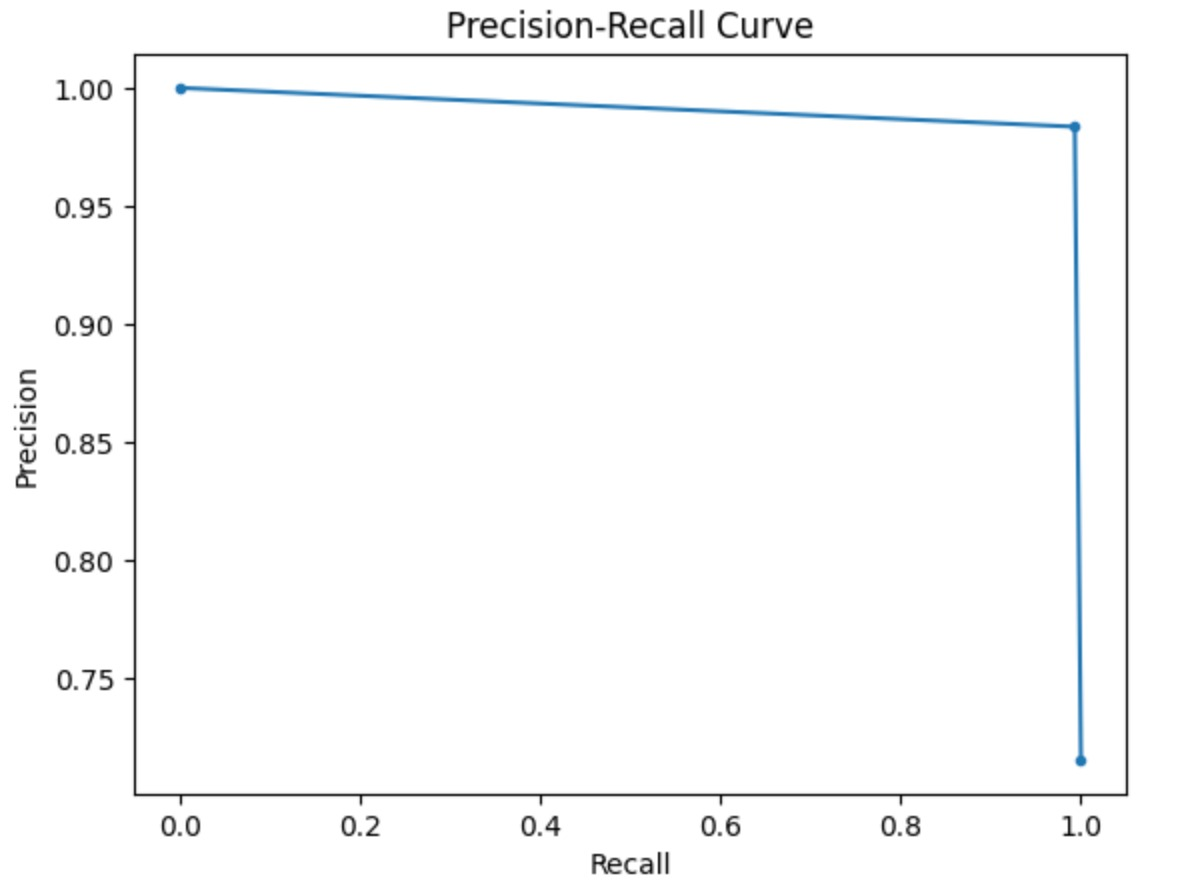


Fig. 8. PR curve for Train/Test Split with class\_weights method

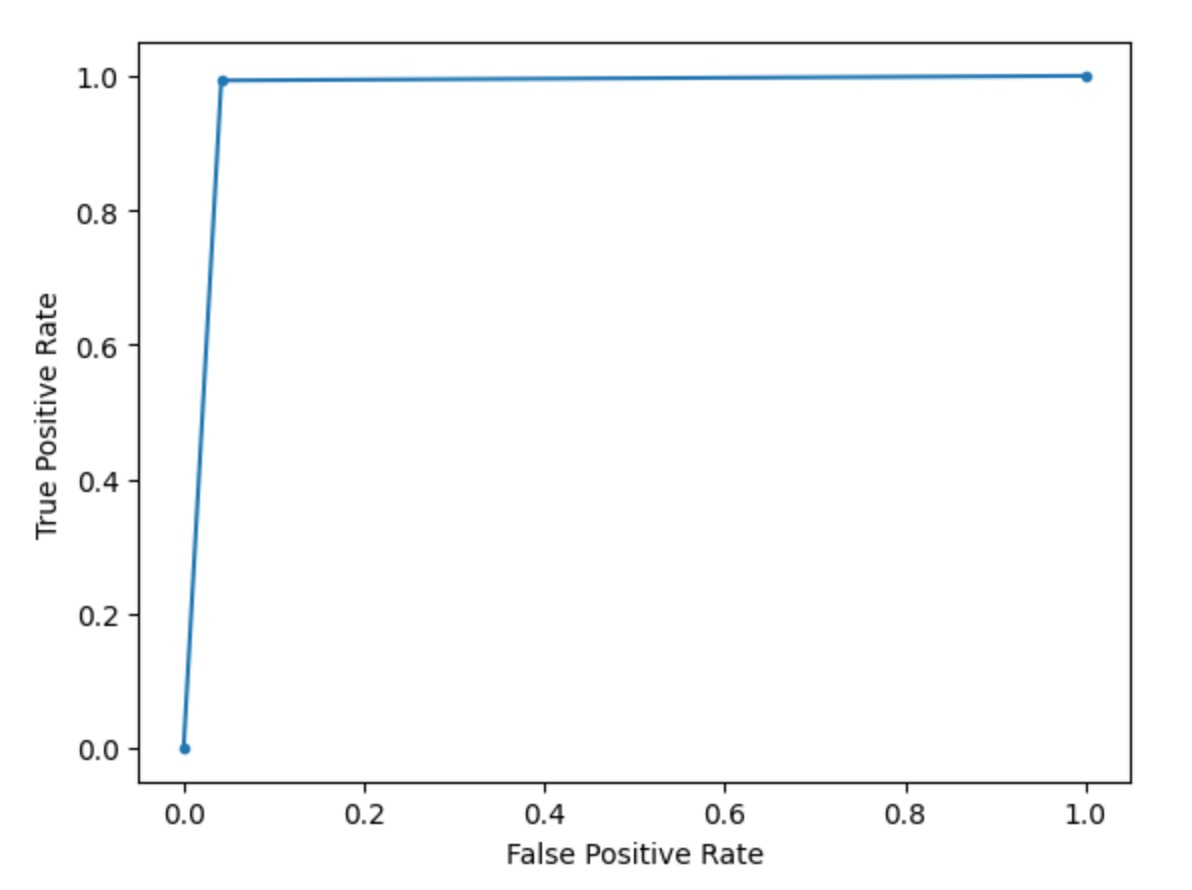


Fig. 9. ROC curve for Train/Test Split with class\_weights method

From the PR and ROC curve, presented in Figure 8 & 9 respectively, it is evident that our model is achieving both high precision and high recall. This balance indicates the model's strong capability in correctly identifying true positive cases while maintaining a low rate of false positives.

