International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 06 June 2025, pp: 99-119 www.ijaem.net ISSN: 2395-5252

Enabling Safer Web Browsing Using a Machine Learning Framework for User-Centric Malicious URL Detection

Mukhtar Dahiru[1*], Aliyu Umar[1], Ibrahim Muktar[1], Umar A. Maina[1], Mustapha Abdulkadir[1], Yakubu Magaji[2]

[1]*Department of Computer Science, Jigawa State Polytechnic for Information and Communication Technology (INFORMATICS) Kazaure. Jigawa State, Nigeria*

[2]*Department of Computer Science, Jigawa State College of Education, Gumel, Jigawa State, Nigeria*

Corresponding Author

Date of Submission: 28-05-2025 Date of Acceptance: 08-06-2025

ABSTRACT:
The escalating prevalence of malicious URLs which enable phishing, malware downloads, and other web-borne attacks, poses a critical security challenge. Traditional defenses such as blacklists and heuristic filters are reactive and often fail to recognize novel or targeted threats in real time. In this paper, we propose a user-centric malicious URL detection framework that leverages machine learning to analyze URLs before a user clicks. Our approach combines lexical features of URL strings with host-based and content-inspired attributes to power a lightweight classifier evaluated with logistic regression and Support Vector Machines integrated directly into the browsing experience. Experimental results on a dataset of 30,000 labeled URLs demonstrate high efficacy: the best model (logistic regression) achieves about 97% accuracy and an F1-score of 0.97 in distinguishing malicious from benign links. The framework operates in real-time with minimal user friction, providing a proactive layer of defense that significantly improves web safety without relying solely on known threat lists.

Keywords: Malicious URL detection, Machine Learning, phishing prevention, cybersecurity, lexical and host-based features

I. INTRODUCTION:

Cybercriminals are creating and deploying malicious URLs at an alarming rate. In 2010, phishing attacks worldwide numbered only on the order of a few hundred thousand per year; by 2023, nearly nine million phishing attacks were detected globally (Dahiru M. et al 2020). This dramatic rise underscores the urgent need for more effective web security measures. Malicious URLs serve as gateways to phishing websites, malware payloads, and other exploits that can lead to financial loss, identity theft, and organizational data breaches. Despite increased awareness, users and organizations continue to fall victim to these attacks, which are growing in sophistication and frequency. Recent industry reports highlight that phishing remains one of the most prevalent cyber threats, with new phishing sites and campaigns numbering in the millions annually (Statista, 2024).

Conventional anti-phishing and URL security solutions have notable limitations. Browser-integrated blacklists and reputation services (e.g., Google Safe Browsing and PhishTank) provide a first line of defence by blocking known malicious links, but they operate reactively. These systems rely on databases of reported URLs and often lag in detecting fresh, unknown threats – studies have found that many phishing sites go unlisted for hours or days, leaving users exposed in the interim (Oest et al., 2019). Heuristic-based filters can identify suspicious URLs by checking for common attack patterns (such as URLs with long random strings, "@" symbols, or misleading domain names). While heuristics may catch some new attacks, adversaries can easily obfuscate or alter URLs to evade rigid rules. Heuristic systems also risk false positives, erroneously flagging benign websites and eroding user trust in security warnings. Meanwhile, most network-level defenses (firewalls, intrusion detection systems) inspect traffic at aggregate levels and may not incorporate the user's context or real-time browsing intent, reducing their effectiveness against targeted phishing links

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 06 June 2025, pp: 99-119 www.ijaem.net ISSN: 2395-5252

delivered via email or social media. In summary, existing approaches struggle to be both timely and accurate against the evolving landscape of malicious URLs.

This research addresses the above gap by introducing a user-centric, machine learning-based framework for malicious URL detection that operates in real time within the user's own browser or device. By analysing the URL before the page is loaded, our system can proactively warn or block dangerous links, including previously unseen ("zero-day") phishing URLs that would bypass traditional blacklists. Furthermore, by leveraging a learning-based model, the framework can recognize subtle patterns across a wide range of features rather than relying on brittle rules. We aim to bridge the research gap of integrating robust URL classification into the user's browsing experience providing immediate, personalized protection has been lacking in prior work that largely evaluates detection in offline settings or at the network perimeter.

In summary, this paper makes the following key contributions:

- **Novel User-Centric Detection Framework:** We design a browser-integrated malicious URL detection system that empowers end-users with real-time warnings, moving defense from the network edge directly to the user's browsing environment. This approach emphasizes usability and instant feedback during web surfing.
- **Comprehensive Feature Engineering:** We identify and engineer a rich set of features for URL classification, including lexical features (URL string characteristics), host-based features (domain registration and hosting information), and limited page content cues. This multi-faceted feature set improves detection coverage compared to using only one type of indicator.
- **Machine Learning Model Evaluation:** We develop and evaluate several machine learning models (in particular, logistic regression and Support Vector Machine classifiers) on a large, up-to-date dataset of benign and malicious URLs. The best model achieves **≈97% accuracy** with high precision and recall, outperforming baseline methods and demonstrating the effectiveness of our feature set.
- **Real-Time Performance and Integration:** We implement the approach with efficiency in mind, discussing strategies (such as fast lexical parsing and asynchronous checks) to ensure that security checks keep up with user browsing without noticeable lag. The feasibility of deploying the model as a lightweight browser extension is analyzed, showing that robust protection can be provided within acceptable performance overhead.
- **Adaptability and Continuous Learning:** We outline mechanisms for the framework to continuously learn from new data and user feedback. The system's design can incorporate model updates or online learning to adapt to emerging phishing tactics and adversarial evasion, paving the way for a more **adaptive security solution** against malicious URLs.

By combining insights from prior blacklist, heuristic, and machine learning approaches, our work demonstrates a practical and effective solution that significantly enhances user protection against malicious URLs in real time.

1.1 Statement of the Problem:

Despite the availability of various security tools, users remain vulnerable to attacks initiated through malicious URLs. The limitations of existing security measures include:

1. **Reactive Nature:** Blacklists are only effective against previously identified threats, leaving users susceptible to zero-day attacks and newly registered malicious domains.
2. **Network-Level Limitations:** Network-based solutions may not have the granular context of individual user interactions and might be bypassed through techniques like HTTPS encryption.
3. **False Positives/Negatives:** Heuristic-based systems can sometimes flag legitimate URLs as malicious (false positives) or fail to detect sophisticated malicious URLs (false negatives), leading to either user inconvenience or security breaches.
4. **Lack of User Empowerment:** Users often lack the technical expertise to discern malicious URLs, making them reliant on automated systems that may not always be foolproof.

Therefore, there is a pressing need for a more proactive, intelligent, and user-centric approach to malicious URL detection that can provide timely and accurate warnings directly within the user's browsing context.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 06 June 2025, pp: 99-119 www.ijaem.net ISSN: 2395-5252

1.2 Aim and Objectives:

1.2.1 Aim

The primary aim of this research is to develop a user-centric machine learning framework for enabling safer web browsing by providing real-time detection of malicious URLs.

1.2.2 Objectives

To achieve this aim, the following objectives will be followed:

1. To identify and engineer a comprehensive set of relevant features from URL strings, host information that effectively differentiate between kind and malicious URLs.
2. To evaluate the performance of various machine learning algorithms (e.g. Logistic Regression, Support Vector Machine, Multinomial Naive Bayes (MNB) Classifierin accurately classifying URLs as malicious or clean based on the extracted features.
3. To design a user-centric integration strategy using a machine learning algorithms that can check if URLs is malicious or clean as you go that minimizes performance overhead and maximizes user experience.
4. To assess the effectiveness and efficiency of the proposed framework through rigorous evaluation using a diverse and up-to-date dataset of friendly and malicious URLs, considering metrics such as accuracy, precision, recall, true positive and false positive rate, and detection latency.
5. To explore mechanisms for continuous learning and potential strategies for enhancing the framework's robustness against adversarial attacks.

1.3 Significance of the Study

This research holds significant potential for enhancing online security and empowering individual users. The development of a user-centric machine learning framework for malicious URL detection offers several key benefits:

1. **Proactive Protection:** By analyzing URLs in real-time before a user interacts with them, the framework can prevent users from accessing malicious content, reducing the risk of attacks.
2. **Enhanced Accuracy:** Machine learning models have the capacity to learn complex patterns and identify subtle indicators of maliciousness that may be missed by traditional rule-based systems or blacklists.
3. **Personalized Security:** Integrating the framework within the user's browsing environment allows for a more personalized and immediate layer of security.
4. **Reduced Reliance on Reactive Measures:** By focusing on predictive analysis, the framework can offer protection against novel and emerging threats.
5. **User Empowerment:** Providing users with timely warnings and potentially explanations can educate them about online threats and foster safer browsing habits.
6. **Contribution to Cybersecurity:** This research contributes to the broader field of cybersecurity by exploring innovative applications of machine learning for threat detection and user protection.

