

Article

A Temporally Dynamic Feature-Extraction Framework for Phishing Detection with LIME and SHAP Explanations

Chris Mayo , Michael Tchuindjang , Sarfraz Brohi *  and Nikolaos Ersotelos 

School of Computing and Creative Technologies, University of the West of England, Bristol BS16 1QY, UK; christopher2.mayo@live.uwe.ac.uk (C.M.); michael.tchuindjang@uwe.ac.uk (M.T.); nikolaos.ersotelos@uwe.ac.uk (N.E.)

* Correspondence: sarfraz.brohi@uwe.ac.uk

Abstract

Phishing remains one of the most pervasive social engineering threats, exploiting human vulnerabilities and continuously evolving to bypass static detection mechanisms. Existing machine learning models achieve high accuracy but often act as opaque systems that lack robustness to evolving tactics and explainability, limiting trust and real-world deployment. In this research, we propose a dynamic Explainable AI (XAI) approach for phishing detection that integrates temporally aware feature extraction with dual interpretability through LIME and SHAP applied to the resulting window-level features. The novelty of this research lies in a temporally dynamic feature framework that simulates a plausible email reading progression using a heuristic temporal model and employs a sliding window aggregation method to capture behavioural and temporal patterns within email content. Using an aggregated dataset of 82,500 phishing and legitimate emails, dynamic features were extracted and used to train four classifiers: Random Forest, XGBoost, Multi-Layer Perceptron, and Logistic Regression. Ensemble models demonstrated strong performance with XGBoost achieving 94% accuracy and Random Forest 93%. This research addresses an important gap by combining dynamically constructed temporal features with transparent explanations, achieving high detection performance while preserving interpretability. These findings demonstrate that dynamic temporal modelling with explainable learning can enhance the trustworthiness and practicality of phishing detection systems, highlighting that temporally structured features and explainable learning can enhance the trustworthiness and practical deployability of phishing detection systems without incurring excessive computational overhead.

Keywords: Explainable Artificial Intelligence (XAI); phishing detection; temporal features; dynamic feature extraction; XGBoost; random forest; LIME; SHAP; cyber threat detection



Academic Editor: Paolo Bellavista

Received: 23 December 2025

Revised: 23 January 2026

Accepted: 9 February 2026

Published: 14 February 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Phishing remains one of the most persistent and successful social engineering threats, with recent reports attributing a substantial number of breaches to human-targeted attacks, including phishing as an initial access point [1], underscoring the need for robust, adaptive detection approaches. As attackers continually vary content, timing, and delivery strategies, the scale and diversity of these attacks highlight that static detection systems are increasingly brittle in the face of evolving phishing campaigns and user contexts.

While there have been significant advances in phishing detection accuracy through machine learning [2,3], widespread reliance on these black-box models, coupled with a

lack of user understanding, erodes trust. It negatively affects operational adoption [4], motivating the integration of Explainable AI (XAI) to enable transparent decision-making and to highlight security-related reasoning [5–7]. This transparent decision-making enables human users to understand, trust, and appropriately act on model outputs, which is critical in domains such as cybersecurity [8]. Existing XAI-enhanced phishing systems largely use static feature sets or text-only vectorisation, leaving a gap in methods that capture temporal dynamics and behavioural context while still providing clear local and global explanations to users. This paper addresses the problem of balancing interpretability with temporally informed modelling by asking: How can a dynamic feature-extraction system, combined with machine learning and XAI, improve phishing detection performance while maintaining transparent explanations suitable for real-world use, even when models are trained in an offline setting?

The main objective of this research is to design a dynamic feature extraction system that utilises simulated temporal progression by combining extracted send timestamps with a heuristic reading-time model, with aggregation into overlapping windows for capturing contextual patterns. In this work, ‘dynamic’ refers to the construction of temporally structured window-level features from email content and timestamps. LIME and SHAP are then applied in their standard form to these dynamic features, with the classifiers being trained in an offline supervised setting. The system aims to train diverse classifiers on these dynamic features and rigorously evaluate them using accuracy, precision, recall, F1, and confusion matrices to determine which classifiers are most suitable for the system. The research also aims to provide both local and global interpretability using Local Interpretable Model-agnostic Explanations (LIME) and SHapley Additive exPlanations (SHAP) methods, highlighting the most influential features to enhance user trust and decision support. Additionally, the paper compares outcomes and insights against recent literature to position the approach among XAI-enhanced and dynamic phishing detection systems, addressing the limitations of static models and contributing to the advancement of real-time solutions in cybersecurity.

The study uses a large, aggregated, open phishing email dataset (82,500 emails, balanced with phishing and legitimate emails) and extracts timestamps using a multi-pattern finding strategy to enable analysis of real temporal context. Emails are segmented and converted into session logs, with simulated reading durations, then grouped via overlapping windows to aggregate temporal features with some linguistic features, before training Random Forest (RF), Extreme Gradient Boosting (XGBoost), Multi-layer Perceptron (MLP) and Logistic Regression (LR) classifiers. LIME provides local instance-level explanations and SHAP beeswarm analyses for global patterns, revealing dominant features for decision-making, such as money-related terms and temporal cues, such as *base_hour* and *base_year*.

The main contributions are stated below:

- Introducing a temporal dynamic extraction method that integrates realistic reading-time progression with overlapping windows to capture dynamic, behavioural and timing signals.
- Delivers a dual XAI setup for the dynamic system (LIME and SHAP), that clarifies the models’ behaviours across classifiers and highlights reasoning, improving transparency and trust.
- Demonstrates strong performance with tree ensemble classifiers with XGBoost achieving 94% accuracy and Random Forest achieving 93%.
- Shows how dynamically constructed temporal features and explainability can be jointly achieved without excessive computational overhead.

The remainder of the paper is organised as follows. Section 2 reviews related work on explainable and dynamic phishing detection. Section 3 describes the proposed methodology,

including the dataset, the dynamic feature extraction process, model training, and the implementation of XAI. Section 4 presents and analyses the experimental results. Section 5 benchmarks the proposed approach against existing works reported in the literature. Section 6 discusses the limitations and future directions of this research. Finally, Section 7 summarises the paper's findings.

2. Related Work

2.1. XAI in Phishing Detection

Recent research demonstrates growing emphasis on interpretability in phishing detection. These works apply machine learning (ML) techniques to detect phishing emails and incorporate XAI to make the results more visually interpretable and user-friendly. Yakan-dawala et al. [9] highlighted that integrating machine learning and XAI can significantly improve phishing detection, thereby developing trustworthy and interpretable systems that help users make informed decisions about email threats. Their research demonstrated notable advances in using XAI to enhance transparency and trust in phishing detection models, with strong results from the support vector machine (SVM) classifier. The methodology employed user-friendly visualisations and a graphical user interface (GUI) to address user distrust in black-box ML models. However, it highlights future work on evolving phishing tactics, which our research addresses by incorporating dynamic features.

In another study, Fajar, Yazid, and Budi [10] demonstrated that effective feature selection is central to accurate phishing detection. They identified crucial elements such as 'length_url,' 'time_domain_activation,' and 'page_rank' as significant indicators of phishing attempts. Their methodology leverages XAI techniques, such as SHAP, to enhance model interpretability and guide classification decisions, yielding robust and efficient models. A notable strength of this approach is its ability to convert complex model behaviour into clear, actionable insights, improving both accuracy and transparency. However, the methodology's reliance on static feature importance has limitations: it does not account for temporal changes or evolving phishing tactics, which can reduce adaptability to new threats. Additionally, focusing on current feature importance may lead to overfitting and limit generalizability in dynamic, real-world environments. These limitations underscore the need for future research on dynamic, context-aware feature analysis and adaptive XAI frameworks for phishing detection.

A framework for using explainable ensemble models for phishing detection was presented in E2Phish [11], which employed hard voting, soft voting, and stacking to improve classification accuracy. To explain the E2Phish model to the end user, SHAP, LIME, and PDP were used. These methods emphasise identifying the key features that influence the model's predictions. With this model introducing stacking, the reported stack ensemble outperformed all individual base classifiers, achieving 97% accuracy, 96% precision, 98% recall, and 97% F1-score, which demonstrates that aggregating diverse models such as Decision Tree, Naive Bayes, Logistic Regression, and SVM leverages their complementary strengths to produce more robust performance. Additional research has validated that ensemble approaches, particularly hybrid models combining decision trees and random forests, consistently achieve accuracies above 96% in phishing detection tasks [12,13]. Comprehensive reviews of ensemble learning methods highlight the effectiveness of techniques such as XGBoost, Random Forest, and gradient-boosting across diverse classification problems [14]. Additionally, using Mutual Information (MI) for feature selection ensured that only the most informative features were retained, reducing noise and potential overfitting. Despite efficient feature selection, the model still relied on statically selected features, which may not capture trends in dynamically evolving phishing URLs and could affect its long-term effectiveness as attacker behaviours shift. Additionally, employing multiple ensemble

techniques and sophisticated XAI methods can increase computational costs and hinder real-time implementation in resource-constrained environments.

Recent systematic reviews have comprehensively evaluated SHAP and LIME as the dominant XAI frameworks in machine learning interpretability [15,16]. These techniques have proven particularly valuable in cybersecurity applications where model transparency is essential for operational trust and regulatory compliance. Similar to XAI techniques such as SHAP and LIME, other methods can provide insights into AI models and improve end-user understanding. A counterfactual explanation (CE) approach was used in Fan et al.'s study [17] to examine the reasoning behind a trained prediction model, which provides users with a level of personalisation in the suggestions and interventions offered to reduce susceptibility to phishing attacks. A key strength of this work lies in its integration of causal reasoning within the counterfactual framework. By leveraging structural causal models and incorporating causal proximity into the counterfactual generation process, the method ensures that proposed interventions are both realistic and contextually meaningful. The framework's application to a real-world dataset comprising psychological, behavioural, and demographic features from thousands of university staff further supports its practical relevance; however, its mathematical sophistication could create barriers to real-world adoption due to resource-constrained security contexts.

2.2. Dynamic Phishing Detection

Arthy et al. [18] created a real-time phishing detection system that uses memory-augmented neural networks with dynamic contextual analysis to identify and block phishing attempts via email. Dynamic contextual analysis was used to evaluate and identify anomalies in email timing, interactions, metadata and sender behaviour, and to employ memory-augmented neural networks to retain knowledge of past detections. This system proved remarkably accurate, achieving 99.27% efficiency. A key advantage of their methodology is its adaptability to evolving phishing tactics, as the system can learn from both historical and contextual data, enhancing resilience against sophisticated attacks. The integration of XAI techniques further supports transparency and user trust. However, there are notable limitations: the computational complexity of combining MANNs and DCA may impact real-time performance and scalability, particularly in large-scale or resource-limited settings. The system's reliance on comprehensive contextual data introduces vulnerabilities if that data are incomplete or inaccurate, and its applicability to unseen real-world phishing scenarios has yet to be thoroughly validated.

Another paper focusing on the features used to detect phishing emails is Chien et al. [19], which conducted experiments to assess the weighted importance of extracted features. Features such as binary, numeric, text-based, as well as HTML tags were used, and HTML tags were among the most accurate, achieving 95% accuracy when trained and tested on phishing email datasets. This work demonstrates that the classifiers lacked accuracy when applied to real phishing emails, underscoring the need for further work to collect real emails rather than relying on the same phishing datasets.