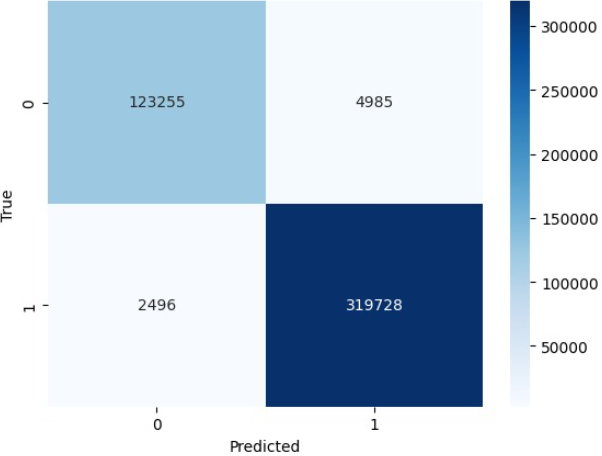


Fig. 10. Confusion Matrix for 5-fold Cross Validation method

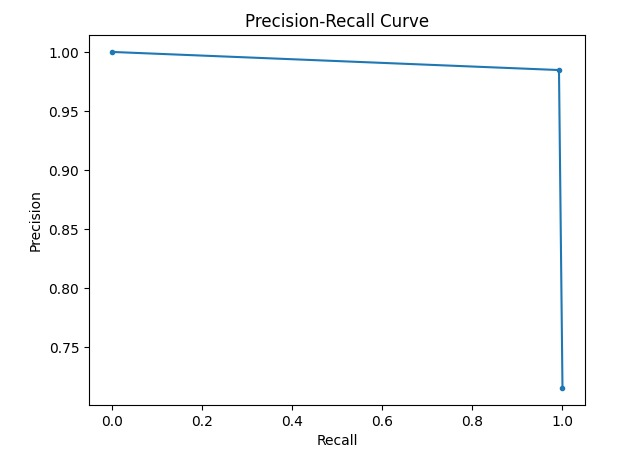


Fig. 11. PR curve for 5-fold Cross Validation method

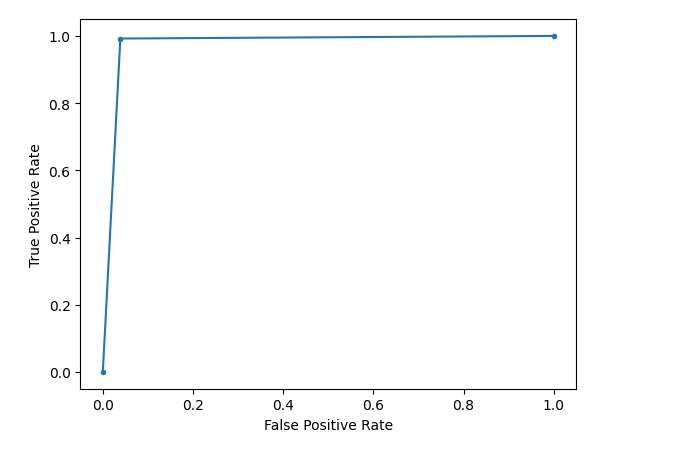


Fig. 11. ROC curve for 5-fold Cross Validation method

From the analysis of the PR curve and the ROC curve shown in Fig. 10 and 11, it is clear that the model is achieving both high precision and high recall.

The model's success in both scenarios is reflected not only in its high accuracy rates but also in the precision-recall curves, ROC curves, and confusion matrices plotted from the results. These metrics indicate the model's robustness in accurately identifying phishing URLs. use of RoBERTa for feature extraction contributed significantly to the success, owing to its ability to capture contextual relationships within URLs. The effective training process, involving optimization techniques and loss functions tailored to the task, further enhanced the model’s performance. The gradual decrease in loss and steady increase in accuracy during training suggest effective learning without overfitting. The confusion matrix and classification report reveal the model's proficiency in correctly identifying phishing URLs with minimal errors.