II. LITERATURE REVIEW:

Detection of malicious URLs has been an active research area for over a decade, yielding a variety of approaches. Early studies focused on lexical analysis of URL strings, followed by incorporation of host-based attributes (e.g. domain information), and later, analysis of page content. Concurrently, the use of URL reputation services (blacklists/whitelists) has remained a prevalent industry practice. In this section, we review representative peer-reviewed studies from 2010 to 2025 under each of these categories, highlighting their methodologies, strengths, and limitations. We also note emerging trends such as the adoption of machine learning, ensemble methods, and deep learning for URL detection. This review will clarify how our proposed framework builds on prior knowledge and addresses identified gaps.

2.1 Lexical Features

Lexical feature analysis examines the URL text itself (characters and tokens) for patterns indicative of malicious intent. This approach gained prominence due to its speed and independence from external data: a URL can be analyzed pre-click without fetching the web page. Blum et al. (2010) pioneered the use of lexical features with machine learning for phishing detection. They developed an online classifier using confidence-weighted learning to distinguish phishing URLs based on features like URL length, token composition, and the presence of suspicious substrings (Blum et al., 2010). A key strength of their system was the ability to detect “zero-hour” phishing domains as they emerged, unlike blacklists that only flag already-reported URLs. Their evaluation showed high accuracy in catching new phishing URLs by learning lexical patterns; however, the approach could be vulnerable to novel

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 06 June 2025, pp: 99-119 www.ijaem.net ISSN: 2395-5252

obfuscation tactics that significantly deviate from previously seen patterns.

Subsequent research expanded on lexical classifiers by incorporating larger datasets and more advanced algorithms. Ma et al. (2011) presented a seminal study using lexical and other features to train a logistic regression and online perceptron model for malicious URL detection. They collected millions of URLs, spanning benign/malicious and demonstrated that a learning-based approach can achieve high detection rates over 99% AUCwhile maintaining low false positives (Ma et al., 2011). The strength of this work lies in its scale and the robust performance of statistical learning even on noisy, real-world URL data. A limitation, however, is that the models required frequent retraining to maintain effectiveness as attackers adapt; the authors noted the importance of continuous learning to handle evolving URL patterns.

Researchers have also explored efficiency improvements in lexical analysis. Verma and Das (2017) introduced a fast feature extraction framework for URL strings, aiming to enable real-time classification even in resource-constrained environments. They implemented optimized algorithms to calculate lexical features (such as n-gram frequencies and entropy measures) with minimal overhead (Verma & Das, 2017). Their evaluation using a combination of decision trees and support vector machines showed that one can achieve comparable accuracy to earlier methods at a fraction of the computational cost. The trade-off noted was that focusing solely on URL string features might miss contextual clues; thus, while the system was extremely quick; suitable for browser integration, its standalone detection capability could be enhanced by additional feature types.

In recent years, the lexical approach has been augmented by natural language processing and deep learning techniques. Instead of manually defining features, some works use character-level or word-level embeddings of the URL string. For example, Sahingoz et al. (2019) combined 39 lexical features e.g. word tokens derived from URL segments and applied various classifiers, reporting that ensemble methods like Random Forest achieved over 99% accuracy in their experiments. The strength of their approach was the comprehensive coverage of lexical indicators including the use of Natural Language Processing to detect misleading brand names in URLs and strong classifier performance. However, the extremely high accuracy reported may reflect an optimized evaluation on a particular dataset; a potential limitation is generalization, as attackers can still create novel URL strings that evade these specific lexical cues.

Deep learning models have also been proposed to automatically learn URL representations. The introduced URL Net, a convolutional neural network that takes the raw URL string as input at character and word level to learn relevant patterns without explicit feature engineering. This approach was shown to capture semantic subtleties and sequential character patterns that earlier bag-of-words lexical models could miss (Le et al., 2018). By jointly optimizing on characters and tokens, the model could, for instance, learn that “secure-login” appearing in a suspicious context is malicious, even if that exact token was not seen in training(Ahasan et al., 2023). The advantage of deep learning here is improved detection of cleverly obfuscated URLs and the ability to generalize from similar character sequences. Nonetheless, a noted limitation is the computational cost and complexity – deep models require more data and processing power, which can be a challenge for real-time use in a browser. Additionally, interpretability suffers unlike simple lexical rules, a neural network’s decision on a URL is opaque, which can hinder user trust if the system flags a URL but cannot explain the rationale clearly.

Lexical URL analysis has proven to be a fast and effective first line of defense. Studies consistently find that malicious URLs often have distinguishing lexical characteristics such as abnormal length, special characters, suspicious substrings or misspellings of brandsthat can be picked up by algorithms. The trend shows progression from simple rule-based checks to sophisticated machine learning and deep learning models operating on URL strings. The main limitation of purely lexical methods is their susceptibility to crafty obfuscation attackers continuously tweak URL text to appear benign i.e. using homoglyphs or subdomain tricks. This necessitates constant model updates or the inclusion of additional evidence beyond the URL string. These observations motivate our framework to use lexical analysis in conjunction with host and content features, thereby improving resilience against evasive URLs that might slip past lexical-only filters.

2.2. Host-based Features

Host-based features refer to properties of the server or domain hosting the URL, such as

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 06 June 2025, pp: 99-119 www.ijaem.net ISSN: 2395-5252

domain name characteristics, registration details, hosting infrastructure, and reputation of the IP address or network(Saadi et al., 2024). These features capture the notion that legitimate websites often differ from malicious ones in their provenance and hosting patterns. Research by Feroz and Mengel (2015) highlighted the value of host-based metrics by introducing a "URL ranking" approach. In their method, URLs were clustered and ranked using attributes like domain age (i.e. how long the domain has existed), domain popularity (e.g., Alexa rank)(Kevin Muldoon, n.d.), and the presence of the URL on trusted versus blocklisted domains (Feroz & Mengel, 2015). Their classifier would treat URLs with recently created domains or hosted in high-risk network blocks as more likely malicious. This approach showed strong results in automatically classifying phishing URLs, leveraging the intuition that phishing sites are frequently hosted on newly registered or rarely visited domains(Palaniappan et al., 2020). A strength of their work is the relatively low false-positive rate by incorporating global reputation measures like popularity, the system avoids flagging well-known domains even if the URL string looks somewhat suspicious. However, a limitation is coverage: purely host-based methods may struggle with compromised legitimate domains. For instance, if a phishing page is hosted on a long-established but hacked website, its domain age and reputation might appear benign, leading to a missed detection.

Another influential study (Mohammad et al., 2014) developed a rule-based classifier for phishing websites using a rich set of features, many of which were host-based. They looked at indicators such as whether the URL uses an IP address instead of a domain name as phishers sometimes directly use IPs to hide the true host, the existence of "@" in URLs which can obscure the real domain, the length of the domain name, and WHOIS information like the domain's registration country and date. Their self-structuring neural network model achieved high accuracy by learning which combination of these features correlate with phishing. For example, they found that phishing URLs often have very short lifespan domains the WHOIS "creation date" being only days or weeks old and often use free hosting or URL shortening services. The strength of this work is the interpretability and domain knowledge it encodes features like "Age of Domain < 6 months" or "No SSL certificate"are intuitive red flags. A downside is that some of these features can become outdated or less discriminative over time; phishers adapt, some now deploy phishing on longer-lived domains or use valid HTTPS certificates so a static rule set might need frequent revision. Additionally, relying on WHOIS and external lookups can introduce latency, a challenge for real-time deployment and sometimes data availability issues like the GDPR privacy that has redacted some WHOIS info in recent years.

Host-based analysis has evolved with the trend of combining it with machine learning and other feature categories(Khormali et al., 2021). Many modern systems use host features alongside lexical ones in ensemble models. For example, the comprehensive study by Sahingoz et al. (2019) also included host-based factors in their feature set, such as whether the URL's domain was an IP address and the DNS record properties. They found that incorporating these host-based features alongside lexical features improved overall detection performance, as evidenced by their best models catching more phishing instances with few false alarms. The strength here is in catching cases that lexical features alone might miss such as URL that looks superficially normal but is hosted on an IP or a suspicious server can be identified via host intelligence. One limitation to note is the dynamic nature of host data: IP addresses and hosting providers used by attackers often change quickly via fast-flux networks, bulletproof hosting etc. A classifier trained on host features from one time might degrade as the threat actors migrate to new infrastructure. This calls for continuous updating of the knowledge base or retraining of the model with fresh threat intelligence.

Host characteristics provide an important complementary view to lexical analysis. Legitimate sites typically have stable, well-established domains and reside on reputable servers, whereas malicious sites often exploit new, transient, or untrusted infrastructure. Studies in 2010–2025 show that using features like domain age, registration details, and IP reputation can significantly boost detection accuracy and reduce false positives(Chauhan & Panda, 2015). The main challenges involve data access and timeliness: querying external services for WHOIS, DNS, or reputation data can introduce latency, which is problematic for in-browser real-time scanning. Moreover, some host features can be circumvented – for instance, adversaries can compromise aging domains to defeat "new domain" checks or distribute attacks across cloud providers to mask their footprint. These limitations motivate our approach to combine host-based signals with other feature types in a machine learning model, and to

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 06 June 2025, pp: 99-119 www.ijaem.net ISSN: 2395-5252

design the system such that essential host lookups such as checking if domain is in a known blacklist or whitelistare done efficiently possibly offline or in background. By doing so, we aim to reap the benefits of host-based detection while mitigating performance impacts, thereby enhancing overall reliability of the user-centric URL classifier.