One approach to creating a system with dynamic features is to use a sliding window technique, which processes sequential data by moving a window over the data. Both CANTINA+ [20] and PhishLang [21] employ sliding-window techniques to address the dynamic nature of phishing attacks and the computational constraints inherent in real-time detection systems. Xiang et al. [20] employed a temporal sliding-window mechanism in CANTINA+ to emulate real-world conditions, training the models on historical data and validating them on future data. Their approach systematically moved a sliding window of length L , measured in days, incrementally along the timeline and utilised a detection algorithm on webpages marked with timestamp T_i . It was done using models trained

on a dataset of phishing information with time labels within the window $[T_{i-L}, T_{i-1}]$. This temporal approach proved highly effective, achieving more than 92% true positive (TP) rate on distinct phishing tests, over 99% TP rate on nearly identical phishing tests, and approximately 1.4% false positive (FP) rate with less than 20% training phish over a two-week sliding window. The sliding window methodology enabled CANTINA+ to continuously incorporate evolving phishing patterns while maintaining computational efficiency, addressing the challenge that phishing tactics change rapidly over time. However, one limitation of this model is its sensitivity to window size; if it is too short, it may miss slower-evolving threats. If it is too large, it will reduce responsiveness to new tactics; thus, when implementing sliding windows, the most effective window size must be chosen.

Similarly, Roy and Nilizadeh [21] implemented a sophisticated sliding-window approach in PhishLang, though under different computational constraints related to Large Language Model token limitations. In both the training and inference stages, each website sample, represented as parsed HTML, is processed using this method, with the window size W adjusted to the maximum token capacity of the LLM. Importantly, this segmentation preserves contextual integrity, ensuring that each section of a webpage is examined within a unified context, and improves detection sensitivity while reducing false negatives when only a minor portion of a webpage is harmful. During inference, their sliding-window strategy employs a dynamic merging technique: if any portion of the website is identified as phishing, the entire website is labelled as phishing. Importantly, if all portions are initially assessed as benign, they are merged incrementally and reassessed to counter evasion tactics in which attackers may spread malicious content across multiple segments. This dynamic merging strategy is advantageous because it enhances robustness by increasing the likelihood of detecting emerging phishing tactics. One limitation of this methodology, however, is the computational overhead of re-evaluating chunks, which could significantly affect the model's scalability. Both implementations in [20,21] demonstrate how sliding window techniques can effectively manage the temporal dynamics of phishing evolution while addressing practical computational constraints, representing significant methodological advances in handling the dynamic features essential for robust phishing detection systems. The effectiveness of sliding window approaches has been validated across multiple domains, with applications in time series forecasting demonstrating superior performance when historical temporal data influences predictions [22,23].

Another dynamic approach was proposed in [24], with its novel contribution lying in the Phishing Email Detection System (PEDS), which can adapt to changes in newly observed behaviours. By integrating reinforcement learning, their framework continuously updates the underlying neural network architecture to handle zero-day attacks and other emerging phishing behaviours. Although their focus is on overall adaptability and robust detection, their approach suggests the potential benefits of dynamically updating feature importance. This insight underscores the need for frameworks that not only adapt during prediction but also adjust explanatory feature attributions transparently over time. Critically, while the PEDS framework introduces a highly adaptive and robust mechanism for phishing detection, it also exhibits several strengths and limitations. A key strength of the framework is its feature evaluation and reduction algorithm, which dynamically selects and ranks important features as new data arrive, ensuring the model focuses on the most relevant indicators while reducing unnecessary complexity. On the other hand, a limitation of this framework is the lack of user interpretability, as it lacks XAI mechanisms, which may erode users' trust and hinder adoption in environments where understanding model decisions is crucial.

Do et al. [25] proposed a phishing detection model based on Temporal Convolutional Network (TCN) to address limitations of conventional Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) approaches. The methodology leverages

temporal patterns in malicious URLs, demonstrating the effectiveness of sequential modelling for capturing evolving phishing tactics, achieving a 98.78% accuracy. However, the approach focuses exclusively on URL-based detection, excludes email content analysis, and lacks interpretability mechanisms, thereby limiting its applicability in scenarios requiring transparent decision-making for end users. In addition to this temporal approach, Mythili et al. [26] introduced a Temporal Naive Bayes classification framework for email spam detection that incorporates temporal dynamics into the classification process. The methodology leverages temporal features, including email arrival times, sender behaviour patterns, and the temporal distribution of keywords, to capture temporal dynamics inherent in email communication. The classifier models conditional dependencies among features within the same class, thereby accounting for sequential temporal dependencies. An experimental evaluation on the Kaggle Spam Filter dataset (5700 emails) achieved 99.13% accuracy, outperforming traditional static classifiers, including KNN, Decision Trees, SVM, and Random Forest. However, the approach focuses on spam detection rather than phishing-specific threats, lacks explainability mechanisms to provide transparency into temporal feature importance, and evaluates on a relatively small dataset compared to large-scale phishing datasets, potentially limiting generalizability to sophisticated phishing campaigns that employ distinct temporal patterns from spam.

Despite notable advances in using XAI to enhance the transparency and trustworthiness of phishing detection systems, a persistent gap remains in balancing interpretability with adaptability to evolving threats. As the literature demonstrates, and as summarised in Table 1, most successful approaches, whether SVM-based models with user-oriented visualisations, ensemble methods leveraging SHAP and LIME, or even frameworks incorporating advanced feature selection, still primarily rely on static features, limiting their ability to capture and counter rapidly changing attack vectors, a challenge frequently highlighted by recent research. When dynamic detection is pursued, such as with memory-augmented neural networks or sliding-window techniques, the integration of effective, user-friendly XAI remains insufficient or absent, often due to computational complexity or trade-offs with detection performance, motivating the need for a new XAI phishing detection system that leverages dynamic, context-aware feature analysis while maintaining interpretable, user-centred explanations, effectively bridging the gap between robust, real-time resilience and genuine transparency in cybersecurity.

Table 1. Comparative Analysis of XAI-Based and Traditional Phishing Detection Models.

Author(s)	Approach	XAI Method	Performance	Key Limitation	Type of Features
Yakandawala et al. (2024) [9]	XAI-enhanced SVM with GUI	LIME	96% (SVM)	Static approach	Static
Fajar, Yazid & Budi (2024) [10]	Feature importance with ensemble models	SHAP	<95%	Static features, high computational cost	Static
Alam et al. (2024) [11]	Explainable ensemble (E2Phish)	SHAP, LIME, PDP	Stacking 97% accuracy	Non-diverse dataset	Static
Hernandes Jr et al. (2021) [27]	URL-based RF/SVM with XAI	LIME, EBM	97–98% accuracy (RF/SVM), (EBM)	URL-only; no temporal dynamics	Static
Lim et al. (2025) [28]	Email phishing LR classifier with LLM layer	LIME, SHAP, LLM (DeepSeek v3)	98.4% accuracy	Email-only; no temporal modelling; LLM latency	Static
Fatima et al. (2025) [29]	IoT/robotics URL phishing RF detector	LIME, SHAP	98.35% accuracy (RF)	URL only; static snapshot	Static
Uddin & Sarker (2024) [30]	DistilBERT phishing email classifier	LIME, Transformer-Interpret	98.48% accuracy (balanced data)	No temporal split	Static
Alotaibi et al. (2025) [31]	XALAOA-WPC (MHA-LSTM with meta-heuristics)	LIME	99.23% accuracy	High complexity	Static

Table 1. Cont.

Author(s)	Approach	XAI Method	Performance	Key Limitation	Type of Features
S & G (2025) [18]	Memory-augmented neural networks	LIME and SHAP	99.27% efficiency	Scalability issues	Dynamic
Chien & Khethavath (2023) [19]	Multi-feature type analysis	None	<70%	Poor real-world performance	Dynamic
Xiang et al. (2011) [20]	Temporal sliding window (CANTINA+)	None	92% TP, 1.4% FP	Window size sensitivity	Dynamic
Roy & Nilizadeh (2024) [21]	LLM-based sliding window (PhishLang)	None	<93%	Computational overhead	Dynamic
Smadi, Aslam & Zhang (2018) [24]	Dynamic evolving neural network (PEDS)	None	98.6%	No interpretability	Dynamic
Do et al. (2025) [25]	Temporal Convolutional Network (TCN)	None	98.78%	No explainability; URL-focused	Dynamic
Mythili et al. (2024) [26]	Temporal Naive Bayes	None	99.13%	Spam-focused; no XAI; small dataset	Dynamic

3. Materials and Methods

To address both the black-box issue of the lack of explainability in machine learning models [32] and the lack of temporally informed phishing detection models, we propose a temporally aware dynamic feature-extraction model with XAI. Our methodology processes phishing emails using a sliding-window analysis, combined with multi-model learning and XAI, as outlined in Figure 1.

To create an XAI-based phishing detection model using dynamic features, an appropriate dataset had to be selected to accurately represent phishing emails from a copyright-free, open-source source. We used the Phishing Email Dataset, created by Naser Abdullah Alam [33,34], which combines multiple other popular phishing datasets, including Enron, Ling, CEAS, Nazario, and SpamAssassin. The dataset contains approximately 82,500 emails, of which 42,891 are phishing, and 39,595 are legitimate. The choice of this dataset is motivated by several considerations. First, it mitigates dataset-specific biases that can arise from relying on a single corpus. Second, given the sensitivity of phishing data, this publicly available dataset offers a comprehensive and ethically accessible alternative to proprietary collections. Third, the dataset was carefully merged to create a unified, near-balanced corpus. The emails are reliably labelled through a rigorous process developed by recognised research teams, including contributors from the Apache Foundation and the Enron project, to address concerns raised in prior studies about poorly validated datasets. Finally, the dataset provides a diverse and representative sample of real-world email communications, spanning multiple sources, time periods, and attack strategies, thereby enhancing the generalizability of the findings. Each email comes with a binary label ('phishing' or 'legitimate') inherited from the original source, these labels are used as-is as ground truth for supervised training and evaluation. While no dataset is entirely free from noise, using this aggregated dataset provides a broader range of phishing email types, spanning from broad-context emails to core content and subject lines, enabling a more dynamic study.

Our system was implemented in Python version 3.13.5 using several key modules. NumPy version 2.2.6 and pandas version 2.3.1 were employed to handle data frames, perform numerical computations, and preprocess data. Regular expressions, dateutil version 2.9, the parser, datetime, and time modules were used to extract email timestamps and construct sliding-window temporal features. Model development relied on scikit-learn version 1.7.1 for dataset partitioning (train_test_split), cross-validation and hyperparameter tuning (GridSearchCV), and classifiers (RandomForestClassifier, MLPClassifier, LogisticRegression), as well as the gradient-boosted tree ensemble XGBClassifier from XGBoost

version 3.0.2. Performance evaluation used scikit-learn's metrics module and matplotlib version 3.10.3 for explanatory plots. At the same time, model interpretability combined local explanations from LIME version 0.2.0.1 with global explanations from SHAP version 0.48.0 to reveal feature attributions across classifiers.