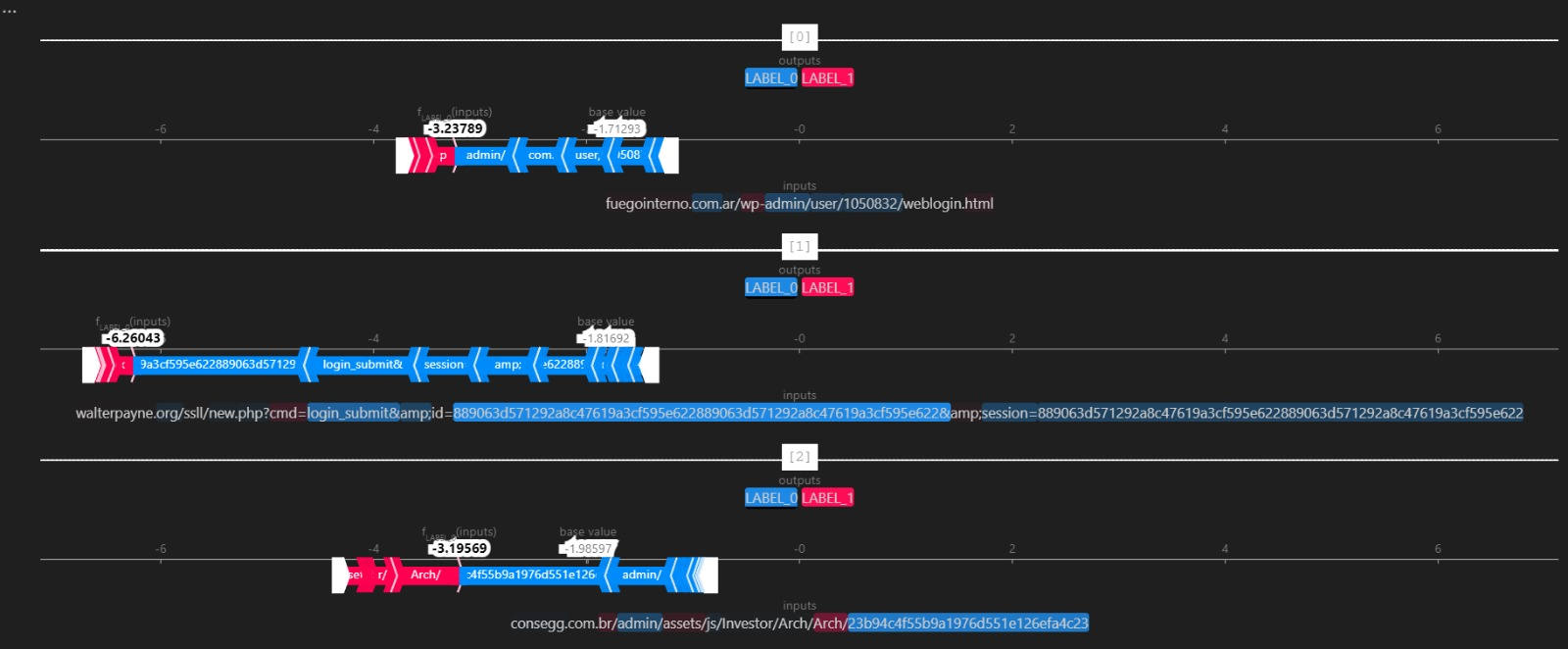


Fig. 12. SHAP's text explainer Output

We utilized SHAP's text explainer to illustrate the output from roberta-base model analysis using SHAP values. This helps understand the impact of each feature on the model's decision-making process [28]. In fig. 12, we see a series of inputs on the left with their corresponding SHAP values, which quantify the contribution of each feature to the model's output. The red and blue bars indicate positive and negative SHAP values, respectively. A positive SHAP value (red) suggests that the presence of that feature pushes the model's prediction higher, while a negative SHAP value (blue) indicates a feature that contributes to a lower prediction value. The base value represents the average model output over the dataset, serving as a reference point from which the contributions of each feature are measured. The outputs at the top right of fig. 12 show two labels, likely the predicted classifications, and the impact of each feature on shifting the prediction from the base value towards one of these labels. The strength and direction of these SHAP values provide insight into why the model makes a particular prediction, enhancing the transparency and interpretability of complex machine learning models. This type of analysis is crucial for validating model behavior and ensuring fair and accurate predictions, especially in high-stakes domains where understanding the rationale behind algorithmic decisions is essential.

## Overall Results and discussion

In our study, we compared our proposed method with established baseline models to validate its effectiveness in detecting phishing URLs (Table II). Jishnu et al.'s model [10], which employs deep learning techniques, demonstrated an accuracy of 97.14%. This approach underscores the effectiveness of deep learning in phishing detection but also sets a high benchmark for our model to surpass. Additionally, we included the DistilBERT model as a baseline, a streamlined variant of the BERT architecture, achieving an accuracy of 96.37%. DistilBERT served as a relevant comparison due to its efficient performance in NLP tasks, while being less resource-intensive than full-scale models.

1. Comparison with Baseline MOdels

|  |  |  |
| --- | --- | --- |
| Models | Applied Approach | Accuracy |
| 1. Jishnu et al. [10] | Deep Learning | 97.14 |
| Baseline Model | DistilBERT | 96.37 |
| Proposed Method | Train/Test Split with class\_weight | 0.9832 |

Our proposed method, which utilized a train/test split with class weighting, achieved an accuracy of 98.32%, significantly outperforming both the deep learning model of Jishnu et al.[10] and the DistilBERT baseline. This superior performance illustrates the efficacy of our approach in handling class imbalance and extracting meaningful features from URLs. Furthermore, under a 5-fold cross-validation scheme, our model maintained a high accuracy of 97.45%, demonstrating its robustness and consistency across different data subsets. These comparisons not only validate the superiority of our proposed method over the baselines but also highlight its potential as a reliable tool for phishing detection in varied scenarios.