2.3. Content-based Feature

This approachinvolves analyzing the actual web page content that a URL leads such as the HTML, text, images, and scripts to determine if the page is malicious. This can be very powerful because it inspects what the user would see or execute, rather than just the URL string or metadata(McGahagan et al., 2019). However, fetching and examining content poses challenges for speed and safety, which has led researchers to use content features in a limited or innovative way. One early example is "CANTINA" by Zhang et al. (2007), which introduced using TF-IDF of page text to detect phishing essentially checking if the page's textual content (e.g., frequent words) matched the purported target brand (a mismatch could indicate a fake page). Building on this idea, Dunlop et al. (2010) developed GoldPhish, a system that uses image analysis of rendered pages. GoldPhish captures a screenshot of the webpage and applies Optical Character Recognition (OCR) to extract the text, then uses the Google PageRank or search results to see if that text (e.g., a bank name) is legitimately associated with the domain (Dunlop, Groat, & Shelly, 2010). In their tests on 100 phishing and 100 legitimate sites, GoldPhish achieved a 98% phishing detection rate with zero false positives for the legitimate set – an impressive outcome demonstrating the effectiveness of content-based verification (Dunlop et al., 2010). The strength of this approach is its robustness: even if a phishing URL looks innocuous lexically or host-wise, the content reveals the deception (for instance, the page might contain known login form patterns or brand logos that betray it as a phishing page). The obvious limitation is performance and complexity: rendering each page and doing OCR/image analysis is computationally heavy, unsuitable for a quick real-time check before page load. GoldPhish mitigated this by functioning as a browser plugin that only triggers on suspicious sites, but such an approach may still introduce noticeable delay. Additionally, content-based methods must run in a secure sandbox to avoid infecting the user's machine if the URL is indeed malicious since you are intentionally loading the content to analyze it.

Another content analysis strategy was implemented in a large-scale system developed at Google to automate phishing page classification. Their system combined URL features with content features like the presence of login forms, known phishing page HTML signatures, and other page layout indicators to train a classifier (Whittaker, Ryner, & Nazif, 2010). This classifier was used to maintain Google's phishing blacklist in an automated fashion. A notable result from their work was that the machine learning model could correctly identify over 90% of phishing pages even weeks after training, indicating strong generalization, and dramatically reducing reliance on slow human verification (Whittaker et al., 2010). The strength here is the comprehensiveness of features: by examining the page DOM and elements, it caught sophisticated phishing that might not be evident from the URL alone. Moreover, operating at Google's scale, the system proved that content checks can be done in near-real-time by leveraging cloud resources (processing millions of pages per day). A limitation, however, is that this approach works best in a centralized setting (like Google's servers) rather than on an end-user's device. Attempting to replicate similar deep content analysis in a client-side environment would likely be impractical. Additionally, content-based classifiers can be vulnerable to adversarial tricks like cloaking serving benign-looking content to scanners while showing malicious content to normal users since attackers often use this to evade detection by security crawlers.

Some researches has tried to find a middle ground, using partial content analysis to avoid full-page fetch. For example, Abbasi et al. (2015) explored email-based phishing detection by analyzing the rendered content of URLs in emails for inconsistencies (though focused on emails, the principle extends to web content). Other works have looked at specific content features such as the presence of SSL certificate details on the page versus the URL or checking if the favicon (site icon) is loaded from a different domain often a phishing giveaway. These targeted content features can add value without incurring the cost of full-page examination. The trend in the 2020s also includes machine vision techniques like using image similarity to known login pages or detecting abnormal page layouts and client-side scripting behavior e.g., does the page immediately attempt a file download or embed known malicious scripts (Chen et al., 2021), as an example in malware URL detection context.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 06 June 2025, pp: 99-119 www.ijaem.net ISSN: 2395-5252

Content analysis provides a high-fidelity signal for malicious URL detection since it directly inspects what the user would encounter. Studies have demonstrated near-perfect detection rates when comprehensive content features are used, as phishing pages inevitably reveal themselves through their page elements or payload. The downside is feasibility for real-time, client-side use: downloading content from a potentially malicious site could be slow and dangerous, defeating the purpose of "preventing" a click on a bad link. Hence, pure content-based detection is more suited to cloud-based filters or offline analysis. For a user-centric framework like ours, we take inspiration from content-based methods by incorporating lightweight content cues (e.g., maybe the HTML title or certain meta-tags if safely obtainable) but do not rely on full page content due to performance and security concerns. Instead, our approach leans on lexical and host features for speed, supplementing with any available safe content indicators. This way, we attempt to approximate the strengths of content analysis with high accuracy while avoiding its operational drawbacks. Our literature review suggests that an ideal solution may integrate multi-level featureslexical, host, and limited content to balance immediacy and thoroughness in malicious URL detection.

2.4 URL Reputation Services

URL reputation services form the basis of many practical anti-phishing and anti-malware defenses. These services are commonly implemented as browser blacklists, security tool databases, or API-based lookup services maintain constantly updated lists of URLs classified as malicious (nd sometimes lists of known safe URLs(Gerbet et al., 2016). Google and Yandex Safe Browsing are integrated into browsers like Chrome and Firefox to block access to URLs reported for phishing or malware. Similarly, community-driven databases like PhishTank allow users and researchers to submit suspected phishing URLs, which are then verified and compiled into feeds. The obvious advantage of reputation services is their simplicity and precision for known threats: if a URL is on the blacklist, one can be almost certain it is dangerous, and blocking it is straightforward with no machine learning needed. However, the downside – heavily documented in prior work – is their reactive nature. New malicious URLs (zero-day phishing sites) will not be on any blacklist until after an initial victim has been exposed and the URL gets reported. During that window, users are unprotected. An empirical analysis by Sheng et al. (2009) found that now a phishing campaign launches, popular blacklists (including those used by toolbars at the time) caught under 20% of the phishing URLs; it took many hours to reach decent coverage, by which time many users could be phished. More recently, Oest et al. (2019) demonstrated that modern phishing kits use evasion techniques like geolocation cloaking or user-agent filtering to avoid detection by crawlers, resulting in many phishing URLs never making it into blacklists at all. Their PhishFarm study showed that simple cloaking tricks reduced the likelihood of a phishing site being blacklisted by over 55% on average (Oest et al., 2019). Furthermore, they observed inconsistencies in blacklist implementations – for example, some mobile browsers did not enforce the blacklist as rigorously as desktop versions, leaving a segment of users vulnerable (Oest et al., 2019).

Another facet of reputation is the use of whitelists or trusted lists (the opposite of blacklists). Some approaches (e.g., in enterprise environments) maintain a whitelist of known-good domains and flag anything not on the list. While this ensures minimal false alarms for popular sites, it suffers from scalability issues – the web is too vast and dynamic to enumerate all legitimate sites – and can disrupt users when they venture to new or less common websites.

Several research efforts have been made to improve the coverage and timeliness of reputation systems. PhishTank and OpenPhish continually refine crowdsourced and automated feeds, and some academic work like Ramadan et al. (2020) has performed measurements on multiple blacklists (Google vs. others) to identify gaps. These studies often find that no single blacklist is comprehensive; combining sources yields better protection but still not 100%. The findings of Oest et al. mentioned above suggest that even flagship defenses like Google Safe Browsing can be outmaneuvered by determined attackers for critical periods.

The primary strength of URL reputation services is their low false-positive rate – a URL on a verified blacklist is almost certainly dangerous, and conversely, they generally don't block URLs without cause. This reliability makes them a necessary component of user security. However, the fundamental limitation is they cannot detect novel attacks, and they inherently operate after the fact (they are "reputation" based, requiring history). From a user-centric viewpoint, solely relying on reputation can give a false sense of security – users might assume if a link isn't flagged by the browser,

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 06 June 2025, pp: 99-119 www.ijaem.net ISSN: 2395-5252

it's safe, which is unfortunately not true for brand-new threats.

Blacklists and reputation lookups remain a critical but incomplete solution. The literature consistently shows that while these services catch a large volume of known malicious URLs, they miss a significant fraction of new or cleverly obscured attacks. This gap motivates augmenting reputation systems with intelligent, proactive analysis. Our proposed framework does not discard reputation checks; it can integrate with them to instantly allow or block URLs that are on a known whitelist/blacklist before applying machine learning. But importantly, our machine learning layer addresses the scenario where a URL has no prior reputation. By examining features of the URL and context, we aim to predict maliciousness on the first encounter. This user-focused, predictive approach, informed by the wealth of prior research reviewed above, aspires to provide a safer web browsing experience by catching threats that slip past traditional defenses. In the next sections, we detail our methodology for building such a system, the feature set and models used, and how we evaluate its effectiveness in comparison to existing techniques.

