UCLouvain

École polytechnique de Louvain

Robustness Analysis of a Multi-Component Phishing Detection Model Against Explainable AI-Guided Adversarial Attacks

Authors: **Liza Denis, Charline Meurant**
Supervisors: **Etienne Rivière, Charles-Henry Bertrand Van Ouytsel, Emilie Deprez**
Reader: **Yves Deville**
Academic year 2024–2025
Master [120] in Computer Science and Engineering

Contents

Acknowlegments

We would like to express our deep gratitude to our supervisor, Prof. Etienne Rivière, for his availability, constructive feedback, guidance, and invaluable advice.

We would also like to thank Charles-Henry Bertrand Van Ouytsel and Emilie Deprez, from the teaching assistant team, for their valuable support, insightful feedback, and helpful discussions throughout this work. A special thanks goes to Charles-Henry for providing us with excellent articles and resources that have greatly supported and enriched our research.

We would like to sincerely thank our family, friends, and roommates for their unwavering support, patience, and understanding throughout this challenging period. A special thanks goes to Albane Denis for her daily support. We also extend our heartfelt thanks to Charlotte Vanneste for her unwavering support and constant encouragement throughout this journey.

We hereby acknowledge the use of ChatGPT exclusively for language refinement, such as grammar and style corrections, as well as occasional assistance in debugging and improving parts of the code used in this thesis. All content and research have been conducted and written entirely by us; ChatGPT was not used to generate or create any textual content.

Abstract

Phishing remains one of the most persistent threats in cybersecurity, and while numerous detection techniques have been developed, machine learning-based approaches now dominate over earlier rules like heuristic-based methods. In this work, we investigate the robustness of a machine learning (ML) model trained on multiple components of phishing emails, when exposed to hand-crafted, explainable AI-guided, model-targeted adversarial attacks. To conduct this analysis, we reproduced and adapted an existing multi-component ML model that processes the structure, text, and URLs of emails separately and combines the results for final classification. The final system achieves a high Area Under the Precision-Recall Curve (AUPRC), reaching 0.9856. Based on model analysis driven by explainable AI (XAI) techniques, we crafted targeted attacks. These achieved a 34% success rate against the component handling the email structure, and up to 100% success rates against the components handling the email text and embedded URLs. The experiments were conducted on a dataset composed mainly of publicly available phishing emails, supplemented with some private data. The results highlight that high-performing models remain vulnerable to adversarial attacks, underscoring the need to balance interpretability with security and to maintain user awareness campaigns as a key layer of defense.

Chapter 1

Introduction

Phishing emails remain one of the most persistent and adaptable cyber threats, leveraging both technical subterfuge and psychological manipulation to deceive users. In recent years, phishing attacks have continued to evolve, not only in volume but also in their methods and targeted sectors [1].

This growing sophistication in attack strategies calls for equally advanced and transparent detection systems. Yet many existing phishing detectors rely on black-box models that make decisions, which are difficult to interpret or justify [2]. This lack of transparency is problematic not only for user trust and system auditing, but also for understanding model limitations. Explainable AI (XAI) techniques offer a promising direction to bridge this gap by shedding light on which features drive predictions of a model.

Another important aspect in phishing detection is the evaluation of model robustness. While a model may perform well on a test set, its reliability in the real world can be compromised by adversarial inputs, i.e., emails that are intentionally crafted to fool the model while still appearing legitimate to users. This is especially relevant because attackers are likely to use targeted, model-aware strategies in practice. By leveraging insights from XAI, adversaries could identify weaknesses of a model and design inputs that exploit them [3]. Thus, assessing how models respond to explainability-guided adversarial attacks is not about fixing their weaknesses, but about uncovering and understanding them, an important step toward developing phishing detectors that can be realistically deployed.

To better illustrate the dynamics at play, Figure 1.1 presents an overview of the actors and interactions involved in this process. It shows how explanations (1) derived from the predictions of a model (2) can be exploited by attackers (3) to craft targeted inputs, and how defenders (4) can use the same explanations to identify vulnerabilities and improve model robustness.

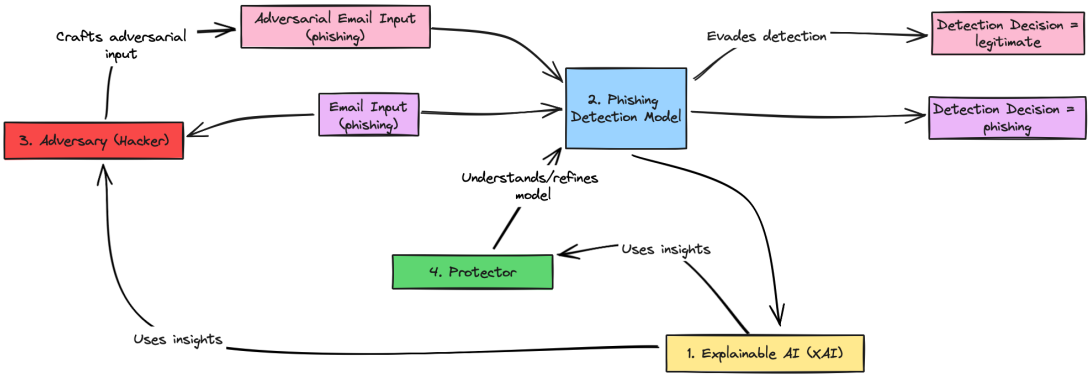


Figure 1.1: This diagram illustrates how model explanations can be used by both defenders and attackers. A phishing "Email Input" is correctly detected as phishing by the model. Using XAI, the protector analyzes the decision process of the model to improve its robustness. However, the adversary can also exploit these insights to craft a modified "Adversarial Email Input" designed to evade detection. The goal of this adversarial input is to remain malicious while being misclassified as legitimate.

Taken together, these limitations lead to a critical research question: **To what extent can a machine learning model, trained on multiple components of phishing emails, remain robust when exposed to hand-crafted, explainability-guided, model-targeted adversarial attacks?**

In response to these challenges, this research aims to leverage explainability techniques to expose and better understand robustness vulnerabilities in phishing detection. For this purpose, we start by reimplementing and adapting a phishing detection model that combines multiple components of an email, including its structure, text, and URLs. Then, we apply explainability methods to enhance the transparency of the model and gain insight into its decision-making process. Finally, we subject the model to systematic robustness testing through adversarial manipulations guided by these XAI insights. The goal is to critically assess whether such a model can withstand real-world, model-targeted attack scenarios.

We can summarize this by enumerating the three objectives this thesis sets out to achieve:

1. **Reimplement and Adapt a Multi-Component Detection Model**: We re-implement and adapt a phishing detection algorithm that integrates and analyzes multiple parts of an email—including metadata, text and URLs—in order to improve classification performance.
2. **Incorporate Explainable AI (XAI)**: We apply explainability techniques to interpret the decisions of the model, gain insight into which features influence its predictions, and enhance the overall transparency of the system.

3. **Evaluate Model Robustness via Adversarial Attacks**: We leverage XAI insights to guide the design of adversarial attacks, allowing us to evaluate the robustness of the model and explore whether machine learning detection alone is sufficient to counter phishing.

The last objective is pursued using phishing emails sourced from public datasets. Their use is carefully controlled and adapted to avoid introducing unfamiliar or low-quality samples that could lead to unpredictable behavior or bias in the results.

The following chapters provide a structured overview of the work presented in this thesis. Chapter 2 covers the State of the Art, introducing phishing, email structure, the datasets used, existing detection methods, evaluation metrics, explainable AI (XAI), and a summary of adversarial attack techniques.

Chapter 3 then focuses on the detection model we adopted and adapted for our study. Based on an existing architecture, the model processes emails through three dedicated modules—structure, text, and URL—each handling a specific aspect of the input. Their outputs are combined using a meta-classifier to produce the final prediction. We describe the original model, the modifications introduced to each module and the overall performance of the system. This chapter also highlights key limitations of the approach.

Finally, the experimental study is detailed in Chapter 4. Each module is independently evaluated through a series of targeted adversarial attacks to assess its robustness. The thesis concludes in Chapter 5 with a summary of the findings and a discussion of potential directions for future research.

Ethical Considerations

Given the nature of this research, focused on adversarial techniques applied to phishing detection, ethical considerations were carefully integrated into every stage of the project. This approach was guided by well-established ethical frameworks, including The Menlo Report [4] and the USENIX Security 2025 Ethics Guidelines [5], which emphasize respect for persons, beneficence, justice, and respect for law and public interest.

Respect for Persons This research involved no interaction with individuals outside the research team, and no third-party personal data was used. Emails were sourced exclusively from public datasets and our own mailboxes, giving us full control over the data and avoiding privacy concerns. Adversarial examples were generated during the attack phase using patterns from public datasets, without involving any real or sensitive communications. As a result, no external individuals were affected, and informed consent principles were respected.

Beneficence This research aims to test the robustness of phishing detection systems by exposing vulnerabilities under adversarial conditions. The attacks targeted a model we

developed ourselves, adapted from an existing design but trained with different datasets and design choices, eliminating disclosure risks. Insights are not shared to support malicious use, but rather to highlight potential weaknesses that need to be addressed.

Justice All experiments were conducted using infrastructure and data fully controlled by the two researchers. Only our own email addresses were used, with no involvement of third-party users or systems. This contained setup ensured no external impact while still producing insights relevant to real-world security improvements.

Respect for Law and Public Interest This research was designed to avoid legal or ethical issues by ensuring that no terms of service were violated and no live or third-party systems were accessed. All experiments ran on a locally hosted, self-contained system, ensuring compliance. While the findings may have dual-use potential, we have made efforts to present them in a responsible way that emphasizes defensive perspectives. Still, we acknowledge that interpretation and use of such results cannot be fully controlled once published.

Chapter 2

State of the Art

2.1 Understanding Phishing

Before exploring how detection models can identify and resist phishing, it is important to understand what phishing is, how it operates, and why it remains so effective. This section outlines its definition, life cycle, recent evolutions, main types of attack, and the psychological factors that make phishing such a persistent threat.

2.1.1 Phishing Definition

A clear definition of phishing is essential to frame the subject before going further. Phishing is the act of trying to steal the personal information or financial data of users by using **social engineering** or **technical subterfuge**. Adversaries use social engineering to trick victims into believing they are interacting with a trusted source, and they use technical subterfuge when malware is planted directly on a device to extract sensitive credentials [6].

There is a difference, however, between a phishing email and spam. While both are unwanted and uninvited emails that attempt to persuade the recipient to take a specific action, spam is not necessarily malicious. More specifically, spam refers to unsolicited emails containing commercial messages, advertisements, or promotional links. The intent is typically to promote a product or service, not to steal data or credentials. Spam is problematic in that it clutters inboxes and slows down email servers [7]. This thesis focuses solely on phishing emails.

2.1.2 Phishing Life Cycle

Phishing attacks are not random or isolated incidents; rather, they follow a well-structured process known as the phishing life cycle. This process helps us understand how attackers operate from start to finish. The life cycle consists of four main stages: planning, preparation, execution, and data extraction [8]. Each of these stages is examined in more detail below, with a visual overview provided in Figure 2.1.

The **Planning Phase** is the initial step in the phishing life cycle, during which attackers gather information on their targets to prepare tailored attacks. Their profiles range from low-skilled individuals using pre-built tools to organized cybercriminal groups and cyberterrorists pursuing political or ideological goals.

During the second stage, known as the **Attack Preparation Phase**, attackers identify exploitable vulnerabilities, such as zero-day flaws which exploit unknown software vulnerabilities. They also select the most effective delivery method (email, website, voice call, or SMS), depending on the target and strategy.

Next comes the **Attack Conducting Phase**, where the phishing attempt is executed by the attacker through a malicious link, attachment, or fake login page. The attacker then waits for the victim to interact with the threat, which exploits vulnerabilities to compromise privacy, system integrity, or data security.

The final stage is the **Valuables Acquisition Phase** which involves collecting the data of the victim, either manually (e.g., through social interactions) or automatically (e.g., via fake login forms) by the attacker. The stolen information is then used for illicit purposes such as fraud or unauthorized transactions.

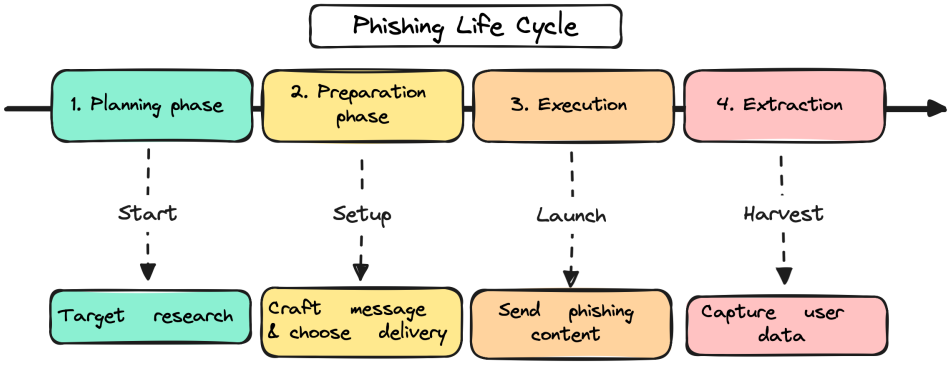


Figure 2.1: Four phases of phishing life cycle.

2.1.3 Evolution of Phishing in 2024

Now that the structure of a phishing attack has been described, it is important to examine how phishing actually manifested in the real world during 2024. This section presents findings from the Anti-Phishing Working Group (APWG), an international coalition of industry, government, and law enforcement organizations that publishes quarterly reports on phishing activity [9]. The data and trends discussed below are based on their analysis [1].

Reported Phishing Attacks in 2024 According to the APWG, after a record high in 2023 with nearly five million phishing attacks, reported volumes dropped sharply in early 2024 — with 963,994 incidents in the first quarter (Q1) and 877,536 in the second quarter (Q2), the lowest levels recorded since 2021. This decline may be partially explained by reporting limitations, as some email providers restricted users from forwarding phishing emails for analysis. In response to these changes, attackers adapted their strategies, leading to a 64% rise in unique campaigns by the fourth quarter (Q4). Activity picked up again in the second half of the year, with attacks reaching 989,123 in Q4. Monthly volumes stabilized between 290,000 and 370,000, confirming a persistent threat. Figure 2.2 shows the overall trend from 2022 to 2024, marked by a sharp rise, a decline, and a partial rebound.

Figure 2.2: Phishing attacks reported by APWG, Q3 2022 - Q3 2024 [10].

Most-Targeted Industry Sectors Beyond the overall phishing volume, the APWG reports a notable shift in the industry sectors most frequently targeted in 2024. Social media platforms, which accounted for 37.6% of attacks in Q1, saw their share fall to 22.5% by Q4. Meanwhile, the SAAS/Webmail sector remained a consistent target throughout the year and ultimately became the most attacked category. *This confirms that webmail remains a key focus for attackers, making email-based phishing a timely subject for research.* Banking, once a primary focus for attackers, dropped significantly—from 24.9% in late 2023 to just 9.8% in early 2024—before ending the year at 11.9%. In contrast, phishing campaigns against eCommerce and retail doubled over the course of the year, reflecting a shift toward sectors with weaker defenses and less user awareness. Payment services remained relatively stable, with around 7% of attacks throughout the year.

2.1.4 Types of Phishing Attacks

According to Alkhalil et al. [8], phishing attacks can be broadly categorized into two types: those that rely on deception and psychological manipulation, and those that exploit technical vulnerabilities. While attacks involving technical subterfuge—such as the use of malicious code or the exploitation of system flaws—represent a significant

area of research, they fall outside the scope of this thesis. The present work focuses on **deceptive phishing**, which is grounded in social engineering and aims to manipulate human psychology rather than technical systems. The most common forms of deceptive phishing are:

- *Phishing Emails*: Still the most common phishing method. More about this type of phishing later.
- *Spoofed Websites*: Websites that imitate the looks of real trusted websites using logos, layouts and domains similar to the initial website. They are often used in emails, advertisements, ...
- *Phone Phishing (Vishing and Smishing)*: Type of attack where they use voice calls (vishing) or text messages (smishing) to impersonate trusted sources. Examples include fraudulent bank alerts or messages containing malicious links.
- *Social Media Phishing (Soshing)*: Attackers target users on platforms like Facebook or Instagram. Different threats can be account hijacking, impersonation attacks, scams, ...

Different Email-Based Phishing Attacks Since this thesis focuses on email-based phishing, it is essential to examine the main types of such attacks. Longtchi et al. [11] identify six distinct categories of email phishing, each varying in method and sophistication:

1. **Generic Phishing**: Emails that have no particular target with no personalization. These messages typically aim to deceive recipients in order to steal money.
2. **Spear Phishing**: Emails that are destined for a specific target. Therefore they are personalized, using the name of the victim, title, ... The attacker makes a big effort to conduct this attack and hardly reuses an email address to send a spear phishing email.
3. **Clone Phishing**: A previously received or sent email is reused but the legitimate links inside the original email are replaced by malicious links. The attacker also fakes the address of the legitimate sender to make it look even more real and unsuspicious.
4. **Whaling**: A specific form of spear phishing where instead of targeting any individual, they target a specific individual, highly ranked individuals such as a CEO. These attacks can cause serious damage to organizations.
5. **Wire Transfer Scams**: Attackers impersonate service providers or executives to pressure victims into sending money under urgent or threatening circumstances.

6. **Business Email Compromise (BEC)**: Spoofed emails that impersonate trusted parties such as CEOs or known customers that try to trick employees into transferring money. BEC has caused billions in losses and is considered the most financially damaging social engineering attack.

2.1.5 The Role of Psychology in Phishing Emails

An essential component of phishing is not just technical execution, but psychological manipulation. The success of a phishing attack often depends on how well it can exploit human cognitive processes. Two key elements are typically manipulated: **believability** and **persuasiveness**. This psychological dimension was explored by Van Der Heijden and Allodi [12], who analyzed how the six principles of influence of Cialdini affect the effectiveness of phishing emails.

To make a phishing message believable, attackers aim to create a sense of authenticity and trust. This often involves *mimicking* logos, tone, or formatting from legitimate sources. *Personalization* further boosts credibility by tailoring content to the context or identity of the recipient. Persuasiveness, in contrast, relies on exploiting cognitive biases—mental shortcuts people use when processing decisions.

The six principles of Cialdini commonly observed in phishing messages include:

- **Authority**: Referencing figures of power, such as "CEO" or "IT Administrator".
- **Scarcity**: Creating urgency or limited-time scenarios.
- **Liking**: Using a friendly tone or mirroring the identity of the recipient.
- **Reciprocity**: People feel obligated to return favors.
- **Consistency**: People align with past commitments or actions.
- **Social Proof**: People look at the behavior of others to guide their own.

In their findings, Van Der Heijden and Allodi reported that *consistency* and *scarcity* were the strongest predictors of user engagement, confirming that certain cognitive triggers significantly increase the likelihood of a click. *Authority* showed limited influence depending on the context, while *social proof* and *liking* had no measurable impact. Surprisingly, *reciprocity* appeared to have a negative effect, possibly undermining the credibility of phishing messages—particularly in financial contexts. These findings highlight the importance of psychological factors in phishing and show how attackers deliberately craft messages to take advantage of how people think and make decisions.

Building on this psychological perspective, Longtchi et al. [11] provide a comprehensive overview of social engineering attacks, emphasizing how attackers systematically exploit human psychology far more effectively than defenders account for it. They

distinguish between Psychological Factors (PFs), such as fear, authority or impulsivity, and Persuasion Techniques (PTs), like impersonation or urgency, which attackers use to exploit those factors. The study identifies a wide asymmetry: *while attackers leverage over 40 PFs and a broad set of techniques, current defense systems only consider a small fraction of these cues.*

The authors categorize PFs into five groups: social, emotional, cognitive, personality-based, and workplace-related factors. Despite the growing sophistication of some psychological-aware defenses, such as BEC Guard or PhishNet-NLP, only a handful of these systems exist, and they still address a limited range of cues. *This mismatch between attacker strategy and defensive coverage may explain why many phishing emails continue to evade detection and succeed in manipulating users.*

2.1.6 Example of Phishing Email

To illustrate the concepts discussed so far, we present in Figure 2.3 a real-world phishing email extracted from the Nazario dataset. In this example, we focus specifically on the content of the email, which demonstrates several of the key elements outlined in the previous sections.

```
Dear USAA Member

We have received multiple account update notifications from you regarding the upgrade
of your account security. Unfortunately we are unable to upgrade your account security
because you didn't complete the process. We urge you to please complete the process for
a good security to be placed on your account. Unable to do this may cause your account
to be compromised by anyone with your information.

CLICK HERE TO COMPLETE YOUR ACCOUNT UPGRADE NOW

Thank you for helping us to server you more better.

Sincerely
USAA.
```

Figure 2.3: Example of body content of phishing email from Nazario dataset.

In terms of phishing type, this is a clear example of **deceptive phishing**, as the email attempts to impersonate an official communication from the institution USAA. It does not qualify as spear phishing—unless it was exclusively sent to actual USAA customers—but is most likely a mass-distributed, **generic phishing** message.

From a psychological perspective, the message appeals to several known influence techniques. It evokes **authority** by referencing a trusted financial institution. It also creates a sense of **scarcity** through implicit urgency—despite the lack of a specific deadline, the tone pushes the recipient to act quickly. Finally, it leverages **consistency**, as the subject matter (account security) aligns with expectations a user might reasonably have from their service provider.

2.2 Understanding Email

In order to better understand and analyze phishing mechanisms, it is essential to have a detailed understanding of the structure of an email and its various components. For this reason, we begin by analyzing its internal composition. Elements such as raw headers and HTML code, which are crucial for this analysis, are only fully visible when accessing the email in its original .eml format.

An email consists of two main parts, each fulfilling distinct roles [13]:

- **Headers:** These contain metadata about the message and are composed of a series of field lines. Each line includes a field name (e.g., `From`, `To`, `Subject`) followed by a colon and a value. Although mostly hidden from the user, headers play a key role in identifying senders and recipients, tracking the route taken by the message through Simple Mail Transfer Protocol (SMTP) relays, and authenticating its legitimacy.
- **Body:** This optional part contains the actual message and can include various types of content, often structured through Multipurpose Internet Mail Extensions (MIME).
  - **Text Message:** A plain text version of the message of the email.
  - **HTML Content:** With MIME, emails can include HTML content, allowing rich formatting, styling, and interactive elements. HTML is structured as a Document Object Model (DOM), a tree-like structure of nested elements.
  - **URLs:** Hyperlinks embedded within the text or HTML content, often used in phishing attacks to redirect users to malicious websites.
  - **Attachments:** MIME also allows the inclusion of files (images, PDFs, executables, etc.), encoded using schemes such as Base64.

Figure 2.4 shows the different components of an email.

2.2.1 Email Header Fields

Email headers are automatically generated metadata fields that provide routing, identification, and authentication information. While certain fields are required (e.g., recipient address, subject), mail servers, using the SMTP protocol, can also append custom or optional headers based on their configuration [15]. The most commonly used fields are detailed in Table 2.1.

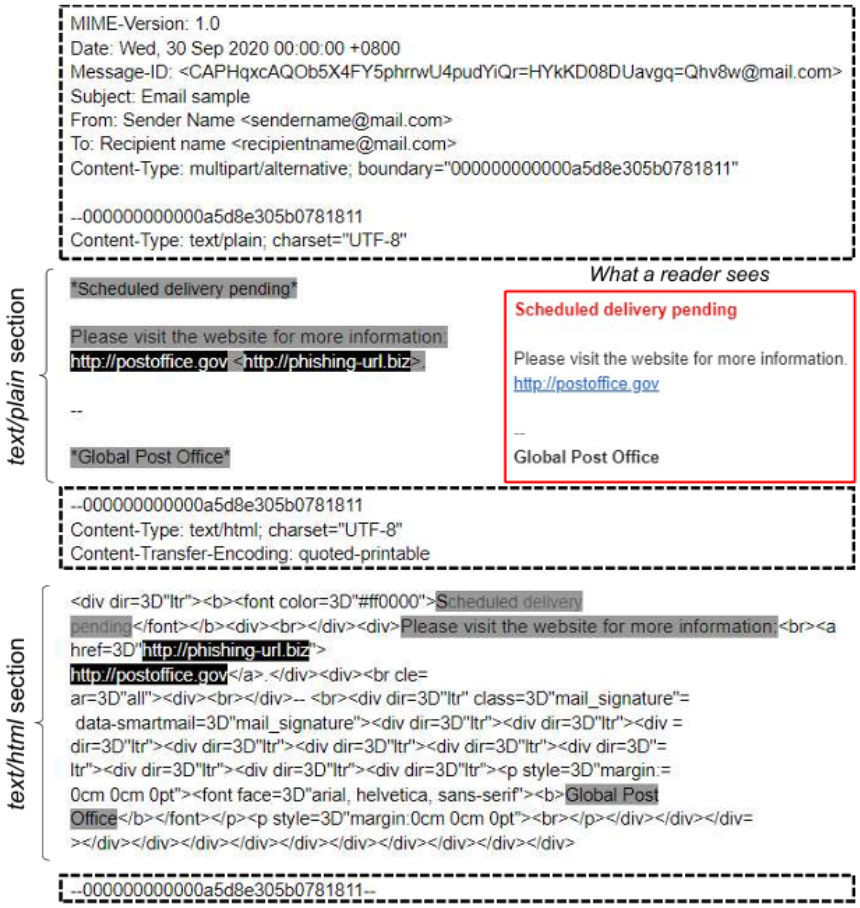


Figure 2.4: Example from Lee et al. [14] of an email showing headers, plain text, HTML, and embedded URLs. Headers are marked with dotted boxes; gray areas represent visible text content; black blocks indicate URLs.

Header	**Description**
`From`, `To`, `Reply-To`, `Date`, `Subject`	Standard fields for sender/recipient identity and message context.
`Message-ID`	Unique identifier for the message.
`Received`	Shows the routing path through SMTP servers.
`In-Reply-To`	Used to follow conversation threads.
`User-Agent`, `X-Mailer`	Indicates the email client or sending software.
`MIME-Version`	Specifies the MIME version used (usually 1.0).
`Content-Type`	Declares the type of content (e.g., text/plain, text/html, multipart).
`Charset`	Character encoding (e.g., UTF-8).
`Boundary`	Separates MIME parts when the content is multipart.
`DKIM`, `SPF`, `DMARC signatures`	SMTP extensions are used to verify the authenticity of the claimed identity of the sender.

Table 2.1: Common header fields found in emails.

