FMDB *Transactions on Sustainable Computer Letters*

PhishGuard: Interpretable Machine Learning for Real-Time Phishing Detection on Mobile Devices

Farhan Tariq[1,*], Maruf Farhan[2]

[1,2]Department of Computer and Information Sciences, Northumbria University, London, England, United Kingdom.
farhan.tariq@northumbria.ac.uk[1], marufrigan9@gmail.com[2]

Abstract: Phishing attempts abuse human behaviour to steal login passwords, financial data, and personal information. Mobile users must quickly and accurately detect phishing emails owing to limited computational resources. Mobile phishing detection app PhishGuard uses machine learning and user-friendly explainability to deliver interpretable feedback. DistilBERT, Logistic Regression, and XGBoost were trained and tested on 57,804 safe and phishing emails. Accuracy, precision, recall, and F1 were assessed. DistilBERT surpassed Logistic Regression at 97.87% and XGBoost at 96.76% with 99.10% accuracy, 99.35% recall, and 99.10% F1-score. XGBoost and Logistic Regression performed well, while DistilBERT had the best predictive performance, mobile application balance, and phishing email detection. PhishGuard delivers a human-readable confidence score, risk indicator, and reason summary for decision-making. Abnormal words, urgency indicators, and sender dissimilarity enable Phishguard to bypass black-box constraints and boost user confidence. After execution-time investigation, DistilBERT achieved real-time mobile inference with strong recall and F1-score. A complete phishing-detection pipeline, including model performance, interpretability, and mobile viability, is provided. Ensemble learning may boost robustness; LIME and SHAP interpretability approaches need more research; continuous learning should adapt to future treatments; and practical implementation and usability user testing are ongoing. The experiment shows that machine learning with practical, interpretable outcomes can help consumers avoid phishing and make informed decisions. DistilBERT improved mobile email security, deployment, and performance.

Keywords: Email Classification; Phishing Detection; DistilBERT Model; Ensemble Learning; Logistic Regression; Confidence Score; Risk Indicator; Conventional Methods; Support Vector Machine; Random Forest.

Received on: 15/04/2025, **Revised on:** 02/07/2025, **Accepted on:** 07/08/2025, **Published on:** 29/06/2026

Journal Homepage: https://www.fmdbpub.com/user/journals/details/FTSCL

DOI: https://doi.org/10.69888/FTSCL.2026.000699

Cite as: F. Tariq and M. Farhan, "PhishGuard: Interpretable Machine Learning for Real-Time Phishing Detection on Mobile Devices," *FMDB Transactions on Sustainable Computer Letters*, vol. 4, no. 2, pp. 65–96, 2026.

1. Introduction

While email is one of the most common ways people communicate in both their professional and personal lives, it's also a key attack vector for cybercriminals. One of the most common and harmful threats users may be exposed to is phishing, in which an attacker composes a message that urges users to provide sensitive information, e.g., usernames and passwords, financial data, or personal information [1]. The emergence of advanced phishing practices, such as spear phishing and domain

[*]Corresponding author.

impersonation, makes it difficult to identify these attacks with conventional methods [2]. The limited screen space and small resolutions of mobile devices create further problems on small screens, and with high user multitasking, subtle signs that something is off in an email are often ignored. Since mobile phones are now the most common method for accessing emails, there's a much higher chance that users will interact with phishing emails. Timely and precise detection of phishing attacks on mobile platforms has thus emerged as an important research issue in cybersecurity. Conventional methods for phishing detection, such as rule-based filters and blocklists, cannot keep pace with ever-changing attack tactics. These techniques use fixed patterns and known bad sources; they have limitations that make them vulnerable to slight content variations introduced by attackers. Machine learning-based solutions are more general, as they can learn from complex patterns in email content, associated metadata, and sender activity [3]. Yet, many of the best-performing machine learning and deep learning models are black boxes that make predictions with no clear explanation. This lack of interpretability can undermine users' trust and potentially impede practical deployment, especially when users are required to act on system warnings. Further, Deep learning models are computationally intensive, raising questions about whether they can be deployed on mobile devices [4].