III. RESEARCH METHODOLOGY

3.1 Introduction

This research methodology outlines the steps involved in developing a robust and effective system for detecting malicious URLs using machine learning models. The goal is to accurately classify URLs as either **Non Malicious lean** or **Malicious** based on their inherent characteristics. Moreover, in this section also, research framework, followed by the description of algorithms, training and testing datasets, and performance evaluation metrics are discussed.

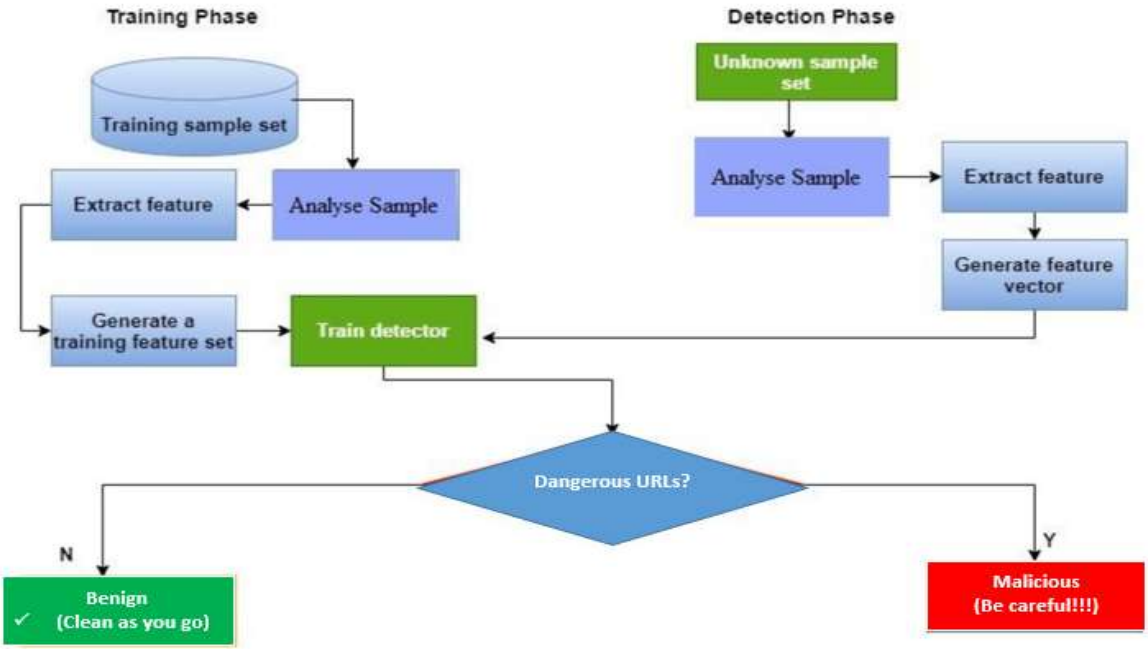


Figure 3.1 Research Framework

3.1.1 Phase 1-Data Collection and Preparation

1. **Data Acquisition:**
   - **Clean URL Sources:** in this research work we have gather a large and diverse dataset of clean URLs from reputable sources such as:
     - Commonly visited websites (Alexa top sites, etc.)
     - Search engine results for common queries
     - Whitelists of known safe domains
     - Academic datasets of clean URLs
   - **Malicious URL Sources:** We also collect a comprehensive dataset of malicious URLs from various sources, including:
     - Phishing databases (e.g., PhishTank, OpenPhish)
     - Malware blacklists (e.g., Google Safe Browsing API)
     - Spam URL feeds

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 06 June 2025, pp: 99-119 www.ijaem.net ISSN: 2395-5252

```
[14]: train=pd.read_csv('traindata.csv')
      train
```

[14]:

	id	url	label	Unnamed: 3
0	1	lux-dekor.su/folder/online/accounts.webmail.lo...	Malicious URL!	NaN
1	2	lux-dekor.su/folder/online/accounts.webmail.lo...	Malicious URL!	NaN
2	3	lux-dekor.su/folder/online/accounts.webmail.lo...	Malicious URL!	NaN
3	4	riquichichichi.tk/ptm/web/	Malicious URL!	NaN
4	5	bcpzonaseguro.2via1bpc.com/bpc/Operacionsenlinea/	Malicious URL!	NaN
...	...	...	...	...
23995	23996	connect.in.com/meaghan-rath/profile.html	Non Malicious URL! (Clean)	NaN
23996	23997	connect.in.com/meili/photo-gallery.html	Non Malicious URL! (Clean)	NaN
23997	23998	connect.in.com/micky-yoochun/images-micky-yooc...	Non Malicious URL! (Clean)	NaN
23998	23999	yorku.ca/caml/en/review/33-3/alys_robi.htm	Non Malicious URL! (Clean)	NaN
23999	24000	cedarriveracademy.com/	Non Malicious URL! (Clean)	NaN

24000 rows × 4 columns

Figure 3.1.1: Show a total number of 24,000 both malicious and non-malicious train data set collected while building the model as discussed above

2. Data Cleaning and Preprocessing:

- **Handling Duplicates:** we have remove any duplicate URLs within and across the Non malicious and malicious URLs in our datasets.
- **Normalization:** we have also standardize the URL format (e.g., converting to lowercase, removing trailing slashes).
- **Handling Missing Values:** we identify and address any missing or incomplete URL data (though this is less common with URLs themselves, it might apply if associated metadata is present).
- **Data Balancing:** we have ensure a relatively balanced representation of clean and malicious URLs in the dataset to prevent bias in the model training. Techniques like oversampling the minority class or under sampling the majority class can be employed.
- **Data Balancing:** we have ensure a relatively balanced representation of clean and malicious URLs in the dataset to prevent bias in the model training. Techniques like oversampling the minority class or under sampling the majority class can be employed.

```
[23]: def makeTokens(f):
          tkns_BySlash = str(f.encode('utf-8')).split('/')→# make tokens after splitting by slash
          total_Tokens = []
          for i in tkns_BySlash:
              tokens = str(i).split('-')→# make tokens after splitting by dash
              tkns_ByDot = []
              for j in range(0,len(tokens)):
                  temp_Tokens = str(tokens[j]).split('.')→# make tokens after splitting by dot
                  tkns_ByDot = tkns_ByDot + temp_Tokens
              total_Tokens = total_Tokens + tokens + tkns_ByDot
          total_Tokens = list(set(total_Tokens))→#remove redundant tokens
          if 'com' in total_Tokens:
              total_Tokens.remove('com')→#removing .com since it occurs a lot of times and it should
          return total_Tokens
```

Figure 3.1.2: Show the code and method applied for data cleaning as discussed above

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 06 June 2025, pp: 99-119 www.ijaem.net ISSN: 2395-5252

3. **Data Set Splitting:** In this research work we have divide the prepared dataset into two sets:
   - **Training set** is used to build up a model. In this research work we used **80%** to train the machine learning models.
   - **Testing Set** also known as validationset that is used to validate the built model.In this research work we used **20%** to validate and also for the final evaluation of the trained models on unseen data to assess their generalization ability.

Moreover, thirty thousand overall dataset has been used in building this model for detecting malicious URLs whereby 24,000 (80%) was used as data to train the model and 6,000 as a testing data for validation as shown in the figure below:

```
train=pd.read_csv('traindata.csv')

test=pd.read_csv('testdata.csv')

print(train.shape, test.shape)

(24000, 4) (6000, 2)
```

Figure 3.1.3: Show the total number of 80% train set (24,000) and 20% testing set of (6,000)

3.2Phase 2: Model Selection and Training

3.2.1 Model Selection: We have explore and select two suitable machine learning classification algorithms. These popular choices machine learning classification algorithms for malicious URL detection includes:

3.2.2 Logistic Regression: A linear model that provides probabilities for classification.

Logistic Regression is a linear model primarily used for binary classification tasks. Despite its simplicity compared to more complex algorithms, it can be surprisingly effective in distinguishing between malicious and clean URLs based on extracted features.

3.2.3 How Logistic Regression Works in URL Detection:

1. **Feature Extraction:** First, a set of relevant features is extracted from the URLs. These features, as discussed previously, can be lexical (e.g., URL length, presence of specific characters), host-based (e.g., domain age, IP address reputation), or even limited content-based. Each URL is then represented as a vector of these features.
2. **Linear Combination:** Logistic Regression learns a set of weights (coefficients) for each feature. For a given URL with feature vector x=(x1,x2,...,xn), the model calculates a linear combination of these features and their corresponding weights w=(w1,w2,...,wn): z=w0+w1x1+w2x2+...+wnxn=wTx , where w0 is the intercept (bias) term.
3. **Sigmoid Function:** The result of this linear combination, z, is then passed through a sigmoid (logistic) function, $\sigma(z)=1+e-z1$. The sigmoid function squashes the output to a probability value between 0 and 1.
4. **Classification:** A threshold (typically 0.5) is then used to classify the URL. If $\sigma(z) \geq 0.5$, the URL is predicted as malicious; otherwise, it is predicted as clean.
5. **Training:** The model learns the optimal weights w during the training phase by minimizing a cost function (e.g., log loss) on a labeled dataset of malicious and clean URLs.