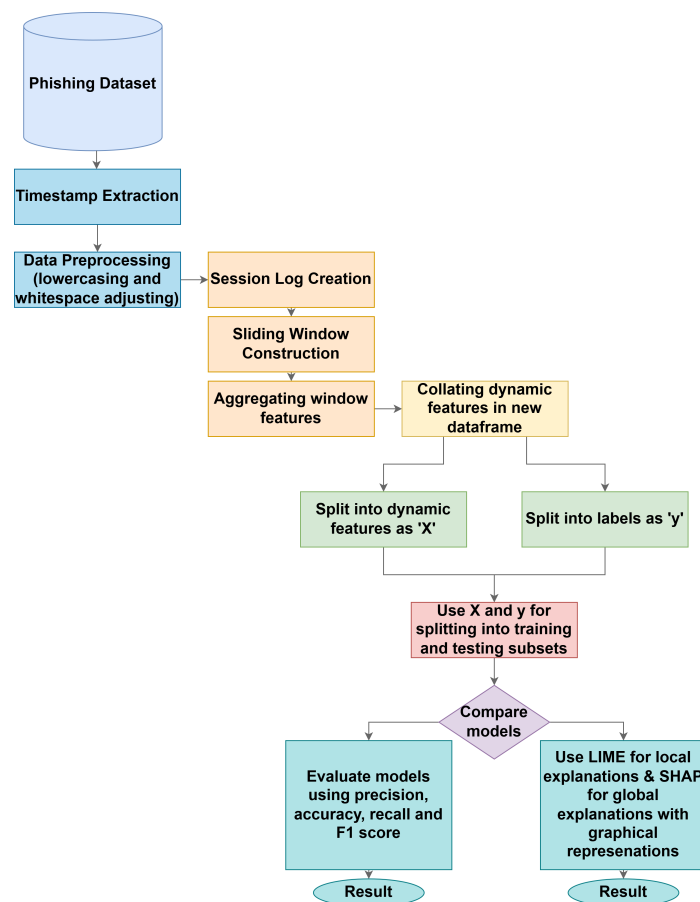


Figure 1. Methodology Overview.

3.1. Dynamic Feature Extraction Framework

3.1.1. Timestamp Extraction

To construct these dynamic temporal features, the timestamps in the dataset needed to be extracted appropriately. Because the dataset comprises multiple email formats and is a combination of datasets, a multi-pattern timestamp extraction approach was implemented, employing six regular expression patterns ordered by reliability. These include: a full language format such as “25 May 2001”; a numerical format with time like “05 30 12 07 pm”; a complete numerical date–time example “05 29 2001 08 37”; a shortened numerical format “05 25 01”; a complete language format including time as in “Thursday 28 June 2001, 9 58”; and a format capturing only the year (“2001”). These formats reflect the variety often encountered in practical timestamp parsing and validation.

An extraction algorithm was implemented, utilising RegEx, employing a first-match strategy to identify specific timestamps, cleaning up spaces and validating each timestamp to ensure it falls between 1990 and 2020. Emails for which no valid timestamp could be reliably extracted were excluded during preprocessing. Handling incomplete timestamps. For emails where only partial temporal information could be extracted (e.g., year-only or date without time), missing components were deterministically normalised to 15 June at 12:00, corresponding to the midpoint of the calendar year and day. This choice minimises systematic temporal skew while preserving relative chronological consistency across records.

3.1.2. Data Preprocessing

Unusually, as this system relies on raw data to extract timestamps, data preprocessing is not the first step in the methodology. Since timestamp extraction relies on pattern matching, it is crucial to apply preprocessing only after extracting the datetimes. Preprocessing involves converting all characters to lowercase to reduce variation, removing leading and trailing whitespace, and replacing multiple internal whitespace characters with a single space to improve text consistency and clarity. These text normalisation and cleaning steps are fundamental preprocessing operations that reduce data variability and standardise representations, significantly improving downstream model performance [35,36]. The preprocessing step largely preserves the original text, as this information is crucial for phishing detection and can serve as features in the detection process.

3.1.3. Session Log Creation

Once the timestamp extraction function was implemented, the next step was to create session logs to simulate realistic email reading behaviour through sentence-level temporal progression. This process addresses the lack of deep temporal analysis in dynamic phishing detection by creating chronologically ordered feature vectors that capture content characteristics and temporal dynamics [37].

The function begins by segmenting emails using a boundary detection mechanism to split them into sentences when encountering a full stop, exclamation mark, or question mark. In addition, a filtering mechanism was added to retain only sentences longer than five characters to ensure sufficient content for meaningful analysis. Because some data lacks punctuation, the system also segments the email by word count. If the email contains fewer than 45 words, it is split into at least three chunks. If it includes more than 45 words, the number of chunks is determined by dividing the total word count by 15. Subsequently, a realistic temporal progression simulation based on conventional reading times [38] is used to process each sentence by multiplying the word count by 0.25 to achieve a reading speed of 240 words per minute, as shown in Equation (1).

$$T_{\text{read}}(s_i) = \frac{W(s_i)}{240} \times 60 \quad (1)$$

This temporal progression does not rely on user reading behaviour that has been objectively measured, it is a heuristic estimate meant to impose a consistent temporal structure over the email text.

Complexity factors were also added because certain aspects of the text require longer reading, such as URLs, longer sentences, and numbers. This time is then added to the email's original timestamp to create an increasing timestamp sequence that models realistic reading progression through the email's content. The features discussed in the next section are then computed for each individual session log, which, along with the timestamp, sentence ID, sentence, and features, are then appended in the structure shown in Listing 1.

Listing 1: Session log creation for temporal progression.

```
# Session LOG
sessions.append({
    'timestamp': currentTime,
    'sentence_id': i,
    'sentence': processed,
    'features': features
})
```

To extract features from session logs, a function was used to identify characteristics in each sentence, and these were appended to each session log. The most basic features included the sentence length, word count, presence of a URL, and the hour of the email. Phishing emails often employ persuasive principles such as authority and scarcity, thereby creating a sense of time pressure that encourages quick, heuristic decision-making [39]. For this reason, urgency-related words such as “urgent,” “immediate,” “ASAP,” “now,” and “quickly” were included as features. Additionally, money-related terms such as “money,” “payment,” “bank,” and “account” were incorporated to capture financially oriented cues commonly observed in phishing attacks [40]. Currency symbols such as \$ and £ were also included as practical indicators of potential financial intent within the message content. Overall, these linguistic and behavioural features help reveal patterns by analysing trends and distributions across sentence groups.

3.1.4. Sliding Window Construction

The sliding window function, as implemented, creates a sophisticated temporal segmentation algorithm that transforms sequential session logs into overlapping analytical windows [41]. This method addresses the fundamental challenge of capturing both sentence-level patterns and broader contextual relationships in email content for dynamic phishing detection systems. The sliding window approach is designed to capture temporal dependencies and patterns in sequences from the email dataset, with each window representing a temporal unit, thereby enabling feature aggregation and pattern recognition.

Algorithm 1 illustrates how sliding windows operate by segmenting the session logs (each with an associated timestamp and feature vector) into overlapping windows. Each window, by default, contains seven session logs, with the windows overlapping by 30% to capture transitional patterns that span boundaries. If the email sequence of session logs is shorter than the specified size of 3, a separate function groups the session logs into a window, preserving available information and context.

In the default configuration, the sliding window size is fixed at 7 session logs with 30% overlap, motivated by the typical length of short paragraphs in phishing emails. To assess sensitivity to this design choice, additional experiments were conducted in which the XGBoost and Random Forest classifiers were trained on dynamic features constructed from session logs using window sizes of 5, 6, and 8. Window sizes of 5 and 6 led to modest performance degradations, with XGBoost accuracy decreasing to 93% in both cases and Random Forest accuracy decreasing to 92% for size 5, while remaining at 93% for size 6. For a window size of 8, Random Forest performance was unchanged, but XGBoost showed slightly lower precision and recall than the default setting. As a result of this ablation, a window size of 7 was selected as it provided the best overall balance between detection performance and computational efficiency.

3.1.5. Aggregating Window Features

As with the sentence-level features in the session logs, features must now be transformed into window-level representations. This is done by aggregating multiple sentence feature vectors within each sliding window into a single feature vector, enabling machine learning models to leverage contextual patterns and temporal information across the windows.

This function first calculates the window duration by subtracting the end time of the window’s last session log (calculated earlier by sequentially adding the sentence’s reading time to the timestamp) from the start time of the window’s first session log. This feature emphasises the dynamic capabilities of the proposed phishing detection system by capturing the temporal span of user engagement within a particular window. The other calculated

features are the sentence count, average sentence length, total words, mean number of URLs, mean number of urgency words, mean number of money terms, and average sentence duration. These aggregated features enable deep pattern recognition across windows by focusing on linguistic and temporal behaviours rather than on individual sentences.

Algorithm 1 Sliding Window Temporal Segmentation for Session Logs

```

1: function SLIDINGWINDOWSEGMENTATION(email_dataset, window_size = 7, overlap = 30%)
2:   Initialize  $W \leftarrow \{\}$  ▷ set of all windows
3:   window_id  $\leftarrow 0$ 
4:   for each email in email_dataset do
5:     Extract session logs  $S = \{s_0, s_1, \dots, s_n\}$  from email
6:     if  $|S| < 3$  then
7:       if true then
8:         Create a single window containing all available logs
9:         Preserve contextual information from partial sequence
10:      end if
11:    else
12:      if true then
13:        for  $i = 0$  to  $|S| - \text{window\_size}$  step stride do
14:          Create window  $w_i = \{s_j | j \in [i, i + \text{window\_size} - 1]\}$ 
15:          Aggregate temporal features within  $w_i$ :
16:          if true then
17:            Extract timestamp metadata
18:            Compute sentence-level patterns
19:            Calculate contextual relationships
20:          end if
21:          Append feature vector (window_id,  $w_i$ , features) to  $W$ 
22:          Increment window_id
23:        end for
24:      end if
25:    end if
26:  end for
27:  return  $W = \{w_0, w_1, \dots, w_n\}$  ▷ set of overlapping windows
28: end function

```

3.1.6. Collating Dynamic Features

To apply all functions to the dataset, the dynamic feature construction function systematically applies the appropriate functions and steps to generate new dynamic window-level feature vectors, which are then aggregated into a new data frame for use in training machine learning models.

The function shown in Algorithm 2 begins by extracting the timestamp from the raw emails to record the sending time, enabling subsequent analysis of session logs to be viewed in a real temporal context. A new column is then added to the dataset, with emails without timestamps filtered out, to align features with their extracted timestamps. Each email's text is divided into sentences, and reading times are simulated to generate a chronological sequence of session logs. Each log captures the sentence content, its local timestamp from simulated reading, and relevant sentence-level features. The ordered session logs are then grouped into overlapping windows using a sliding window function, with a default window size of 7 session logs and 30% overlap, thereby preserving the order of sentences and their temporal context and enabling the detection of short-range behavioural patterns, such as monetary and urgency terms.

The windowed sentence features are aggregated into a single vector using the designated feature aggregation function. Figure 2 shows that the resulting vector contains window and email identifiers, the true label of each email (0 or 1), associated temporal metadata, and the extracted timestamp. The aggregation ensures that each window cap-

tures essential structural and contextual information necessary for downstream analysis and classification.