1.1. Aims and Objectives

This study presents the development of a mobile-based phishing-detection application, PhishGuard, designed to deliver high detection performance, interpretability, and actionable outputs for end users [2]. The application adopts a user-centric design, ensuring that phishing predictions are not only accurate but also understandable and practical for real-world decision-making. The primary objective is to evaluate the effectiveness of contemporary machine learning models in achieving high precision, recall, and F1-score for phishing email classification using a large, balanced dataset. Particular emphasis is placed on recall and F1-score, as phishing detection requires minimising false negatives to reduce potential security risks. In addition to detection performance, the study investigates the integration of explainable artificial intelligence (XAI) features, including confidence scores, risk indicators, and concise explanatory feedback, within a mobile interface to improve user comprehension, trust, and decision-making. Furthermore, the study examines the trade-offs between model complexity, classification performance, computational efficiency, and real-time deployment on resource-constrained mobile devices. For practical deployment, the DistilBERT model is selected for online inference due to its lightweight architecture, enabling efficient, low-latency phishing detection while maintaining competitive classification performance [1].

1.2. Work Done and Key Results

To achieve these results, PhishGuard is a phishing detection app that integrates machine learning and mobile app design. Three models, DistilBERT, Logistic Regression, and XGBoost, were trained and evaluated on an augmented dataset with 57,804 emails consisting of phishing as well as legitimate samples. By increasing the data, the model's generalisation to a wider range of phishing tactics was improved and made more robust. Model performance was measured in terms of accuracy, precision, recall and F1-score. Given that phishing emails belonged to the minority class, recall and F1-score were favoured over avoiding false negatives [3]. DistilBERT recorded the best overall performance, with accuracy, recall, and F1-score all at 99.10%, 99.35%, and 99.10%, respectively, among the models tested. These findings demonstrate DistilBERT's promising ability to detect phishing emails while preserving computational efficiency for deployment on mobile devices. Besides predictive accuracy, PhishGuard provides interpretable results that aim at guiding the user's decision. The system reports confidence scores, risk indicators, and brief validity reason summaries that highlight suspicious language, urgency terms, or anomalous sender behaviour. Such features mitigate the limitations of black-box models, providing clear explanations of predictions and improving user trust and usability, at the cost of mobile responsiveness [5].

2. Literature Review

The background work for this paper covers the main areas concerning email phishing detection through machine learning and user interaction [4]. It reviews phishing threats, machine learning and deep learning approaches, including transformer-based models, real-time detection studies, and deployment challenges [6]. The user-perception and explainability literature is reviewed to identify limitations in existing systems and to inform the creation of a mobile detection system.

2.1. Phishing and Email-Based Cyber Threats

Phishing is a cyberattack in which attackers try to trick users into revealing sensitive information, e.g., user credentials or financial or personal information, by pretending to be legitimate sources. Email phishing remains one of the most popular and successful types of phishing attacks, exploiting the pervasive use of email for personal and business communications. On the social engineering side, these attacks prey on human emotions such as urgency, authority, or fear to influence recipients into clicking malicious links or opening harmful attachments. Email has remained the most common way for phishing attacks due to its cost-effectiveness, reach, and credibility [8]. The Anti-Phishing Working Group (APWG) reports that phishing has grown significantly over the last decade, targeting individuals, businesses, and critical infrastructure worldwide. Successful

phishing attacks can result in direct financial losses, identity theft, system damage, and unauthorised access [7]. Data breaches are another source of these costs, which is a major concern. Consequently, phishing-induced problems represent one of the high-impact, ongoing threats. The growing professionalism of phishing campaigns, supported by automation tools and readily available phishing kits, has further reduced the technical barrier for attackers.