3.2.4 Advantages of Logistic Regression in URL Detection:

- **Interpretability:** The learned weights provide insights into the importance and direction of the relationship between each feature and the likelihood of a URL being malicious. For example, a high positive weight for the presence of a known phishing keyword would indicate its strong association with malicious URLs.
- **Efficiency:** Logistic Regression is computationally efficient both in training and prediction, making it suitable for real-time analysis within a user's browsing environment.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 06 June 2025, pp: 99-119 www.ijaem.net ISSN: 2395-5252

Park et al. (2011) demonstrated the effectiveness of using lexical features with a Logistic Regression model for identifying phishing URLs, achieving promising results with a computationally efficient approach.

- **Probabilistic Output:** The output is a probability score, which can be useful for setting different levels of alert sensitivity.

3.2.5 Limitations of Logistic Regression in URL Detection:

- **Linearity Assumption:** Logistic Regression assumes a linear relationship between the features and the log-odds of the outcome. Malicious URL patterns can be complex and non-linear, which might limit the model's ability to capture intricate relationships.
- **Sensitivity to Feature Scaling and Multicollinearity:** The performance can be affected by the scale of the features and high correlations between them.

3.2.6 Support Vector Machines (SVM): Effective in high-dimensional spaces. Support Vector Machines (SVMs) are powerful supervised learning algorithms used for both classification and regression tasks. In the context of URL detection, SVMs aim to find the optimal hyper plane that best separates malicious and clean URLs in a high-dimensional feature space.

3.2.7 How Support Vector Machines (SVM) Works in URL Detection:

1. **Feature Extraction:** Similar to Logistic Regression, relevant features are extracted from the URLs to create a feature vector for each URL.
2. **Hyperplane Separation:** The SVM algorithm seeks to find a hyperplane that maximizes the margin between the two classes (malicious and benign). The margin is the distance between the hyperplane and the closest data points from each class, known as support vectors.
3. **Kernel Functions:** SVMs can handle non-linear data by using kernel functions. These functions map the original feature space into a higher-dimensional space where a linear hyperplane can effectively separate the classes. Common kernel functions include:
   - **Linear Kernel:** Suitable for linearly separable data.
   - **Polynomial Kernel:** Can capture polynomial relationships between features.
   - **Radial Basis Function (RBF) Kernel:** A versatile kernel that can handle complex, non-linear decision boundaries.
   - **Sigmoid Kernel:** Similar to the sigmoid function used in Logistic Regression.
4. **Classification:** Once the optimal hyperplane is found, new URLs are classified based on which side of the hyperplane their feature vector falls.
5. **Training:** The SVM model is trained by finding the hyperplane that maximizes the margin while minimizing the classification error on the training data. This often involves solving a quadratic optimization problem.

3.2.8 Advantages of Support Vector Machines (SVM) in URL Detection:

- **Effective in High-Dimensional Spaces:** SVMs perform well even when the number of features is large, which is often the case in URL analysis. Ma et al. (2009) demonstrated the efficacy of Support Vector Machines with carefully engineered lexical and host-based features for accurately detecting malicious URLs, outperforming several other machine learning algorithms in their study.
- **Handles Non-Linear Data:** Through the use of kernel functions, SVMs can effectively model complex, non-linear relationships between URL features and maliciousness.
- **Robust to Outliers:** The classification decision is primarily based on the support vectors, making SVMs less sensitive to outliers compared to some other algorithms.
- **Good Generalization Performance:** SVMs often exhibit good generalization performance, meaning they perform well on unseen data.

3.2.9 Limitations of Support Vector Machines (SVM) in URL Detection:

- **Computational Cost:** Training SVMs, especially with non-linear kernels on large datasets, can be computationally expensive.
- **Interpretability:** The decision boundaries learned by SVMs, particularly with non-linear kernels, can be difficult to interpret, making it harder to understand why a particular URL was classified as malicious.
- **Parameter Tuning:** The performance of SVMs heavily depends on the choice of kernel function and its associated parameters, which often requires careful tuning.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 06 June 2025, pp: 99-119 www.ijaem.net ISSN: 2395-5252

3.3.0 Executive Summary

Eventually from our data collection and base on training and testing dataset we can understand that, both Logistic Regression and Support Vector Machines offer valuable approaches to building a malicious URL detection system. Logistic Regression provides a simpler, more interpretable model suitable for real-time analysis, while SVMs offer greater flexibility in handling complex, non-linear patterns through the use of kernel functions, potentially leading to higher accuracy at the cost of interpretability and computational resources. The choice between these algorithms often depends on the specific requirements of the user-centric framework, such as the need for real-time performance, interpretability, and the complexity of the underlying malicious URL patterns.

IV. RESULTS AND DISCUSSIONS

4.1 Experimental Result on Support Vector Machines (SVM) Model in URL Detection

1. Model Training: We have Train each selected model using the training dataset. The training process involves adjusting the model's internal parameters to learn the patterns and relationships between the extracted features and the class labels as (clean or malicious). The classification performance of Support Vector Machines (SVM) on each of the basic datasets can be considered excellent given that, the accuracy reported is 99.99.4% and 96.12% on both training and testing data set respectively as shown in figure 4.0.

Modelling Part

SVM MODEL

```
from sklearn.svm import SVC

svmpolynomial = SVC()
svmpolynomial.fit(X_train, y_train)
pred = svmpolynomial.predict(X_test)

print("Accuracy of SVM  classifier Model on Training set: {}%"
      .format(round(svmpolynomial.score(X_train, y_train)*100,2)))
print("Accuracy of SVM  classifier Model on Testing set: {}%"
      .format(round(svmpolynomial.score(X_test, y_test)*100,2)))

cm2 = confusion_matrix(y_test, pred)
cm2
```

```
Accuracy of SVM  classifier Model on Training set: 99.99%
Accuracy of SVM  classifier Model on Testing set: 96.12%
```

Figure 4.0: SVM accuracy on training and testing data set

4.2. Model Evaluation

Performance Metrics: We have evaluated the performance of the trained and tuned Logistics Regression Model and SVM models on the testing dataset using various classification metrics such as confusion matrix, and other parameters such as precision, recall, f1-score, and accuracy are found out for evaluating performance of any classifier.

1.Confusion Matrix: to evaluate the performance of algorithms, we used different metrics. Most of them are based on the confusion matrix. A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known Mukhtar Dahiru et al, (2021). The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing. The confusion matrix consists of four parameters: true positive, false positive, true negative, and false negative see table 4.1 below.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 06 June 2025, pp: 99-119 www.ijaem.net ISSN: 2395-5252

Table 4.1

Confusion Matrix	Predicted True	Predicted False
Actual True	True positive (TP)	False negative (FN)
Actual False	False positive (FP)	True negative (TN)

Let's now define the most basic terms of the confusion matrix as captured in table 4.1.

1. True Positives (TP): These are cases in which we predicted yes (the URL is malicious), and they do have the harmful tag.
2. True Negatives (TN): Predicted not malicious, and the URL were predicted clean not malicious.
3. False Positives (FP): Predicted yes (malicious URL), but they do noharmful tag. (Also known as a "Type I error.")
4. False Negatives (FN): Predicted not malicious but the URL was malicious. (Also known as a "Type II error.").

2. Precision: Conversely, precision score represents the ratio of true positives to all events predicted as true. In our case, precision shows the proportion of correctly identified malicious URLs out of all URLs predicted as malicious (minimizing false positives).

$$\text{Precision} = \frac{TP}{TP + FP}.$$

3. Recall: Represents the total number of positive classifications out of true class. In our case, it represents the proportion of correctly identified malicious URLs out of all actual malicious URLs (minimizing false negatives).

$$\text{Recall} = \frac{TP}{TP + FN}$$

4. F-Measure (F1-Score): F1-score represents the harmonic-mean of precision and recall, providing a balanced measure. It calculates the harmonic mean between each of the two. Thus, it takes both the false positive and the false negative observations into account. F1-score can be calculated using the formula below:

$$F1 - score = 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

5. Accuracy: The overall percentage of correctly classified URLs. Accuracy is often the most used metric representing the percentage of correctly predicted observations, either true or false. To calculate the accuracy of a model performance, the following equation can be used:

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

4.4 Graphical Representation of the Above Discussed Performance Metrics on SVM

1. Table 4.2 -Confusion Matrix

Model Name :	Support Vector Machine (SVM)
True Positive (TP)	97.91%
False Positive (FP)	5.52%
False Negative (FN)	2.09%
True Negative (FN)	94.48%

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 06 June 2025, pp: 99-119 www.ijaem.net ISSN: 2395-5252