However, the study faced limitations due to the constraints of the free version of Kaggle, which restricted the runtime to 12 hours. This limitation necessitated a reduction in the dataset size and the number of training epochs, which might have impacted the depth of model training and its generalizability. Despite these constraints, the model demonstrated remarkable effectiveness in detecting phishing URLs, as indicated by the improvement in accuracy compared to the baseline models. This suggests that even under restricted conditions, the model was able to learn significant features pertinent to phishing detection. The high accuracy rates support the claim that deep learning models can effectively distinguish between legitimate and malicious URLs, offering a robust tool against online threats.

The study also highlights the need for continuous model updates and adaptations. The dynamic nature of phishing tactics requires models to evolve and incorporate new data to maintain their effectiveness. Overcoming the limitations of dataset size and training time could further refine the model's accuracy and reliability. The lessons learned from this study point towards a continuous effort in improving and updating phishing detection methodologies.

# Conclusion and Future work

In this study, we presented a comprehensive approach to phishing URL detection through the application of the RoBERTa model, which included the integration of SHAP values for enhanced interpretability, has been pivotal in achieving a high accuracy rate of 98.34% on a diverse dataset of 550,000 URLs. This performance not only signifies a substantial improvement over traditional phishing detection methods but also highlights the effectiveness and reliability of our proposed model. The utilization of RoBERTa's advanced feature extraction capabilities, combined with SHAP's interpretative analysis, has provided deep insights into the decision-making process of the model, thereby increasing its trustworthiness and applicability in real-world scenarios. Moreover, the model's ability to effectively interpret complex patterns and contextual information inherent in URLs positions it as a challenging tool against phishing attacks.

The future work involves extending SHAP values across the full dataset to enhance model interpretability. This will be achieved by deploying optimized computational strategies for large-scale data analysis. The goal is to gain a thorough understanding of how individual features influence model predictions. Another area of focus is the integration of additional machine learning models following the RoBERTa layer to improve the overall predictive performance and robustness. Potential models for integration include SVM, Random Forest, and other neural networks, aimed at refining the outputs from RoBERTa. This strategy will lead to a multi-model system where each model contributes to a more nuanced analysis. Additionally, increasing the number of training epochs will provide the model with more opportunities to learn and adapt to complex patterns in the data. To balance this approach, regularization techniques can be implemented to mitigate the risks of overfitting, particularly crucial when extending training duration and dataset size.