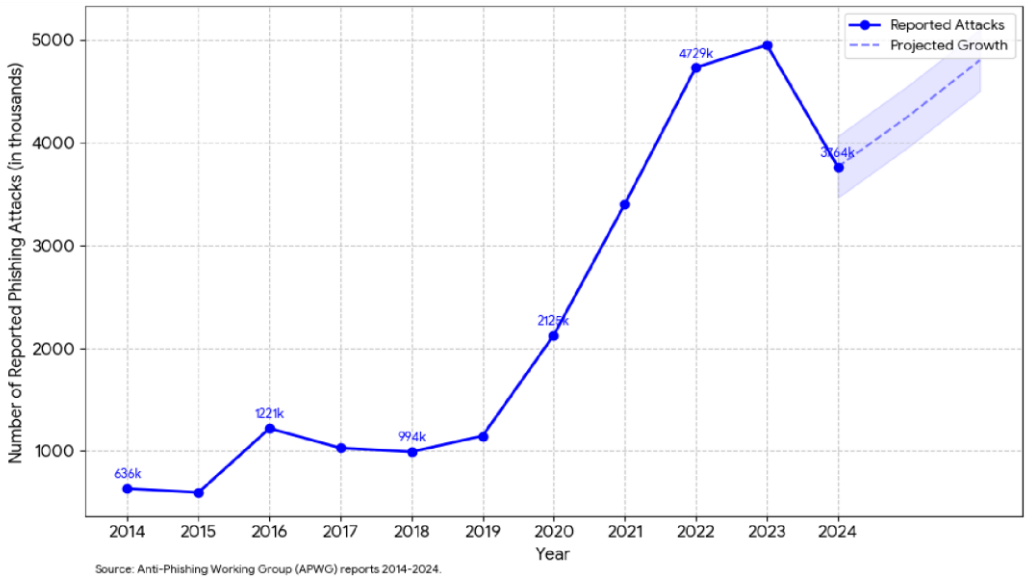


Figure 1: Phishing attacks (2014–2024) from APWG reports

Figure 1 presents long-term trends in reported phishing activity from 2014 to 2024, aggregated from APWG reports. Figure 1 shows a clear upward trend. The increase aligns with a worldwide move towards remote work and digitisation during COVID-19. And although there is a minor dip in 2024, the total number of phishing attacks across years is still much higher than what was observed before 2020. This implies that phishing isn't just a temporary phenomenon but a continuous and evolving threat. Despite awareness and user education, phishing attacks remain remarkably successful. Existing studies reveal that user awareness alone is insufficient, as phishing emails increasingly resemble legitimate ones and leverage context-aware personalisation. When users are suffering from cognitive overload, time pressure or minimal technical knowledge, then they can still be caught out. As a consequence, relying solely on human judgment is futile against sophisticated phishing tactics. This limitation has led to the development of technical countermeasures, such as rule-based filters and machine-learning detection systems [7]. Classic methods fail to keep up with dynamic threat vectors or perform in real time. These difficulties underscore the need for intelligent, automated detection software that can analyse email content dynamically and provide users with immediate, actionable advice. This motivation forms the backbone of the subsequent sections, in which researchers discuss machine learning- and deep learning-based approaches for classifying email phishing attacks and their application in real-time user-interacting systems.

2.2. Evolution of Phishing Detection Techniques

Early detection methods in phishing were mostly rule-based and blacklist-driven. These systems used manually created heuristic rules, a keyword-based approach to detect inline phishing indicators, and lists of known malicious URLs or domains to flag potential phishing. Methods including URL tokenisation and pattern matching have been extensively employed to detect suspicious links and e-mail content. Although efficient against earlier known threats, such procedures have been essentially reactive [9]. They needed to be manually updated regularly and couldn't adequately detect zero-day phishing attacks, in which malicious emails or websites hadn't yet been reported. To overcome some of these constraints, heuristic and signature-based detection approaches have been introduced. These techniques examined behavioural and structural features, e.g., anomalous email headers, mismatches in sender domains, suspicious anchor text, and irregular URL structures. While heuristics improved detection performance over static rule-based systems, they remained susceptible to evasion techniques. Creating detection based on fixed heuristics soon became a low-success-rate defence as attackers adopted techniques such as concealing URLs, randomising text content, and misusing legitimate cloud or email services [10].