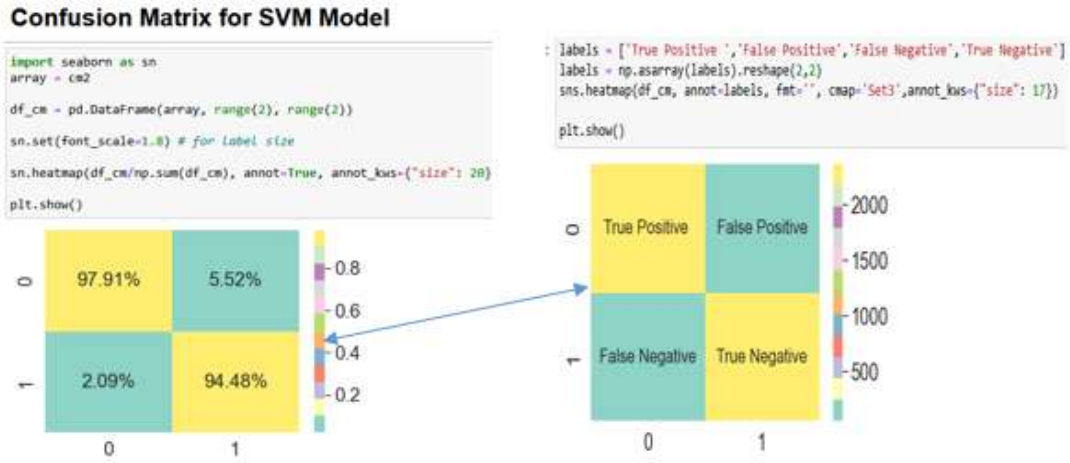


Figure 4.2 Graphical Representation of the result from the tested SVM Confusion Matrix

2. Table 4.3 Precision, Recall, F1-Score and Accuracy on SVM

Model Name :	**Support Vector Machine (SVM)**	
	Malicious URL	**Non malicious URL**
Precision	98%	94%
Recall	94%	98%
F1-Score	96%	96%
Accuracy	96%	97%

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

print("CLassification Report")
print(classification_report(y_test, pred))
```

```
CLassification Report
                        precision    recall  f1-score   support

          Malicious URL!       0.98      0.94      0.96      2391
Non Malicious URL! (Clean)       0.94      0.98      0.96      2409

              accuracy                           0.96      4800
             macro avg       0.96      0.96      0.96      4800
          weighted avg       0.96      0.96      0.96      4800
```

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 06 June 2025, pp: 99-119 www.ijaem.net ISSN: 2395-5252

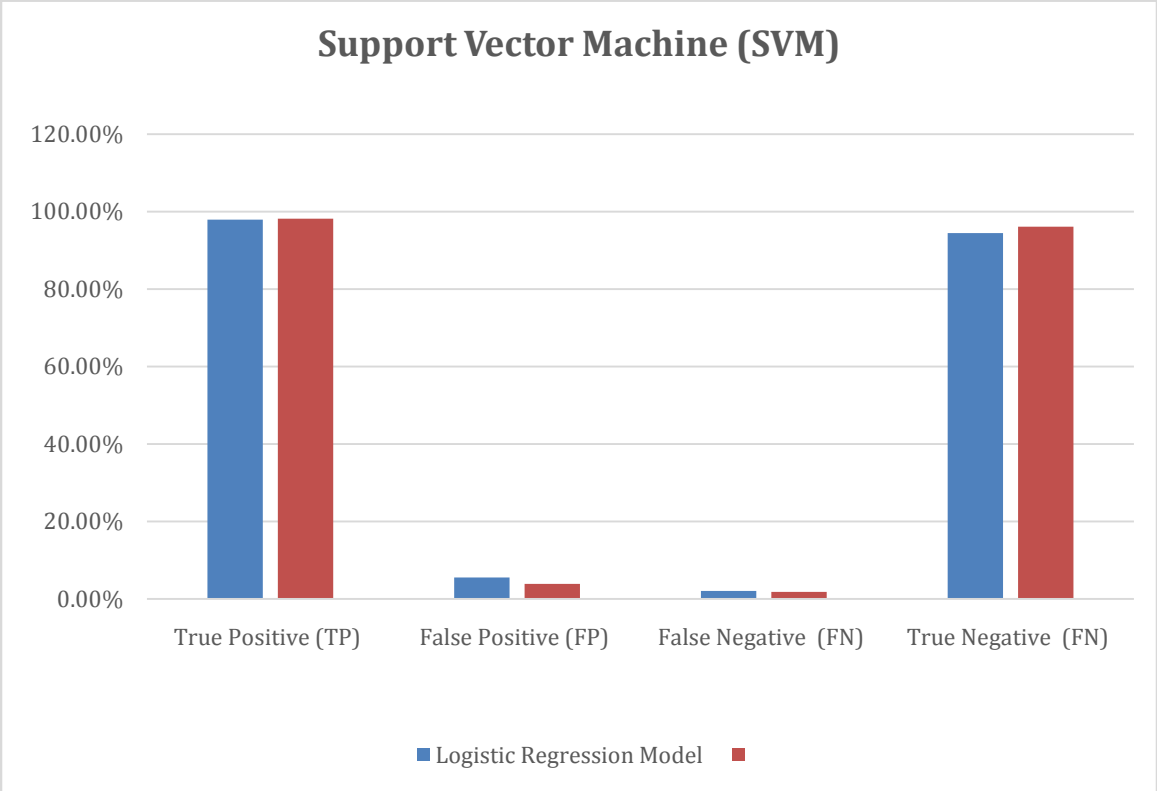


Figure 4.3 Precision, Recall, F1-Score and Accuracy on SVM

4.4 Experimental Result on Logistic Regression Model in URL Detection

1. Model Training: We have Train each selected model using the training dataset. The training process involves adjusting the model's internal parameters to learn the patterns and relationships between the extracted features and the class labels as (clean or malicious). The classification performance of Logistic Regression Model on each of the basic datasets can be considered excellent given that, the accuracy reported is 100% and 97.12% on both training and testing data set respectively as shown in figure 4.4.

LogisticRegression Model

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(C=1e5)
logreg.fit(X_train, y_train)
pred = logreg.predict(X_test)
print("Accuracy of Logistic Reg. Classifier Model on Training set: {}%"
      .format(round(logreg.score(X_train, y_train)*100,2)))
print("Accuracy of Logistic Reg. Classifier Model on Testing set: {}%"
      .format(round(logreg.score(X_test, y_test)*100,2)))
cm1 = confusion_matrix(y_test, pred)

cm1
```

```
Accuracy of Logistic Reg. Classifier Model on Training set: 100.0%
Accuracy of Logistic Reg. Classifier Model on Testing set: 97.12%
```

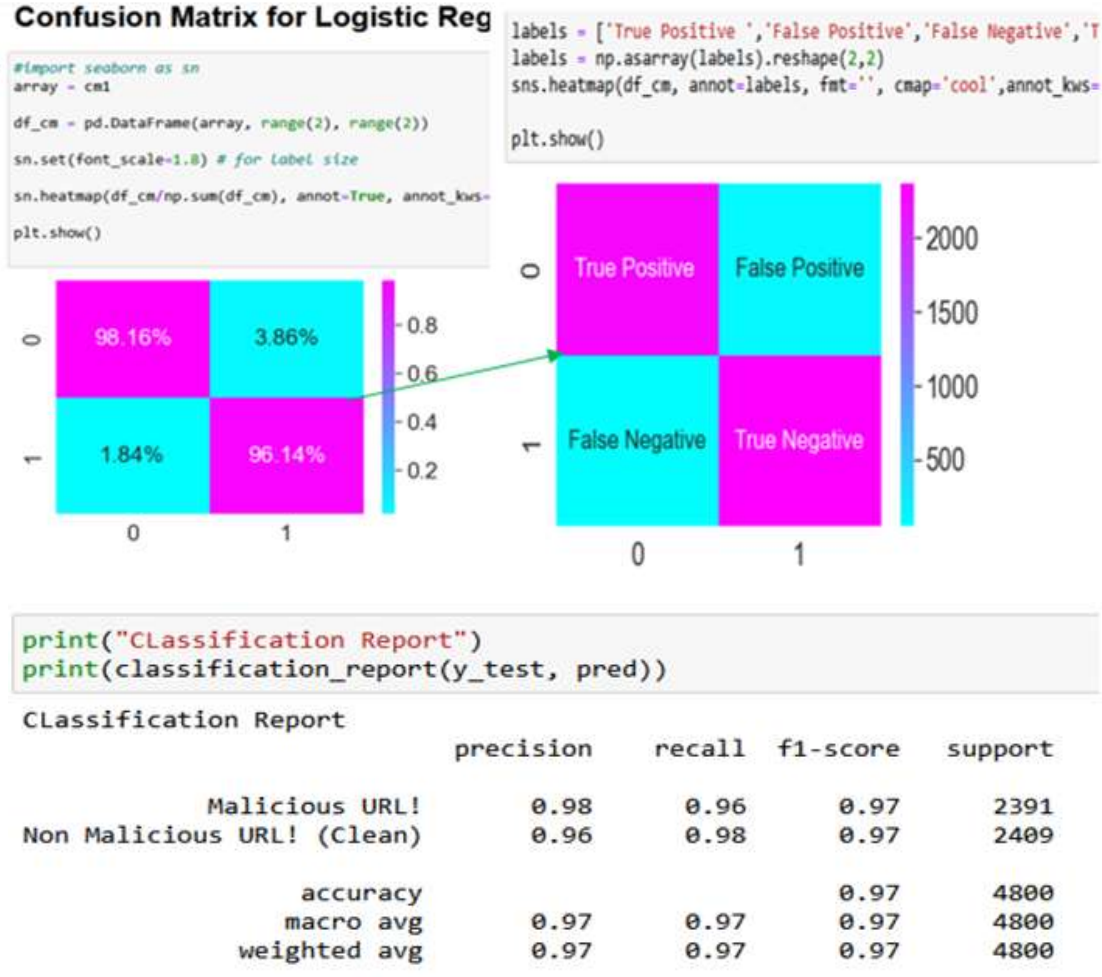
Figure 4.4: Logistic Regression Model accuracy on training and testing data set