2.2.2 Email Body

As previously mentioned, the email body can contain the text message, HTML content, URLs, and attachments. The plain text part is, by nature, just simple text and attachments are out of the scope of this thesis. Therefore, in the following sections, we will focus on the HTML content and URLs.

HTML Content The HTML body of an email follows the structure of a Document Object Model (DOM), which organizes elements in a hierarchical, tree-like form. Each node represents a part of the content (e.g., paragraphs, links, styles), and DOM methods allow for dynamic manipulation of this structure. This makes HTML emails more visually rich and interactive, but also more exploitable for phishing. Table 2.2 lists common HTML nodes found in email messages [16].

Node	Description
`<html>`	Root of the DOM tree.
`<head>`	Contains metadata (title, styles, scripts).
`<style>`	Declares styling rules for content.
`<body>`	Contains visible content of the message.
`<a href>`	Defines a hyperlink to another URL.
`<a data-saferedirecturl>`	Hyperlink with embedded redirection metadata.

Table 2.2: Common HTML nodes found in emails.

URL Structure URLs embedded in email bodies are among the most common phishing vectors. A typical URL starts with the protocol (e.g., `https`), followed by the domain name, which includes the subdomain, second-level domain (SLD), and top-level domain (TLD). The rest of the URL may contain a path with query parameters indicating a specific page or resource on the server. Understanding the structure of URLs is essential in detecting obfuscation or domain spoofing techniques [17]. The components of a typical URL are illustrated in Figure 2.5.

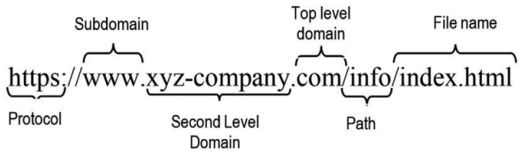


Figure 2.5: Structure and components of a URL from Sahingoz et al. [17].

2.3 Database Analysis

Phishing detection, like many tasks in machine learning, is only as effective as the data that powers it. After understanding the anatomy of an email, we now shift our focus to

the raw material used to train and evaluate detection models: the datasets themselves. This section presents a detailed analysis of the email corpora used in our study.

The databases used for training our models and evaluating their performance were carefully selected based on trustworthiness and usability. We prioritized datasets containing raw emails — including both metadata and body — and those frequently referenced in scientific studies on phishing detection. Since most of the available phishing corpora are in English, we primarily used English-language datasets. In addition, we relied on trusted datasets focused on specific email components, such as URLs, headers, or text content, depending on the task.

Datasets deemed untrustworthy or unusable were excluded, often due to missing information about email origins or because they were provided only in processed formats, like simplified CSV files, leading to the loss of crucial metadata. We also discarded datasets composed solely of spam messages that had not been used in prior phishing studies, as they were originally intended for spam detection and did not align with the objectives of phishing detection in this work. Based on these selection criteria, we retained the following datasets for our study:

1. **SpamAssassin** [18]: The SpamAssassin corpus is a publicly available collection of emails used for spam detection research. It includes both spam and legitimate messages, collected from real users and mailing lists, and organized into subsets like Easy Ham, Hard Ham, and Spam. Despite its original focus on spam, this dataset is included in our study because it has frequently been used in phishing detection research, making it a well-established reference in this domain [14].

2. **Enron** [19]: This dataset was compiled by the Cognitive Assistant that Learns and Organizes (CALO) Project and includes emails from approximately 150 users, primarily Enron senior executives. The messages are structured into folders and represent a total of around 500,000 legitimate emails. The dataset was initially released to the public by the Federal Energy Regulatory Commission as part of its investigation and made accessible online.

3. **Nazario** [20]: Hosts a comprehensive collection of phishing email datasets curated by Jose Nazario.

4. **Nigerian** [21]: The Nigerian dataset collects the fraudulent emails known as the Nigerian Letter or the "419" Fraud. This dataset consists of phishing emails designed to trick a person into providing financial information or transferring money to the attacker. Unsuspecting victims may fall for this social engineering trick by transferring a certain amount of money or bank account numbers in the belief that they will receive a much larger sum of money.

5. **Personal Advertising Emails**: We extracted personal emails where all components are exploitable. These emails are legitimate advertising messages, but

when taken out of context, they could be mistaken for phishing by a human. An example of advertising email is illustrated in Figure 2.6. This makes them not only particularly valuable for testing and training our model on more ambiguous cases, but also quite important to add more recent email samples to our datasets.

```
Subject: Free 2025 Digital Planner is here!
From: mydailyplanners <hello@mydailyplanners.com>

Your Free 2025 Digital Planner is here!
50% off Digital Planners, Limited time offer!
Get ready to transform your planning game in 2025!
Get the free digital planners and matching stickers set!
```

Figure 2.6: Illustrative example from the Personal Advertising Email dataset.

6. **Personal Emails**: We extracted legitimate personal emails that are suitable for analyzing email headers. They provide more diversified and up-to-date examples of email metadata.

7. **Ling** [22]: The Ling-Spam dataset consists of email messages, spams and hams, collected from academic mailing lists related to linguistics. These messages cover topics such as job postings, research opportunities, and software discussions. The dataset was introduced by Sakkis et al. [23] in their work on memory-based anti-spam filtering for mailing lists. The Ling dataset was chosen because it was adopted in previous phishing detection research [24].

8. **Ebubekirbbr** [25]: The Ebubekirbbr dataset was introduced by Sahingoz et al. [17], using two classes of URLs: phishing and legitimate. Phishing URLs were mainly collected from PhishTank (2018). Legitimate URLs were gathered using the Yandex Search API by submitting a list of predefined query terms and retrieving the top-ranked results.

Table 2.3 provides details about the age and composition of the datasets.

Database	**Period**	**Legitimate**	**Phishing**	**Composition**
SpamAssassin	2002–2005	4,153	1,899	Raw Email
Enron	1999–2002, v.2015	500,000	0	Raw Email
Nazario	2005–2024	0	2,985	Raw Email
Nigerian	1998–2007	0	3,978	Raw Email
Personal Advertising Emails	2023–2025	1,052	0	Raw Email
Personal Emails	2023–2025	1,692	0	Headers
Ling	2003	2,412	481	Body
Ebubekirbbr	2019	36,400	37,175	URLs

Table 2.3: Composition of the selected datasets.

Limitation Despite our efforts to carefully select reliable and well-documented datasets, we encountered a limitation related to the age of most datasets. Many of them are outdated and do not accurately reflect the structure or content of modern emails, except for the recent additions of personal emails we collected ourselves.

2.4 Detection Methods

As phishing techniques continue to evolve, the methods developed to detect them have also progressed. Over time, various approaches have been proposed to identify phishing emails. To gain a comprehensive understanding of the current state of phishing email detection, it is essential to review both the detection techniques — based on various types of data — and the evaluation metrics used to assess their effectiveness. Figure 2.7 depicts an overview of phishing detection methods.

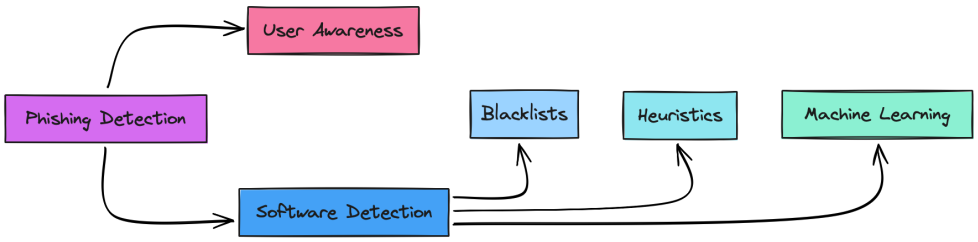


Figure 2.7: An overview of phishing detection approaches inspired by Khonji et al. [26].

2.4.1 User Awareness

User training approaches aim to educate end users about the nature of phishing attacks and to enhance their ability to differentiate between legitimate and malicious messages. However, while phishing is indeed a form of social engineering and user education is often seen as the first line of defense, research has shown that awareness alone is not enough. Rather than solely informing users, it may be more effective to regulate their behavior. Notably, even trained professionals and security experts have fallen victim to phishing attacks, highlighting that the growing complexity of systems may exceed human cognitive limits.

A more promising approach involves improving system usability, particularly through better user interfaces. For instance, modern browsers have evolved from passive to active security warnings, which block content and use clear visual cues to convey risk, recognizing that most users tend not to read traditional warning messages.

Given these limitations, relying exclusively on user behavior is not sufficient. This is why automated and intelligent software-based detection mechanisms are essential to provide scalable and reliable protection against phishing threats [26].

2.4.2 Software Detection

To complement user-focused strategies, software-based mitigation approaches aim to automatically classify phishing and legitimate messages on behalf of the user. These methods address the gap created by human error or limited awareness, which is especially important given that user training can be costly and impractical in some contexts. Phishing detection can rely on blacklists, heuristics or various machine learning algorithms. While infrastructure-based solutions exist and operate upstream — for example, by filtering or rewriting emails before they reach the user, such as Microsoft Safe Links [27] — this work focuses on software-based detection methods.

Blacklists One common defense against phishing relies on blacklists, which are databases that are continuously updated to store previously identified phishing URLs, IP addresses, or keywords. Most modern web browsers such as Google Chrome, Mozilla Firefox, and Microsoft Edge include blacklist-based protection by automatically checking accessed URLs against a database of known phishing websites. When a match is found, the browser either issues a warning or blocks access to the site entirely [28]. Prakash et al. [29] proposed a blacklist-based approach specifically designed to prevent phishers from bypassing traditional blacklist systems.

Despite their usefulness, blacklist-based methods offer limited protection against zero-hour phishing attacks—newly launched campaigns that have not yet been reported or added to the list. Since a malicious site must first be discovered and documented before being included, these early-stage attacks often manage to evade blacklist-based filters [26].

Heuristics While blacklist-based approaches depend on known phishing indicators, they fall short when facing zero-hour attacks. To overcome this limitation, heuristic-based detection offers an alternative by relying on manually defined rules or thresholds drawn from expert knowledge. These heuristics are designed to capture patterns frequently found in real-world phishing attempts, such as suspicious wording, unusual header configurations, or misleading link structures. For instance, the following rules can be applied to detect phishing behavior based on URL or email header analysis:

- **Rule 1:** If the host name in a URL is an IP address rather than a domain name, classify the email as phishing.
- **Rule 2:** If the received SMTP header does not contain the domain name of the organization, raise a phishing alert.
- **Rule 3:** If the email is formatted as HTML and a displayed URL uses Transport Layer Security (TLS) (e.g., `https`), but the actual `href` attribute points to a non-TLS link (e.g., `http`), flag it as phishing.

This capacity for generalization comes with a drawback: an increased risk of false positives, where legitimate messages are incorrectly flagged as phishing. In addition, heuristic rules require manual updates to remain effective against evolving attack techniques [13] [26]. To address these challenges, researchers have increasingly shifted towards machine learning, which provides a more adaptive and data-driven approach.

Machine Learning Methods While heuristic approaches rely on manually crafted rules, machine learning methods offer a more dynamic alternative capable of detecting zero-hour phishing attacks. Their key advantage lies in their adaptability to evolving phishing strategies. This flexibility can be achieved either through reinforcement learning or by periodically retraining models on updated datasets, allowing them to better capture new patterns and characteristics of emerging campaigns. As represented in Figure 2.8, these methods include supervised learning (relying on labeled data) and unsupervised learning (detecting suspicious behaviors or anomalies in unlabeled data).

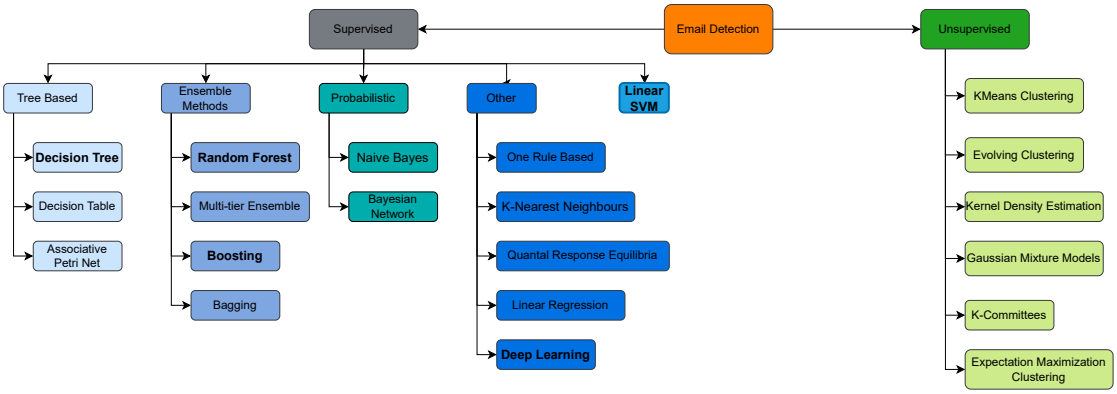


Figure 2.8: Algorithms used in phishing email detection adapted from Das et al. [13]. The most used algorithms are highlighted in bold.

To enable supervised models to distinguish between phishing and legitimate emails, features are extracted from various components of the message. According to Das et al. [13], these features can be grouped into the following main categories:

- **Email Headers:** Focus mainly on sender-related fields (e.g., `From`, `Sender`, `Reply-To`) and `Received` headers to trace email origin.
- **HTML Content:** Structural analysis of the HTML body using the Document Object Model (DOM), which reflects the organization and presentation of the email; for example, Lee et al. [14] tested various ML algorithms on 63 features extracted from email headers and HTML content.
- **Text Content:** Lexical analysis of phishing keywords (e.g., “Click here,” “Urgent”), increasingly replaced by deep learning approaches, discussed later.

- **URLs:** Semantic and lexical features (e.g., domain structure, use of obfuscation techniques, similarity to legitimate domains); for example, Sahingoz et al. [17] trained models on such features to classify URLs.

Common machine learning algorithms include decision trees, Support Vector Machines (SVM), Random Forests, Gradient Boosting, and Deep Learning (DL) models such as neural networks. Among these, Random Forest and XGBoost are widely used in phishing detection. **Random Forest** uses bagging, combining multiple decision trees trained on random data subsets and aggregating their predictions to reduce overfitting and improve generalization. **XGBoost** applies boosting, building trees sequentially to correct prior errors, with built-in regularization, efficient handling of missing values, and strong scalability, making it highly effective in real-world phishing detection tasks [30]. While these methods perform well, they depend heavily on feature engineering and high-quality datasets. A key limitation is reduced generalization to unseen patterns: when email features or structures (e.g., headers, phrasing, URLs) evolve, model performance can decline [31].

To address the limitations of traditional machine learning approaches, various **deep learning-based methods**—particularly Convolutional Neural Networks (CNNs) and Large Language Models (LLMs)—have been developed in recent years. These models have proven highly effective for phishing detection, especially in tasks involving URL analysis and text classification.

Convolutional Neural Networks (CNNs), initially designed for visual recognition tasks, are now widely applied to text classification due to their strong performance. Concerning URL detection, CNNs can analyze URLs at the character level by embedding each character and converting the URL into a fixed-size matrix, where convolution operations detect discriminative patterns. However, CNNs struggle with long-range dependencies, which sequential models like Long Short-Term Memory (LSTM) networks address by capturing contextual relationships over longer sequences. Hybrid CNN–LSTM architectures, combining both approaches, have shown superior performance, as demonstrated by Divakaran and Oest [28] and Lee et al. [14] for phishing URL detection.

Transformer-based architectures have become central to Natural Language Processing (NLP), with models like OpenAI GPT and Google BERT recognized as leading examples. These models are referred to as **Large Language Models (LLMs)** due to their ability to understand and generate humanlike texts. Unlike generative models like GPT, some LLMs focus on understanding and encoding context rather than producing text. Bidirectional Encoder Representations from Transformers (BERT) is one such model, pre-trained on large-scale text using a Masked Language Modeling (MLM) strategy and bidirectional self-attention to capture both left and right context, making it highly effective at grasping semantic nuances. Once pre-trained, BERT can be fine-tuned for specific tasks by adding a classification layer, achieving state-of-the-art results across NLP tasks such as question answering, inference, and text classification [32]. Due to its

ability to capture deep contextual meaning, BERT has also been successfully applied to phishing and spam detection, as demonstrated by Lee et al. [14] and Riftat et al. [33]. Variants like RoBERTa, ALBERT, and DistilBERT offer trade-offs between accuracy and computational efficiency [34].

Limitations of Detection Methods Today, ML and DL methods are more used than blacklist- and heuristic-based approaches to detect phishing emails, delivering significantly higher performance in phishing detection. However, they rely heavily on high-quality labeled datasets, which are scarce, often outdated, and difficult to supplement, particularly because private datasets are rarely available for academic research. Additionally, most models focus on a single email component (e.g., headers, text, or URLs), and few leverage multi-component analysis. Moreover, while these advanced models push performance forward, they also introduce a major challenge: their black-box nature makes them difficult to interpret and explain, raising concerns about transparency, trust, and robustness, as emphasized by Gaurav et al. [2]. Therefore, a robust approach should not only aim for high performance but also ensure interpretability, multi-component integration, and robustness against adversarial manipulation. As noted by Kavya et al. [35], incorporating explainable AI (XAI) is increasingly viewed as a key means to enhance the transparency and trustworthiness of phishing detection systems, especially as their complexity grows with deep learning.

2.4.3 Evaluation Metrics

Once the models for phishing detection have been introduced, the next step is to evaluate how well they perform. Evaluating the performance of phishing detection models requires metrics that go beyond simple accuracy. While accuracy is widely used in classification tasks due to its ease of interpretation, it can be misleading in imbalanced settings—such as phishing detection—where legitimate emails vastly outnumber phishing ones. In such contexts, more nuanced metrics are needed to assess how well the model identifies the minority class. To address this, we rely on a combination of precision, recall, F1-score, and the Area Under the Precision-Recall Curve (AUPRC). These metrics provide a more reliable and informative view of model performance, especially when the goal is to detect phishing emails without misclassifying legitimate ones [14] [36].

Confusion Matrix and Accuracy Many performance metrics are derived from the confusion matrix, which summarizes the predictions of the model, as detailed in Table 2.4.

	Positive Prediction	**Negative Prediction**
Positive Class	True Positive (TP)	False Negative (FN)
Negative Class	False Positive (FP)	True Negative (TN)

Table 2.4: Confusion matrix layout.

Based on this, *accuracy* is defined as:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Accuracy is easy to compute, but it is not always a reliable measure—especially on imbalanced datasets. In binary classification tasks where one class (e.g., legitimate emails) is much more frequent than the other (e.g., phishing emails), a model could predict only the majority class and still reach high accuracy. This would give the illusion of good performance, even though the model completely fails to detect the minority class, which is often the real objective. For this reason, accuracy alone should be interpreted with caution in such cases [37].

Precision, Recall and F1-Score To better assess performance in imbalanced scenarios, we turn to metrics that focus on the positive class:

- *Precision* measures the proportion of predicted phishing emails that are actually phishing:
$$Precision = \frac{TP}{TP + FP}$$
- *Recall* (or True Positive Rate) measures how many actual phishing emails are correctly detected:
$$Recall = \frac{TP}{TP + FN}$$
- *F1-score* is the harmonic mean of precision and recall:
$$F = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

The F1-score captures the balance between precision and recall by combining them into a single metric. It ranges from 0 to 1, with higher values indicating better overall performance. In the context of phishing detection, a high F1-score suggests that the model effectively identifies phishing emails while keeping false positives low, making it a particularly relevant evaluation metric [36].

Area Under the Precision-Recall Curve (AUPRC) The AUPRC represents the area under the precision-recall (PR) curve, which illustrates the trade-off between precision and recall at various decision thresholds. It is particularly useful in imbalanced classification problems, such as phishing detection, where identifying the minority class (phishing emails) is critical.

A perfect AUPRC is achieved when the model detects all phishing emails (high recall) without mistakenly labeling any legitimate emails as phishing (high precision). This is especially important in phishing detection, where a false positive could prevent a user

from receiving a legitimate email. As a result, a higher AUPRC score reflects better performance in identifying phishing emails accurately, making it a desirable evaluation metric. Figure 2.9 illustrates this by comparing a model with a low AUPRC to one with a high AUPRC.

One important characteristic of the PR curve is that it ignores true negatives, allowing the AUPRC to focus entirely on the performance of the positive class. This makes it well suited for situations where the negative class dominates the dataset [38].

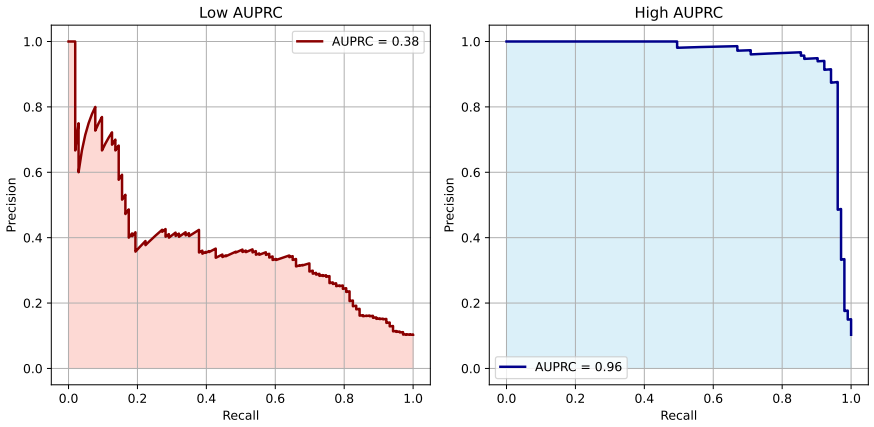


Figure 2.9: Hypothetical illustration of the Area Under the Precision-Recall Curve (AUPRC). Synthetic data was used to illustrate what such a curve might look like.

2.5 Explainable Artificial Intelligence (XAI)

After exploring the various machine learning models employed in phishing detection, it is essential to address a critical challenge that arises when using these tools: understanding how they make decisions. Many of these models, especially the more complex ones, function as black boxes. They deliver accurate predictions, but offer little transparency about their inner workings.

Explainability is the concept of extracting meaningful insights from a machine learning model. While achieving high performance is important, it is no longer sufficient for many real-world applications. There is a growing need to understand *why* a model made a specific decision. This desire for transparency is driven by several motivations: gaining insights about the data and the task itself, detecting and correcting biases in the model, building trust in automated systems, ensuring that decisions are safe (especially in high-stakes applications) and promoting social acceptance of AI systems [39].

Several interpretability approaches have been developed, as illustrated in Figure 2.10.

Broadly, these can be divided into two main categories:

- **Interpretability by design**, which refers to models that are inherently interpretable due to their simple and transparent structure, such as linear/logistic regression or decision trees.
- **Post-hoc interpretability**, which involves applying interpretability techniques after a complex model has been trained. This approach is typically used with black-box models. Within the post-hoc category, we can further distinguish between:
  - **Model-agnostic methods**, which do not rely on the internal workings of the model. Instead, they analyze how the output of the model changes in response to variations in the input features, focusing solely on input-output behavior. This category includes:
    - **Local methods**, which aim to explain individual predictions (e.g., LIME, SHAP).
    - **Global methods**, which provide insights into how features influence predictions on average.
  - **Model-specific methods**, which are tailored to a particular model architecture. These techniques aim to interpret specific components of the model in order to understand exactly what the model has learned [40].

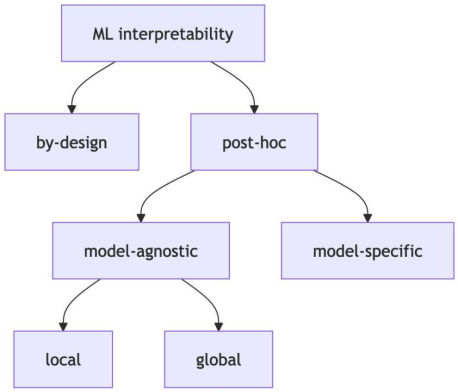


Figure 2.10: Overview of interpretability approaches from Molnar et al. [40].

In the following chapters, during our experiments, we adopted a post-hoc interpretability approach using local, model-agnostic methods, specifically LIME and SHAP, to explain our models. We will therefore provide a more detailed explanation of these methods here.

2.5.1 Local Interpretable Model-Agnostic Explanations (LIME)

The goal of LIME is to provide a method for explaining individual predictions (local) made by any classifier (model-agnostic) in a way that is both human interpretable and locally faithful. It achieves this by fitting a simple, interpretable model—such as sparse linear regression—around the prediction instance. LIME relies on an interpretable representation of the input, such as the presence or absence of words in a text, instead of more complex and less understandable features like word embeddings [41].

LIME has several strengths. It is model-agnostic, meaning it can be applied to any classifier. It supports various data types, including text, tabular data, and images. The features used in the explanations are interpretable—even if the original model operates on abstract features like embeddings.

One notable application of LIME beyond phishing detection is in the field of medical diagnosis. For instance, when predicting whether a patient has the flu based on clinical features such as sneezing, headache, or the absence of fatigue, LIME can identify which symptoms contributed most to the decision of the model. This allows doctors to interpret the reasoning of the model for individual cases, assess whether it aligns with their clinical judgment, and decide whether to trust the prediction [41].

Despite these advantages, LIME also has some limitations. It ignores feature correlations, treating all features as independent. The results can be unstable, meaning that explanations may vary with each run, and instances that are very close in input space can receive quite different explanations. Moreover, the complexity of the explanation model must be set manually by specifying the number of features to include [42].

2.5.2 Shapley Additive Explanations (SHAP)

SHAP is an explainability method grounded in cooperative game theory, designed to provide interpretable and consistent explanations for any model prediction. Introduced by Lundberg and Lee [43], SHAP unifies several existing interpretability methods, including LIME, into a single coherent framework. In their work, the authors argue that an effective explanation method should satisfy three key properties:

1. *Local accuracy*: the sum of feature attributions should match the model output for a given instance.

2. *Missingness*: features that are absent from the input (i.e., set to zero in the simplified representation) should receive zero attribution.

3. *Consistency*: if a model changes in a way that increases the contribution of a feature, the attribution for that feature should not decrease.

According to Lundberg and Lee, **Shapley values**, derived from cooperative game theory, are the only solution that satisfies all three properties.

Shapley values explain the prediction of a model by fairly distributing the prediction difference between the input features, based on their contributions. Each feature is seen as a player in a game, and the prediction is the outcome of their cooperation. The Shapley value of a feature is its average contribution to the prediction across all possible feature combinations [44].