Algorithm 2 Temporal Session Log Windowing and Feature Aggregation

```

1: Initialize processed_df  $\leftarrow$  empty, window_id  $\leftarrow$  0
2: function SESSIONLOGWINDOWING(email_dataset)
3:   for each email in email_dataset do
4:     Extract timestamp; if missing, skip email
5:     for each sentence do
6:       Simulate local timestamp based on reading speed
7:       Extract sentence-level features (e.g., monetary/urgency terms)
8:       Store sentence logs with content, local timestamp, and features
9:     end for
10:    Group sentence logs (ordered by local timestamp) into overlapping windows
11:    Set window_size = 7, overlap = 30%
12:    for each window do
13:      Aggregate sentence features within the window
14:      Construct feature vector:
15:      if true then
16:        window_id, email_id
17:        ground-truth label
18:        window temporal metadata
19:        original email timestamp
20:      end if
21:      Append feature vector to processed_df
22:      Increment window_id
23:    end for
24:  end for
25:  return processed_df
26: end function
27: Invoke SESSIONLOGWINDOWING(email_dataset)
28: Process and structure output for downstream machine learning

```

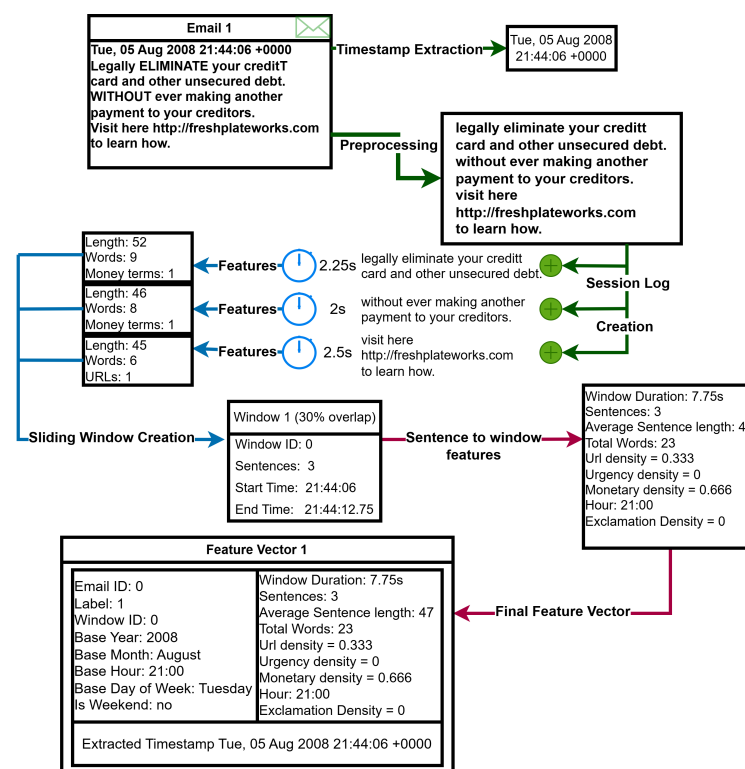


Figure 2. Dynamic Feature Extraction Process

3.2. Model Training Pipeline

3.2.1. Dataset Preparation

After applying the dynamic features function to the original dataset, a comprehensive data frame was created, with each row corresponding to a windowed segment of an email and containing temporal and aggregated phishing trigger features. After this, to ensure model stability, metadata was removed from the column, which would be used for training, as including the email_id, label, window_id and timestamp could cause the model to cheat by being provided the label, as well as the rest of the metadata being irrelevant to the prediction, so it was not needed. It then formed the feature matrix X , representing the input variables for each sample, and extracted the 'label' column to create the target vector y , which contained the true labels for the phishing emails.

The data were then split into training and testing subsets using a stratified split to preserve class distribution, with 80% of samples used for training and 20% reserved for testing. A random state of 42 is used to fix the random seed, ensuring reproducibility across experiments, and stratification is used to preserve class proportions in both the training and testing subsets. All models are trained once on the dynamic feature matrix, no online learning or continual retraining is performed in this study. All results are reported on a held-out test set for consistency across configurations. As a result, the reported results and conclusions correspond to point estimates from a controlled, reproducible experimental setting.

3.2.2. Training Models

To classify emails as phishing or legitimate, four supervised machine learning algorithms were used: Random Forest (RF), Extreme Gradient Boosting (XGBoost), Multi-layer Perceptron (MLP), and Logistic Regression, all from the scikit-learn library. These classifiers were chosen because they encompass a range of algorithms, including ensemble methods, gradient-boosting, neural networks, and statistical methods, demonstrating a diverse set of classification approaches. These models are commonly used in the literature, with random forests and XGBoost often achieving very high accuracies [42–44].

During model selection, Scikit-learn's GridSearchCV was employed to optimise hyperparameters for each classifier, ensuring the highest classification accuracy as supported in related literature [45,46]. The Random Forest (RF) model was implemented with 200 estimators to balance performance and computational efficiency, while maintaining reproducibility through a fixed random state. The XGBoost classifier was trained with 300 boosting rounds, a maximum tree depth of 10, and a learning rate of 0.1 to achieve stable, controlled improvements and avoid overfitting. The Multi-Layer Perceptron (MLP) model featured a two-layer structure with 200 and 50 neurons, respectively, and was trained for 1000 iterations to capture nonlinear relationships while minimizing overfitting. Lastly, Logistic Regression employed the liblinear solver with an L1 penalty ($C = 0.1$) for regularisation and interpretability in binary classification.

To rigorously assess model performance in phishing email detection, four complementary evaluation metrics were employed: accuracy, precision, recall, and F1-score. Accuracy was included as a general indicator of overall classifier performance; however, given the potential imbalance between phishing and legitimate emails, it alone was insufficient for reliable evaluation. Precision was selected to quantify the model's ability to avoid false positives, which is critical in minimising the misclassification of legitimate emails as phishing, thereby reducing user disruption and unnecessary alerts.

Recall was chosen to measure how effectively the classifier identifies actual phishing attempts, helping ensure that real threats are not overlooked, an essential requirement in security contexts. Finally, the F1-score was used to balance precision and recall, providing

a single, harmonised metric that reflects the trade-off between detecting phishing emails and maintaining classification precision. Together, these metrics offer a comprehensive and reliable evaluation of each model's practical effectiveness in phishing detection.

3.2.3. Computational Environment

All experiments were implemented in Jupyter Notebook and executed on a desktop PC with an AMD Ryzen 7 7800X3D 8-core processor (16 logical threads), 32 GB RAM, and an NVIDIA GeForce RTX 4070 Ti SUPER GPU. On this configuration, the dynamic feature-extraction pipeline required approximately 29 s to process the full dataset, while the Random Forest model required around 14 s to train and the XGBoost model approximately 2 s to train.

3.3. Implementing XAI

3.3.1. Setting up LIME

To improve the interpretability of the trained classifiers' results, XAI was implemented using the LIME framework. LIME provides local explanations of classification outcomes and offers users deeper insight into how the classifier makes decisions, either in textual or graphical form. The explanations approximate the classification decision with a simple, interpretable model in the specific instance, highlighting which features most strongly support a given prediction [47]. LIME finds the optimal interpretable model by solving the following optimisation problem seen in Equation (2) [47]:

$$E(x) = \arg \min_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g) \quad (2)$$

In this formulation, g represents a simple linear model selected from the class of interpretable models G . The term $\mathcal{L}(f, g, \pi_x)$ denotes a locality-aware loss function that measures how accurately g approximates the predictions of the complex model f within the local region defined by the proximity measure π_x . The complexity term $\Omega(g)$ acts as a regulariser that penalises overly complex explanations, thereby maintaining interpretability. The optimisation ensures that the resulting explanation faithfully represents the model's local decision boundary while remaining accessible and comprehensible to human users. Due to its ability to provide robust and interpretable explanations of complex models, LIME was chosen as the XAI of choice [48].

To set up LIME, shown in Listing 2, the LimeTabularExplainer was initialised using the training feature matrix, along with the feature names, the class names of 'Phishing' and 'Legitimate', as well as setting it in classification mode. Subsequently, the explainer was applied to each classifier model to generate local explanations for specific emails by perturbing the input features and observing changes in the classifier's output probabilities. LIME then fitted a simple, interpretable graphical representation of horizontal bars that showed which features were most important in its classification decisions.

Listing 2: LIME explainer setup for the Random Forest model.

```
# Initialize LIME explainer
explainer = LimeTabularExplainer(
    X_train.values,
    feature_names=feature_columns,
    class_names=['Legitimate', 'Phishing'],
    mode='classification'
)

# Generate explanation for an email instance
instance = X_test.iloc[0].values
```

```

exp = explainer.explain_instance(
    instance,
    rf_model.predict_proba,
    num_features=10
)

# Creating the graph
fig = exp.as_pyplot_figure()
plt.title('LIME Explanation - Random Forest')
plt.show()

```

3.3.2. Setting up SHAP

To view explanations of the classifiers at a global rather than local level, SHAP was implemented, utilising beeswarm visualisations to reveal feature importance and impact patterns across the dataset. For each email prediction, SHAP values explain the model's decision by quantifying how much each feature contributes to the classification of phishing versus legitimate, accounting for feature interactions and dependencies [49]. The SHAP value for feature i is defined in Equation (3);

$$\phi_i(f, x) = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} [f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S)] \quad (3)$$

where F is the set of all features, $S \subseteq F \setminus \{i\}$ is a coalition excluding i , and $f_S(x_S)$ is the model's prediction using only features in S . This computes the average marginal contribution of feature i across all coalitions [49].

The SHAP values were computed using the explainer object, with each trained classifier and a sample size of 10,000 from the training data (due to computational constraints, the whole dataset was not used). These SHAP values quantify each feature's contribution to a specific prediction; once calculated by the explainer, they can be applied to multiple visual and graphical representations.

The beeswarm plot is used to visualise the SHAP values for this method. It was chosen because it is a benchmark representation used in many studies and is crucial for understanding feature importance, model bias, and potential areas for improvement [50]. The beeswarm plot shows dots representing instances, positioned horizontally according to their SHAP values: positive SHAP values extend to the right, indicating that the feature increases the probability of phishing, whereas negative values decrease it. These dots are also colour-coded, with blue indicating low values and red indicating high values, allowing identification of feature-prediction relationships. The vertical distribution of density also provides insights: dense clusters indicate consistent feature impact, whereas wider spreads indicate more variable influence.

4. Results

4.1. Dataset Preprocessing and Feature Extraction Results

The dynamic, temporally aware preprocessing pipeline demonstrated strong performance across the diverse dataset. The extraction process successfully generated 164,302 windows, each containing an average of 7 session logs, yielding 2.6 windows per email. It produced a dynamic feature matrix comprising 164,302 samples $\times$ 14 features, including dynamic temporal features such as *window_duration*, *average_sentence_duration*, *base_hour*, *base_month*, *base_year*, *base_day_of_week*, alongside content-based features encompassing *sentence_count*, *average_sentence_length*, *total_words*, *URL_density*, *urgency_density*, and *money_density*.

4.2. Classifier Performance Comparison

The results obtained from the models are presented in Table 2. The precision, recall, and F1 score reported in Table 2 correspond to weighted averages across classes, as computed using the standard classification report. Overall, XGBoost and Random Forest achieve the most balanced and accurate classification results, with weighted performance scores of 94% and 93%, respectively.

Table 2. Performance comparison of machine learning models using weighted average precision, recall, and F1 score (best results in bold).