##### References

1. A. Ali, et al. "An effective phishing detection model based on character level convolutional neural network from URL." Electronics 9.9 (2020): 1514. J.
2. Z. Mouad, and B. Outtaj. "A novel lightweight URL phishing detection system using SVM and similarity index." Human-centric Computing and Information Sciences 7.1 (2017): 1-13.
3. A. Rômulo, and C. Westphall. "Heuristic phishing detection and URL checking methodology based on scraping and web crawling." 2020 IEEE International Conference on Intelligence and Security Informatics (ISI). IEEE, 2020.
4. B. Eda, et al. "A comprehensive study of features and algorithms for URL-based topic classification." ACM Transactions on the Web (TWEB) 5.3 (2011): 1-29.
5. A. El Ayman, et al. "An in-depth benchmarking and evaluation of phishing detection research for security needs." IEEE Access 8 (2020): 22170-22192.
6. S. Asadullah, and S. Singh. "A systematic literature review on phishing website detection techniques." Journal of King Saud University-Computer and Information Sciences (2023).
7. D. David., and A. Abbasi. "Phishmonger: A free and open source public archive of real-world phishing websites." 2016 IEEE conference on intelligence and security informatics (ISI). IEEE, 2016.
8. B. Aaron, et al. "Lexical feature based phishing URL detection using online learning." Proceedings of the 3rd ACM Workshop on Artificial Intelligence and Security. 2010.
9. D. Rachna, D. Tygar, and M. Hearst. "Why phishing works." Proceedings of the SIGCHI conference on Human Factors in computing systems. 2006.
10. K. Jishnu, and B. Arthi. "Phishing URL detection by leveraging RoBERTa for feature extraction and LSTM for classification." 2023 Second International Conference on Augmented Intelligence and Sustainable Systems (ICAISS). IEEE, 2023.
11. “Phishing Activity Trends Report” <https://apwg.org/trendsreports/>
12. A. Nureni Ayofe, et al. "Adopting automated whitelist approach for detecting phishing attacks." Computers & Security 108 (2021): 102328.
13. D. Avisha, et al. "SoK: a comprehensive reexamination of phishing research from the security perspective." IEEE Communications Surveys & Tutorials 22.1 (2019): 671-708.
14. X. Guang, et al. "Cantina+ a feature-rich machine learning framework for detecting phishing web sites." ACM Transactions on Information and System Security (TISSEC) 14.2 (2011): 1-28.
15. R. Srinivasa, T. Vaishnavi, and A. Roshan . "CatchPhish: detection of phishing websites by inspecting URLs." Journal of Ambient Intelligence and Humanized Computing 11 (2020): 813-825.
16. F. Mark, C. Kreibich, and V. Paxson. "On the Potential of Proactive Domain Blacklisting." LEET 10 (2010): 6-6.
17. S. Ozgur Koray, et al. "Machine learning based phishing detection from URLs." Expert Systems with Applications 117 (2019): 345-357.
18. R. Routhu Srinivasa, and A. Roshan. "Two level filtering mechanism to detect phishing sites using lightweight visual similarity approach." Journal of Ambient Intelligence and Humanized Computing 11.9 (2020): 3853-3872.
19. M. Pranav, et al. "URLTran: Improving phishing URL detection using transformers." MILCOM 2021-2021 IEEE Military Communications Conference (MILCOM). IEEE, 2021.
20. N. Riyadh Rahef & M. Selvakumar. (2023). A Critical Review: A New Taxonomy for Phishing Attacks Based on Phishing Techniques Used. Wasit Journal for Pure sciences. 2. 251-269. 10.31185/wjps.143.
21. A. Mohammed & A. Mohammad & M. Khaled & B. Sherenaz. (2021). URL Phishing Detection using Machine Learning Techniques based on URLs Lexical Analysis. 147-152. 10.1109/ICICS52457.2021.9464539.
22. M. Aljabri and S. Mirza, "Phishing Attacks Detection using Machine Learning and Deep Learning Models," 2022 7th International Conference on Data Science and Machine Learning Applications (CDMA), Riyadh, Saudi Arabia, 2022, pp. 175-180, doi: 10.1109/CDMA54072.2022.00034.
23. F. Salahdine, Z. El Mrabet and N. Kaabouch, "Phishing Attacks Detection A Machine Learning-Based Approach," 2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 2021, pp. 0250-0255, doi: 10.1109/UEMCON53757.2021.9666627.
24. A. Zafar, M., Liu, et al. A comprehensive survey of AI-enabled phishing attacks detection techniques. Telecommun Syst 76, 139–154 (2021). <https://doi.org/10.1007/s11235-020-00733-2>
25. X. Ding and Y. Ding, "Research on roBERTa-based fraudulent information identification and interpretable AI," 2021 International Conference on Computer Information Science and Artificial Intelligence (CISAI), Kunming, China, 2021, pp. 960-964, doi: 10.1109/CISAI54367.2021.00193.
26. P. Maneriker, J. W. Stokes, E. G. Lazo, D. Carutasu, F. Tajaddodianfar and A. Gururajan, "URLTran: Improving Phishing URL Detection Using Transformers," MILCOM 2021 - 2021 IEEE Military Communications Conference (MILCOM), San Diego, CA, USA, 2021, pp. 197-204, doi: 10.1109/MILCOM52596.2021.9653028.
27. S. Dhiman & H. Sohrab & C. Rana & M. Tanni. (2020). Phishing Attacks Detection using Deep Learning Approach. 10.1109/ICSSIT48917.2020.9214132.
28. R. Alenezi and S. A. Ludwig, "Explainability of Cybersecurity Threats Data Using SHAP," 2021 IEEE Symposium Series on Computational Intelligence (SSCI), Orlando, FL, USA, 2021, pp. 01-10, doi: 10.1109/SSCI50451.2021.9659888.
29. N. Puri, P. Saggar, A. Kaur and P. Garg, "Application of ensemble Machine Learning models for phishing detection on web networks," 2022 Fifth International Conference on Computational Intelligence and Communication Technologies (CCICT), Sonepat, India, 2022, pp. 296-303, doi: 10.1109/CCiCT56684.2022.00062.