It is worth noting that SHAP is not simply equal to Shapley values. It builds upon the original concept by introducing efficient estimation techniques, practical implementations, and theoretical connections to methods like LIME. It extends the use of Shapley values to a broader range of models, including those for text and image data, and introduces new ways to visualize and aggregate explanations. As a result, SHAP made Shapley-based explanations more practical and popular, serving as a modern and extended approach to model interpretability.

A concrete example of SHAP in practice comes from the financial sector, where it was used to interpret a regression model predicting future returns of Goldman Sachs stock. SHAP identified the daily low and opening prices as key factors influencing the predictions of the model. The resulting summary plot provided clear insights into feature impact, enhancing transparency—a critical requirement in finance for compliance, risk assessment, and decision support [45].

SHAP values estimate each feature contribution to a model prediction by averaging its impact across all possible combinations of features. Since this computation becomes highly complex with many features, SHAP relies on approximation methods. These include model-agnostic approaches like Kernel SHAP, and model-specific variants such as Linear SHAP, Deep SHAP, and Tree SHAP, the latter being particularly efficient for tree-based models. While SHAP supports both local and global interpretability and offers rich visualization tools, it has limitations. It assumes feature independence, which may not hold in domains with strong feature interactions, and its attributions can be misinterpreted if not carefully analyzed [46].

2.5.3 The Double-Edged Nature of Explainability: XAI as a Vector for Adversarial Attacks

While XAI methods like LIME and SHAP improve model transparency and trust, they can also expose vulnerabilities. By revealing which features drive predictions, these tools can help attackers craft adversarial examples that exploit model weaknesses. Local explanations make such attacks more precise by highlighting sensitive features. This risk has been demonstrated by Kozik et al. [3], who showed that explainability can be exploited to conduct successful adversarial attacks in the context of fake news detection.

2.6 Deception Techniques and Adversarial Attacks

With the previous sections having examined existing phishing detection models and the role of explainability in understanding them, the next step is to explore how these models can be challenged. Adversarial attacks in the context of phishing email detection refer to deliberately crafted inputs (emails) that contain small, strategic modifications designed to fool a machine learning model into making incorrect predictions. These perturbations are minimal enough to preserve the meaning or readability of the email, yet sufficient to alter the outcome of the classification model [47]. Since an email is composed of multiple components, different types of adversarial attacks can be applied to each part individually.

2.6.1 Adversarial Attacks on Email Headers

Email headers, particularly those related to sender identity, are common targets in adversarial phishing strategies. These manipulations are designed to either deceive the user interface or to bypass email authentication mechanisms (e.g., SPF, DKIM, DMARC). We distinguish between two main categories of header-based attacks: sender-based deception and authentication bypass.

Sender-Based Deception Attacks By manipulating how sender information is displayed to the recipient, attackers aim to trick users into trusting a forged identity. This can involve forging the `From` email address (email spoofing), altering the display name to mimic a trusted entity (name spoofing), or using lookalike domains (mangled domains) to deceive the recipient [48].

Authentication Bypass Attacks Instead of directly targeting the user, these attacks exploit weaknesses or misconfigurations in email authentication protocols. Two representative examples are authentication result injections, where attackers falsify SPF, DKIM, or DMARC outputs to make unauthenticated messages appear legitimate, and replay attacks, where validly signed emails are reused with malicious modifications to bypass authentication [49].

2.6.2 Adversarial Attacks on Text

Adversarial attacks in Natural Language Processing (NLP) can be applied to either *white-box models* or *black-box models*. The former are models for which everything is known: the hyperparameters, the training dataset, and so on. In contrast, black-box models are those for which only the prediction on the input data is accessible. Attacks on the latter are considered more realistic.

NLP Adversarial Attacks Many adversarial attacks exist in Natural Language Processing (NLP). A notable framework, *TextAttack* [50], published in 2020, unifies 16 widely used attacks and provides a standardized platform to generate and analyze them. It

supports reproducibility and facilitates research in this area. Among the included attacks is *TextFooler* [51], which was used to test the robustness of phishing email detection models [47]. TextFooler ranks words by their influence on model predictions and replaces the most impactful with semantically similar synonyms, using cosine similarity and the Universal Sentence Encoder to preserve meaning and context.

Imperceptible Adversarial Attacks Another type of adversarial attack is the imperceptible adversarial attack, which uses visually unnoticeable encoding tricks to fool NLP models in a black-box setting. These include invisible characters that disrupt tokenization without altering appearance, and homoglyphs that replace letters with similar-looking ones from other scripts. Even minimal use of such perturbations can break commercial NLP systems [52].

XAI-Guided Adversarial Attacks Another line of research explores other techniques for performing adversarial attacks on text, such as using XAI methods to guide adversarial attacks on a fake news detection model [3]. In this approach, the SHAP explainer is used to identify the most influential words, which are then replaced in a way that alters the model prediction without changing the semantic meaning. *However, to the best of our knowledge, this technique has not yet been applied to a phishing detection model.*

2.6.3 Adversarial Attacks on URL

Several types of adversarial attacks can be crafted against URL-based phishing detection systems. These attacks aim to deceive models or users by manipulating the appearance, structure, or behavior of URLs. Below are four major categories of such adversarial strategies.

Hidden Malicious URL Attacks To obscure the true destination, attackers conceal malicious URLs within HTML elements. For example, attackers may use custom CSS or HTML attributes to display a fake tooltip showing a benign URL when the user hovers over a malicious link (fake tooltip), or present trusted visible link text (e.g., amazon.com) while the underlying hyperlink points to a malicious domain (link mismatch) [48].

Altered Malicious URL Attacks The URL is modified in appearance while maintaining the same final redirection target. One common technique is *Redirect*, where attackers use legitimate redirection services (e.g., `https://trusted.com/redirect?q=malicious.com`) to hide the final malicious destination, often bypassing detection by leveraging trusted intermediary domains. Another widely used method is the *URL Shortener*, which employs services like TinyURL or Bit.ly to obfuscate the final target, preventing users and detection systems from easily recognizing the malicious domain [48].

Behavior-Based URL Attacks Some phishing techniques exploit not just the URL structure but also its behavior over time or in response to user actions, evading detection

through dynamic changes or deceptive interactions. For example, Ropemaker uses externally hosted CSS to control the link destination after email delivery: the first click may lead to a safe site, encouraging trust and sharing, while later clicks silently redirect to a malicious website [48].

URL Composition Attacks The attacker crafts a deceptive URL that appears more legitimate by embedding well-known brand names or trusted domains within subdomains or path segments. For example, they may register a domain with a different top-level domain (e.g., .net instead of .com), place a trusted brand name in the subdomain (e.g., paypal.login.attacker.com), or use domain extensions and structures that mimic trusted domains (e.g., secure-paypal.com or bank-login.net). However, the actual destination differs entirely, often pointing to a malicious domain. This technique is used to construct misleading URLs from scratch, rather than modifying existing malicious ones [48].

2.6.4 Final Remarks

This section has highlighted the diversity and sophistication of adversarial and deception techniques used to hijack phishing detection systems. Whether through subtle manipulation of email headers, modifications to text, or URL obfuscation strategies, attackers leverage a wide range of vectors to bypass both human and machine-based filters.

Importantly, these attacks can combine multiple strategies across different components of an email—such as the headers, text, and URLs—demonstrating that each element contains potential vulnerabilities. *This reinforces the need for detection models that analyze and secure the email as a whole, rather than relying on isolated features.*

Chapter 3

Detection Model

As a starting point, our approach builds upon an existing model from the literature. Specifically, Lee et al. [14] introduced D-Fence, a multi-modular phishing detection system consisting of structure, text, and URL components—each focusing on a different part of the email for independent analysis. By combining insights from these separate views, D-Fence aims to cover a broader attack surface and improve detection accuracy. As an initial step in our own research, we replicated their implementation and subsequently adapted it to align with our specific objectives, particularly the integration of explainability into the detection pipeline. However, some of the models used in the original modules were not the most suitable for applying explainability techniques; in practice, more interpretable results were obtained when we replaced them with alternative models. In the following sections, we first describe the dataset used to train and evaluate the model. We then present the original architecture proposed in the paper, followed by a detailed account of our own implementation of the three modules and the meta-classifier. We conclude this chapter with a discussion of the limitations of the model.

3.1 Dataset Composition

To train and evaluate our three individual modules, the meta-classifier, and the complete end-to-end detection system, we used a combination of public and private datasets, as previously described in the database analysis section (see Section 2.3).

As a **common foundation** across all three modules, we relied on five datasets: the public *Enron*, *SpamAssassin*, *Nazario*, and *Nigerian* corpora, along with the private *Personal Advertising Emails* dataset, which was added to introduce more recent and realistic email samples.

To promote fair training and avoid model bias, the training set was balanced, containing equal numbers of phishing and legitimate emails [53]. In contrast, the test set was intentionally left unbalanced to better reflect real-world distributions and assess model robustness in practical deployment scenarios.

The final base dataset was partitioned using a 60:20:20 split. Specifically:

- $\boldsymbol{D_{tr}}$: 60% used to train the three individual modules (structure, text, and URL),
- $\boldsymbol{D_{mt}}$: 20% used to train the meta-classifier that combines the module outputs,
- $\boldsymbol{D_{ts}}$: 20% held out as the test set to evaluate the performance of the full end-to-end system.

In summary, 80% of the dataset is allocated for training—comprising both the individual modules and the meta-classifier—while the remaining 20% is reserved for testing, which aligns with common practices in machine learning.

To ensure a balanced representation of the various data sources, we carefully distributed emails from the five datasets (SpamAssassin, Enron, Nigerian, Nazario, and Personal Advertising Emails) across the three main subsets: the primary training set D_{tr}, the meta-classifier training set D_{mt}, and the final test set D_{ts}. Each training split maintains a balance between legitimate and phishing emails while preserving a relatively consistent proportion of each source. A detailed breakdown of the dataset distribution is provided in Table 3.1.

D_{tr} : Train Set (60%)

Category	**Count**
Total	11,724
Legitimate	5,862
Phishing	5,862
Source Breakdown	
SpamAssassin	3,528
Enron(L)	3,020
Nigerian(P)	2,642
Nazario(P)	1,955
PersoAdv(L)	579
SA (L/P)	2,263/1,265

D_{mt} : Train Set (20%)

Category	**Count**
Total	4,000
Legitimate	2,000
Phishing	2,000
Source Breakdown	
SpamAssassin	1,204
Enron(L)	1,045
Nigerian(P)	870
Nazario(P)	694
PersoAdv(L)	187
SA (L/P)	768/436

D_{ts} : Test Set (20%)

Category	**Count**
Total	4,000
Legitimate	3,000
Phishing	1,000
Source Breakdown	
SpamAssassin	1,320
Enron(L)	1,592
Nigerian(P)	466
Nazario(P)	336
PersoAdv(L)	286
SA (L/P)	1,122/198

Table 3.1: Email distribution across the three datasets. SA = SpamAssassin (Legitimate / Phishing), P = phishing, L = legitimate. PersoAdv = Personal Advertising Emails dataset.

Alongside D_{tr}, we incorporated **module-specific datasets** to enrich each component with more targeted examples:

- $\boldsymbol{D_{tr\_s}}$: Final training dataset for the structure module. The *Personal Emails* dataset was added, contributing 1,052 legitimate emails. To preserve class balance, an equal number of legitimate emails were removed from the Enron corpus. This addition brings more recent and diverse header formats.

- $\boldsymbol{D_{tr\_t}}$: Final training dataset for the text module. The *Ling Spam* dataset was integrated to diversify the content of email bodies. Specifically, 481 legitimate and 481 phishing emails were included to maintain class balance while enriching textual variation.
- $\boldsymbol{D_{tr\_u}}$: Final training dataset for the URL module. The training set was expanded with the *Ebubekirbbr* URL dataset, which contains 36,400 legitimate and 37,175 phishing URLs. This augmentation introduces a wider variety of both benign and malicious links, improving the ability of the model to detect suspicious URLs.

3.2 Overview of D-Fence

D-Fence is composed of three independent analysis modules: the structure module, the text module, and the URL module. Each of these modules is trained on a dedicated portion of the dataset, referred to as D_{tr} in Figure 3.1. Once trained, their outputs—specifically, the probability scores assigned to each email—are passed to a meta-classifier. The meta-classifier is trained separately using a second, non-overlapping subset of the training data, denoted D_{mt}. It uses the three probability outputs from the individual modules as input features to produce a final prediction about whether an email is phishing or legitimate. A visual representation of the D-Fence architecture is provided in Figure 3.1.

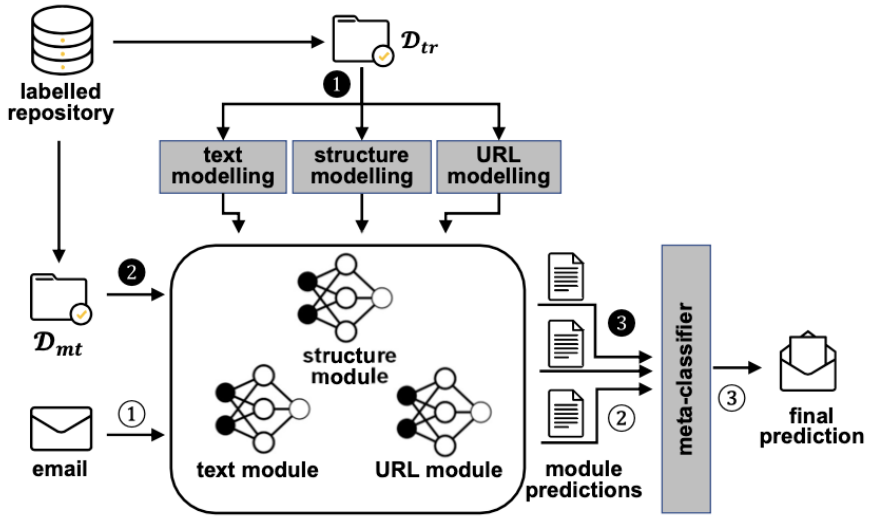


Figure 3.1: Overall architecture of D-Fence from Lee et al. [14].

In our work, we retain the overall structure and operating principle of D-Fence. The modifications were made concerning the internal design of individual modules, which we adapted as needed to reduce overfitting or to obtain a model better suited for applying explainable AI techniques. The code used to train and evaluate our models is available in the associated repository listed in Appendix A.

3.2.1 Structure Module

We first introduce the original structure module of D-Fence, specify the modifications applied along with their purpose, and subsequently present the performance of the model.

Original Version of D-Fence The structure analysis module in the D-Fence model focuses on analyzing the email headers and HTML content structure to identify phishing characteristics. This analysis includes features such as the number of hyperlinks, unique domain names in HTML assets, DOM tree structure, and email header patterns. The 63 features and their descriptions are detailed in Appendix B.1.1. The structural features are extracted using a single-pass scan of the email, including the text/plain sections and URLs. These structural features are then used as input for the classification model. Additionally, the extracted text/plain content and URLs will be used as input for the two other classification modules.

Lee et al. [14] evaluated multiple machine learning algorithms, including Random Forest and XGBoost, for phishing email detection within their D-Fence framework. Both classifiers were trained using their default hyperparameter. The models were compared in terms of training time, inference time, and classification accuracy. Based on this evaluation, the authors selected Random Forest as the final classifier due to its slightly better overall performance.

Our Structure Classification Module Like in D-Fence, this module performs the initial read of the email to extract the 63 structural features from each input email, along with the text and URLs, which will serve as input data for the text module and the URL module.

For the classification model, we opted for the XGBoost classifier for the structure model instead of the Random Forest classifier used in D-Fence. The choice of XGBoost was made because it is significantly faster compared to the Random Forest classifier, particularly when generating SHAP-based graphs for experimental analysis. Our XGBoost model adopts the default configuration used in D-Fence, with a few key adjustments aimed at mitigating overfitting — a point we revisit in Section 3.3. Initial results motivated slight hyperparameter tuning to promote better generalization [54]:

- max_depth = 5 — The default is 6. We reduced it to 5 to limit model complexity and reduce the risk of overfitting.
- learning_rate = 0.1 — Lowered from the default value of 0.3 to slow down the learning process and improve generalization.
- eval_metric = "logloss" — A standard evaluation metric for binary classification problems.

Our model is trained on structural features extracted from the D_{tr_s} dataset.

Model Evaluation The model is then tested on a dataset composed of the D_{mt} and the D_{ts}, totaling 8,000 structures. The results are very promising, as we achieved a high AUPRC of **0.9969**, which is the most relevant metric for our unbalanced test set. Such a high level of performance warrants further investigation using explainability techniques, which are explored in the following chapter. The classification report with more details can be found in Table 3.2.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Benign (0)	0.98	0.99	0.99	5,000
Phishing (1)	0.98	0.97	0.98	3,000
Accuracy	0.98			8,000
AUPRC	0.9969			8,000

Table 3.2: Classification report using XGBoost for the structure module.

3.2.2 Text Module

In the text module, we first present the original version of D-Fence, then describe the modifications made to integrate explainable AI. We detail the fine-tuning process of our new BERT-based model and conclude with an evaluation of its performance.

Original Version of D-Fence In the original D-Fence architecture, the authors proposed using BERT to model the textual content of emails by extracting contextual embeddings from the email body as input features for a classifier. The pipeline begins with language detection to retain only English emails, followed by a preprocessing phase that removes non-informative elements such as URLs, email addresses, and special characters. The cleaned text is then tokenized using the BERT tokenizer, and contextual embeddings are generated using the pre-trained `bert-base` model, which includes 12 transformer layers and 768 hidden units. For each email, token-level embeddings are averaged to form a fixed-size vector representation, which is then passed to a classification model—typically Random Forest or XGBoost—to determine whether the email is phishing or legitimate.

Integrating Explainability into the Text Module One aspect not addressed in the original D-Fence implementation is the use of explainable AI to gain deeper insight into the inner workings of BERT, which typically functions as a black box. In the initial version of our text analysis module, BERT was used solely to generate embeddings, which were then passed to a classifier. However, this architecture prevents the direct use of SHAP, as the model receives vector representations rather than raw textual input. As a result, we cannot trace the decision of the model to individual words or tokens in the original text. To address this limitation, we revised our pipeline. Instead of using static BERT embeddings with an external classifier, we fine-tune a BERT model directly for the classification task. This integrated setup allows us to apply SHAP to the BERT model itself, enabling token-level interpretability and offering a more transparent view of how the model processes email text to distinguish phishing from legitimate messages [55].

Text Preprocessing To prepare the email texts for fine-tuning the BERT model and generating reliable SHAP explanations, we applied several preprocessing steps similar to those described in the original D-Fence paper. These included removing non-English emails, URLs, email addresses, numbers, and excess whitespace in order to reduce noise and focus on the core textual content. As in D-Fence, we used the `bert-base-uncased` model, which ignores case distinctions, making it well suited for the informal and inconsistent capitalization often found in phishing emails. However, unlike D-Fence, we chose to retain punctuation, as it can provide important semantic cues and is meaningfully handled by the BERT tokenizer [56].

Model Training - Fine-Tuning BERT To fine-tune BERT for phishing email classification, we developed a training pipeline using PyTorch and the Hugging Face Transformers library. The dataset D_{tr_t} is first preprocessed as described in the previous paragraph. The best hyperparameters are selected via grid search (discussed later). We then use the `bert-base-uncased` tokenizer to convert the cleaned email texts into token IDs compatible with BERT. A custom PyTorch Dataset class handles dynamic tokenization and returns dictionaries containing input IDs, attention masks, and labels. These are passed to a DataLoader that manages batching and shuffling. The model is built using `BertForSequenceClassification`, which appends a binary classification head to the pre-trained encoder. Training is performed using the AdamW optimizer, combined with a linear learning rate scheduler with warm-up to stabilize the updates. For each batch, the model computes logits; the loss is calculated using cross-entropy, and gradients are backpropagated to update the parameters. The average loss per epoch is logged to track progress [57]. Once the training is complete, the final model and tokenizer are saved for later evaluation and interpretability analysis.

Hyperparameter Tuning - Grid Search To optimize the performance of the model, a grid search was conducted as a second step following model definition. The goal was to systematically explore different combinations of key hyperparameters and identify the configuration that yielded the best results. The search focused on three hyperparameters commonly tuned in BERT-based models [32]:

- Batch size: {16, 32}
- Learning rate: {2e-5, 3e-5, 5e-5}
- Number of training epochs: {2, 3, 4}

This resulted in a total of $2 \times 3 \times 3 = 18$ combinations, tested using nested `for`-loops in a grid-like fashion. Each model was trained on 80% of D_{tr_t} and evaluated on the remaining 20%, using the Area Under the Precision-Recall Curve (AUPRC) as the evaluation metric. Based on the results of this search, the best-performing configuration was identified as batch size = 16, learning rate = 3e-5, and number of epochs = 4. These hyperparameters were subsequently used for training the final model.

Model Evaluation Following hyperparameter tuning, the final model was evaluated on D_{mt} and D_{ts}. Key metrics computed include Accuracy, Precision, Recall, F1-score, and Area Under the Precision-Recall Curve (AUPRC). The model achieved an AUPRC score of **0.9968**, indicating excellent predictive performance. These high results naturally raise the question of why the model performs so well. To investigate this, we turn to explainable AI methods, which will be explored in the next chapter. A detailed classification report with the remaining metrics is presented in Table 3.3.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Benign (0)	0.99	0.99	0.99	4,794
Phishing (1)	0.97	0.99	0.98	2,090
Accuracy	0.99			6,884
AUPRC	0.9968			6,884

Table 3.3: Evaluation metrics of the best model for the text module.

3.2.3 URL Module

We first introduce the original structure module of D-Fence, detail the modifications made to enhance XAI usability, and then present the performance of the model.

Original Version of D-Fence The D-Fence system detects phishing URLs using a CNN-LSTM architecture. First, URLs are extracted from the text/plain and text/html sections of emails. Then, these URLs are encoded into numerical sequences, capturing lexical patterns indicative of phishing. The CNN component identifies local patterns, while the LSTM component captures long-term dependencies. MaxPooling layers reduce dimensionality and retain key features. If an email contains multiple URLs, the system classifies it as phishing based on the highest detected probability. However, we chose not to use this deep learning algorithm, as applying SHAP to it would yield explanations at the character level, which is not meaningful in the context of our experiments. As with the text module—where we adapted our model to ensure SHAP would operate on raw text rather than word embeddings—an explanation highlighting the index of a single impactful character offers little interpretative value. Instead, we opted for a model that relies on more interpretable features, better suited for explainability analysis.

Our URLs Classification Module Our classification model is therefore based on tabular features, allowing SHAP to rely on well-defined, clear features rather than on numerical indices.

1. **URLs Set Composition**: The URLs set for our URL model consists of the extracted URLs from D_{tr}, augmented with the *Ebubekirbbr* dataset, resulting in D_{tr_u}. Duplicate entries were removed to ensure data quality.

2. **Features Extraction**: We extracted 40 features for each URL using the techniques and features described by Sahingoz et al. [17] in their study on machine learning-based phishing detection. This set of features constitutes the training data for our model. The features and their description are summarized in Appendix B.1.2.

3. **Classification Model**: XGBoost is chosen once again for its high performance and compatibility with SHAP. To optimize the performance of the model, a grid search was conducted. The search focused on the following hyperparameters: n_estimators (100, 200, 300), max_depth (3, 5, 7), learning_rate (0.001, 0.01, 0.1, 0.2), subsample (0.6, 0.8, 1.0), colsample_bytree (0.8, 1.0), and eval_metric (logloss). The best-performing configuration combined 300 estimators, a maximum depth of 7, a learning rate of 0.2, a subsample of 1.0, and 0.8 column sampling, evaluated with logloss.

4. **Aggregating Results**: Similar to the D-Fence model, when an email contains multiple URLs, the system selects the highest probability among all the URLs to determine the classification of the email.

Model Evaluation The model is then tested on the URLs extracted from the training set for the meta-classifier D_{mt} and the test set D_{ts}. This model produces good results, as the AUPRC reaches **0.9399**. Although this result is slightly lower than the others, it remains strong and motivates further investigation through XAI, which we will explore in the following chapter. Detailed results are provided in Table 3.4.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Benign (0)	0.97	0.97	0.97	12,427
Phishing (1)	0.87	0.87	0.87	3,201
Accuracy	0.95			15,628
AUPRC	0.9399			15,628

Table 3.4: Classification report for the URLs module.

3.2.4 Meta-Classifier

Our meta-classifier follows the same approach as described in D-Fence. It takes as input the output probabilities (ranging from 0.0 to 1.0) generated by the three independent analysis modules—structure, text, and URL—and produces a final classification decision. In some cases, a module may not return a prediction (e.g., if an email contains no URL); such missing values are encoded as -1 to indicate the absence of information. These module-level probabilities form the feature vector for the meta-classifier. A supervised learning algorithm—specifically XGBoost, as used in the original study—is trained to combine these features and predict whether an email is phishing.

For training, we load the dedicated dataset reserved for the meta-classifier D_{mt}. Each sample is first passed through the three pre-trained modules to extract their respective output probabilities. These are then used as input features for the meta-classifier. Missing values are handled appropriately before training. The XGBoost model is then trained on this processed data. The model is then saved for future use in evaluation or interpretability analysis.

We then evaluated the model on the test set D_{ts} by passing it through the three individual modules and using their output probabilities as input to the meta-classifier. The final performance is summarized below, with an AUPRC of **0.9856**. As shown in Table 3.5, the model achieves excellent overall performance, with high precision, recall, and F1-scores across both classes.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Benign (0)	1.00	1.00	1.00	3,000
Phishing (1)	0.99	0.99	0.99	1,000
Accuracy	1.00			4,000
AUPRC	0.9856			4,000

Table 3.5: Evaluation metrics for the final model with the meta-classifier.

The results are encouraging, but the almost-perfect performances suggest the possibility of overfitting. This calls for a closer examination of the behavior of the meta-classifier, which will be addressed in the following section.

3.3 Limitations

The meta-classifier performs well, yet its perfect accuracy (1.00) suggests possible overfitting, likely caused by the structure module—highlighted by the SHAP analysis (Figure 3.2) as the most influential component. The bar plot shows that `structure_proba` contributes more to the predictions than `text_proba` and `url_proba`, indicating that the structure module may be the main source of this overfitting. This observation motivates a closer investigation of how overfitting emerges within the structure module in order to better understand and mitigate its impact on the overall model.