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	93%	93%	93%	93%
XGBoost	94%	94%	94%	94%
MLP	87%	87%	87%	87%
Logistic Regression	81%	81%	81%	80%

In addition, the confusion matrices shown in Figures 3 and 4 reveal clear performance hierarchies and specific strengths and weaknesses of each approach in processing temporal phishing features. They show that XGBoost achieved the fewest false positives and false negatives, with 874 and 1181, respectively. In security, false negatives are critical because they represent missed threats; thus, showing that logistic regression had 4348 false negatives compared with XGBoost (1181) demonstrates the practical value of tree-based approaches such as Random Forest and XGBoost for real-world deployment versus other methods, such as neural networks and linear approaches.

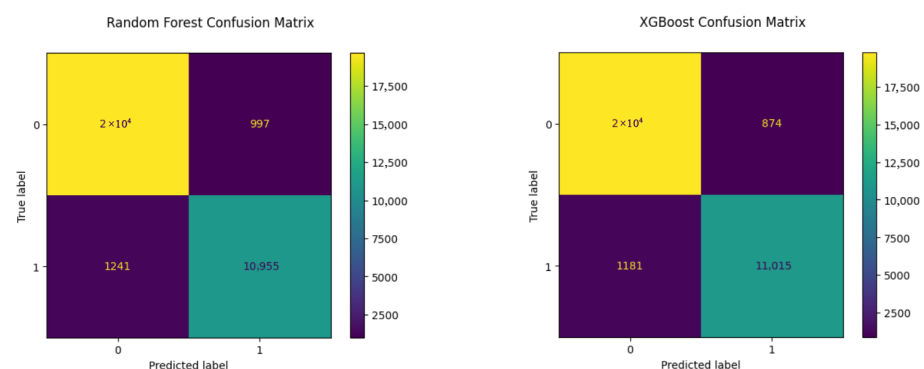


Figure 3. Random Forest and XGBoost Confusion Matrices.

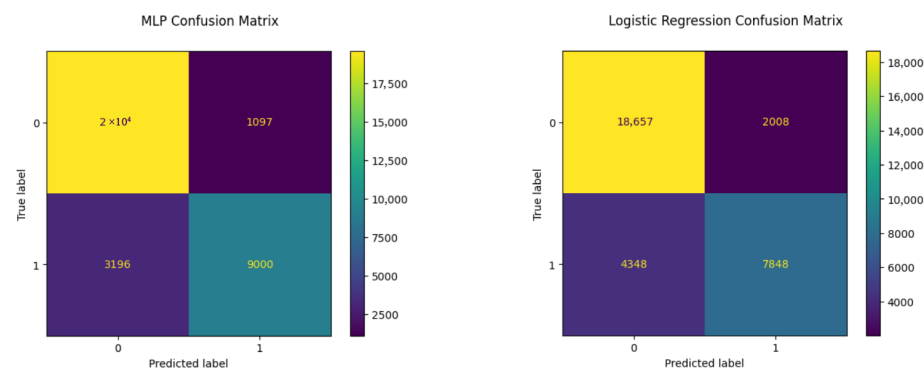


Figure 4. MLP and Logistic Regression Confusion Matrices.

4.3. XAI LIME Results

LIME was applied to each model using the same instance to explain its window-level aggregated feature hierarchy, thereby improving usability. The LIME explanations reveal consistent patterns in feature importance across all four classifiers, supporting the robustness of the dynamic feature extraction methodology. *Money_density* emerges as the strongest positive indicator across Random Forest, XGBoost, MLP, and Logistic Regression, consistently contributing the most significant positive influence toward phishing classification (see Figures 5–8). This alignment demonstrates that the temporal sliding window approach successfully captures universal phishing behavioural patterns. It also shows how urgency density consistently appeared as a negative contributor across all models, indicating that the absence of urgency terms is causing the emails to be classified as legitimate.

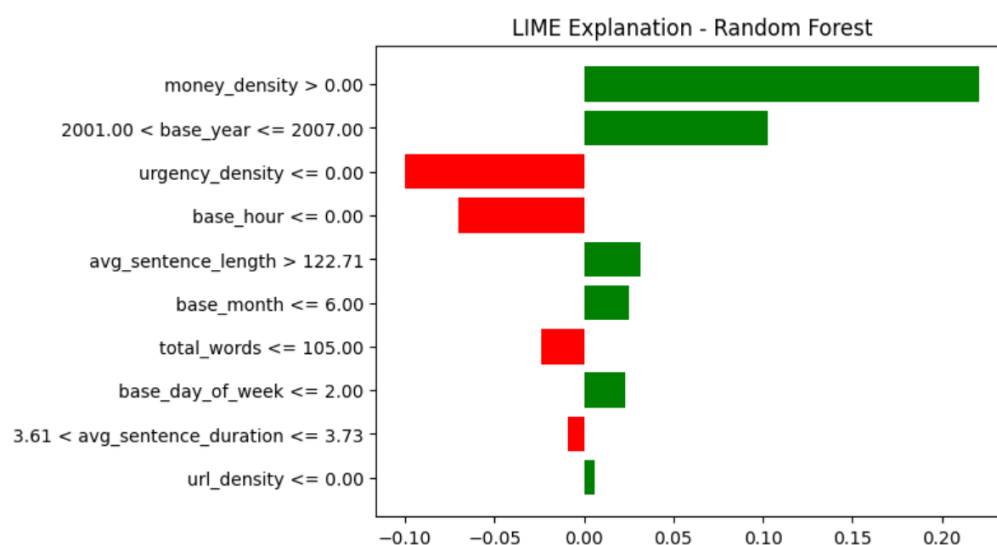


Figure 5. Random Forest LIME Graph showing LIME feature attributions (x-axis).

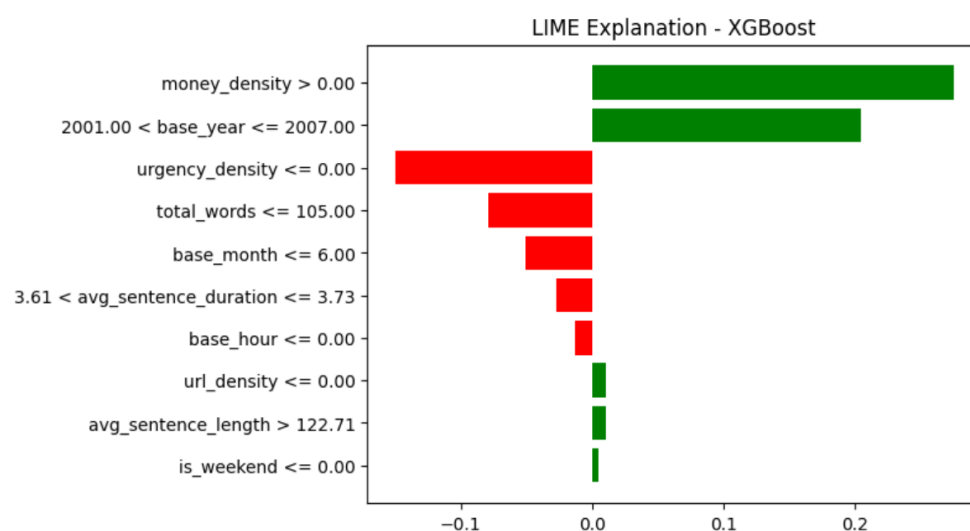


Figure 6. XGBoost LIME Graph showing LIME feature attributions (x-axis).

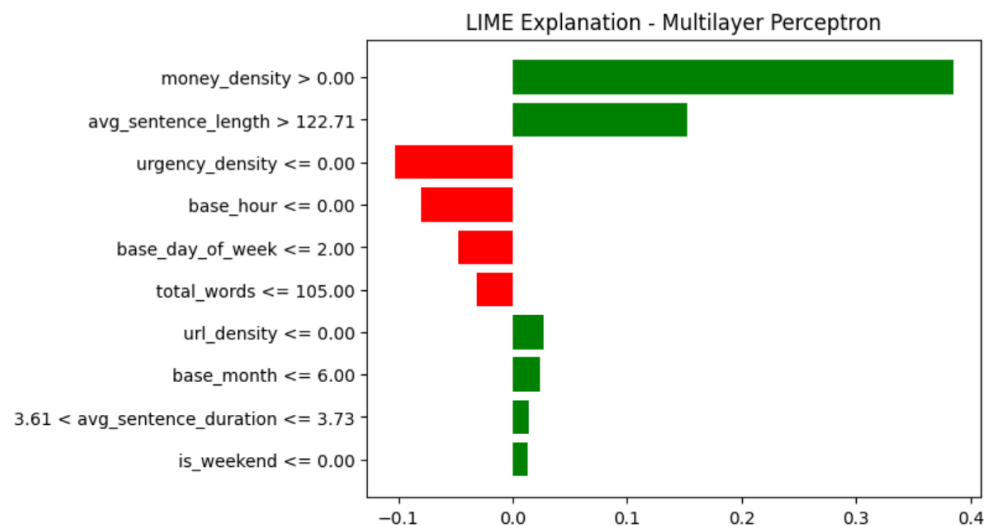


Figure 7. MLP LIME Graph showing LIME feature attributions (x-axis).

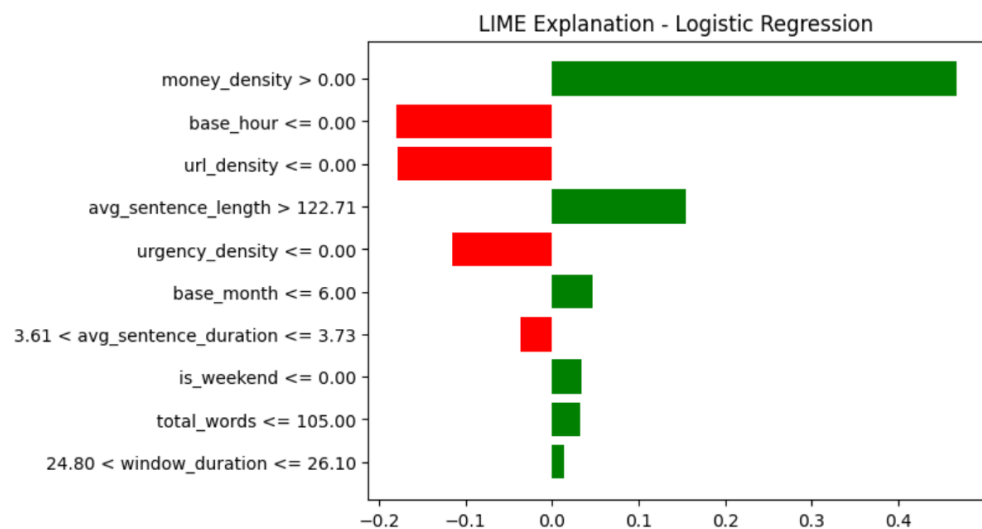


Figure 8. Logistic Regression LIME Graph showing LIME feature attributions (x-axis).

The Random Forest and XGBoost models, shown in Figures 5 and 6, had similar feature rankings, with *money_density*, *base_year*, and *urgency_density* as the top features for classification. The temporal catching features of the *base_year* and *base_hour* have a strong influence on these models, suggesting that they better capture temporal decision boundaries.

The multilayer perceptron model, LIME explanation shown in Figure 7, places average sentence length higher than the other models and exhibits more balanced feature contributions to classification, indicating that neural networks use a broader feature set than other classifiers. The LIME explanation of the logistic regression model, shown in Figure 8, illustrates the most diverse pattern in how the features have been utilised. *URL_density* shows a greater negative contribution than the other models, which may explain its lower accuracy.