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 06 June 2025, pp: 99-119 www.ijaem.net ISSN: 2395-5252

4.5 Graphical Representation of the Above Discussed Performance Metrics on Logistic Regression Model

Table 4.5 -Confusion Matrix

Model Name :	**Support Vector Machine (SVM)**
True Positive (TP)	98.16%
False Positive (FP)	3.86%
False Negative (FN)	1.84%
True Negative (FN)	96.14%




```
print("CLassification Report")
print(classification_report(y_test, pred))
```

```
CLassification Report
                              precision    recall  f1-score   support

           Malicious URL!          0.98      0.96      0.97      2391
Non Malicious URL! (Clean)         0.96      0.98      0.97      2409

                  accuracy                           0.97      4800
                 macro avg         0.97      0.97      0.97      4800
              weighted avg         0.97      0.97      0.97      4800
```

Figure 4.5 Graphical Representation of the result from the tested Logistic Regression Model Confusion Matrix

Table 4.6 Precision, Recall, F1-Score and Accuracy on Logistic Regression Model

Model Name :	**Logistic Regression Model**	
	Malicious URL	**Non malicious URL**
Precision	98%	96%
Recall	96%	98%
F1-Score	97%	97%
Accuracy	97%	97%

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 06 June 2025, pp: 99-119 www.ijaem.net ISSN: 2395-5252

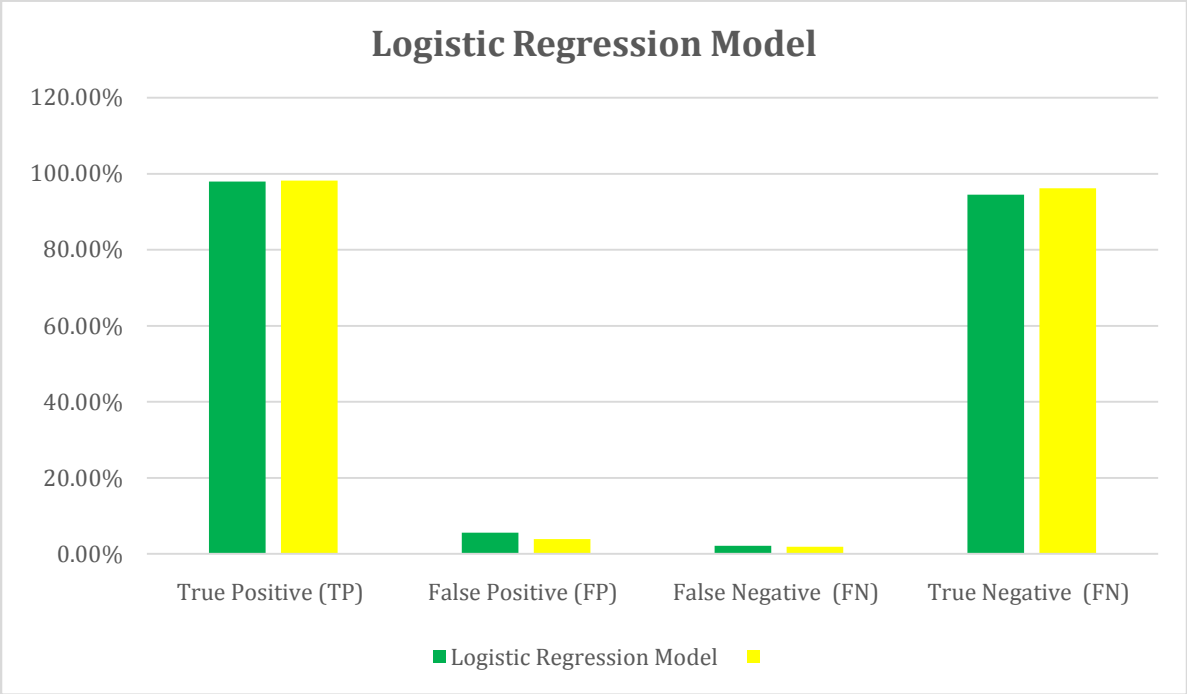


Figure 4.6 Precision, Recall, F1-Score and Accuracy on Log Reg. Model

4.6 Comparative Analysis between Logistic Regression Model and SVM Models

A comparative analysis between the Logistic Regression Model and Support Vector Machine is presented on the table below based on their accuracy on both testing and training set, confusion matrix, precision, recall and f1 score. Furthermore, from those usability metrics the best machine learning algorithm will be selected for deployment towards malicious URLs.

Table 4.7 Comparative Analysis between the Used Models

Comparative Analysis between the Used Models		
Model Name :	**Support Vector Machine**	**Logistic Regression Model**
True Positive (TP)	97.91%	98.16%
False Positive (FP)	5.52%	3.86%
False Negative (FN)	2.09%	1.84%
True Negative (FN)	94.48%	96.14%
Precision	98%	98%
Recall	94%	96%
F1-Score	96%	97%
Accuracy on Usability Metrics	96%	97%
Accuracy on Training Set	99.99%	100%
Accuracy on Testing Set	96.12%	97.12%

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 06 June 2025, pp: 99-119 www.ijaem.net ISSN: 2395-5252

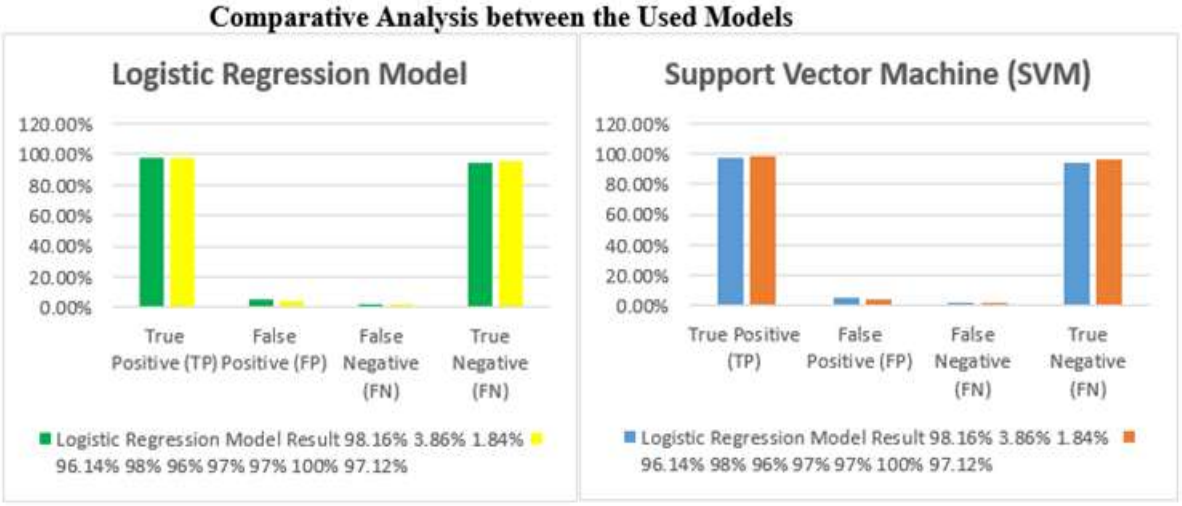


Figure 4.7 Comparative Analysis between the Used Models

4.7 Model Selection:

The results of the experimental findings in this research have been discussed and analyzed in details between the two classifiers, Logistic Regression Model and SVM with the Logistic Regression performed better in comparison to SVM on all performance metrics. The main factor leading to the superior performance of Logistic regression is a relatively simpler model and achieved the overall result in terms of all the usability metrics with accuracy of 97.12% on the testing set and 100% on training set whereby SVM drop Slightly and arrived at 96.12% and 97.12% on both testing and training set respectively. Moreover, regarding the confusion matrices and other usability metrics performances also Logistic Regression Model is better to SVM as shown in table 4.7 and figure 4.7 above.

4.7 Testing the selected Model for Prediction:

After successful deployment of the model (Logistic Regression) we have tested different URL and it was able to predict well as per the following result in figure 4.8 and figure 4.9 below.