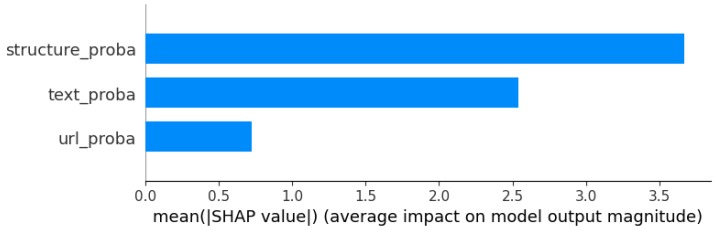


Figure 3.2: SHAP bar plot explainaing meta-classifier of D_{ts}.

3.3.1 Observations about Structure Module

Early in our experiments, we quickly recognized the importance of maximizing dataset diversity. Emails from the same source tend to share highly similar structures, which can lead the model to overfit—particularly when trained on a single legitimate and a single phishing dataset, as we initially did. SHAP value analyses confirmed this issue, revealing that the model was relying heavily on a few dataset-specific features.

To mitigate this, we progressively expanded the dataset to include a broader variety of sources. The final structure model was trained on a diverse combination of *Enron*, *SpamAssassin*, *Personal Advertising Emails*, *Personal Emails*, *Nigerian*, and *Nazario* datasets. This broader composition improved the ability of the model to generalize and reduce overfitting, as confirmed by the SHAP-based analyses. Detailed graphs and supporting metrics reinforcing these findings are provided in Appendix B.2.

Notably, the inclusion of more recent sources—such as *Advertising Personal Emails* and *Personal Emails*—led to a significant shift in feature importance, highlighting the limitations of older datasets. These datasets may no longer reflect the structure of real-world emails, raising concerns about their continued relevance and reliability. The effect of this shift is evident in how the dependence of the model on certain structural features changed after the dataset was augmented. For instance, the presence of dots in the domain part of the `Message-ID`, which was initially a strong indicator of phishing, had a reduced impact after the dataset was expanded. Likewise, the influence of the `Received` header count decreased with the introduction of more diverse data. These changes are clearly illustrated in the SHAP analysis graphs presented in Appendix B.2, which show how the influence of these features declined as the dataset became more representative of real-world email structures.

To summarize, incorporating a diverse mix of datasets proved essential to prevent overfitting in the structure module, as models trained on limited sources tended to learn dataset-specific patterns. Despite our efforts to increase dataset diversity, the overfitting observed in the meta-classifier suggests that the current dataset diversity remains insufficient. This highlights the need for access to more recent and varied data sources in order to better capture the structural diversity of real-world emails and further improve generalization.

Chapter 4

XAI-Guided Adversarial Attacks

This chapter presents the full set of experiments conducted as part of this study. We evaluated all components of our system individually—the structure, text, and URL modules. Each of these elements was tested under adversarial conditions, using attacks specifically crafted with the help of explainability techniques (XAI). The goal was to assess their robustness when exposed to targeted manipulations. The following sections are dedicated to these evaluations. Each section includes one or more experiments, with the corresponding methodology introduced beforehand to provide context and clarity.

Importantly, we focused on attacks aiming to make phishing emails appear legitimate in the eyes of the model. This choice reflects a more realistic threat scenario: an attacker is generally not interested in generating random misclassifications, but in crafting phishing emails that successfully bypass detection systems while retaining their malicious intent - for instance, to trick the victim into clicking a malicious link or providing sensitive information.

To illustrate the components targeted in our experiments, Figure 4.1 presents an example email in .eml format. This pseudo-email has been simplified for clarity, with some headers and the HTML part intentionally removed. In the experimental sections, we focus on modifying specific parts of the email: the orange-framed sections cover the headers, where we modify the `Message-ID` (light orange) and add forged `Received` headers (dark orange), as detailed in Section 4.1. The blue-framed section represents the main body content, on which the experiments in Section 4.2 are performed. The green-framed section highlights the embedded URL, which is specifically analyzed and modified in Section 4.3. The code used to run these experiments, along with the raw result files, is available in the GitHub repository referenced in Appendix A.

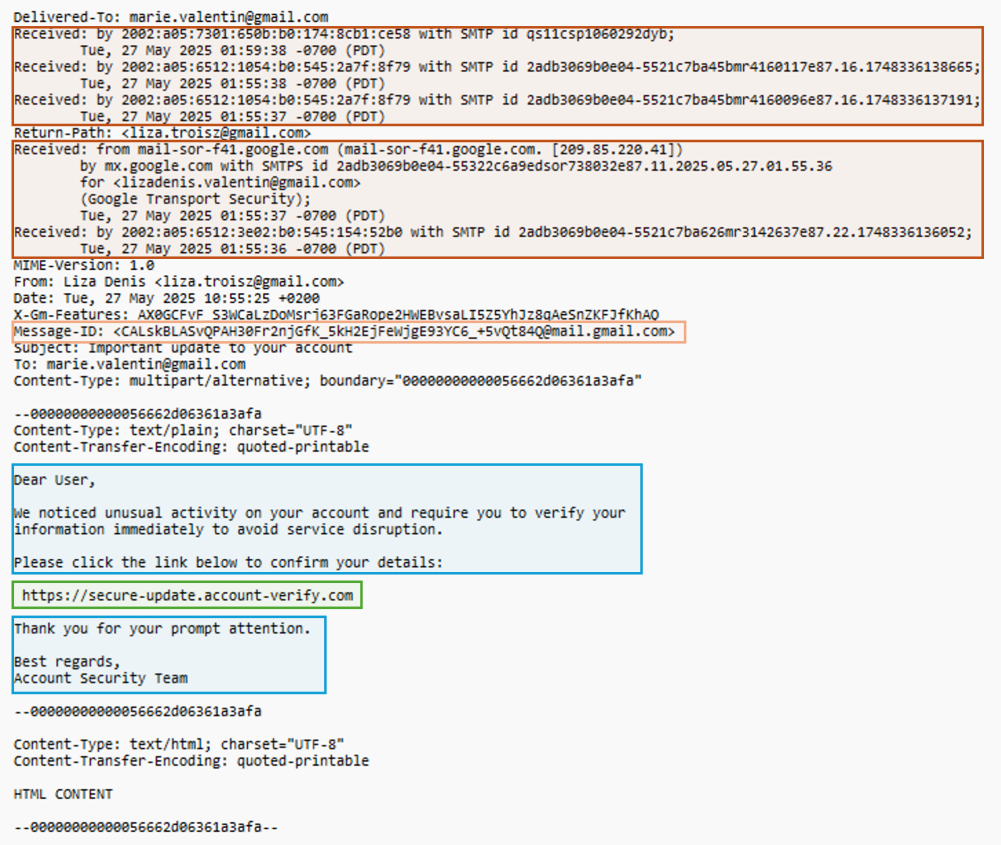


Figure 4.1: Example of a simplified .eml phishing email hypothetically constructed using ChatGPT and sent via Gmail. Orange frames indicate the header components (light orange: Message-ID; dark orange: Received headers); the blue frame marks the message body; the green frame highlights the embedded URL.

4.1 Experiments on the Structure Module

In this section, we present the experiments specifically targeting the robustness of the structure-based classification module. Relying on SHAP-based analyses, we identify the most impactful features and apply targeted adversarial manipulations to evaluate their influence on the prediction of the model. Beyond measuring statistical effects, we also discuss the practical feasibility of such attacks in real-world scenarios.

4.1.1 Methodology Overview

The structure-based classification model is first analyzed to understand the influence of each feature in the decision process. This analysis is based on SHAP-generated values and visualizations. Next, we randomly extract 100 structures from the D_{ts} dataset that are both correctly classified as phishing and are genuinely phishing. We apply targeted

adversarial modifications to these components, then re-evaluate them using the model.

We focus on manipulating the most impactful features to induce the misclassification of phishing emails as legitimate. Our analysis highlights two key features: the number of `Received` headers and the number of dots in the domain part of the `Message-ID`. These features are targeted in two dedicated experiments to evaluate their influence on classification outcomes. Finally, we assess the real-world feasibility of such manipulations by examining how an attacker could apply them in practice.

4.1.2 Analysis of Structure Module with SHAP

A SHAP explainer, generated from the model and its training data, is used to compute SHAP values for each feature in the test set. These values quantify the contribution of individual features to the predictions of the model:

- High negative SHAP values indicate that the feature contributes to a legitimate classification.
- High positive SHAP values suggest that the feature pushes the prediction toward phishing.

Two types of visualizations are used in the analysis: beeswarm plots and bar plots. These graphs provide insights into the distribution and magnitude of feature importance, helping to identify patterns that distinguish legitimate from phishing structures. Such features can be strategically manipulated to design adversarial attacks.

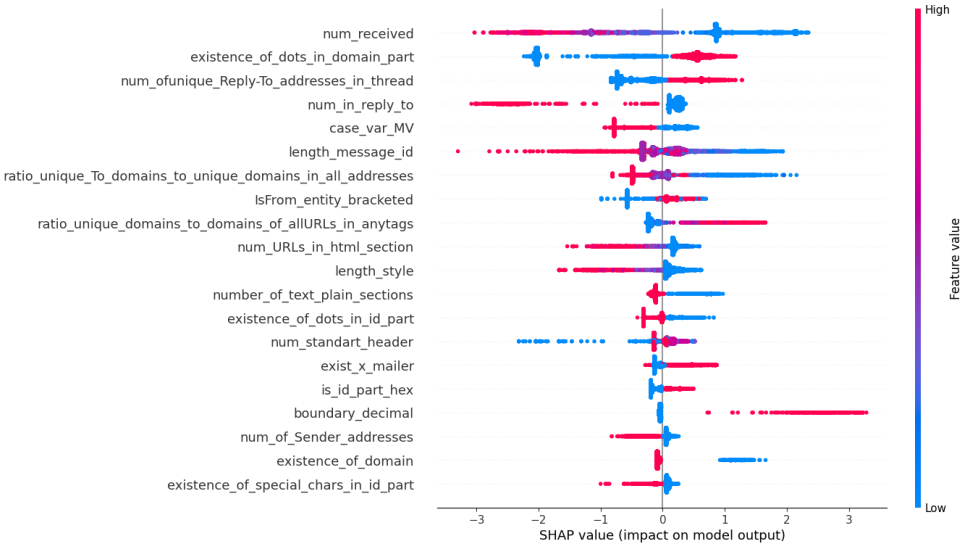
The beeswarm plot displays each structure as a dot, positioned horizontally according to its SHAP value for a specific feature. This SHAP value reflects how much the feature contributes to pushing the prediction toward either phishing or legitimate for that particular structure. The color of the dot represents the raw value of the feature (blue for low values, red for high values). Dots clustered on the left indicate that the feature tends to reduce the phishing probability, while those on the right indicate an increase. This visualization enables detailed observation of the distribution of feature effects and their interactions across the dataset.

The bar plot summarizes the mean SHAP value for each feature. The length of each bar represents the average importance of the feature, while the color indicates whether its effect tends to push predictions towards phishing (red) or legitimate (blue). This provides a global overview of the most influential features across all samples.

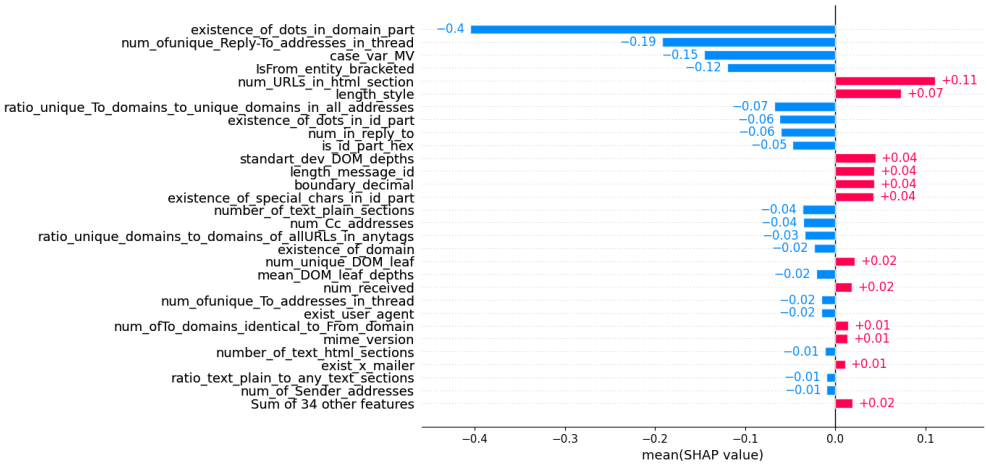
For example, the beeswarm plot (Figure 4.2a), displayed with the structures from the D_{ts} and D_{mt} datasets, reveals that legitimate structures (with negative SHAP values) are characterized by several recognizable features: a high number of `Received` headers, a low or nonexistent number of dots in the domain part of the `Message-ID`, a high number

of `Reply-To` headers, and a high variance in the `MIME-Version` header.

Similarly, the bar plot (Figure 4.2b) shows that features like `existence of dots in domain part`, `number of unique Reply-To addresses`, and `case variance in MIME Version` influence legitimate classification, while features like `number of URLs in the HTML section` and `style length` have a greater impact on phishing classification.



(a) Beeswarm plot for the XGBoost model for the structure classification of the D_{ts} and D_{mt} datasets.



(b) Bar plot for the XGBoost model for the structure classification of the D_{ts} and D_{mt} datasets.

Figure 4.2: Plots for the XGBoost model for the structure classification of the D_{ts} and D_{mt} datasets.

To conclude, since the number of dots in the domain part of the `Message-ID` and the number of `Received` headers appear to be among the most influential features, we will focus on manipulating them in the next two experiments in an attempt to cause the model to misclassify phishing emails as legitimate.

4.1.3 Message-ID Manipulation

One example of a possible adversarial attack involves manipulating the number of dots in the domain part of the `Message-ID`, which emerged as a strong discriminative feature: a low number of dots is often associated with legitimate emails, while a high number tends to correlate with phishing.

To carry out this attack, it is first necessary to understand how a `Message-ID` is structured. As shown in Figure 4.3, the `Message-ID` follows the format `<ID@Domain>`, where the domain usually corresponds to the Fully Qualified Domain Name (FQDN), often beginning with the local host name, followed by subdomains separated by dots to indicate domain hierarchy [58].

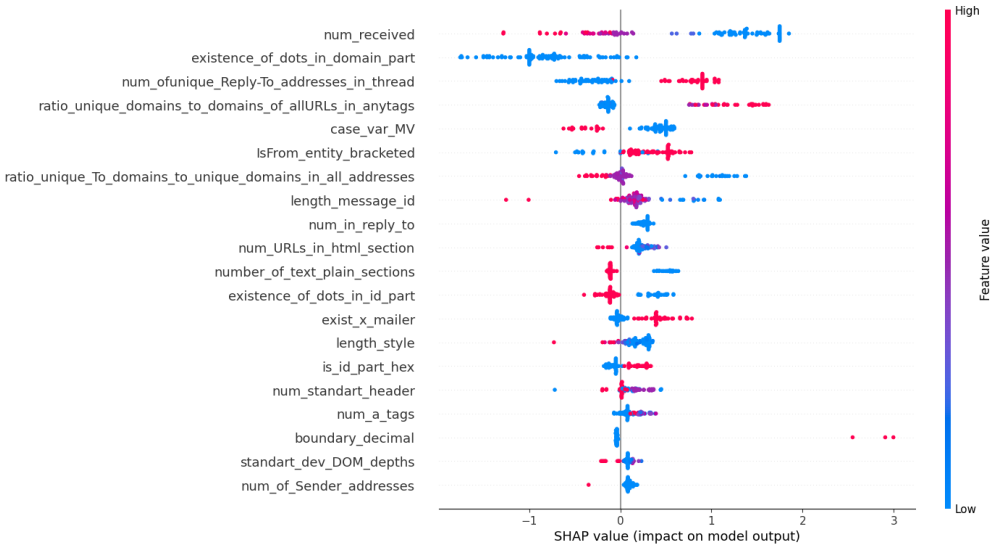
```
Message-ID: <1437947123938l0953a450162@mlsend.com>
```

Figure 4.3: Message-ID example.

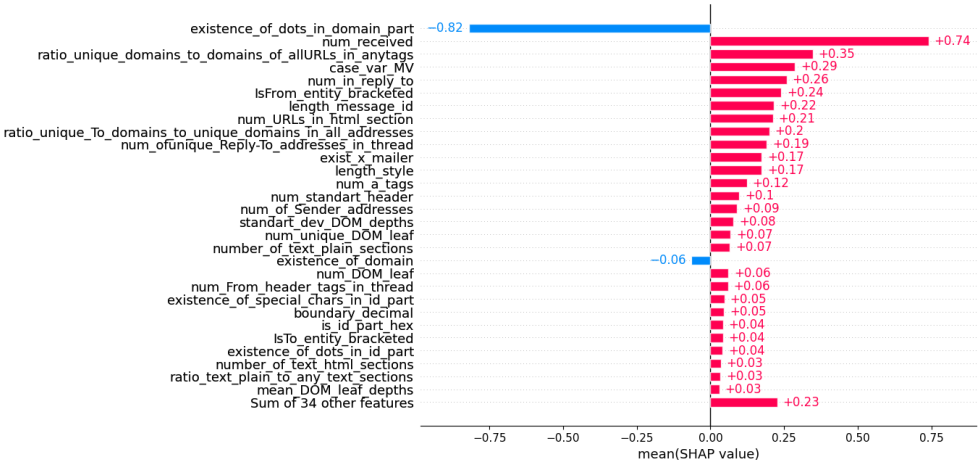
In this attack, we replaced the domain part of the `Message-ID` in the 100 randomly selected structures with a simplified domain containing no dots: `test-friendly`. The feasibility of this attack is discussed in Section 4.1.5.

Results and Analysis Only 6 of the modified structures were reclassified as legitimate by the model, while the remaining 94 were still identified as phishing. Among these, the average decrease in phishing probability was 0.0529. These results suggest that this modification alone has a limited impact on the prediction of the model but confirm that removing all dots from the domain part of the `Message-ID` aligns with a feature typically associated with legitimate emails.

Based on the analysis of the SHAP visualizations in Figure 4.4 , the next promising feature to manipulate appears to be the number of `Received` headers. As shown in the bar plot Figure 4.4b, this feature exhibits the highest mean SHAP value contributing to phishing classification, with an average impact of +0.74. In the corresponding beeswarm plot Figure 4.4a, low values for this feature are associated with phishing classifications, whereas higher values tend to contribute to legitimate ones.



(a) Beeswarm plot for the structure classification of the 100 structures with the modified ID domain.



(b) Bar plot for the structure classification of the 100 structures with the modified ID domain.

Figure 4.4: SHAP plots for the structure classification of the 100 structures with the modified ID domain.

4.1.4 Received Header Manipulation

As observed in the previous results, a low number of `Received` headers in the metadata is typically associated with phishing emails, while a high number is more indicative of legitimate ones. This feature is determined by the SMTP servers involved in the transmission of the email.

Based on this observation, we conducted an attack by artificially increasing the value of this feature across the 100 selected structures, in order to assess its influence on the classification decisions of the model.

Results and Analysis By maintaining a dotless domain in the `Message-ID` and modifying the number of `Received` headers to 4 for each structure, 6 additional structures were classified as legitimate, bringing the total to 12% of structure classified as legitimate. Among those still identified as phishing, the average decrease in phishing probability was 0.1345. By setting the number of `Received` headers to 7, a total of 34% of the structures were classified as legitimate, with an average probability drop of 0.1503 among those still identified as phishing. An overview of the top five SHAP values is provided to highlight the most influential features in the classification of two selected structures. This comparison offers insight into the outcome of the previous attack and helps explain why some structures remain classified as phishing, despite the modification of the two most impactful features.

Table 4.1 presents the structure with the **highest** predicted phishing probability. It is still classified as phishing due to a high `number of unique domains across all email addresses`, along with a low `ratio of unique "To" domains relative to all unique domains`. Although the number of `Received` headers has a negative SHAP value—indicating a contribution toward legitimate classification—this influence is outweighed by the strong positive contributions of other features. In contrast, Table 4.2 shows the SHAP values for the structure with the **lowest** phishing probability. Notably, the two most impactful features in this case are those previously manipulated: the number of `Received` headers and the absence of dots in the domain part of the `Message-ID`. The feasibility of these attacks is discussed in the next section.

Feature	**Value**	**SHAP Value**
Number of unique domains in all email addresses	10.0	1.1821
Ratio of unique To domains to the unique domains in all email addresses	0.1	0.9641
Number of Received headers	7.0	-0.9158
Length of Message-ID	31.0	0.8280
Number of unique Reply-To addresses in the mail thread	1.0	0.6173

Table 4.1: Structure with the **highest** phishing probability: 0.9969.

Feature	**Value**	**SHAP Value**
Number of Received header	7.0	-1.1809
Existence of dots in domain part of Message-ID	0.0	-0.8588
Number of unique Reply-To addresses in the mail thread	0.0	-0.5175
Number of In-Reply-To header	0.0	0.2595
Ratio of unique domains to total domains across all URLs	0.0	-0.2249

Table 4.2: Structure with the **lowest** phishing probability: 0.0454.

Detailed graphs about the final impact of the features related to this experiment on the 100 structures are presented in Appendix C.2.

4.1.5 Feasibility of those Attacks

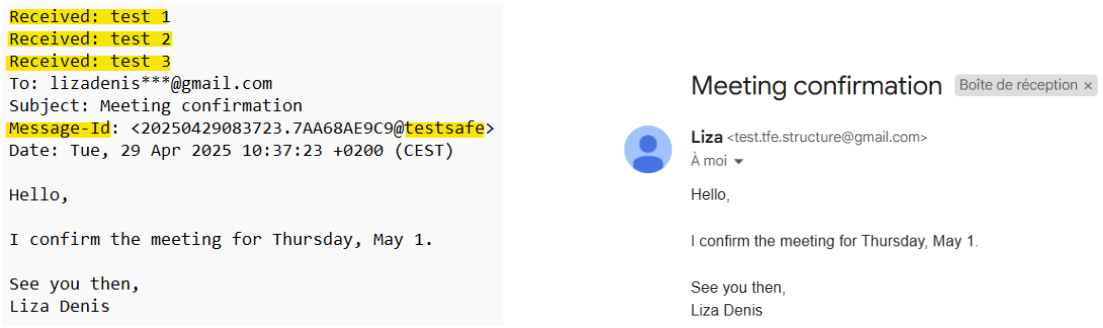
These attacks naturally raise the question of how malleable these features are in real-world scenarios. While modifying the structure of an email is straightforward when working with datasets that contain emails in their raw format, the real challenge lies in determining how much control a sender actually has over these elements during real email transmission.

While the email body is entirely under the control of the sender, the header adheres to stricter standards and is not fully manipulable. In particular, spoofing the `Message-ID` field requires advanced technical expertise, unlike other header fields, which are more easily forged. Only technically proficient attackers are capable of altering the `Message-ID` in a convincing way. Such spoofing involves significant effort and risk: the attacker would either need to remove the field—an action that would likely raise suspicion, or replicate a legitimate `Message-ID` from a previous email. In the latter case, this would also require disabling integrity checks within the mail client [58].

As for the `Received` headers, according to RFC 5321 [59] and SpamAssassin header forgery detection rules [60], each SMTP relay independently appends its own `Received` field. However, only the `Received` header fields inserted by trusted hosts are considered reliable. This allows inconsistencies in network paths, timestamps, or authentication records to be detected.

Interestingly, by using Postfix—an open-source Mail Transfer Agent (MTA)—we were able to craft an email in .eml format with forged `Received` headers and a custom `Message-ID`. The identifier part of the `Message-ID` was extracted from an existing legitimate email, while the domain part was customized to contain no dots, mimicking the structure observed in legitimate samples. This email (Figure 4.5a) was successfully delivered to a Gmail inbox (Figure 4.5b). The forged headers were interpreted as genuine by our model, suggesting that a total of seven `Received` headers may in fact be achievable in practice.

Our experiment confirms that manually injecting fake `Received` headers is feasible, although such headers may appear suspicious to a human user upon inspection. In contrast, generating legitimate-looking `Received` headers—by actually routing the email through multiple SMTP relays—would likely be more complex and less controllable for an attacker. Furthermore, we demonstrated that modifying the domain part of the `Message-ID` is also possible and can be leveraged to influence model predictions.



(a) Email in .eml format.

(b) Received email in the Gmail inbox.

Figure 4.5: Sending of an email in .eml format with Postfix.

4.1.6 Conclusion

This section demonstrated that structural features—such as `Received` headers and the format of the `Message-ID`—can be manipulated to reduce phishing detection scores, based on insights from SHAP analysis. However, unlike text-based and URL-based attacks, which will be explored in Section 4.2 and Section 4.3, these manipulations require more advanced technical skills and are less trivial to execute. These results highlight the importance to integrate structure-based detection within a broader phishing defense strategy.

4.2 Experiments on the Text Module

To evaluate the robustness of our phishing detection model against textual adversarial attacks, we conducted a series of experiments using automatic attacks, explainability-guided rewriting, and Large Language Models (LLMs). All experiments were performed on the same set of 10 phishing emails randomly selected from the D_{ts} dataset, covering common phishing themes such as fake system alerts, financial scams, and impersonation. Each email was assigned a unique identifier and remained unchanged across experiments to ensure comparability. A short description of the content of each email can be found in Appendix C.1.1.

Three types of adversarial modifications were used. First, TextFooler introduced synonym substitutions to mislead the model. Second, ChatGPT rewrote emails freely to make them seem more legitimate. Third, in the explainability-guided approach, LIME and SHAP were used to identify key features influencing model decisions. Based on these, ChatGPT rephrased or removed suspicious terms while keeping legitimate ones. SHAP was applied both locally and globally to capture feature importance.

ChatGPT (paid version) was used as the rewriting agent to simulate an attacker crafting adversarial emails. The prompts used are detailed in Appendix C.1.2. Each

transformation was evaluated along three criteria: the phishing probability predicted by the model, the fluency and naturalness of the rewritten text (assessed informally by the author), and feature importance analysis for explainability-based attacks.

These experiments provide insights into how various adversarial strategies, particularly those informed by explanation methods, can undermine the robustness of phishing detection systems.

4.2.1 Initial Robustness Test Against TextFooler and ChatGPT

To establish a baseline, we begin with simple adversarial attacks, either by applying the NLP attack TextFooler or by prompting ChatGPT to generate adversarial versions of the emails without providing specific guidelines.