4.4. XAI SHAP Results

SHAP was also applied to each classifier using a sample size of 10,000 to obtain a global overview of feature importance. In the Random Forest SHAP beeswarm, shown in Figure 9, *base_hour* is the most influential feature, with a strongly bidirectional effect: some hours show negative SHAP values (blue, left) while others contribute positively (red, right), indicating pronounced temporal patterns learned by the model. *Base_year* exhibits a

large, asymmetric spread with higher years pushing SHAP values to the right, consistent with temporal drift toward higher predicted risk in more recent messages. *Money_density* shows a clear pattern, where a greater density of monetary terms produces a long red right tail of positive contributions. *Avg_sentence_length* has a moderate, mixed-direction impact: unusually short or long sentences nudge predictions, but most points cluster near zero. *Url_density* is generally low overall, yet a subset of high-value cases shifts predictions to the right, implying that links increase risk in specific emails. Seasonal and weekly effects from *base_month* and *base_day_of_week* are present but modest, centred near zero with small left/right shifts. Finally, *window_duration* displays a small-to-moderate positive skew, suggesting certain durations increase predicted risk, though its overall contribution is secondary.

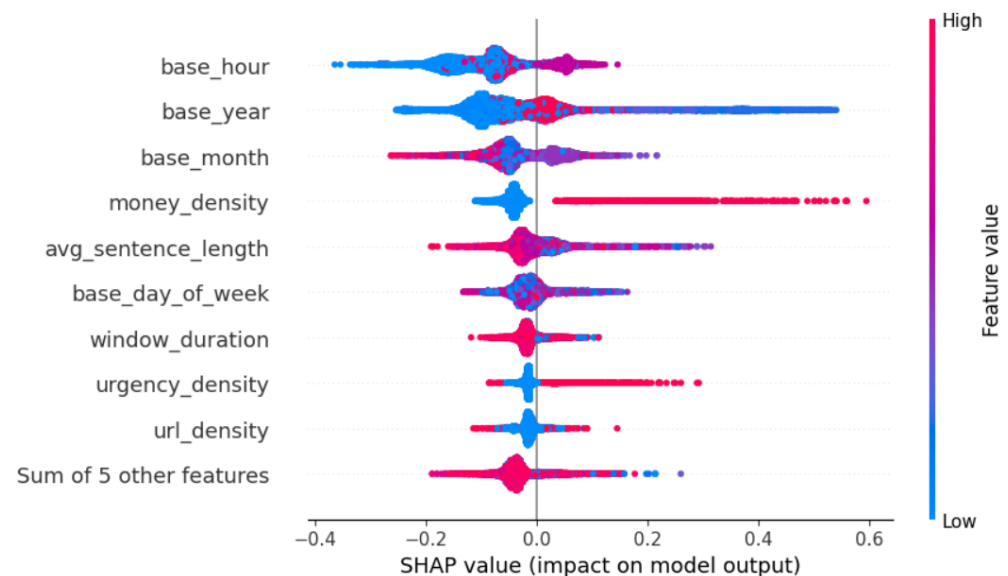


Figure 9. Random Forest SHAP Plot.

The XGBoost SHAP beeswarm, shown in Figure 10, indicates that *base_year* is the dominant driver: more recent years (high values) strongly push predictions positive, while earlier years pull them negative, indicating temporal drift. *Money_density* exhibits a clear monotonic right shift, with higher concentrations consistently increasing the prediction. *Base_month* and *base_hour* provide modest, bidirectional nudges reflecting seasonal and diurnal patterns without large fluctuations. *Avg_sentence_length* has limited influence overall; both very short and very long sentences can matter slightly, with a mild right tail for longer sentences. *Base_day_of_week* stays centred near zero, suggesting minimal weekly cyclic effects. Finally, *total_words* and *window_duration* have small, mixed impacts, indicating secondary roles in the model's decisions.

Beeswarm plots were also generated for MLP and logistic regression classifiers. The models echoed similar results but spread influence more evenly across length and window features, yielding smaller, mixed contributions. Across models, temporal features (particularly *base_year* and, to a lesser extent, *base_hour*/*base_month*) are the principal determinants of predictions, with money-related language providing the most consistent content-based lift.

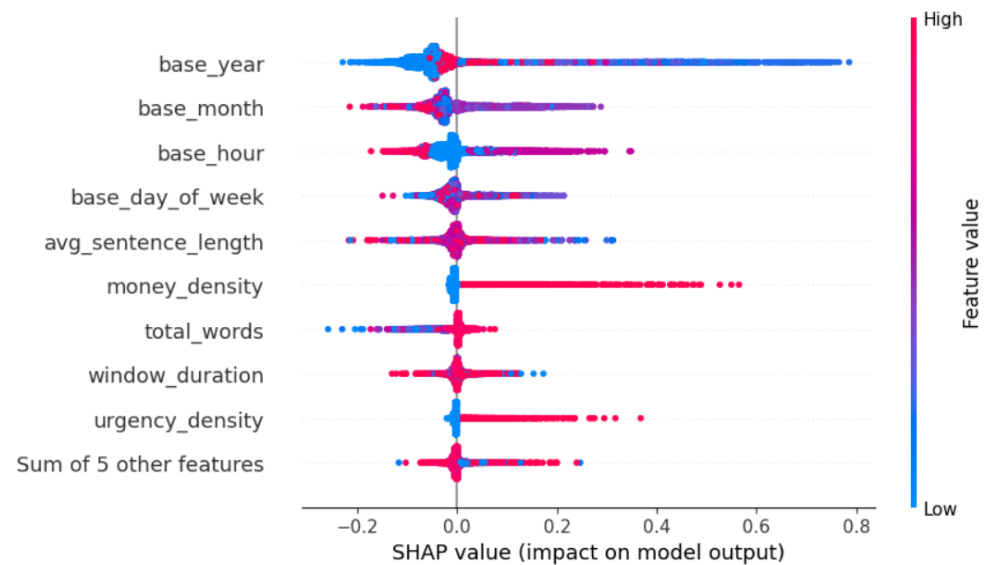


Figure 10. XGBoost SHAP Plot.

5. Benchmarking with Existing Works

This section benchmarks the proposed dynamic and explainable phishing detection approach against existing works in the literature. The comparison focuses on detection performance, computational practicality, and explainability, highlighting how the proposed method balances accuracy and interpretability relative to static, deep learning, and hybrid approaches.

This study shows that the dynamic features derived from the sliding-window representation of phishing emails, when paired with explainable tree ensembles, provide a balance of accuracy, operational reliability, and transparency. We demonstrated that the dynamic features used in the classifiers achieved strong and consistent performance across multiple classifiers, with tree ensembles performing best (XGBoost 94%, Random Forest 93%) and reducing false negatives, which is a critical aspect of phishing detection (see Section 4.2). High levels of explainability accompanied these high-performance results at both the local (LIME) and global (SHAP) levels. The explanations in Sections 4.3 and 4.4 highlighted that money-related content and temporal signs (*base_year*, *base_hour*) are dominant drivers for classification, indicating that the method captures both evolving phishing dynamics and content in a way that is transparent and actionable.

The strong base-year and base-hour effects highlighted by the XAI feature-importance visualisations indicate temporal drift and time-of-day patterns in phishing that static feature sets miss, underscoring the importance of dynamic features, as a sliding window preserves short-range sequential cues and improves sensitivity to evolving tactics. Ensemble classifiers were also found to enhance the use of dynamic features by reducing false negatives across classifiers, suggesting that interactions between temporal metadata and content characteristics are key to practical detection.

Recent studies provide a useful benchmark for evaluating the performance of our proposed dynamic feature-extraction framework, as summarised in Table 3. Al-Subaiey et al. [34] reported notably high accuracies of 99.1% using Support Vector Machine (SVM) and 98.4% using Random Forest (RF) with their static Term Frequency-Inverse Document Frequency (TF-IDF)-based approach. However, the variant of their approach using word2vec embeddings produced substantially lower results, achieving 83.8% with RF and 82.1% with SVM. These findings highlight the strong influence of feature representation on model behaviour.

They also show that while dynamic methods introduce valuable contextual information, traditional static text representations can still capture important phishing indicators.

Table 3. Comparison with existing works.

Study	Dataset	Methods	Accuracy	Precision	Recall	F1-Score
(Al-Subaiey et al., 2024) [34]	Phishing Email	RF (W2Vec),	83.8%,	83%,	84%,	83%,
		SVM (W2Vec),	82.1%,	82%,	81%,	82%,
		RF (TF-IDF),	98.4%,	98%,	99%,	98%,
		SVM (TF-IDF)	99.1%	99%	99%	99%
(Damatie, Eleyan & Bejaoui, 2024) [51]	Phishing Email	DistilBERT	99.3%,	99.3%,	99.3%,	99%,
		(controlled), DistilBERT (real world)	95.4%	87.5%	100%	93.3%
(Salian, 2024) [52]	Phishing Email	Hybrid Distil- BERT + SVM,	97%,	97%, 94%	97%,	97%,
		SVM	94%		94%	94%
Our Study	Phishing Email	RF,	93%,	93%,	93%,	93%,
		XGBoost,	94%,	94%,	94%,	94%,
		MLP,	87%,	87%,	87%,	87%,
		LR	81%	81%	81%	80%

Damatie, Eleyan, and Bejaoui [51] further illustrated the sensitivity of model performance to deployment conditions. In their work, DistilBERT’s accuracy decreased from 99.3% in controlled experiments to 95.4% in real-world evaluations, demonstrating both the robustness and the limitations of deep contextual models in real-world settings. Their dynamic framework provided comparable performance, albeit slightly lower, while offering enhanced interpretability, a key consideration for operational use.

Salian [52] showed that hybrid classification approaches can achieve strong results (97%) but require substantially higher computational cost. This performance is only marginally higher than that achieved by standard SVM or XGBoost classifiers using dynamic features (94%), suggesting that the accuracy gains from hybrid pipelines may not justify the additional complexity in many practical scenarios.

In contrast, our study demonstrates that a carefully engineered dynamic extraction pipeline, when combined with efficient classical models such as Random Forest and XGBoost, achieves competitive performance, with accuracies of 93% and 94%, respectively. It also outperforms the word2vec-based model of Al-Subaiey et al. [34] while remaining lightweight, scalable, and far easier to train, deploy, and interpret. The combination of accuracy, efficiency, and practicality positions our approach as a compelling alternative to more complex deep-learning architectures, particularly in real-world environments where computational constraints, transparency, and operational scalability are essential considerations.

6. Discussion and Future Directions

This study advances phishing detection research by moving beyond conventional performance-centric evaluation toward a feature-driven, temporally informed understanding of model behaviour. While prior studies predominantly report aggregate metrics such as accuracy and F1-score, our results demonstrate that these measures alone are insufficient for assessing trustworthiness, adaptability, and operational viability of phishing detection systems in dynamic or real-world settings. By integrating dynamic feature extraction with model interpretability techniques, this work introduces a principled framework for analysing why phishing detection models succeed or fail.

A key contribution is the systematic application of LIME and SHAP to quantify feature importance at both local and global levels. This dual interpretability enables granular, per-email explanations alongside dataset-wide insights, revealing the temporal and content cues that consistently drive model decisions.