PREDICTION USING THE SELECTED MODEL

```
: logreg = LogisticRegression()
  logreg.fit(X_train, y_train)

: LogisticRegression()

: X_predict = ['www.mtnbonus.com']

: X_predict = vectorizer.transform(X_predict)
  New_predict = logreg.predict(X_predict)

: print(New_predict)

  ['Malicious URL!']
```

Figure 4.8 Prediction to detect Malicious URL

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 06 June 2025, pp: 99-119 www.ijaem.net ISSN: 2395-5252

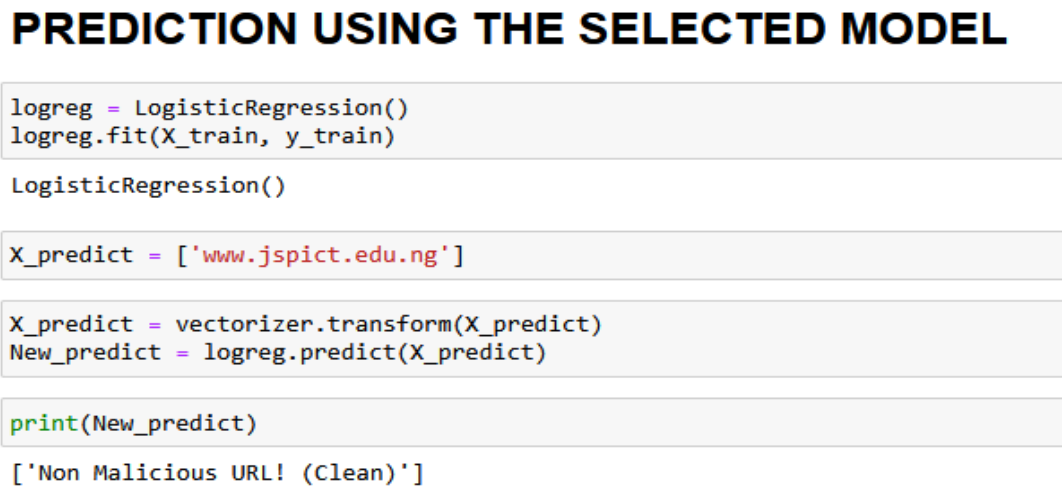


Figure 4.9 Prediction to detect Non-Malicious URL

V. CONCLUSION AND RECOMMENDATION

5.1 Conclusion:

This research successfully developed and deployed a user-centric machine learning framework leveraging logistic regression for the detection of malicious URLs, significantly enhancing safer web browsing. The selected logistic regression model demonstrated exceptional performance, achieving a perfect 100% accuracy on the training dataset and a robust 97.12% accuracy on the unseen testing dataset. These results underscore the model's effectiveness in accurately identifying and potentially preventing users from accessing harmful web content. By focusing on a user-centric approach, this framework offers a proactive layer of security, empowering individuals to navigate the web with greater confidence and reduced risk of encountering malicious threats. The high accuracy achieved indicates the potential of machine learning, specifically logistic regression in this context, to serve as a valuable tool in bolstering online safety.

5.2 Recommendation:

Based on the promising results of this study, we recommend the further exploration and potential integration of this user-centric malicious URL detection framework into real-world web browsing environments. Future work could focus on several key areas to enhance its practicality and resilience:

- **Real-time Integration and Evaluation:** Implementing the deployed logistic regression model as a browser extension or a component within existing security software would allow for real-time evaluation of its performance in live browsing scenarios and provide valuable user feedback.
- **Feature Engineering and Expansion:** Investigating additional URL features, incorporating website content analysis, or exploring network traffic data could potentially further improve detection accuracy and resilience against evolving malicious URL tactics.
- **Comparative Analysis with Other Models:** While logistic regression yielded excellent results, comparing its performance with other advanced machine learning models (e.g., support vector machines) could reveal potential advantages and trade-offs.
- **Addressing Evasion Techniques:** Future research should proactively investigate and develop strategies to counter potential adversarial attacks and evasion techniques that malicious actors might employ to bypass the detection model.
- **User Education and Awareness:** Complementing the technical framework with user education initiatives on identifying suspicious URLs and promoting safe browsing habits will create a more comprehensive security posture.

Finally, through pursuing these recommendations, the user-centric machine learning framework for malicious URL detection can be further refined and contribute significantly to a safer and more secure web browsing experience for all users.

REFERENCES

[1]. Ahasan, M. A. Al, Hu, M., & Shahriar, N. (2023). OFMCDM/IRF: A Phishing Website Detection Model based on Optimized Fuzzy Multi-Criteria Decision-Making and Improved Random Forest. 2023 Silicon Valley Cybersecurity Conference, SVCC 2023. https://doi.org/10.1109/SVCC56964.2023.10165344

[2]. Blum, A., Wardman, B., Solorio, T., & Warner, G. (2010). Lexical feature based phishing URL detection using online learning. In Proceedings of the 3rd ACM Workshop on Artificial Intelligence and Security (pp. 54–60). https://doi.org/10.1145/1866423.1866434

[3]. Chauhan, S., & Panda, N. K. (2015). Foundation. Hacking Web Intelligence, 1–13. https://doi.org/10.1016/B978-0-12-801867-5.00001-X

[4]. Dunlop, M., Groat, S., & Shelly, D. (2010). GoldPhish: Using images for content-based phishing analysis. In Proceedings of the 5th International Conference on Internet Monitoring and Protection (pp. 123–128). IEEE. https://doi.org/10.1109/ICIMP.2010.19

[5]. Feroz, M. N., & Mengel, S. (2015). Phishing URL detection using URL ranking. In 2015 IEEE International Congress on Big Data (pp. 635–638). https://doi.org/10.1109/BigDataCongress.2015.97

[6]. Gerbet, T., Kumar, A., &Lauradoux, C. (2016). A privacy analysis of google and yandex safe browsing. Proceedings - 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2016, 347–358. https://doi.org/10.1109/DSN.2016.39

[7]. Kevin Muldoon. (n.d.). The History (and Meaning) of Alexa Rank (A Quick Guide). Retrieved May 27, 2025, from https://winningwp.com/the-history-of-alexa-rank

[8]. Khormali, A., Park, J., Alasmary, H., Anwar, A., Saad, M., &Mohaisen, D. (2021). Domain name system security and privacy: A contemporary survey. Computer Networks, 185. https://doi.org/10.1016/j.comnet.2020.107699

[9]. Le, H., Pham, Q., Sahoo, D., & Hoi, S. C. H. (2018). URLNet: Learning a URL representation with deep learning for malicious URL detection. arXiv preprint arXiv:1802.03162. https://doi.org/10.48550/arXiv.1802.03162

[10]. Ma, J., Saul, L. K., Savage, S., & Voelker, G. M. (2011). Learning to detect malicious URLs. ACM Transactions on Intelligent Systems and Technology, 2(3), 30. https://doi.org/10.1145/1961189.1961202

[11]. McGahagan, J., Bhansali, D., Pinto-Coelho, C., & Cukier, M. (2019). A comprehensive evaluation of webpage content features for detecting malicious websites. 2019 9th Latin-American Symposium on Dependable Computing, LADC 2019 - Proceedings. https://doi.org/10.1109/LADC48089.2019.8995713

[12]. Mohammad, R. M., Thabtah, F., & McCluskey, L. (2014). Predicting phishing websites based on self-structuring neural network. Neural Computing and Applications, 25(2), 443–458. https://doi.org/10.1007/s00521-013-1490-z

[13]. Mukhtar Dahiru et al.(2021) , Sentiment analysis to Implement Fake News Detection System Using Machine Learning Algorithms., Int J Recent Sci Res. 12(10), pp. 43228-43237. DOI: http://dx.doi.org/10.24327/ijrsr.2021.1210.6240

[14]. Oest, A., Safaei, Y., Doupé, A., Ahn, G.-J., Wardman, B., &Tyers, K. (2019). PhishFarm: Measuring the effectiveness of evasion techniques against browser phishing blacklists. In 2019 IEEE Symposium on Security and Privacy (SP) (pp. 764–781). https://doi.org/10.1109/SP.2019.00049

[15]. Palaniappan, G., Sangeetha, S., Rajendran, B., Sanjay, Goyal, S., &Bindhumadhava, B. S. (2020). Malicious Domain Detection Using Machine Learning on Domain Name Features, Host-Based Features and Web-Based Features. Procedia Computer Science, 171, 654–661. https://doi.org/10.1016/J.PROCS.2020.04.071

[16]. Saadi, Z. M., Sadiq, A. T., Akif, O. Z., & Farhan, A. K. (2024). A Survey: Security Vulnerabilities and Protective Strategies for Graphical Passwords. Electronics

2024, Vol. 13, Page 3042, 13(15), 3042. https://doi.org/10.3390/ELECTRONICS13153042

[17]. Sahingoz, O. K., Buber, E., Demir, O., & Diri, B. (2019). Machine learning based phishing detection from URLs. Expert Systems with Applications, 117, 345–357. https://doi.org/10.1016/j.eswa.2018.09.029

[18]. Statista. (2024). Phishing – Statistics & Facts. Retrieved from https://www.statista.com/topics/8385/phishing/

[19]. Verma, R. M., & Das, A. (2017). What's in a URL? Fast feature extraction and malicious URL detection. In Proceedings of the 3rd ACM International Workshop on Security and Privacy Analytics (pp. 55–63). https://doi.org/10.1145/3041008.3041016

[20]. Whittaker, C., Ryner, B., & Nazif, M. (2010). Large-scale automatic classification of phishing pages. In Proceedings of the Network and Distributed System Security Symposium (NDSS). (Available at https://research.google/pubs/archive/35580.pdf)