Methodology To establish a baseline for model robustness, we evaluated two attack strategies on the fixed set of 10 phishing emails. First, we applied the TextFooler attack (see Section 2.6.2) from the TextAttack framework, which operates by replacing highly influential words with semantically similar synonyms. This method was selected for its word-level focus and compatibility with our implementation. Second, we used ChatGPT to freely rewrite the emails to make them appear more legitimate, without providing any specific instructions. The prompt given to ChatGPT can be found in Appendix C.1.2. Both approaches were then evaluated in terms of phishing probability and the perceived quality of the rewritten emails.

Results and Analysis The following two tables present the results, which serve as the basis for further analysis and discussion.

Email	Label	Original Proba	TextFooler Proba	LLM-Altered Proba
1	1	0.9998	0.4115	0.9998
2	1	0.9998	0.2932	0.9998
3	1	0.9997	0.2668	0.9997
4	1	0.9986	0.4963	0.9987
5	1	0.9998	0.4563	0.9998
6	1	0.9998	0.4509	0.9997
7	1	0.9998	0.4889	0.9998
8	1	0.9998	0.4026	0.9998
9	1	0.9907	0.0253	0.9998
10	1	0.9997	0.4921	0.9998
Mean	—	**0.9996**	**0.3784**	**0.9998**

Table 4.3: Results of the experiment showing the output of the model probability when given the adversarial inputs generated with TextFooler, and the output probability when given the LLM-altered inputs.

Attack Results	
Number of successful attacks	10
Number of failed attacks	0
Number of skipped attacks	0
Original accuracy	100.0%
Accuracy under attack	0.0%
Attack success rate	100.0%
Average perturbed word %	11.23%
Average num words per input	277.4
Avg num queries	1,161.1

Table 4.4: Summary of TextFooler attack results.

Comparison in Terms of Phishing Probability Table 4.3 shows that the TextFooler attack was highly effective, with a 100% success rate and an average drop in phishing probability from 0.9996 to 0.3784—a reduction of over 62%. Table 4.4 shows that, although the attack achieved a 100% success rate, it required a high number of queries and modified on average 11.23% of the words to deceive the model. This indicates that overcoming the initial robustness of the model demanded a meaningful amount of effort. In contrast, ChatGPT without specific instructions failed entirely, producing no adversarial input capable of fooling the model.

Comparison of Email Quality Before analyzing how the emails change through each transformation, we start with a brief look at the original messages. Some resemble realistic phishing attempts, like fake LinkedIn alerts or security warnings. Others follow older scam patterns, involving stories about African presidencies or wealthy benefactors—typical of so-called Nigerian scams. These differ notably from modern, targeted phishing tactics, raising concerns about how accurately public datasets reflect current threats. We will now examine how email quality evolves after transformation.

- **Adversarial Emails with TextFooler:** Emails generated with TextFooler tend to lack linguistic coherence and are not particularly convincing from a human perspective. Word substitutions frequently introduce contextual errors or disrupt the tone, sometimes making the text nonsensical or harder to read. An example is shown in Figure 4.6a:
  - *Dear* → *Pumpkin* — an overly affectionate term that is completely out of place in a formal phishing email.
  - *year* → *olds* — a grammatically incorrect substitution.
  - *Account* → *Compte* — a French word that disrupts the flow of an otherwise English email.

  This aligns with Chiang et al. [61], who show that attacks like TextFooler often produces adversarial samples that are semantically incoherent and grammatically flawed. While effective at deceiving models, these messages remain easily detectable

by humans due to their unnatural phrasing. Overall, the quality of such adversarial emails is low—they fool the model but would likely still raise suspicion for a human reader.

********** Blocked Incoming Messages **********

Dear , Incoming messages were blocked due to Email Account Verification Failure by you for the year. You have new messages in your email quarantine pending to be delivered into your Inbox. * * *Date:* // :: a.m. Click on Release, to free these messages to your inbox and to Verify your Mail Account : Deliver Messages *Quarantined email* Subject: Subject: Date: *Release* ( ) Re: Enquiry/Consultation RE: Send PI Asap for payment // :: a.m. *Release* ( ) Re: Balance Payment RE: T/T Payment Copy // :: a.m. *Release* ( ) Re: Account Statement RE: New Purchase order // :: a.m. *Deliver all messages ()* ( ) E-Mail Support NW Th Ave, BLDG , Suite , Miami , FL Unsubscribe ( ) – Unsubscribe Preferences ( )

********** Blocked Incoming Messages **********

Pumpkin , Incoming messages were blocked due to Email Account Verification Failure by you for the olds. You have new messages in your email quarantine pending to be delivered into your Inbox. * * *Date:* // :: a.m. Click on Release, to free these messages to your inbox and to Verify your Mail Compte : Deliver Messages *Quarantined email* Subject: Subject: Date: *Release* ( ) Re: Enquiry/Consultation RE: Send PI Asap for payment // :: a.m. *Release* ( ) Re: Balance Payment RE: T/T Payment Copy // :: a.m. *Release* ( ) Re: Account Statement RE: New Purchase order // :: a.m. *Deliver all messages ()* ( ) E-Mail Support NW Th Ave, BLDG , Suite , Miami , FL Unsubscribe ( ) – Unsubscribe Preferences ( )

(a) Adversarial output generated from Email 1 using TextFooler. Words in green indicate the original terms that were replaced by the red ones. Email 1 is shown in its preprocessed form to avoid modifying parts that are not used by the model during prediction.

Email Quarantine Notification

Dear jose@monkey.org,

We were unable to verify your email account for the year 2022. As a result, several incoming messages have been temporarily held in quarantine.

You currently have 3 pending messages that are awaiting delivery to your inbox.

Date: 5/4/2022 – 9:14:19 AM

To release these messages and complete the verification process, please click the link below:

👉 Release Messages to Inbox

——

Quarantined Emails:
Subject - Date - Action:
Re: Enquiry/Consultation – Send PI ASAP for payment
5/4/2022 – 9:14:19 AM
Release
Re: Balance Payment – T/T Payment Copy
5/4/2022 – 9:14:19 AM
Release
Re: Account Statement – New Purchase Order
5/4/2022 – 9:14:19 AM
Release

🔓 Deliver All Messages (3)

If you believe you've received this message in error, or have questions, please contact Email Support.

——

Email Support
2360 NW 66th Ave, BLDG 701, Suite 102
Miami, FL 33122

Unsubscribe | Manage Preferences

(b) Email 1 after simple reformulation by ChatGPT without any targeted adversarial intent.

Figure 4.6: Comparison of outputs for Email 1 using TextFooler and ChatGPT.

- **LLM-Generated Emails (ChatGPT):** The quality of these emails is significantly higher. A human reader would be much more likely to be deceived by emails generated by ChatGPT compared to those produced by TextFooler. ChatGPT improves grammar and spelling, adds a professional tone, and refines the overall layout and structure of the message. However, despite these improvements, the emails were still not sufficient to fool the model. An example of such an output can be found in Figure 4.6b.

Conclusion In summary, the adversarial attack with TextFooler was effective at misleading the model but produced emails of low quality from a human perspective. Conversely, while the large language model did not manage to fool the classifier, it generated realistic and coherent messages. This suggests that there is potential for continuing to explore the use of ChatGPT, but with additional guidance. Specifically, we plan to incorporate explainability results (XAI) to steer the generation process, rather than leaving the model to generate content without any targeted instructions.

4.2.2 LIME Guided Adversarial Rewriting via LLMs

In the previous experiment, ChatGPT made phishing emails appear more legitimate without guidance, but not enough to fool the model. We now examine whether explainable AI can improve its effectiveness.

Methodology We applied the LIME method (see Section 2.5.1) locally to each of the fixed set of 10 phishing emails to extract the 25 most influential features driving the classification of the model. The number 25 was chosen, as it corresponds to approximately 10% of the total word count across the emails (see Table 4.6), aligning with the previously observed average word perturbation rate of around 11% required to alter the prediction of the model. ChatGPT was then instructed to rewrite each email by removing or replacing words contributing to a phishing prediction, while preserving those associated with legitimate classification. This process was repeated over three rounds. The rewritten emails were evaluated based on their updated phishing probabilities and the overall quality of the generated content.

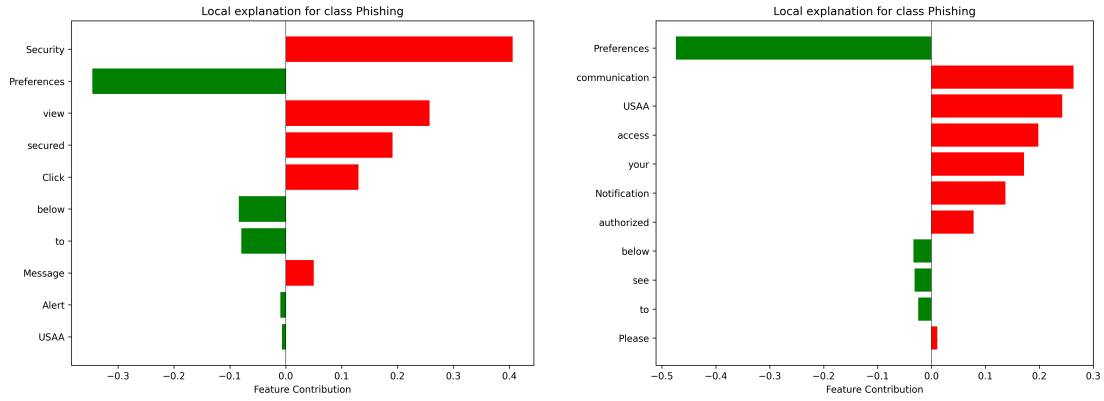
Results and Analysis To illustrate the transformation process, Email 9 was selected for closer examination. Due to its brevity, it allows for a more detailed step-by-step analysis. Presented below are the three versions of the email:

1. **Original**: *USAA Security Preferences Message Alert. Click below to view secured Message*
2. **After Round 1**: *USAA Preferences Notification. Please see below to access your authorized communication.*
3. **After Round 2**: *USAA Preferences Update. Kindly refer to the information provided further down for more details.*

Email	Label	Original Prob.	Round 1 Prob.	Round 2 Prob.	Round 3 Prob.
1	1	1.000	0.997	0.135	0.135
2	1	1.000	1.000	0.999	0.000
3	1	1.000	0.999	0.671	0.001
4	1	0.999	0.926	0.511	0.066
5	1	1.000	1.000	0.940	0.001
6	1	1.000	0.998	0.046	0.046
7	1	1.000	1.000	0.037	0.037
8	1	1.000	1.000	0.999	0.001
9	1	0.991	0.966	0.001	0.001
10	1	1.000	0.999	0.919	0.015
Mean	—	**0.9991**	**0.9885**	**0.5258**	**0.0303**

Table 4.5: Phishing probabilities for each email across successive rewriting rounds using LIME-guided ChatGPT.

These changes were made based on the LIME output tables, which identified the most influential words driving the phishing prediction of the model. ChatGPT used this information to guide its rewriting strategy. The corresponding LIME plots for each round can be seen in Figure 4.7.



(a) LIME plot of original version.

(b) LIME plot of first round.

Figure 4.7: LIME explanations for Email 9. The plots highlight the most influential words used by the model to classify the email as phishing. Red bars on the right (positive SHAP values) push the prediction toward phishing, while green bars on the left (negative values) push it toward legitimate.

During the rewriting process, most phishing-related words were systematically removed—except for "USAA". Although LIME identified it as contributing to the phishing classification, it was retained in all versions. This likely reflects its importance in maintaining the meaning of the message and plausibility. From a human perspective, removing the brand name would make the email less coherent and credible. ChatGPT seemed to implicitly prioritize preserving this narrative integrity over eliminating all flagged terms.

Analysis of Phishing Probability As shown in Table 4.5, using only 25 features in the first rewriting round, leads to limited impact—ChatGPT struggles to legitimize the emails, with phishing probability dropping by just 1.1%. In the second round, results improve significantly with a 46.3% drop and 4 (almost 5) emails classified as legitimate. By the third round, phishing probability drops by 96.97%, and all emails are misclassified as legitimate. This demonstrates the effectiveness of iterative rewriting guided by targeted feature manipulation.

Analysis of Email Quality The quality of each email is assessed based on its credibility, similarity to original messages, and length (reflecting readability). This evaluation is carried out subjectively by the author.

- **Credibility**: The emails generated in the third round appear significantly more credible. While the original versions were overtly suspicious, these revised versions seem more legitimate at first glance.
- **Similarity**: The third-round emails can be viewed as refined versions of the originals. They preserve the underlying intent while presenting the content in a clearer and more polished manner.
- **Length of the email**: As shown in Table 4.6, the number of words decreased significantly after rewriting, except for Email 9. On average, the word count dropped by 34.13%, improving readability and making the emails clearer and more concise. The original versions were often long and poorly structured, whereas the rewritten versions by ChatGPT are easier to understand and appear more legitimate. This suggests that shorter emails may appear more legitimate. However, this assumption is challenged by Email 9, which is already very short yet still classified as nearly 100% phishing.

Index	Original Word Count	Modified Word Count	Difference Ratio
1	144	117	-0.1875
2	354	147	-0.5847
3	266	185	-0.305
4	120	114	-0.0500
5	400	190	-0.5250
6	143	92	-0.3566
7	472	182	-0.6144
8	451	181	-0.5998
9	11	14	+0.2727
10	489	268	-0.4520
Mean	**284.9**	**149.0**	**-0.3413**

Table 4.6: Word count comparison before and after rewriting, with a relative difference ratio.

Conclusion This experiment demonstrates that combining explainable AI with large language models enables highly effective adversarial rewriting. While the first round had limited effect, by the second round nearly half of the phishing emails were misclassified, and by the third, all evaded detection. The rewritten emails were shorter, clearer, and more credible, yet still retained their phishing intent. Notably, ChatGPT sometimes preserved certain phishing-related terms to maintain the meaning and credibility of the message, revealing a trade-off between following instructions and preserving coherence. These results highlight a key vulnerability: explanation tools can help attackers iteratively craft emails that bypass detection.

4.2.3 SHAP Guided Adversarial Rewriting via LLMs

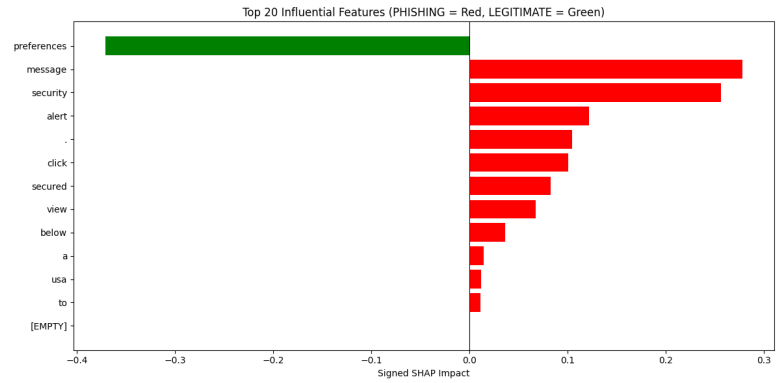
Following the review of how LIME can be used to guide ChatGPT, we now turn to the second explainability technique introduced in the State of the Art, which is SHAP (Section 2.5.2).

Methodology We applied SHAP locally to each of the 10 phishing emails to extract 40 key features: the 20 most influential phishing features and the 20 most influential legitimate ones. This broader set aimed to improve editing efficiency, building on the LIME-based experiment, where only 25 features required three rounds to reach full evasion. By increasing feature coverage, we aimed to achieve similar or better results in fewer steps. Guided by SHAP, ChatGPT rewrote the emails over two rounds, preserving phishing intent while enhancing legitimacy. We then evaluated the rewritten emails based on changes in phishing probability and quality, and aggregated the most common phishing-related features across all emails to gain deeper insight into the behavior of the model.

Results and Analysis We now re-examine the transformation process of Email 9, this time focusing on a modification guided by SHAP explanations. The original version of the message was:

> *USAA Security Preferences Message Alert. Click below to view secured Message*

The SHAP feature importance plot that was used to guide the rewriting of ChatGPT in this single transformation round is shown in Figure 4.8a. The final output produced by ChatGPT after incorporating the SHAP indications is presented in Figure 4.8b.



(a) SHAP feature relevance plot for the original version of Email 9.

Notification: Update to Your Preferences

A recent adjustment has been registered in your settings.

To review the recent update and related information, proceed through the following link:
🔗 [Access Preferences Dashboard]

If you have not initiated any changes or would like to verify configuration details, we encourage you to consult your account settings.

Thank you for staying connected.
— USAA Member Services

(b) Email 9 rewritten by ChatGPT using SHAP guidance.

Figure 4.8: SHAP-guided transformation process of Email 9.

Email	Label	Original Prob.	Round 1 Prob.	Round 2 Prob.
1	1	1.000	0.997	0.013
2	1	1.000	1.000	0.022
3	1	1.000	0.995	0.050
4	1	0.999	0.981	0.982
5	1	1.000	1.000	0.236
6	1	1.000	0.999	0.007
7	1	1.000	1.000	0.004
8	1	1.000	1.000	0.046
9	1	0.991	0.085	0.085
10	1	1.000	0.999	0.001
Mean	—	**0.9991**	**0.9056**	**0.1446**

Table 4.7: Probabilities output by the phishing detection model for each email across successive rewriting rounds.

Analysis of Results in Terms of Phishing Probability As shown in Table 4.7, only two rewriting rounds were sufficient to cause 9 out of 10 phishing emails to be misclassified as legitimate. The average phishing probability dropped from 0.9991 to 0.9056 after the first round, a modest decrease of 9.3%, reflecting limited initial impact. However, the second round was significantly more effective, reducing the average to

0.1446—an overall drop of 85.5% compared to the original. This sharp decline confirms the strength of iterative, explainability-guided rewriting.

Email 9 was the only one misclassified as legitimate after the first round. As noted in the LIME experiment, its extreme brevity (11 words) limited the number of usable features, making the larger feature set of SHAP (40) only marginally more informative. Despite this, SHAP-based rewriting still proved more effective. Email 4 was the only failure in round two. Seven phishing-related terms identified by SHAP—*©, you, are, file, registered, office,* and *unlimited*—remained in the text. ChatGPT had not removed them, which explains the result. Once these were manually replaced, the phishing probability dropped to 0.22.

Analysis of Email Quality The email quality observations closely mirror those from the LIME experiment. Third-round emails appear more credible, better written, and easier to read than the original versions, which were often lengthy, poorly structured, and clearly suspicious. While the core intent remained, the phrasing became more natural and polished. Most emails were shortened, improving readability—except for Email 9, which was significantly lengthened. While the LIME-based rewriting added only 12 characters, the SHAP-guided version added 333 characters. Despite this, it retained the original intent while adding clarity, structure, and content, making it seem more legitimate. A final detail is the occasional use of smileys of ChatGPT, likely intended to make the messages appear more friendly and aligned with modern digital communication norms.

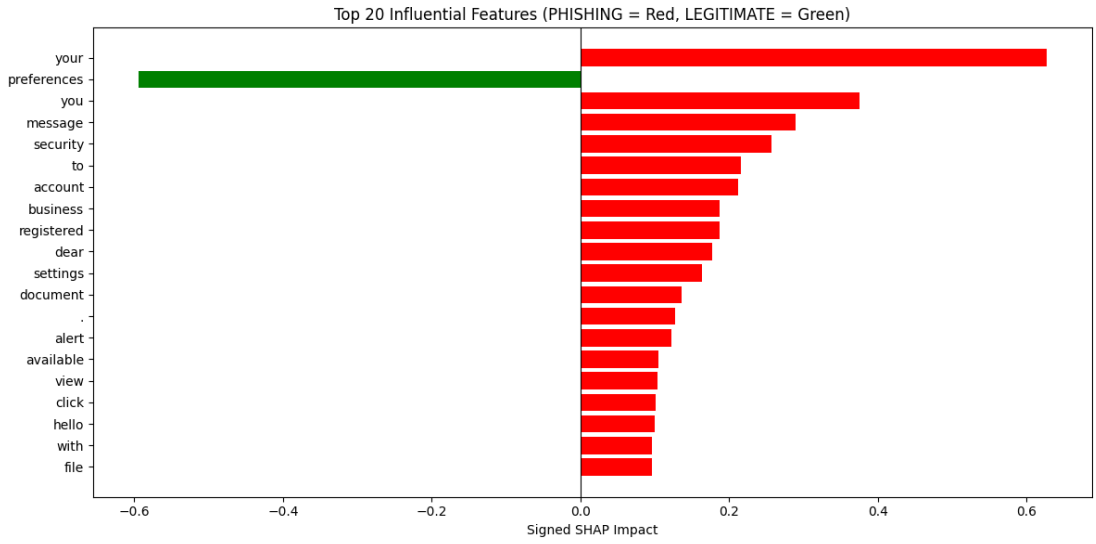


Figure 4.9: Bar plot with the 20 most influent features across the 10 phishing emails analyzed over two rounds.

Analysis of Top 20 Global Phishing Features To better understand the decision making of the model, we aggregated the top 20 most influential phishing features across all 10 emails. This global analysis reveals consistent patterns used by the classifier. As shown in Figure 4.9, frequent indicators include terms like *message*, *security*, *account*, *registered*, and *dear*—all commonly associated with phishing content.

Conclusion This experiment confirms that SHAP can effectively guide a language model in rewriting phishing emails to evade detection. After two rounds of SHAP-based rewriting, 9 out of 10 emails were misclassified as legitimate, with the second round proving especially impactful. As with LIME, the rewritten emails became more concise, readable, and credible. SHAP showed better performance on very short emails, however, unlike LIME, it failed to convert all emails into legitimate ones, revealing some limitations. Finally, aggregating the most influential features of SHAP provided a global view of common phishing patterns and helped shape the next experiment.

4.2.4 SHAP Guided Rewriting via Global Feature Importance

Expanding on the initial approach of identifying the top 20 most influential features across a set of 10 emails, we extended the analysis to a broader dataset in order to examine which features consistently exert the greatest influence on the prediction of the model at a larger scale.

Methodology We began by randomly selecting 500 phishing and 500 legitimate emails from the test set D_{ts} and applied SHAP to extract the 50 most globally influential features from each class. We selected 50 features per class to maximize coverage of recurrent patterns identified across the dataset, while still maintaining manageable prompt length. These features were then used to guide the rewriting of the fixed set of 10 phishing emails (excluded from the 1,000-email SHAP analysis). In the first rewriting round, ChatGPT was provided with the top 50 phishing-related features and asked to modify the emails to appear more legitimate. In the second round, it was guided by the top 50 legitimate-related features to further refine the messages. We first analyzed the global feature importance graphs to identify broader patterns in the behavior of the model, and then evaluated each rewritten email based on phishing probability and content quality.

Analysis of Top 50 global Phishing Features The analysis begins with the global feature graphs. Figure 4.10 and Figure 4.11 show the top 30 most influential features for the 500 legitimate and 500 phishing emails, respectively (limited to 30 features for readability purposes, instead of 50).

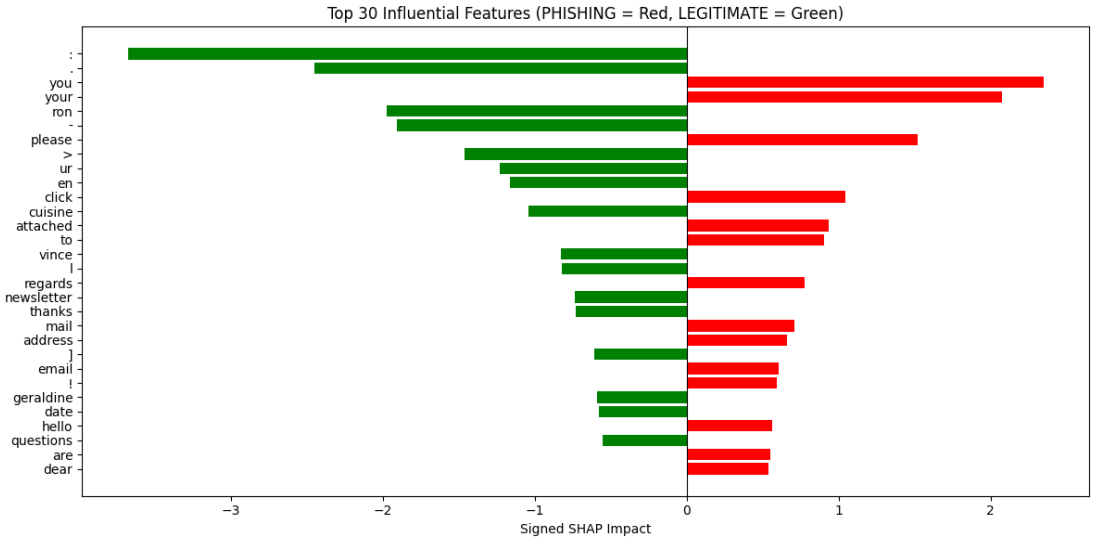


Figure 4.10: Bar plot with the 30 most influencial features across the 500 **legitimate** emails analyzed.

Figure 4.10 reveals a strong presence of punctuation in legitimate emails, with ":" and "." being the top features—suggesting punctuation can outweigh specific words in influence. Other frequent symbols include "-", ">", "]", ",", "[", and "/". Notably, the exclamation mark is absent, appearing instead as a phishing indicator. We also observe first names like *Geraldine*, *Vince*, and *Ron*, while most other tokens lack clear significance. However, some features raise concerns. Terms like *cuisine*, *newsletter*, and *Geraldine* come from a specific source: a recurring English-language newsletter titled "La Cuisine de Geraldine". About 40 such emails appear in the *Personal Advertising Emails* dataset of 1,052 mails. While intended to diversify the dataset, this inclusion introduced unintended bias.

The phishing side of the analysis, shown in Figure 4.11, highlights a set of features that are much more expected and can be grouped into several intuitive categories:

- **Personalization**: *your*, *you*
- **Credentials & financial terms**: *account*, *password*, *money*, *bank*, *payment*, *name*
- **Overly formal/unnatural greetings and politeness**: *dear*, *mr*, *kindly*, *please*
- **Typical phishing vocabulary**: *click*, *download*, *here*

Compared to legitimate emails, punctuation (aside from the exclamation mark) is much less important, whereas specific words play a major role in pushing the model toward a phishing classification. These findings are aligned with common expectations and help reinforce the reliability of our model.