The interpretability-driven results have direct operational implications. SHAP-based global explanations guide feature prioritisation and retraining strategies by identifying attributes that consistently mitigate false negatives, while LIME enables instance-level auditing of model decisions. By integrating temporal analysis and explainability into phishing detection pipelines, this work enables continuous monitoring of feature relevance, cross-model interpretability, and adaptive defense against evolving phishing strategies, which are largely absent in prior studies.

The use of sliding-window temporal preprocessing is a critical component of this study. By segmenting email activity into overlapping temporal windows, the proposed preprocessing strategy preserves both local temporal continuity and global dataset coverage. This design enables models to capture patterns in message frequency, timing irregularities, and burst-like behaviour, phenomena commonly associated with coordinated phishing campaigns. From an operational perspective, sliding-window preprocessing also offers practical advantages. It supports scalable, streaming-friendly implementations suitable for real-time email filtering systems, where decisions must be made continuously as new messages arrive.

The proposed temporally dynamic feature extraction framework and dual XAI analysis are generalizable beyond phishing emails to other sequential, timestamped data domains where understanding key patterns that occur within a sequence is valuable. This approach could be used for social media threat detection, where timestamped posts are entered into the system for analysis of harassment and threats, as well as the temporal campaign involved. It could also be used for spam emails rather than phishing to capture campaign timing and windows to identify evolving promotional or urgency patterns. Finally, it could also be used in log-based anomaly detection, where it treats system logs as session logs, with XAI identifying anomalous temporal sequences and when they occur, such as rapid login/logout.

Limitations and Future Directions

Despite these strengths, several limitations remain. The temporal progression in our framework is synthetically constructed from a fixed reading-speed heuristic rather than derived from user-behaviour data. The timestamp extraction process failed to recover timestamps for all emails, potentially introducing bias into the temporal feature distributions. Our deterministic normalisation using fixed temporal midpoints preserves chronological order but provides only coarse temporal context. Although absolute timestamp filtering yields higher apparent accuracy, it substantially reduces dataset coverage and risks temporal shortcut learning, underscoring the importance of robustness over peak performance. This trade-off is acknowledged as a limitation of this work. In addition, using fixed window sizes and overlap limits the system's ability to adapt to phishing campaigns that evolve across varying temporal scales. Although the input representation is temporally dynamic, the explanations generated by LIME and SHAP are static in the sense that they operate on individual windows or collections of windows, and do not yet provide explicit sequence-level or transition-level explanations across the full temporal trajectory of an email. As well as this the present work employs static batch training and does not explicitly model or mitigate concept drift. These constraints highlight opportunities for refinement rather than fundamental shortcomings of the approach.

Future research directions include calibrating the temporal model using real interaction logs or user studies and examining the sensitivity of detection performance to different temporal assumptions. Further work should improve the robustness of timestamp extraction and explore adaptive windowing strategies [53] that can respond to both slow- and fast-evolving phishing behaviours. Adaptive strategies such as online learning, contin-

ual learning, or drift-aware retraining have been shown to be beneficial in maintaining long-term performance. Extending the proposed temporally structured feature framework with such adaptive mechanisms is, therefore, a key avenue for future work. Incorporating TF-IDF vectorisation alongside dynamic features may further enhance detection performance, provided that the balance between static textual representations and temporal dynamics is carefully maintained. Moreover, investigating lightweight hybrid classifiers or ensemble-stacking approaches could improve robustness while preserving explainability. Finally, extending the framework to support more adaptive temporal modelling within XAI-enhanced systems may facilitate deployment in real-time environments, addressing ongoing challenges in achieving efficient, interpretable, and resilient phishing detection.

7. Conclusions

This study addressed the gap between high-performing but non-explanatory phishing detection systems and the need for methods that both adapt to evolving tactics and remain interpretable. The proposed temporal-aware dynamic feature methodology, built on session-level progression, sliding-window aggregation, machine-learning classifiers, and XAI explanations, leverages realistic reading-time simulation and windowed feature aggregation to capture both behavioural patterns and temporal dynamics. The proposed framework enabled classifiers to integrate content-based indicators with contextual progression, providing a robust foundation for detecting sophisticated phishing attempts.

The results confirm the effectiveness of the proposed approach, with tree ensemble models delivering strong performance: XGBoost achieved 94% accuracy and Random Forest 93%. Both local (LIME) and global (SHAP) explanations revealed that money-related content and temporal cues, such as the *base_year* and *base_hour*, are dominant drivers of classification decisions. These findings demonstrate that adaptability and interpretability are not mutually exclusive, showing that compact, explainable dynamic features can support real-time phishing resilience while maintaining transparency and manageable computational cost. Overall, the findings suggest that future phishing detection research should move beyond static performance benchmarks and incorporate dynamic, explainable analysis as a core criterion for evaluation.

Author Contributions: Conceptualisation, C.M., M.T., S.B. and N.E.; methodology, C.M.; software, C.M.; validation, C.M., M.T., S.B. and N.E.; formal analysis, C.M.; Investigation, C.M.; data curation, C.M.; writing—original draft preparation, C.M.; writing—review and editing, M.T., S.B. and N.E.; visualisation, C.M.; supervision, M.T., S.B. and N.E.; project administration, M.T., S.B. and N.E. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The source code and artefacts of this research are available online at: <https://github.com/chrismayooo/XAI-Phishing-Detection-System-using-Dynamic-Features>. All resources are released under an open-source license for academic and research purposes, facilitating reproducibility and enabling further investigation in this domain.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Singh, P.; Hasija, T.; Ramkumar, K. Machine Learning Algorithms for Phishing Detection: A Comparative Analysis of SVM, Random Forest, and CatBoost Models. In Proceedings of the 2024 Second International Conference on Intelligent Cyber Physical Systems and Internet of Things (ICoICI), Coimbatore, India, 28–30 August 2024; pp. 1421–1426. <https://doi.org/10.1109/ICoICI62503.2024.10696365>.
2. Akinyelu, A. Machine Learning and Nature Inspired Based Phishing Detection: A Literature Survey. *Int. J. Artif. Intell. Tools* **2019**, *28*, 1930002. <https://doi.org/10.1142/S0218213019300023>.

3. Wilk-Jakubowski, J.; Pawlik, Ł.; Wilk-Jakubowski, G.; Sikora, A. Machine Learning and Neural Networks for Phishing Detection: A Systematic Review (2017–2024). *Electronics* **2025**, *14*, 3744. <https://doi.org/10.3390/electronics14183744>.
4. Sarker, I.; Janicke, H.; Mohsin, A.; Gill, A.; Maglaras, L. Explainable AI for cybersecurity automation, intelligence and trustworthiness in digital twin: Methods, taxonomy, challenges and prospects. *ICT Express* **2024**, *10*, 935–958. <https://doi.org/10.1016/j.ict.2024.05.007>.
5. Brohi, S.; Mastoi, Q.u.a. AI Under Attack: Metric-Driven Analysis of Cybersecurity Threats in Deep Learning Models for Healthcare Applications. *Algorithms* **2025**, *18*, 157. <https://doi.org/10.3390/a18030157>.
6. Zhang, Z.; Hamadi, H.A.; Damiani, E.; Yeun, C.Y.; Taher, F. Explainable Artificial Intelligence Applications in Cyber Security: State-of-the-Art in Research. *IEEE Access* **2022**, *10*, 93104–93139. <https://doi.org/10.1109/ACCESS.2022.3204051>.
7. Brohi, S.; Mastoi, Q.u.a. From Accuracy to Vulnerability: Quantifying the Impact of Adversarial Perturbations on Healthcare AI Models. *Big Data Cogn. Comput.* **2025**, *9*, 114. <https://doi.org/10.3390/bdcc9050114>.
8. Srivastava, G.; Jhaveri, R.H.; Bhattacharya, S.; Pandya, S.; Rajeswari; Maddikunta, P.K.R.; Yenduri, G.; Hall, J.G.; Alazab, M.; Gadekallu, T.R. XAI for Cybersecurity: State of the Art, Challenges, Open Issues and Future Directions. *arXiv* **2022**, arXiv:2206.03585.
9. Yakandawala, Y.L.D.H.; Madushanka, M.K.P.; Ilmini, W.M.K.S. Explainable AI for Transparent Phishing Email Detection. In Proceedings of the 2024 International Conference on Advances in Technology and Computing (ICATC), Kelaniya, Sri Lanka, 19 December 2024; pp. 1–6. <https://doi.org/10.1109/ICATC64549.2024.11025302>.
10. Fajar, A.; Yazid, S.; Budi, I. Enhancing phishing detection through feature importance analysis and explainable ai: A comparative study of catboost, xgboost, and ebm models. *arXiv* **2024**, arXiv:2411.06860. <https://doi.org/10.48550/arXiv.2411.06860>.
11. Alam, R.; Khune, A.; Kalal, T.V.; Nautiyal, A. E2Phish: Explainable Ensemble Machine Learning Model for Enhanced Phishing URL Detection. In Proceedings of the 2024 IEEE 8th International Conference on Information and Communication Technology (CICT), Prayagraj UP, India, 6–8 December 2024; pp. 1–6. <https://doi.org/10.1109/CICT64037.2024.10899721>.
12. Makkar, A.; Kumar, N.; Sama, L.; Mishra, S.; Samdani, Y. An intelligent phishing detection scheme using machine learning. In Proceedings of the Sixth International Conference on Mathematics and Computing: ICMC 2020, Gangtok, Sikkim, India, 18–20 March 2020; pp. 151–165. https://doi.org/10.1007/978-981-15-8061-1_13.
13. Maturure, P.; Ali, A.; Gegov, A. Hybrid machine learning model for phishing detection. In Proceedings of the 2024 IEEE 12th International Conference on Intelligent Systems (IS), Varna, Bulgaria, 9 October 2024; pp. 1–7. <https://doi.org/10.1109/IS61756.2024.10705257>.
14. Omari, K. Phishing Detection using Gradient Boosting Classifier. *Procedia Comput. Sci.* **2023**, *230*, 120–127. <https://doi.org/10.1016/j.procs.2023.12.067>.
15. Roshinta, T.A.; Gábor, S. A Comparative Study of LIME and SHAP for Enhancing Trustworthiness and Efficiency in Explainable AI Systems. In Proceedings of the 2024 IEEE International Conference on Computing (ICOCO), Kuala Lumpur, Malaysia, 12–14 December 2024; pp. 134–139. <https://doi.org/10.1109/ICOCO62848.2024.10928183>.
16. Desai, B.; Patil, K.; Mehta, I.; Patil, A. Explainable AI in Cybersecurity: A Comprehensive Framework for enhancing transparency, trust, and Human-AI Collaboration. In Proceedings of the 2024 International Seminar on Application for Technology of Information and Communication (iSemantic), Semarang, Indonesia, 21–22 September 2024; pp. 135–150. <https://doi.org/10.1109/iSemantic63362.2024.10762690>.
17. Fan, Z.; Li, W.; Laskey, K.B.; Chang, K.C. Towards Personalized Anti-Phishing: Counterfactual Explanation Approach-Extended Abstract. In Proceedings of the 2024 IEEE 11th International Conference on Data Science and Advanced Analytics (DSAA), San Diego, CA, USA, 6–10 October 2024; pp. 1–2. <https://doi.org/10.1109/DSAA61799.2024.10722801>.
18. Arthy, S.; Sakthi Priya, G. A Hybrid Machine Learning Approach for Securing Emails: Phishing Detection and Prevention. In Proceedings of the 2025 3rd International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA), Coimbatore, India, 4–5 April 2025; pp. 1–6. <https://doi.org/10.1109/ICAECA63854.2025.11012305>.
19. Chien, A.; Khethavath, P. Email Feature Classification and Analysis of Phishing Email Detection Using Machine Learning Techniques. In Proceedings of the 2023 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE), Nadi, Fiji, 4–6 December 2023; pp. 1–8. <https://doi.org/10.1109/CSDE59766.2023.10487729>.
20. Xiang, G.; Hong, J.; Rosé, C.; Cranor, L. CANTINA+: A Feature-Rich Machine Learning Framework for Detecting Phishing Web Sites. *ACM Trans. Inf. Syst. Secur.* **2011**, *14*, 21. <https://doi.org/10.1145/2019599.2019606>.
21. Roy, S.; Nilizadeh, S. PhishLang: A Lightweight, Client-Side Phishing Detection Framework Using MobileBERT for Real-Time, Explainable Threat Mitigation. *arXiv* **2024**, <https://doi.org/10.48550/arXiv.2408.05667>.
22. Norwawi, N.M. 29-Sliding window time series forecasting with multilayer perceptron and multiregression of COVID-19 outbreak in Malaysia. In *Data Science for COVID-19*; Kose, U., Gupta, D., de Albuquerque, V.H.C., Khanna, A., Eds.; Academic Press: Cambridge, MA, USA, 2021; pp. 547–564. <https://doi.org/10.1016/B978-0-12-824536-1.00025-3>.