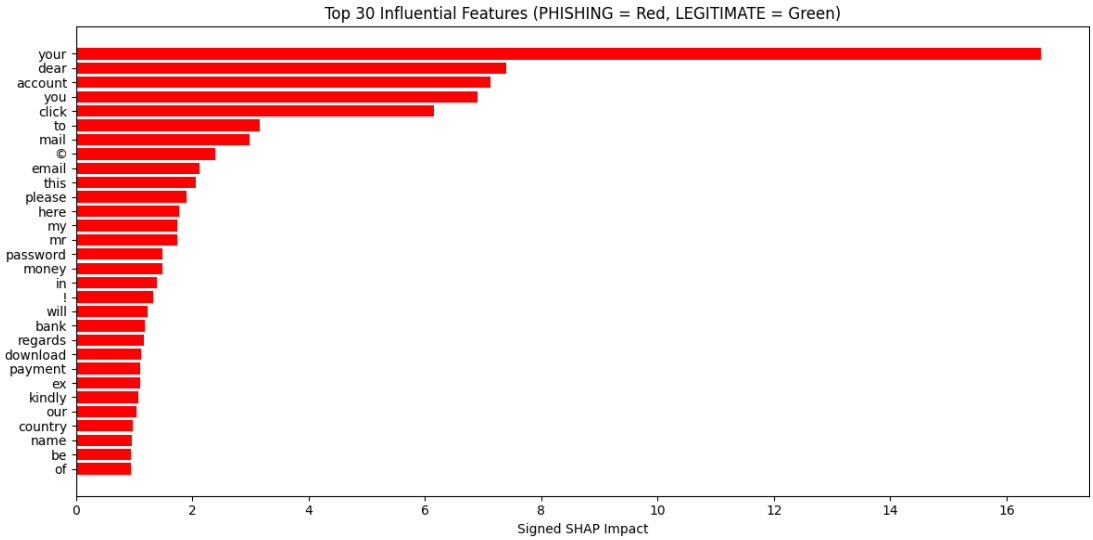


Figure 4.11: Bar plot with the 30 most influential features across the 500 **phishing** emails analyzed.

Email	Label	Original Prob.	Round 1 Prob.	Round 2 Prob.
1	1	1.000	0.997	0.946
2	1	1.000	1.000	0.018
3	1	1.000	0.996	0.002
4	1	0.999	0.999	0.949
5	1	1.000	0.993	0.001
6	1	1.000	1.000	0.000
7	1	1.000	1.000	0.003
8	1	1.000	1.000	0.001
9	1	0.991	0.999	0.255
10	1	1.000	0.363	0.363
Mean	—	**0.9991**	**0.9347**	**0.2538**

Table 4.8: Model probabilities for each email before and after SHAP-guided rewriting. Used global features and not local features based on a specific input.

Analysis of Results in Terms of Phishing Probability The results in Table 4.8 show that this approach performs nearly as well as computing SHAP or LIME values individually. After two rewriting rounds, 8 out of 10 emails bypassed detection, with an overall phishing probability reduction of 74.6%. In the first round, focused on removing phishing-related features, only Email 10 was misclassified as legitimate, with an average drop of just 6.4%. The second round, which emphasized legitimate indicators, was far more effective, suggesting that reinforcing legitimacy may be more robust than simply removing phishing cues.

Emails 4 and 1 remained flagged as phishing. A closer inspection of Email 4 revealed

that 13 phishing-related words remained in the message. When ChatGPT was explicitly asked to replace these specific terms with synonyms outside the top 50 phishing features, the phishing probability dropped dramatically to 0.0017. Applying the same strategy to Email 1 reduced its probability to 0.0085. This highlights a limitation in prompt effectiveness: despite general instructions, ChatGPT often left phishing terms untouched. However, when asked to identify them or replace a specific list, it responded effectively and significantly reduced the phishing score.

Subject: Member Notification – Security Hub Update

Hi Jennifer,

A new security event has been logged within your USAA Communications Dashboard.
This item is part of your routine alerts and is now available for member access.

—

Event Details
• Classification: Security Preference Update
• Availability: 22 April 2025
• Location: Member Access Panel → Security Notifications

To view this entry, visit your secure member interface.
[Access Notifications Center]

—

This is an automated notice from USAA. No reply is required.
For help understanding this alert, visit: [FAQ] | [Notification Settings]

—

USAA Member Support Services
9800 Fredericksburg Road, San Antonio, TX 78288
[Manage Preferences] | [Privacy Policy]
© 2025 USAA. All trademarks belong to their respective holders.

Figure 4.12: Transformation of Email 9 with global SHAP features. This email includes several elements from the top 50 phishing features list, such as *©*, *your*, and *you*. The core intent is well preserved. Notably, the name *Jennifer* appears, along with the use of emojis.

Analysis of Email Quality Beyond phishing probability, we also conducted a qualitative analysis of the rewritten emails. Several consistent patterns emerged, revealing both the strengths and limitations of using ChatGPT. Figure 4.12 illustrates this through the transformation of Email 9.

- As previously mentioned, ChatGPT does not always follow instructions strictly. In some cases, it retained words like *dear*, despite them being among the top features contributing to phishing detection. This suggests that the adherence of the model to constraints is not absolute.

- The core intent of each email is generally preserved, even though the surface-level wording often changes significantly. In some cases, the content is even restructured to make the message appear more credible.
- The rewritten emails tend to be well formulated and convincing. First names such as *Geraldine*, *Vince*, *Ron*, and *Jennifer* are frequently inserted, either in the signature or to personalize the message.
- Email 9, being originally very short, grows substantially with each rewriting round. By the second iteration, over 700 characters were added, with ChatGPT generating a verbose and heavily rephrased version of the message.
- ChatGPT also tends to introduce emojis in the rewritten emails to increase their perceived legitimacy—even when explicitly instructed not to do so.

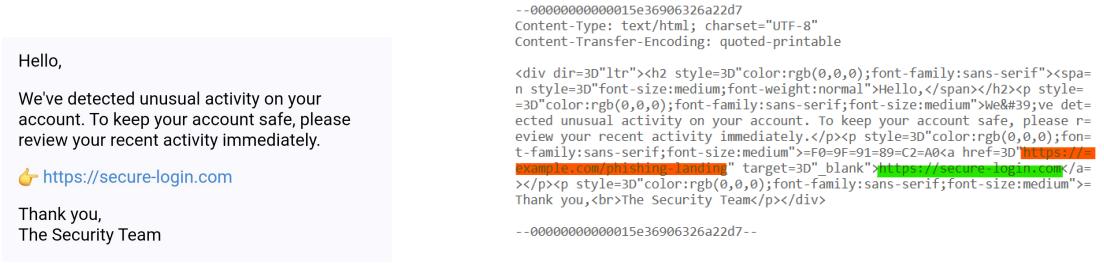
Conclusion This experiment shows that globally aggregated SHAP features can effectively guide large language models in rewriting phishing emails to evade detection. Although slightly less effective than local methods, this approach is more scalable and requires less customization. Reinforcing legitimate signals proved more robust than simply removing phishing cues. Feature distribution analysis confirmed this, with legitimate emails marked by structured punctuation and phishing ones by urgent, financial, or overly polite language—validating the output of SHAP and offering insight into the logic of the model. While ChatGPT sometimes ignored instructions or added unwanted content, such issues could be bypassed in real-world scenarios through repeated prompting, reinforcing the practicality of LLMs in iterative adversarial attacks.

4.3 Experiments on the URL Module

Several types of deceptive attacks can be applied to URLs, as discussed in Section 2.6.3 of the State of the Art. However, only the **Altered Malicious URL Attacks**— attacks that change the appearance of the URL without altering its actual destination — are relevant to our work, and more specifically the **URL Shortened** and **Redirect** attacks. Both attacks achieved a 100% success rate after targeted manipulations, guided by SHAP-based feature importance analysis.

The **Hidden Malicious URL Attacks** — attacks that conceal the malicious URL within the HTML (making it invisible to the user unless inspecting the raw HTML source), but still correctly leading to the hidden malicious destination — proved ineffective against our model. Although the malicious URL is hidden in the HTML part of the email, our system extracts all URLs from both the plain text and HTML sections. As a result, the malicious URL is still detected and classified by the model. While such deceptive techniques may successfully mislead users, they are not effective against our model. Figure 4.13 illustrates a link mismatch attack. The email as received in the Gmail inbox—shown in Figure 4.13a—displays what the user sees, while Figure 4.13b shows

the corresponding .eml file structure that supports this analysis. Figure 4.14 displays the URLs extracted from the email. As shown, the malicious URL is correctly extracted and will still be analyzed by the model.



(a) Received email in the Gmail mailbox.

(b) Received email in .eml format. In red the hidden malicious URL. In green the fake legitimate URL.

Figure 4.13: Link attack.

```
{'https://example.com/phishing-landing', 'https://secure-login.com', 'https://example.com/phishing-landing>'}
```

Figure 4.14: Extracted URLs from the email by our model.

Deceiving our URL detection model therefore requires modifying the visual appearance of URLs while keeping their actual destination unchanged. Similarly, **URL Composition Attacks** and **Behavior-Based Attacks** fall outside the scope of this evaluation, as they alter the underlying behavior.

4.3.1 Methodology Overview

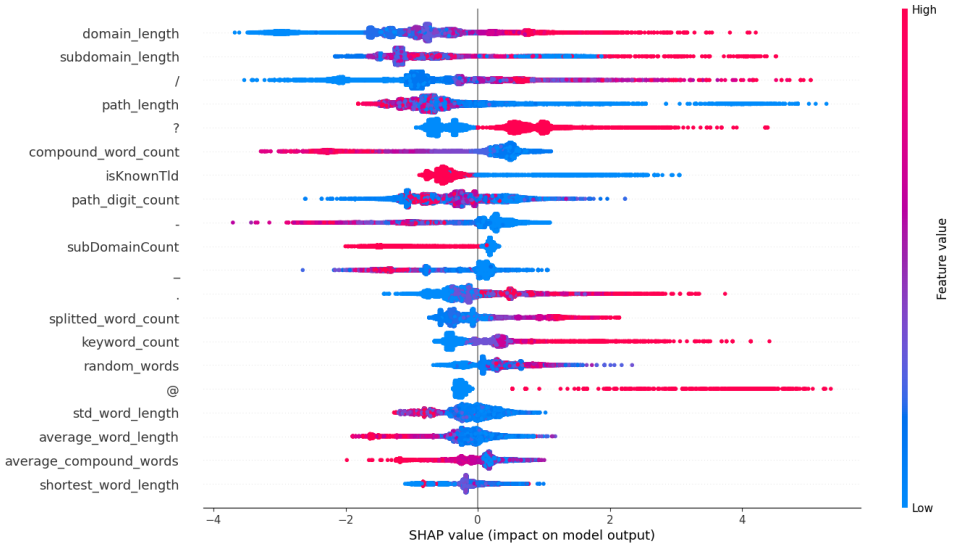
The URLs-based classification model is first analyzed to understand the influence of each feature in the decision process. This analysis is based on SHAP-generated values and visualizations. Next, 100 phishing URLs are randomly selected from the test set D_{ts}, all of which are correctly classified as phishing by the model. These URLs serve as a baseline for evaluating the effect of adversarial manipulations.

Each attack consists of modifying the URLs in a targeted way while preserving their redirection behavior. The modified URLs are then re-evaluated by the detection model, and their new classification probabilities are recorded. The design of the adversarial attacks is informed by a SHAP-based analysis of the URL classification module. This analysis allows us to identify the most influential features contributing to phishing predictions and to select those that are most promising to manipulate.

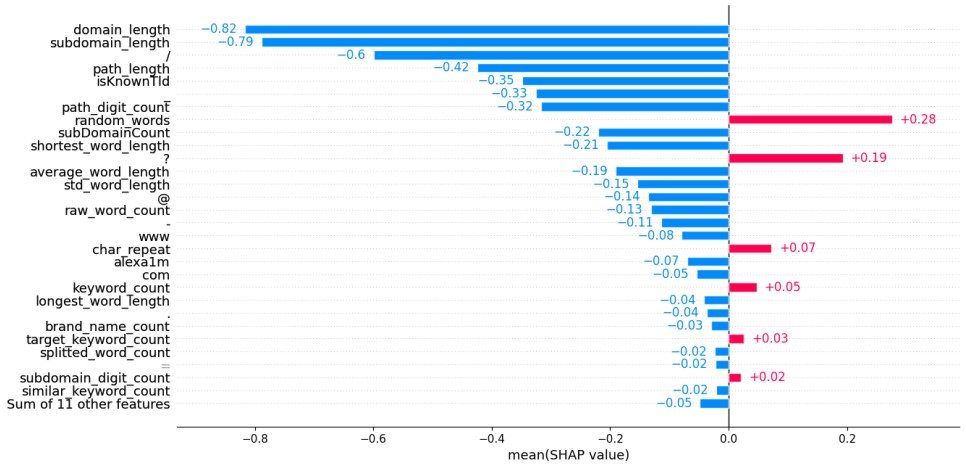
4.3.2 Analysis of URL Module with SHAP

Before applying any attack, the behavior of the model is analyzed using SHAP in order to determine which URL features have the greatest impact on classification decisions.

Figure 4.15 show the bar plot and beeswarm plot generated from D_{mt} and D_{ts}. The features most associated with legitimate classification include a moderate `subdomain length`, a short `domain length`, a low number of `slashes` (/), a non-empty `path`, and the use of a `known Top-Level Domain` (TLD). A TLD is considered "known" if it belongs to a predefined list embedded in the features extraction code (e.g., .com, .net, etc.). In contrast, the features most strongly linked to phishing classification are the presence of `random words` and one or more `question marks` (?) in the URL.



(a) Beeswarm plot for the URL classification of D_{mt} and D_{ts} datasets.



(b) Bar plot for the URL classification of the D_{mt} and D_{ts} datasets.

Figure 4.15: SHAP plots for the URL classification of the D_{mt} and D_{ts} datasets.

On the basis of this analysis, three characteristics appear to be relevant targets for manipulation. First, the number of `slashes` (/) in the path can be reduced using a URL shortener service. Second, the `domain` and `subdomain` structure can be modified through the use of redirection services. These two strategies form the basis of the attacks presented in the following sections.

4.3.3 URL Shortener-Based Attack

URL shorteners are services that convert long URLs into significantly shorter ones, which redirect to the original destination when accessed. These services are commonly used to simplify URL sharing—especially on platforms with character limits—or to conceal the true destination of a link.

In this experiment, the TinyURL[1] service was selected because of its public API and free availability. TinyURL generates a unique alias that maintains the redirection functionality of the original URL. During testing, approximately 11% of the phishing URLs were rejected by the service, likely due to blacklist filtering. As a result, these URLs could not be shortened. The following results are therefore based on the 89 successfully shortened URLs.

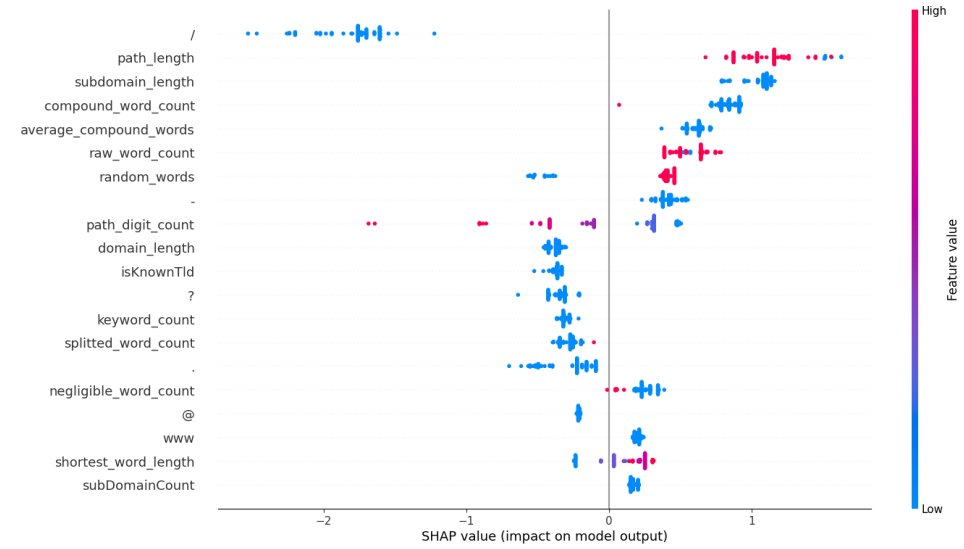
Results and Analysis of the URL Shortener Attack After URL shortening, 12.86% of the shortened URLs were classified as legitimate by the model, with an average probability drop of 0.2331 among those still identified as phishing. Figure 4.16 shows that one of the key features influencing the classification of phishing URLs is the absence of a subdomain. Given this observation, a second experiment was conducted in which a custom domain including a subdomain—such as `safe.tinyURL.com`—was simulated. This setup reflects the type of configuration made possible by the premium option of TinyURL.

Results and Analysis of the URL Shortener Attack with a Custom Domain All shortened URLs using the custom domain were classified as legitimate by the model, resulting in a 100% evasion rate. This confirms that reducing the number of path segments through shortening, combined with the presence of a subdomain, is sufficient to bypass the detection mechanism. The final SHAP plots, showing the detailed contribution of each feature, are provided in Appendix C.2.

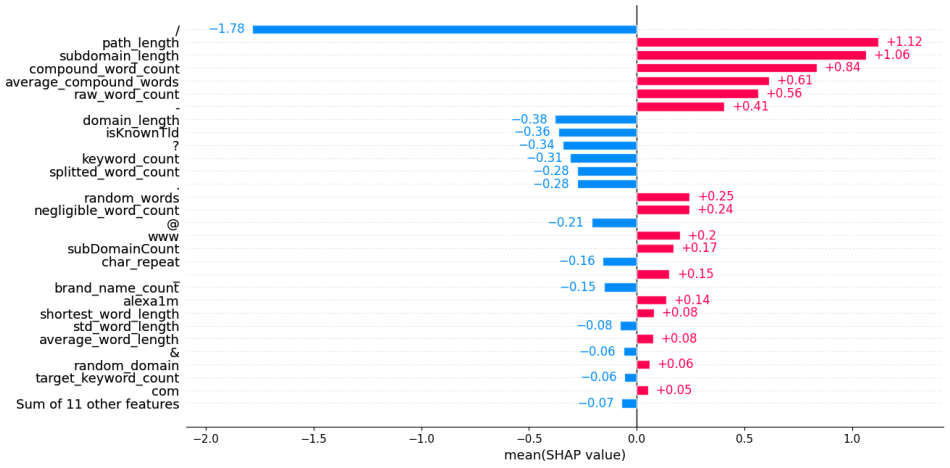
4.3.4 Redirection-Based Attack

This attack leverages URL redirection by embedding a phishing link as a parameter within a URL that appears legitimate. Although the visible domain may belong to a trusted provider, the full URL redirects the user to a malicious destination. Redirection services might seem like obvious channels for attacks, but they also serve essential functions on

[1] `https://tinyurl.com/`



(a) Beeswarm plot of the URL classification model of the selected shortened URLs.



(b) Bar plot of the URL classification model of the selected shortened URLs.

Figure 4.16: SHAP plots of the URL classification model of the selected shortened URLs.

the web. Redirects are commonly used to guide users and search engines to updated or canonical URLs, manage domain changes, merge websites, or handle removed pages by pointing visitors to relevant alternatives. These practices ensure smooth navigation and maintain Search Engine Optimization (SEO) integrity [62]. The general principle of this technique is illustrated in Figure 4.17.

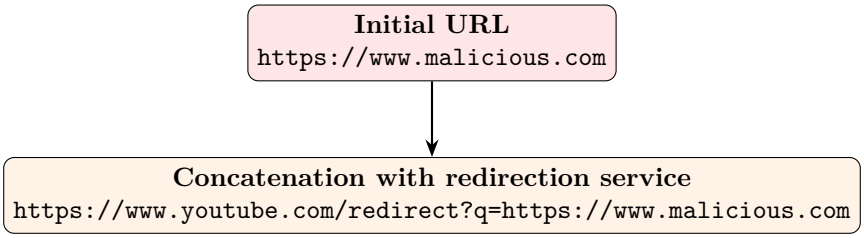


Figure 4.17: Combining a legitimate redirection service URL with a malicious URL.

In practice, redirection is only possible when the URL structure follows a specific format supported by the hosting service. Not all platforms offer this functionality, as it requires a dedicated redirection mechanism. To build a realistic test set, a collection of redirection URLs was gathered from trusted providers such as YouTube, Google, Facebook, LinkedIn, and Slack. The first set of experiments was conducted by concatenating these redirection URLs with the 100 selected phishing URLs.

Results and Analysis of the First Redirect Attack Only 3 out of 100 modified URLs were classified as legitimate; the others remained flagged as phishing, with an average probability reduction of 0.066. These results demonstrate that the redirection strategy slightly deceived the model, although its overall impact remains limited. To better understand which features contributed most to these predictions, SHAP values were examined. Table 4.9 and Table 4.10 present the five most impactful features for the URLs with the highest and lowest phishing probabilities, respectively. As a reminder, positive SHAP values increase the likelihood of a phishing classification, while negative values contribute to a legitimate prediction.

Feature	**Value**	**SHAP Value**
@	1.0	3.0382
?	2.0	2.6534
/	6.0	1.6995
.	2.0	1.6303
Number of detected keyword	3.0	1.2217

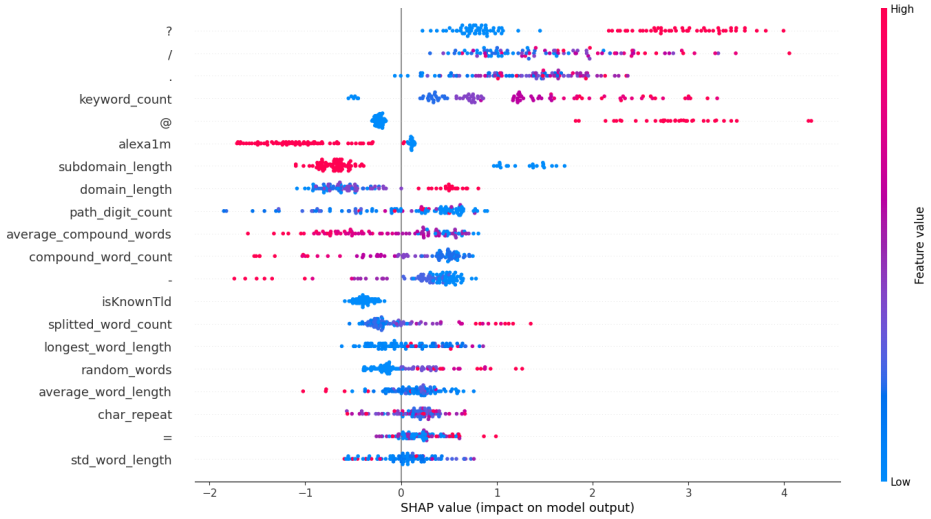
Table 4.9: Most impactful features for the URL with the **highest** phishing probability: `https://slack-redir.net/link?url=https:/.../?email=jose@monkey.org` with a probability of **1.0000**.

Feature	**Value**	**SHAP Value**
/	6.0	1.0403
?	1.0	0.8866
Number of detected random words	19.0	0.8330
Length of the path	648.0	-0.8194
Length of the domain	8.0	-0.7449

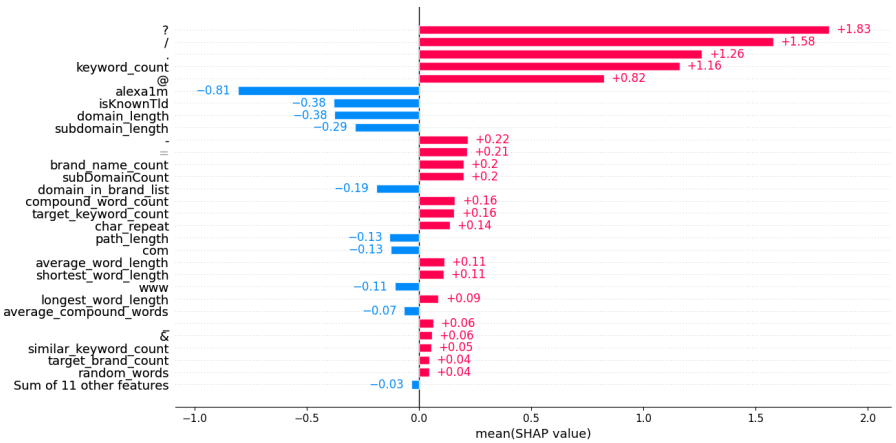
Table 4.10: Most impactful features for the URL with the **lowest** phishing probability: `https://www.facebook.com/l.php?u=https://t.pvboxorange...` with a probability of **0.3599**.

To improve this success rate, the most influential features observed in this attack are analyzed below, based on SHAP explanations. As shown in the various graphs (see Figure 4.18), most of the impactful features are related to the URL path, which often

contains the malicious payload. To conceal this part, a shortening technique can be applied. This approach reduces the influence of the top contributing features, such as the characters `/`, `?`, and `.`, and also lowers the `number of detected keyword` and other indicators, like the number of `@` or `-`. This observation is consistent with the characteristics found in the URL that had the highest phishing probability.



(a) Beeswarm plot of the URL classification model of the 100 tested URLs for the first redirection attack.



(b) Bar plot of the URL classification model of the 100 tested URLs for the first redirection attack.

Figure 4.18: SHAP plots of the URL classification model of the 100 tested URLs for the first redirection attack.

To evaluate the effect of this strategy, a second experiment was performed where the malicious URL is first shortened before being embedded within a redirection link.

Results and Analysis of the Second Redirect Attack To build on the previous result, the malicious URLs are first shortened before being embedded within the redirection link. The process is illustrated in Figure 4.19. Shortening was performed using the publicly available TinyURL service. For URLs that were rejected—likely due to blacklisting—shortened versions were simulated to maintain consistency across the dataset.

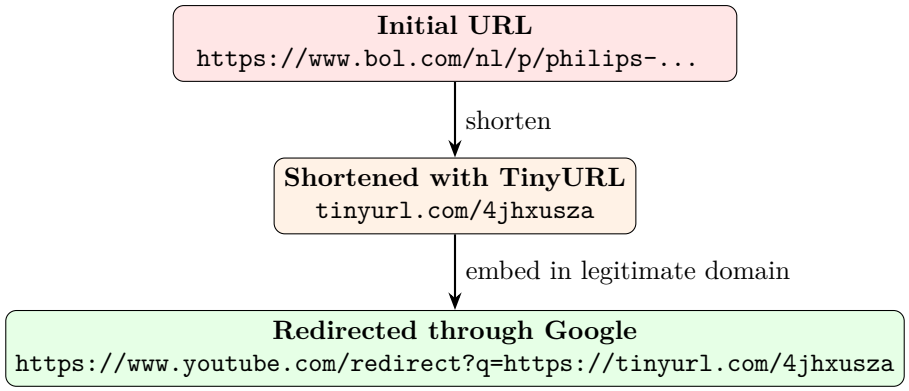
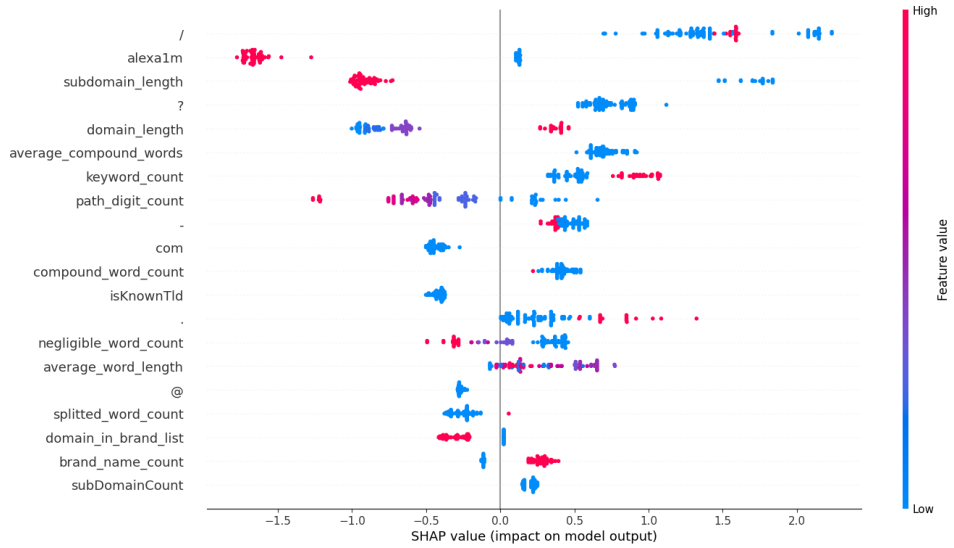


Figure 4.19: Transformation of a phishing URL using shortening and redirection to evade detection.

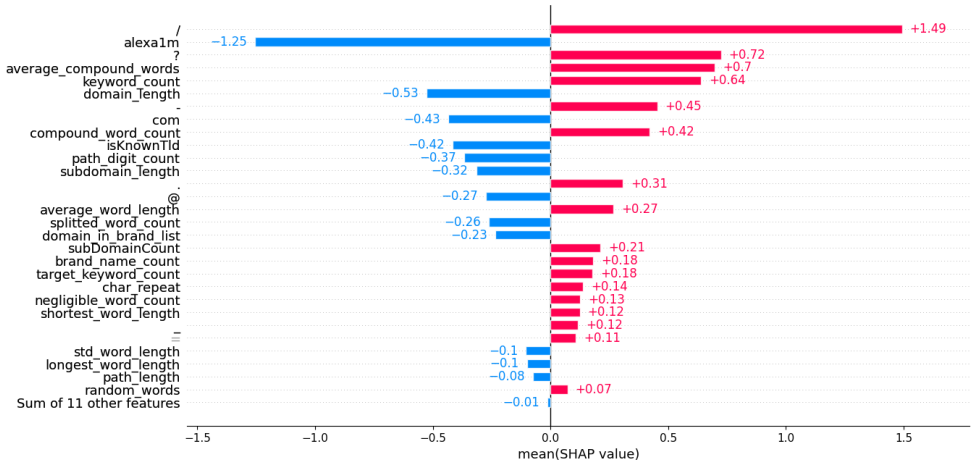
As a result, 30% of the modified URLs are classified as legitimate by the model. This increase can be attributed to the reduced number of `slashes` (/) in the URL path. Although this feature still plays an important role in phishing detection, its average SHAP value slightly decreased in this scenario.

As shown in Figure 4.20, all major features previously associated with the URL path have experienced a noticeable drop in influence. The number of `question marks` (?) was reduced to just one, which also helped decrease the impact of that feature. Similarly, a reduction in the `number of dots` (.) contributed to a lower phishing score. Additionally, shortened URLs often contain more `digits in the path`—an element that tends to favor legitimate classification. The absence of the `@` symbol and the presence of a single `.com` further reinforce the legitimate classification. At this stage, the most influential features favoring a legitimate classification are those related to the domain component, including `alexa1m`, `domain length`, `isKnownTld`, and `subdomain length`. The `alexa1m` feature indicates whether the domain appears in the Alexa Top 1 Million list — a listing of the top 1 million websites. This record includes well-known legitimate domains like `youtube.com`, `facebook.com`, `wikipedia.org`, `yahoo.com`, and `amazon.com`. It is therefore expected that this feature has a strong impact on the legitimate detection, especially given that many of the domains used in the modified URLs belong to this list. These

observations suggest that URLs with favorable domain characteristics can be used to bypass the model, revealing a potential vulnerability in the current detection system.



(a) Beeswarm plot of the URL classification model of the 100 tested URLs for the second redirection attack.



(b) Bar plot of the URL classification model of the 100 tested URLs for the second redirection attack.

Figure 4.20: SHAP plots of the URL classification model of the 100 tested URLs for the second redirection attack.

To further improve the success rate, the next step involves customizing the domain

names used in the redirection URLs. The goal is to ensure that the domain-related features align with those typically associated with legitimate URLs, while still preserving the redirection mechanism.

Results and Analysis of the Third Redirect Attack Constructing URLs with a `known TLD`, a short `domain`, and a `subdomain` of reasonable length -such as the following examples- results in 99% of the URLs being classified as legitimate.

- https://secure.log.com/redirect?q=
- https://check.amz.com/redirect?q=
- https://www.ucl.edu/redirect?q=
- https://auth.pro.net/redirect?q=

The URL still identified as phishing is detected due to the existence of `?` and the `number of detected keyword` feature, which has a value of 2 as detected in Table 4.11. Specifically, the words `"redirect"` and `"com"` appearing in the path are recognized as keywords during feature extraction.

Feature	Value	SHAP Value
Length of the domain	3.0	-2.6692
?	1.0	1.1646
Number of digit in the path	6.0	-0.9525
Number of detected keyword	2.0	0.9214
Length of the subdomain	3.0	-0.9198

Table 4.11: Top 5 features for the URL with the **highest** phishing probability (0.5592) in the redirection attack: `https://www.ucl.edu/redirect?q=https://tinyurl.com/23x347a4`.

The `?` character cannot be removed from the URL, as it is necessary for the redirection to function correctly. However, by restructuring the link to mimic the redirection format of Google—using `"url?q="` instead of `"redirect?q="`—we reduce the `number of detected keywords` from 2 to 1, while maintaining the same redirection behavior. As a result, 100% of the tested URLs were classified as legitimate by the model. The detailed impact of each feature on the prediction of the model is shown in Appendix C.2. These findings show that by combining a custom domain, a redirection service and a URL shortener, it becomes highly feasible to bypass the detection model.

4.3.5 Conclusion

These experiments demonstrate that a URL-based detection model can be effectively bypassed by exploiting features revealed through XAI techniques. Shortening, redirection, and domain customization take advantage of specific model biases, such as an overemphasis

on path complexity or domain patterns, to reduce phishing scores while leaving the actual malicious intent intact. The results show that attackers can combine simple manipulations to construct deceptive URLs that evade detection entirely. This emphasizes the limitations of relying solely on static URL features and calls for more robust approaches that incorporate behavioral and content-based signals.

4.4 Summary of the Experiments

To conclude, Table 4.12 provides a comprehensive overview of all adversarial experiments conducted across the three core components of the phishing detection system: structure, text, and URL. Each row details a specific attack strategy, the tools or methods used, the targeted features, and the resulting success rate. This summary helps to identify which parts of the model are most robust, and where targeted adversarial manipulation is most effective.

As the results show, structure-based attacks are more complex and demand greater technical expertise, particularly regarding email headers and mail transfer behavior. In contrast, text and URL attacks are easier to execute—especially using tools like ChatGPT or public URL shorteners—and generally achieve higher success rates, making them more appealing to real-world attackers.

Targeted Component	**Attack Type**	**Method / Tool**	**Targeted Feature(s)**	**Success Rate**
Structure	Message-ID manipulation	Manual & Postfix	Dots in domain part	6%
	Received headers (4) manipulation	Manual	Number of Received headers	12%
	Received headers (7) manipulation	Manual & Postfix	Number of Received headers	34%
Text	TextFooler	TextAttack	Word substitutions	100%
	Free-form rewriting	ChatGPT	—	0%
	LIME-guided rewriting	ChatGPT + LIME	Top 25 local words	100%
	SHAP-guided rewriting (Local)	ChatGPT + SHAP	20 phishing + 20 legitimate features	90%
	SHAP-guided rewriting (Global)	ChatGPT + SHAP	50 phishing + 50 legitimate features	80%
URL	Hidden malicious URL	Manual	HTML	0%
	Shortener only	TinyURL	Slashes in path	12.86%
	Shortener + custom domain	TinyURL & Simulated	Slashes + domain name	100%
	Redirection only	Domain with redirection service	Domain name	3%
	Redirection + shortener	TinyURL	Domain name + Path	30%
	Redirection + shortener + domain	Simulated	Domain name + path + keyword count	100%

Table 4.12: Summary of the adversarial attacks by component, strategy, and effectiveness.

Chapter 5

Conclusion and Future Work

This work aimed to investigate to what extent a machine learning model, trained on multiple components of phishing emails, can remain robust when exposed to hand-crafted, explainability-guided, model-targeted adversarial attacks.

Our experiments demonstrate that even models with strong baseline performance —measured using the AUPRC metric—remain vulnerable to adversarial attacks driven by explainable AI (XAI). These attacks were crafted to exploit specific components of phishing emails. In the case of the structure-based module, the highest observed success rate is 34%, whereas attacks on the text-based and URL-based modules achieve success rates of up to 100%. These results highlight the vulnerability of high-performing classifiers to manipulation once their decision mechanisms are made transparent.

These findings demonstrate that explainable AI can serve as a powerful tool for guiding adversarial attacks. By identifying and manipulating the features that most influence the decisions of the model, attackers can craft inputs that bypass detection. This strategy, although requiring varying levels of technical effort depending on the target component, raises serious concerns about the robustness of phishing detection systems. Models trained on multiple, diverse features remain vulnerable when adversaries can exploit the transparency provided by XAI.

A key limitation of this study lies in the quality, recency, and diversity of the dataset used, which mainly relies on publicly available sources. While many public datasets exist, they are often outdated or incomplete, limiting the realism of evaluations under real-world conditions. We can reasonably assume that large industry players, such as Microsoft, possess high-quality, up-to-date phishing datasets, but unfortunately, these are not publicly available, restricting access for academic research and hindering progress in developing more robust detection systems.

Future research should focus on developing more resilient models through adversarial training. In addition, detection systems should shift their focus toward features that are

less susceptible to manipulation. As our results show, URL-based features are highly vulnerable to adversarial attacks, suggesting that future models may benefit from directly analyzing the content and structure of linked web pages rather than relying on URLs.

Improvements can also be made in the text module. For example, our experiments revealed that legitimate emails often contain first names, which the model tends to associate with trustworthiness. Exploring strategies such as generalizing, removing, or substituting these names could help reduce overfitting to such superficial patterns and improve robustness.

Further analysis should explore the impact of email authentication mechanisms, specifically SPF, DKIM, and DMARC, on phishing detection from a structural perspective. Investigating these features could clarify how effectively these authentication protocols help prevent phishing attacks.

This study also underlines the dual role of explainable AI: while XAI is essential for understanding model behavior and uncovering biases, it enables targeted attacks against the model as well. Striking the right balance between interpretability and robustness remains a major challenge in securing ML-based detection systems.

A key next step involves combining attacks across different email components (headers, text, URLs) and passing the resulting adversarial emails through the meta-classifier to observe the overall decline in phishing detection performance. This approach would provide a more realistic evaluation of system-wide vulnerabilities.

Machine learning models can significantly enhance phishing detection, but they are not sufficient on their own. Users must remain vigilant, and awareness efforts should continue to play a critical role. Educating users on the limitations of automated systems and encouraging cautious behavior are essential elements in a holistic defense strategy against phishing.

References

[1] Anti-Phishing Working Group (APWG). Apwg phishing activity trends reports, n.d. Accessed: 2025-02-28.

[2] Devottam Gaurav and Sanju Tiwari. Interpretability vs explainability: the black box of machine learning. In *2023 international conference on computer science, information technology and engineering (ICCoSITE)*, pages 523–528. IEEE, 2023.

[3] Rafał Kozik, Massimo Ficco, Aleksandra Pawlicka, Marek Pawlicki, Francesco Palmieri, and Michał Choraś. When explainability turns into a threat-using xai to fool a fake news detection method. *Computers & Security*, 137:103599, 2024.

[4] David Dittrich, Erin Kenneally, Michael Bailey, Aaron Burstein, KC Claffy, Shari Clayman, John Heidemann, Douglas Maughan, Jenny McNeill, Peter Neumann, Charlotte Scheper, Lee Tien, Christos Papadopoulos, Wendy Visscher, and Jody Westby. The menlo report: Ethical principles guiding information and communication technology research. Technical report, U.S. Department of Homeland Security, Science and Technology Directorate, Cyber Security Division, August 2012. CORE Technical Report.

[5] USENIX Association. Usenix security '25 ethics guidelines, 2025. Accessed: 2025-05-05.

[6] Phishing activity trends report, 4th quarter 2024. Technical report, Anti-Phishing Working Group (APWG), March 2025. Published March 19, 2025.

[7] Mimecast. The difference between phishing vs. spam emails, n.d. Accessed: 2025-02-28.

[8] Zainab Alkhalil, Chaminda Hewage, Liqaa Nawaf, and Imtiaz Khan. Phishing attacks: A recent comprehensive study and a new anatomy. *Frontiers in Computer Science*, 3:563060, 2021.

[9] Anti-Phishing Working Group. About us, 2025. Accessed: 2025-05-24.

[10] Anti-Phishing Working Group (APWG). Phishing activity trends report: 3rd quarter 2024. Technical report, Anti-Phishing Working Group, 2024. Accessed: 2025-05-02.

[11] Theodore Tangie Longtchi, Rosana Montañez Rodriguez, Laith Al-Shawaf, Adham Atyabi, and Shouhuai Xu. Internet-based social engineering psychology, attacks, and defenses: A survey. *Proceedings of the IEEE*, 2024.

[12] Amber Van Der Heijden and Luca Allodi. Cognitive triaging of phishing attacks. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1309–1326, 2019.

[13] Avisha Das, Shahryar Baki, Ayman El Aassal, Rakesh Verma, and Arthur Dunbar. Sok: a comprehensive reexamination of phishing research from the security perspective. *IEEE Communications Surveys & Tutorials*, 22(1):671–708, 2019.

[14] Jehyun Lee, Farren Tang, Pingxiao Ye, Fahim Abbasi, Phil Hay, and Dinil Mon Divakaran. D-fence: A flexible, efficient, and comprehensive phishing email detection system. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 578–597. IEEE, 2021.

[15] Mailtrap. Email headers explained: What they are and how to read them, 2021. Accessed: 2024-04-30.

[16] Mozilla Developer Network. Document object model (dom), n.d. Accessed: 2025-04-30.

[17] Ozgur Koray Sahingoz, Ebubekir Buber, Onder Demir, and Banu Diri. Machine learning based phishing detection from urls. *Expert Systems with Applications*, 117:345–357, 2019.

[18] Apache Software Foundation. Spamassassin public corpus, 2004.

[19] Bryan Klimt and Yiming Yang. The enron email dataset.

[20] Jose Joseph. Phishing research resources, 2025.

[21] Rachael Tatman. Fraudulent e-mail corpus, 2019.

[22] Ion Androutsopoulos, John Koutsias, Konstantinos V. Chandrinos, Georgios Paliouras, and Constantine D. Spyropoulos. The Ling-Spam dataset, 2000. Accessed: 2025-02-08.

[23] Georgios Sakkis, Ion Androutsopoulos, Georgios Paliouras, Vangelis Karkaletsis, Constantine D Spyropoulos, and Panagiotis Stamatopoulos. A memory-based approach to anti-spam filtering for mailing lists. *Information retrieval*, 6:49–73, 2003.

[24] Abdulla Al-Subaiey, Mohammed Al-Thani, Naser Abdullah Alam, Kaniz Fatema Antora, Amith Khandakar, and SM Ashfaq Uz Zaman. Novel interpretable and robust web-based ai platform for phishing email detection. *Computers and Electrical Engineering*, 120:109625, 2024.

[25] Ebubekir Buber. Phishing detection dataset (pdd), 2023.

[26] Mahmoud Khonji, Youssef Iraqi, and Andrew Jones. Phishing detection: a literature survey. *IEEE Communications Surveys & Tutorials*, 15(4):2091–2121, 2013.

[27] Microsoft. Learn about safe links in microsoft defender for office 365, 2025. Accessed: 2025-05-24.

[28] Dinil Mon Divakaran and Adam Oest. Phishing detection leveraging machine learning and deep learning: A review. *IEEE Security & Privacy*, 20(5):86–95, 2022.

[29] Pawan Prakash, Manish Kumar, Ramana Rao Kompella, and Minaxi Gupta. Phishnet: Predictive blacklisting to detect phishing attacks. In *2010 Proceedings IEEE INFOCOM*, pages 1–5, 2010.

[30] GeeksforGeeks. Difference between random forest and xgboost, 2023. Accessed: 2025-05-19.

[31] Yongjie Huang, Qiping Yang, Jinghui Qin, and Wushao Wen. Phishing url detection via cnn and attention-based hierarchical rnn. In *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 112–119. IEEE, 2019.

[32] Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[33] Nafiz Rifat, Mostofa Ahsan, Md Chowdhury, and Rahul Gomes. Bert against social engineering attack: Phishing text detection. In *2022 IEEE International Conference on Electro Information Technology (eIT)*, pages 1–6. IEEE, 2022.

[34] Mohammad Amaz Uddin and Iqbal H Sarker. An explainable transformer-based model for phishing email detection: A large language model approach. *arXiv preprint arXiv:2402.13871*, 2024.

[35] S Kavya and D Sumathi. Staying ahead of phishers: a review of recent advances and emerging methodologies in phishing detection. *Artificial Intelligence Review*, 58(2):50, 2024.

[36] Appsilon. Machine learning evaluation metrics for classification. `https://www.appsilon.com/post/machine-learning-evaluation-metrics-classification`, 2023. Accessed: 2025-05-06.

[37] Jason Brownlee. Failure of accuracy for imbalanced class distributions, 2019. Accessed: 2025-05-05.

[38] Glassbox Medicine. Measuring performance: Auprc (area under the precision-recall curve), 2019. Accessed: 2025-02-28.

[39] Christoph Molnar. *Interpretable Machine Learning*, chapter Interpretability. Self-published, 2022. Accessed: 2025-05-04.

[40] Christoph Molnar. *Interpretable Machine Learning*, chapter Methods Overview. Self-published, 2022. Accessed: 2025-05-04.

[41] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[42] Christoph Molnar. Lime - local interpretable model-agnostic explanations. Online book chapter in "Interpretable Machine Learning", 2022. Accessed: 2025-02-28.

[43] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[44] Christoph Molnar. *Interpretable Machine Learning*. Leanpub, 2 edition, 2022. `https://christophm.github.io/interpretable-ml-book/shapley.html`.

[45] The AI Quant. Machine learning interpretability in finance: Investigating shap and lime, 2023. Accessed: 2025-05-24.

[46] Christoph Molnar. Shap - shapley additive explanations. Online book chapter in "Interpretable Machine Learning", 2022. Accessed: 2025-02-28.

[47] Parisa Mehdi Gholampour and Rakesh M Verma. Adversarial robustness of phishing email detection models. In *Proceedings of the 9th ACM International Workshop on Security and Privacy Analytics*, pages 67–76, 2023.

[48] Maxime Fabian Veit, Oliver Wiese, Fabian Lucas Ballreich, Melanie Volkamer, Douglas Engels, and Peter Mayer. Sok: The past decade of user deception in emails and today's email clients' susceptibility to phishing techniques. *Computers & Security*, 150:104197, 2025.

[49] Jianjun Chen, Vern Paxson, and Jian Jiang. Composition kills: A case study of email sender authentication. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 2183–2199, 2020.

[50] John X Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. *arXiv preprint arXiv:2005.05909*, 2020.

[51] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025, 2020.

[52] Nicholas Boucher, Ilia Shumailov, Ross Anderson, and Nicolas Papernot. Bad characters: Imperceptible nlp attacks. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1987–2004. IEEE, 2022.

[53] Juan C. Olamendy. Tackling the challenge of imbalanced datasets: A comprehensive guide. `https://medium.com/@juanc.olamendy/tackling-the-challenge-of-imbalanced-datasets-a-comprehensive-guide-2feb11ca2fa0`, n.d. Accessed: 2025-05-11.

[54] XGBoost Developers. Xgboost parameters — xgboost 1.7.4 documentation, 2025. Accessed: 2025-05-08.

[55] Angelo Sotgiu, Maura Pintor, and Battista Biggio. Explainability-based debugging of machine learning for vulnerability discovery. In *Proceedings of the 17th international conference on availability, reliability and security*, pages 1–8, 2022.

[56] Hugging Face. Tokenizer summary, 2024. Accessed: 2025-05-27.

[57] Ashish Rao. Fine-tuning a pre-trained bert model for classification using native pytorch, 2023. Accessed: 2025-02-28.

[58] Rakesh Verma and Nirmala Rai. Phish-idetector: Message-id based automatic phishing detection. In *2015 12th International Joint Conference on e-Business and Telecommunications (ICETE)*, volume 4, pages 427–434. IEEE, 2015.

[59] J. Klensin. Simple mail transfer protocol. `https://datatracker.ietf.org/doc/html/rfc5321`, 2008. RFC 5321, IETF.

[60] Apache spamassassin: Header checks documentation.

[61] Cheng-Han Chiang and Hung-yi Lee. Are synonym substitution attacks really synonym substitution attacks? *arXiv preprint arXiv:2210.02844*, 2022.

[62] Google Search Central. Redirects and google search. Online; accessed 26-May-2025, 2024. `https://developers.google.com/search/docs/crawling-indexing/301-redirects`.

Appendix A

GitHub Repository

The code used to train and evaluate the models, as well as the scripts for running the experiments and the corresponding result files (which include all raw experimental outputs), are available in our GitHub repository: `https://github.com/Charline-meu/TFE_Phishing_Ch-Li.git`.

Certain resources—such as the fine-tuned BERT model, the *Enron* dataset and private email datasets (i.e., *Personal Advertising Emails* and *Personal Emails*)—are not publicly accessible due to confidentiality and storage limitations.

Appendix B

Additional Details on Detection Model

B.1 Tables of the Extracted Features

The following are the different features employed in the structure module and the URL module.

B.1.1 Extracted Features for the Structure Module

The extracted features from the email structures can be found in Table B.1.

Table B.1: Structural features used in the model.

Category	**Feature**
Section statistics (mandatory)	Number of text/plain sections Number of text/html sections Number of image sections Number of application sections Ratio of text/plain to any text sections Length of texts in text/html section
Header statistics	Number of standard headers Number of In-Reply-To entities Number of Received entities Existence of User-Agent Existence of X-mailer
MIME version	MIME version of the first section Case variance in MIME-Version header tag
Message-ID	Length of Message-ID ASCII number of Message-ID boundary character (if any) Existence of the domain of Message-ID in ID part Is ID part hexadecimal (except special characters) Is ID part decimal (except special characters) Existence of dots in ID part Existence of special characters in ID part

Continued from previous page

Category	Feature
	Is domain part hexadecimal (except special characters) Is domain part decimal (except special characters) Existence of dots in domain part
Mail address and domain	Number of From header tags in the mail thread Number of To header tags in the mail thread Number of unique To addresses in the mail thread Number of unique Reply-To addresses in the mail thread Number of unique domains in all email addresses Ratio of unique To domains to the unique domains in all email addresses Number of Cc addresses Number of Bcc addresses Number of Sender addresses Is Return-Path address identical to Reply-To address Is From domain same as Reply-To domain Is From entity bracketed
Section boundary	Length of the first boundary ID Does the first boundary ID start with an equal symbol (=) Existence of "=" symbol in middle of first boundary ID Existence of underscores in the first boundary ID Existence of dots in the first boundary ID Existence of other special characters in the first boundary ID Is the first boundary ID hexadecimal (except special characters) Is the first boundary ID decimal (except special characters)
Character set	Index of charset in the first section Number of unique charset in all the sections
Cascading Style Sheets (CSS)	Length of <style> bodies and inline style bodies Existence of direction: rtl style for backward rendering
JavaScript (JS)	Existence of inline <script>
Document Object Model (DOM)	Depth of DOM object tree Number of DOM leaf nodes Number of unique DOM leaf node types Mean of DOM leaf node depths Standard deviation of DOM leaf node depths
Links	Number of email addresses in text/html section Number of <a href> and <a data-saferedirecturl> tags Number of URLs in text/html section Ratio of unique domains to the domains of all URLs in any tags

B.1.2 Extracted Features for the URL Module

The extracted features from the URLs can be found in Table B.2.

Feature	Description
domain_digit_count	Number of digits in the domain name
subdomain_digit_count	Number of digits in the subdomain
path_digit_count	Number of digits in the URL path
domain_length	Length of the domain name
subdomain_length	Length of the subdomain
path_length	Length of the path in the URL
isKnownTld	Whether the TLD is known (binary)
www	Presence of 'www' (binary)
com	Presence of 'com' (binary)
punnyCode	Whether the domain uses Punycode
random_domain	Whether the domain seems randomly generated
subDomainCount	Number of subdomain levels
char_repeat	Maximum number of repeated characters
alexa1m_tld	Whether the TLD appears in Alexa top 1M
alexa1m	Whether the domain appears in Alexa top 1M
-	Count of hyphens in the URL
.	Count of dots in the URL
/	Count of slashes in the URL
@	Count of '@' symbols in the URL
?	Count of question marks in the URL
&	Count of '&' symbols in the URL
=	Count of '=' symbols in the URL
_	Count of underscores in the URL
domain_in_brand_list	Whether the domain matches a known brand
raw_word_count	Number of words extracted from raw tokens
splitted_word_count	Number of words obtained by splitting compound words
average_word_length	Average length of all extracted words
longest_word_length	Length of the longest word
shortest_word_length	Length of the shortest word
std_word_length	Standard deviation of word lengths
compound_word_count	Number of compound words found
keyword_count	Number of detected keywords
brand_name_count	Number of detected brand names
negligible_word_count	Number of detected negligible/common words
target_brand_count	Number of targeted brands found
target_keyword_count	Number of targeted keywords found
similar_keyword_count	Number of words similar to known keywords
similar_brand_count	Number of words similar to known brand names
average_compound_words	Average length of compound words
random_words	Number of randomly generated words detected

Table B.2: Description of the 40 handcrafted URL features used for phishing detection.