23. Ndungi, R.; Stanislavovich, L.I. Improving time series forecasting by applying the sliding window approach. In Proceedings of the International Conference on Intelligent and Fuzzy Systems (INFUS 2025), Istanbul, Turkey, 29–31 July 2025; pp. 294–302. https://doi.org/10.1007/978-3-031-98304-7_34.
24. Smadi, S.; Aslam, N.; Zhang, L. Detection of online phishing email using dynamic evolving neural network based on reinforcement learning. *Decis. Support Syst.* **2018**, *107*, 88–102. <https://doi.org/10.1016/j.dss.2018.01.001>.
25. Do, N.Q.; Selamat, A.; Krejcar, O.; Fujita, H. Detection of malicious URLs using Temporal Convolutional Network and Multi-Head Self-Attention mechanism. *Appl. Soft Comput.* **2025**, *169*, 112540. <https://doi.org/10.1016/j.asoc.2024.112540>.
26. Mythili, J.; Deebeshkumar, B.; Eshwaramoorthy, T.; Ajay, J.N. Enhancing Email Spam Detection with Temporal Naive Bayes Classifier. In Proceedings of the 2024 International Conference on Communication, Computing and Internet of Things (IC3IoT), Chennai, India, 17–18 April 2024; pp. 1–6. <https://doi.org/10.1109/IC3IoT60841.2024.10550229>.
27. Hernandez, P.R.G.; Floret, C.P.; De Almeida, K.F.C.; Da Silva, V.C.; Papa, J.P.; Da Costa, K.A.P. Phishing detection using URL-based XAI techniques. In Proceedings of the 2021 IEEE Symposium Series on Computational Intelligence (SSCI), Orlando, FL, USA, 24 January 2021; pp. 01–06. <https://doi.org/10.1109/SSCI50451.2021.9659981>.
28. Lim, B.; Huerta, R.; Sotelo, A.; Quintela, A.; Kumar, P. EXPLICATE: Enhancing Phishing Detection through Explainable AI and LLM-Powered Interpretability. *arXiv* **2025**, <https://doi.org/10.48550/arXiv.2503.20796>.
29. Fatima, Z.; Tanveer, M.H.; Voicu, R.C.; Chakravarty, S.; Ashfaq, M.; Khan, M.; Zaib, A.; Desa, H. Explainable AI for IOT Devices and Robotic Communication Phishing Detection: A Machine Learning Approach Using LIME and SHAP. *Eng. Technol. Appl. Sci. Res.* **2025**, *15*, 26478–26486. <https://doi.org/10.48084/etasr.11595>.
30. Uddin, M.A.; Mahiuddin, M.; Sarker, I.H. An Explainable Transformer-based Model for Phishing Email Detection: A Large Language Model Approach. *arXiv* **2025**, <https://doi.org/10.48550/arXiv.2402.13871>.
31. Alotaibi, R.; Alkahtani, H.; Aljebreen, M.; Alshuhail, A.; Saeed, M.; Ebad, S.; Almukadi, W.; Alotaibi, M. Explainable artificial intelligence in web phishing classification on secure IoT with cloud-based cyber-physical systems. *Alex. Eng. J.* **2024**, *110*, 16. <https://doi.org/10.1016/j.aej.2024.09.115>.
32. Brohi, S.; Mastoi, Q.u.a.; Jhanjhi, N.Z.; Pillai, T.R. A Research Landscape of Agentic AI and Large Language Models: Applications, Challenges and Future Directions. *Algorithms* **2025**, *18*, 499. <https://doi.org/10.3390/a18080499>.
33. Alam, N.A. Phishing Email Dataset. 2024. Available at: <https://www.kaggle.com/ds/5074342> (accessed on 16 May 2025).
34. Al-Subaiey, A.; Al-Thani, M.; Abdullah Alam, N.; Antora, K.F.; Khandakar, A.; Uz Zaman, S.A. Novel interpretable and robust web-based AI platform for phishing email detection. *Comput. Electr. Eng.* **2024**, *120*, 109625. <https://doi.org/10.1016/j.compeleceng.2024.109625>.
35. Gonzalez Zelaya, C.V. Towards Explaining the Effects of Data Preprocessing on Machine Learning. In Proceedings of the 2019 IEEE 35th International Conference on Data Engineering (ICDE), Macao, China, 8–11 April 2019; pp. 2086–2090. <https://doi.org/10.1109/ICDE.2019.00245>.
36. Rao, S.; Poojary, P.; Somaiya, J.; Mahajan, P. A Comparative Study Between Various Preprocessing Techniques for Machine Learning. *Int. J. Eng. Appl. Sci. Technol.* **2020**, *5*, 431–438. <https://doi.org/10.33564/IJEAST.2020.v05i03.069>.
37. Peng, T.Q.; Zhu, J.J.H. Understanding Online Behaviors through a Temporal Lens. *arXiv* **2023**, arXiv:2301.05996.
38. Brysbaert, M. How many words do we read per minute? A review and meta-analysis of reading rate. *J. Mem. Lang.* **2019**, *109*, 104047. <https://doi.org/10.1016/j.jml.2019.104047>.
39. Auton, J.C.; Sturman, D. Persuasion under pressure: The influence of persuasion principles and time constraints on phishing email susceptibility. *Inf. Comput. Secur.* **2025**, *33*, 845–859. <https://doi.org/10.1108/ICS-07-2024-0163>.
40. Ferreira, A.; Teles, S. Persuasion: How phishing emails can influence users and bypass security measures. *Int. J. Hum.-Comput. Stud.* **2019**, *125*, 19–31. <https://doi.org/10.1016/j.ijhcs.2018.12.004>.
41. Keogh, E.; Chu, S.; Hart, D.; Pazzani, M. Segmenting Time Series: A Survey and Novel Approach. *Data Min. Time Ser. Databases* **2003**, *57*, 1–22. https://doi.org/10.1142/9789812565402_0001.
42. Navaneethakannan, B.; Gokul, C.; Swathi, R.; Mohanraj, K.C. Phishing Detection and Response System with XGBOOST Machine Learning. In Proceedings of the 2025 International Conference on Emerging Smart Computing and Informatics (ESCI), Pune, India, 5–7 March 2025; pp. 1–6. <https://doi.org/10.1109/ESCI63694.2025.10988346>.
43. Sadaf, K. Phishing Website Detection using XGBoost and Catboost Classifiers. In Proceedings of the 2023 International Conference on Smart Computing and Application (ICSCA), Hail, Saudi Arabia, 5–6 February 2023; pp. 1–6. <https://doi.org/10.1109/ICSCA57840.2023.10087829>.
44. Keerthana, B.P.; Siva, A.; Senthilkumar, T.; Palanisamy, T.; Prabhu, N.; Srinivasan, K. Web Phishing Detection Using Decision Tree Random Forest and XGBoost. In Proceedings of the 2025 International Conference on Inventive Computation Technologies (ICICT), Kirtipur, Nepal, 23–25 April 2025; pp. 1079–1083. <https://doi.org/10.1109/ICICT64420.2025.11005064>.
45. Sawe, L.; Gikandi, J.; Kamau, J.; Njuguna, D. Sentence Level Analysis Model for Phishing Detection Using KNN. *J. Cyber Secur.* **2024**, *6*, 25–39. <https://doi.org/10.32604/jcs.2023.045859>.
46. Dasgupta, S.; Sen, J. A Comparative Study of Hyperparameter Tuning Methods. *arXiv* **2024**, arXiv:2408.16425.

47. Ribeiro, M.T.; Singh, S.; Guestrin, C. “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, San Francisco, CA, USA, 13–17 August 2016; pp. 1135–1144. <https://doi.org/10.1145/2939672.2939778>.
48. Prayogo, R.D.; Alfisyahrin, A.R.; Gambetta, W.; Karimah, S.A.; Nambo, H. An Explainable Machine Learning-Based Phishing Website Detection using Gradient Boosting. In Proceedings of the 2024 International Conference on Information Technology Research and Innovation (ICITRI), Jakarta, Indonesia, 5–6 September 2024; pp. 76–81. <https://doi.org/10.1109/ICITRI62858.2024.10698870>.
49. Lundberg, S.M.; Lee, S.I. A unified approach to interpreting model predictions. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 4768–4777. <https://doi.org/10.48550/arXiv.1705.07874>.
50. Sivasankar, C.; Tamilarasan, A.; Christy, S.; Parthiban, S. Towards Practical Phishing Detection: Addressing Challenges with Hybrid Machine Learning Architectures. In Proceedings of the 2025 International Conference on Computational Robotics, Testing and Engineering Evaluation (ICCRTEE), Virudhunagar, India, 28–30 May 2025; pp. 1–4. <https://doi.org/10.1109/ICCRTEE64519.2025.11053024>.
51. Damatie, E.M.; Eleyan, A.; Bejaoui, T. Real-Time Email Phishing Detection Using a Custom DistilBERT Model. In Proceedings of the 2024 International Symposium on Networks, Computers and Communications (ISNCC), Washington, DC, USA, 22–25 October 2024; pp. 1–6. <https://doi.org/10.1109/ISNCC62547.2024.10759011>.
52. Salian, S.S. Enhancing Phishing Email Detection with Sentiment Analysis: A Hybrid Approach. Master’s Thesis, National College of Ireland, Dublin, Ireland, 2024. Available online: <https://norma.ncirl.ie/8363/> (accessed on 15 May 2025).
53. Bifet, A.; Gavalda, R. Learning from time-changing data with adaptive windowing. In Proceedings of the 2007 SIAM International Conference on Data Mining. SIAM, Minneapolis, MN, USA, 26–28 April 2007; pp. 443–448. <https://doi.org/10.1137/1.9781611972771.42>.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.