B.2 Limitations of Detection Model

You can find here a detailed reflection on the overfitting observed in our structure module.

B.2.1 Observations about the Structure Module

At the beginning of our experiments, before composing the final dataset, we tested different dataset compositions. During this process, we realized that using datasets from diverse sources is crucial to prevent overfitting. The various dataset compositions we conducted are detailed below.

The process begins with features extraction from the selected datasets, followed by training and testing the XGBoost model on these data. The training set used is always balanced, while the test set contains 75% legitimate instances and 25% phishing instances. Next, a SHAP explainer is generated based on the model and training data. For each dataset composition, two SHAP graphs are generated: a beeswarm plot and a bar plot.

B.2.1.1 Dataset Composition 1

The first tested dataset is composed of emails from the **Enron** and **Nigerian** datasets. As shown in Table B.3, the model trained on this dataset is completely overfitted, as all the evaluation metrics reach a perfect score of 1.

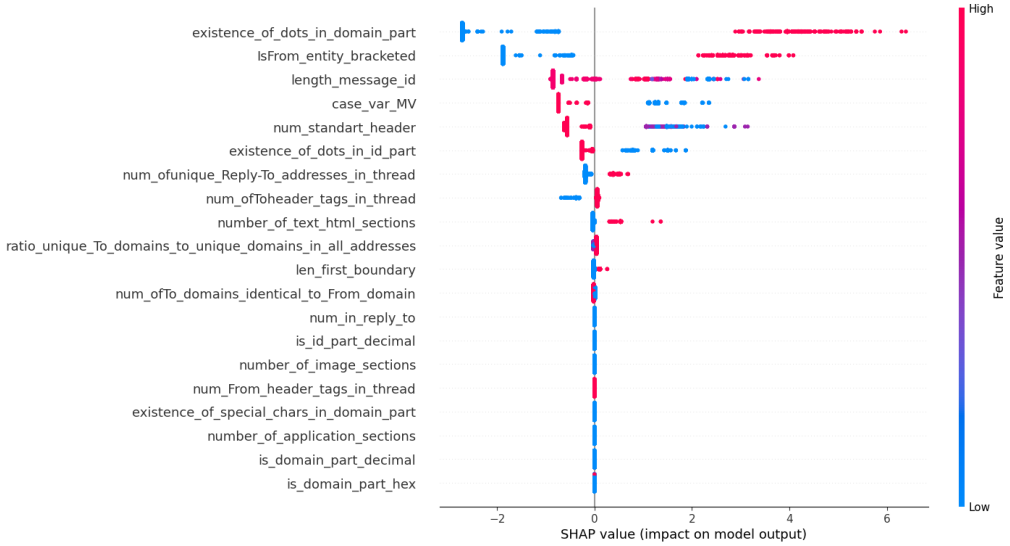
Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	1.00	1.00	1.00	1,326
1	1.00	1.00	1.00	442
Accuracy	1.00			1,768

Table B.3: Classification report using XGBoost on Enron and Nigerian datasets.

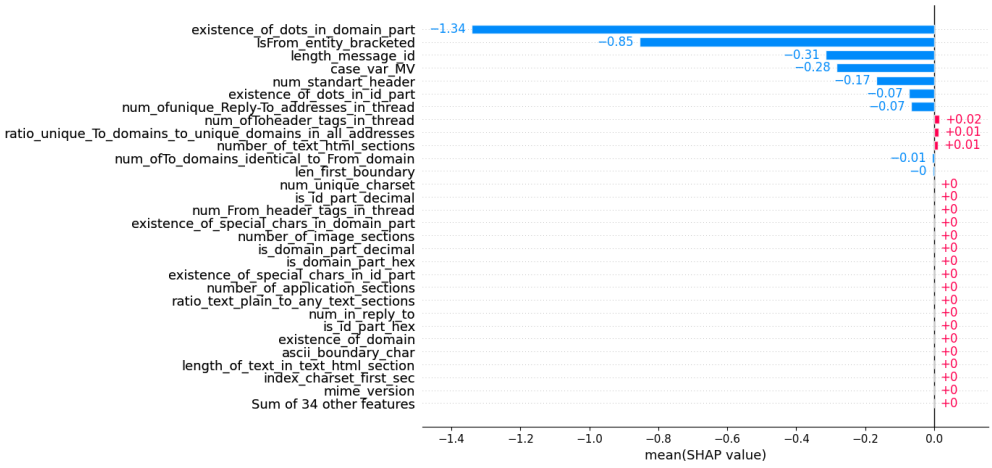
SHAP analysis helps interpret these results. In Figure B.1a, the email classified as legitimate—with a negative SHAP value—exhibits highly recognizable features: a low number of dots in the domain part of the `Message-ID`, no `bracketed From` entity, inconsistent `MIME-version` format ("Mime-Version" instead of "MIME-Version"), and the presence of dots in the ID part of the `Message-ID`. A SHAP value of 0 means that a feature has no influence on the decision of the model, neither contributing to phishing nor legitimate classification. This is the case for several features, such as the number of `In-Reply-To` headers or the number of `image sections`.

In the bar plot (Figure B.1b), features with a mean SHAP value close to 0 have no overall effect on the output of the model. We observe that only features related to legitimate structures have a strong impact, indicating that the model relies heavily on those characteristics to detect legitimate emails. However, this also highlights a potential issue: the model may rely excessively on these features, making it less effective in more diverse, real-world scenarios. To address this, we decided to expand the dataset by including SpamAssassin to diversify legitimate and phishing structures, and Nazario to increase the variety of phishing examples.

The goal is to reduce the dominance of overly specific patterns and build a more balanced, generalizable model that performs well across realistic and diverse datasets.



(a) Beeswarm plot for the XGBoost model of the dataset composed of Enron and Nigerian.



(b) Bar plot for the XGBoost model of the dataset composed of Enron and Nigerian.

Figure B.1: SHAP plots for the XGBoost model of the Enron and Nigerian dataset.

B.2.1.2 Dataset Composition 2

The second dataset includes emails from the **Enron**, **Nigerian**, **SpamAssassin**, and **Nazario** datasets. As shown in Table B.4, the model still exhibits signs of overfitting.

The updated dataset has only a minor effect on the recall and F1-score for phishing detection, and no noticeable impact on the recall for legitimate emails.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	1.00	1.00	1.00	2,952
1	1.00	0.99	0.99	984
Accuracy	1.00			3,936

Table B.4: Classification report using XGBoost on dataset composition.

In Figure B.2a, we observe a broader distribution of SHAP values, showing that more features contribute to the decision of the model. This suggests that increasing data diversity helps the model learn more complex internal patterns and consider a wider range of features, resulting in more balanced and robust predictions.

Figure B.2b shows that certain features, like the existence of dots in the domain part `Message-ID`, have gained importance for legitimate classification, while others, such as `case var MV`, have lost influence. We also note an increase in features that support phishing detection, indicating that the model now better balances its decision-making by relying on a more diverse set of features.

These shifts confirm that introducing more varied datasets improves the model by increasing feature diversity. This positive outcome encouraged us to further extend the dataset by adding recent emails from the Personal Advertising Emails and Personal Emails datasets, aiming to further enhance robustness.

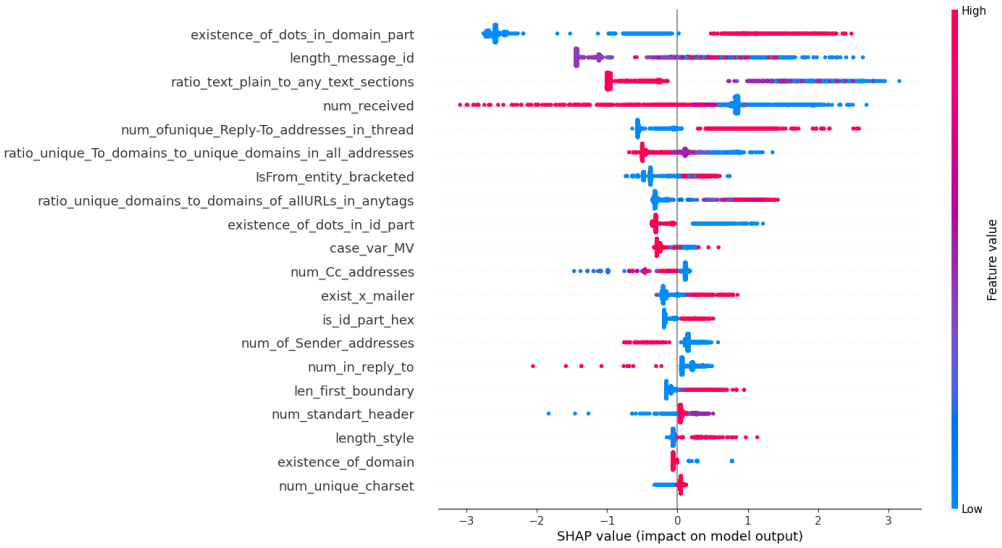
B.2.1.3 Dataset Composition 3

This final dataset used for the structure model, presented in Section 3.2.1, combines emails from the **Enron**, **SpamAssassin**, **Nazario**, **Nigerian**, and recent datasets such as **Personal Advertising Emails** and **Personal Emails**.

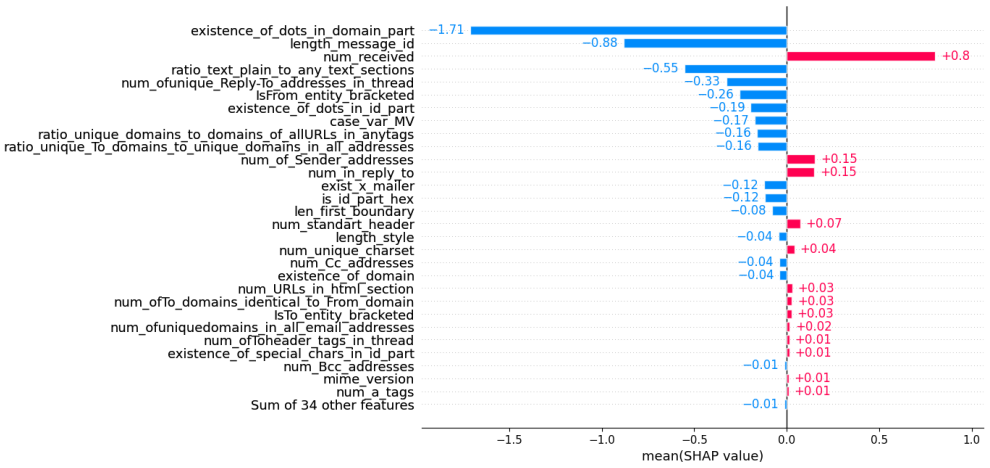
The classification report presented in Table 3.2, in Section 3.2.1, displays the values of the various evaluation metrics. These metrics indicate no clear signs of overfitting.

Concerning the impact of the features, we observe a noticeable drop in the ones associated with legitimate emails. The addition of more recent and diverse data reduces the reliance of the model on dominant characteristics and increases the variability of feature importance—key elements for improving robustness and generalization.

The shift observed in the impact of the features, discussed in Section 3.3, becomes evident when examining how the influence of specific features changes after data augmentation. For example, the SHAP value for `Message-ID` length moves from a negative



(a) Beeswarm plot for the XGBoost model of the dataset composed of Enron, SpamAssassin, Nigerian and Nazario.



(b) Bar plot for the XGBoost model of the dataset composed of Enron, SpamAssassin, Nigerian and Nazario.

Figure B.2: SHAP plots for the XGBoost model of the dataset composed of Enron, SpamAssassin, Nigerian and Nazario.

influence of -0.88 to a slightly positive value of +0.04, suggesting that long identifiers—once seen as suspicious—may now be common in legitimate emails. Similarly, the impact of the number of `Received` headers drops from +0.80 to +0.02, and the influence of dots in the domain part of the `Message-ID` decreases from -1.71 to -0.40.

These changes are illustrated in Figure B.2b and Figure 4.2b (Section 4.1.2).

Based on SHAP analysis, we conclude that the final structure model does not show significant overfitting, as it draws on a more balanced and varied set of features in its decision-making process.

Appendix C

Additional Details on XAI-Guided Adversarial Attacks

C.1 Experiments on the Text Module

In this appendix, we will briefly describe the emails used and the prompts employed to guide the attacks for the text-based experiments.

C.1.1 Description of Email Selection

We randomly selected a set of 10 emails, which remained the same across all experiments. Each email was assigned an unique identifier from 1 to 10. Here you have a short description of the content of the emails:

- **Email 1**: An email claiming that incoming messages were blocked due to verification failure, prompting the user to act.
- **Email 2**: A message from someone posing as the son of a former president, seeking partnership for transferring funds.
- **Email 3**: An unsolicited offer promoting bulk email marketing services as a way to gain exposure.
- **Email 4**: A fake notification from LinkedIn claiming a file is shared and urging the recipient to view it.
- **Email 5**: A formal letter from a Nigerian board member proposing a confidential business opportunity.
- **Email 6**: An update from a supposed partner claiming success in a fund transfer, now proposing investments.
- **Email 7**: An email from a political dissident in Zimbabwe asking for assistance and offering a share of hidden funds.

- **Email 8**: A message from individuals in Côte d'Ivoire proposing a business relationship and requesting contact.
- **Email 9**: A fake security alert urging the recipient to click a link to view a secure message.
- **Email 10**: An email from a fake bank representative in the UK offering access to a dormant foreign account.

The full content of these emails is available in the repository referenced in Appendix A.

C.1.2 Prompts Provided to ChatGPT for Email Rewriting

This appendix presents the prompts used in our experiments to guide ChatGPT in rewriting phishing emails.

C.1.2.1 Initial Robustness Test Against TextFooler and ChatGPT

LLM-Based Rewriting *I am gonna give you phishing emails, and I want you to make them look legitimate for the model, but they still have to remain phishing emails. You need to make them look as legitimate as possible in the eyes of the model. I want to test the robustness of my model. Try to stay as close as possible to the original structure and flow. I will give you input that represents the content of the email. Do not bother changing the URL—my model does not take the URL into account—and do not bother adding emojis; same thing for emojis. Can you also always tell me why this looks more legitimate?*

Second Iteration *Can you make all the emails that I will give you appear even more legitimate.*

C.1.2.2 LIME Guided Adversarial Rewriting via LLMs

First Iteration *Ok I am going to give you a phishing email (from a public dataset). I applied an explainable AI on this email to get to know what parts of the email have the most influence on the decision-making, i.e., which words in the email influence the classification to phishing email or legitimate email. I will give you emails and the output of LIME (which is an explainable AI that tends to explain the decision-making of the model based on the input). I will give you the LIME output and I want you to replace all the words that push the model to predict phishing by some other word and to keep all the words that push the model towards legitimate classification. Try to stay as close as possible to the original email.*

Second Iteration *Ok, I have applied LIME once again on every email you just generated because they keep being detected as phishing emails by my model. We will do the same all over again. I will give you the email you generated and the LIME analysis on this email, and you will do the same as before, right?*

Third Iteration *Excellent! Let's do a third round.*

C.1.2.3 SHAP Guided Adversarial Rewriting via LLMs

Prompts Used Across Both Rounds *I have created an algorithm capable of detecting whether an email is a phishing email or not. I will give you 10 phishing emails. Your goal is to try to make them appear legitimate to the model (while keeping their phishing intent) using the help of explainable AI, specifically SHAP. SHAP has provided, for each email, the 20 features that contribute most to the phishing classification according to the model. SHAP has also given the features that, conversely, push the model to classify the email as legitimate. I will give you 10 phishing emails and, using the data provided by SHAP, I want you to modify each email by removing/replacing the words that are characteristic of phishing. I want you to absolutely keep the words that are characteristic of legitimate emails. The idea is to stay relatively close to the original email, while applying the modifications suggested by SHAP. Do you understand? One small note: all emails are in English.*

C.1.2.4 SHAP Guided Rewriting via Global Feature Importance

First Round *I have created an algorithm capable of detecting whether an email is a phishing email or not. I will give you 10 phishing emails. Your goal is to try to make them appear legitimate to the model (while keeping their phishing intent) using the help of explainable AI, specifically SHAP. SHAP has provided the 50 most influential features based on 500 phishing emails. I will give you 10 phishing emails and, using the data provided by SHAP, I want you to modify each email by removing/replacing the words that are characteristic of phishing (and keeping the words that are characteristic of legitimate emails). The idea is to stay relatively close to the original email, while applying the modifications suggested by SHAP. Do you understand? I will proceed by giving you the 50 top features. You should keep them in mind for the 10 next emails, I will give them to you at every iteration! You should remove every word from the top 50 list if it pushes toward phishing, right?*

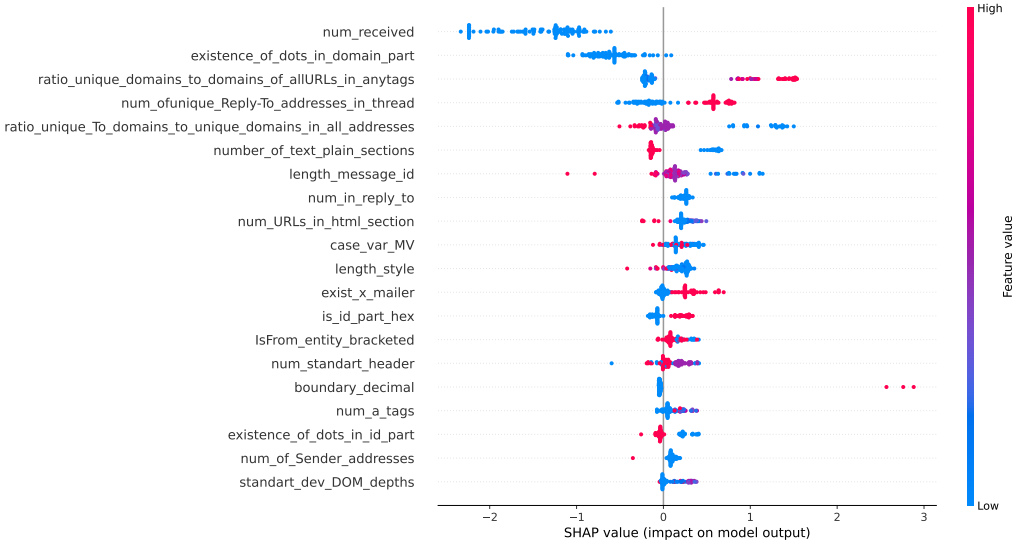
Second Round *SHAP also provided the 50 most influential features, based on 500 legitimate emails. Can you help me making them look even more legitimate? I am gonna give you the emails that were still detected as phishing by the model.*

C.2 Final Figures

This section presents the SHAP-generated graphs illustrating the final impact of each feature on the models applied to the structure and URL modules

C.2.1 Graphs of the Structure Experiments

Figure C.1 shows the final graphs obtained after manipulating the two selected features, `Received` and `Message-ID`, across the 100 randomly chosen structures.



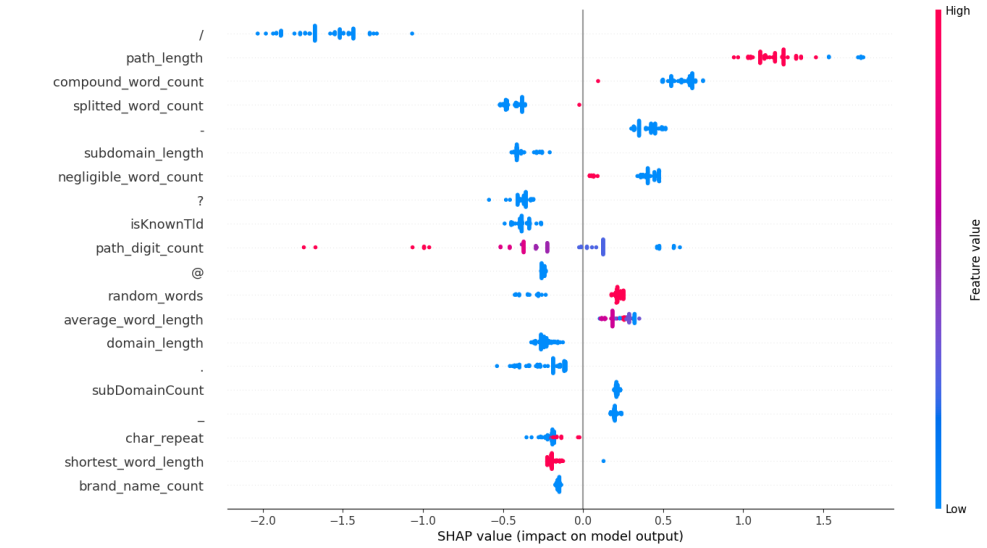
(a) Beeswarm plot of the modified `Message-ID` domain structures and 7 `Received` headers.

(b) Bar plot of the modified `Message-ID` domain structures and 7 `Received` headers.

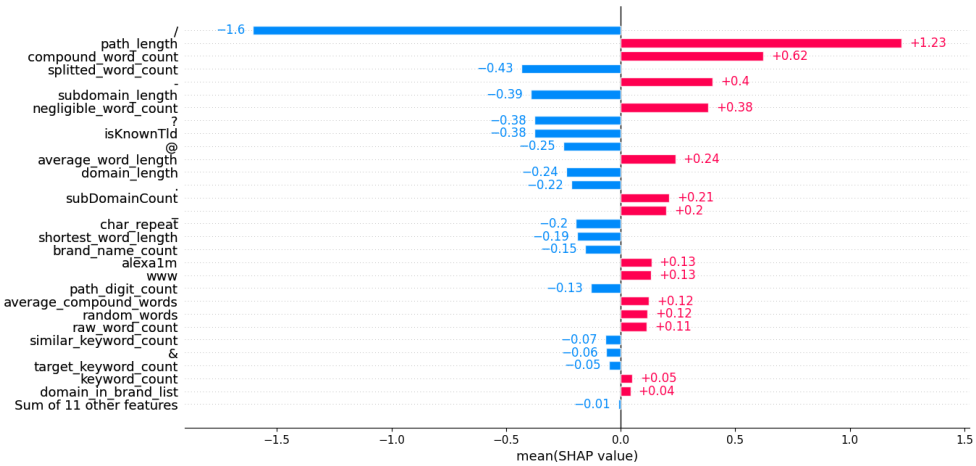
Figure C.1: SHAP plots of the modified `Message-ID` domain structures and 7 `Received` headers.

C.2.2 Graphs of the URL Shortener Attack with a Custom Domain

Figures C.2 present the final graphs for the shortened URLs attack using a custom domain.



(a) Beeswarm plot of the URL classification model of the selected shortened URLs with a customized domain.

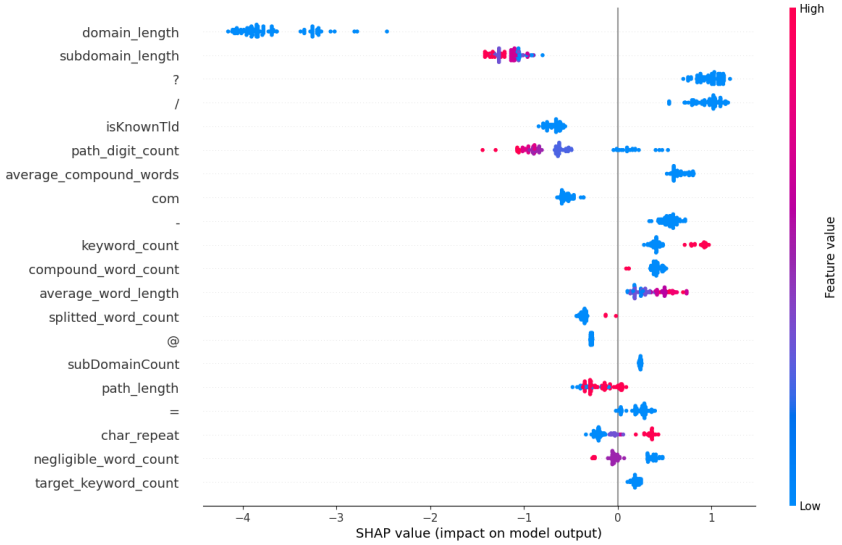


(b) Bar plot of the URL classification model of the selected shortened URLs with a customized domain.

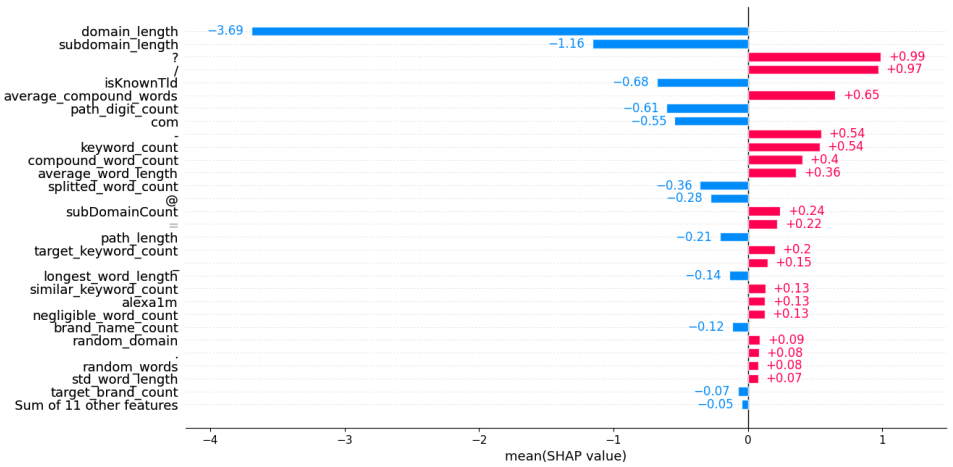
Figure C.2: SHAP plots of the URL classification model of the selected shortened URLs with a customized domain.

C.2.3 Graphs of the Third Redirect Attack with the Reformat

These are the final graphs obtained after shortening and concatenating the URLs using the `url=?` format, represented in Figure C.3.



(a) Beeswarm plot of the URLs classification model on the 100 tested URLs for the third redirection attack.



(b) Bar plot of the URLs classification model on the 100 tested URLs for the third redirection attack.

Figure C.3: SHAP plots of the URLs classification model on the 100 tested URLs for the third redirection attack.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**