Furthermore, because maintaining and growing databases of such signatures is so time- and effort-intensive, scalability was severely limited. The inefficiencies of traditional phishing-detection methods have driven a shift toward machine-learning-based approaches. Machine learning-enabled systems can learn discriminatory patterns directly from historical datasets, rather

than relying on present rules. It is widely accepted that supervised classifiers such as Naive Bayes, Support Vector Machine (SVM), Logistic Regression, Random Forests, and ensemble models perform better than rule-based techniques because they generalise well. These methods commonly involved feature engineering, e.g., lexical, structural, and statistical features extracted from email content and URLs. Although machine learning approaches greatly enhanced detection capability, they were still affected by feature quality, sample imbalance, and concept drift over time. Phishing detection was further improved using deep learning techniques that reduce reliance on handcrafted features [11]. Architectures of neural networks, including Convolutional Neural Networks and Long Short-Term Memory networks, showed promising results in capturing intricate textual and sequential patterns embedded in phishing emails (Figure 2).

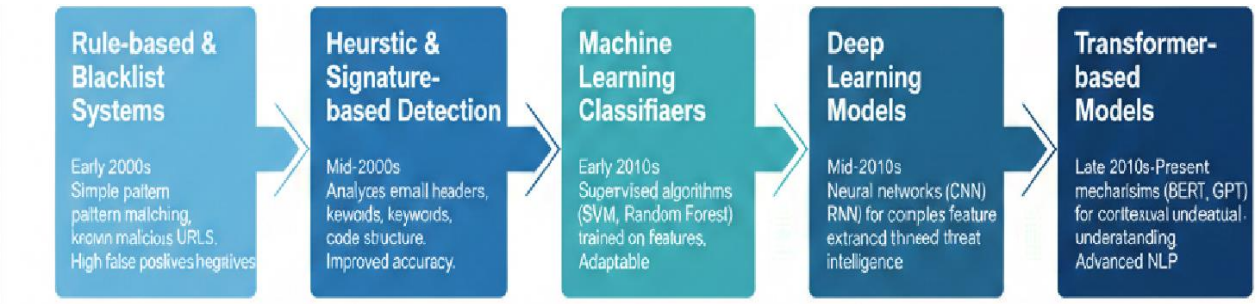


Figure 2: Progression of phishing detection techniques: rule-based → ML → DL → transformer models

These methods allowed the models to learn features automatically but raised concerns about computational cost, interpretability, and deployment complexity. These limitations were especially important for real-time systems and resource-constrained environments. More recently, transformer-based models have become the state of the art in phishing detection. Models utilising BERT and its derivatives employ attention to capture deep semantic information in email text, thereby achieving improved detection performance compared to prior approaches. However, the transformer models also have an inherent trade-off between performance and efficiency, especially required in mobile and real-time scenarios. This development clearly points to progress from static, rule-based detection methods to adaptive, data-driven methods, but also reveals some research challenges regarding lightweight deployment and user-friendliness. And all of them affect the design decisions made in this paper.

2.3. Machine Learning Approaches for Email Phishing Detection

The limitations of rule-based and heuristic-based approaches to phishing detection were the primary drivers of the emergence of machine learning-based approaches to email phishing detection. Machine learning approaches learn discriminative patterns from a history of detected emails and can generalise beyond known signatures. Classical supervised algorithms are thus widely applied in the literature to detect phishing emails [12]. Traditional machine learning methods often use probabilistic and linear classifiers, e.g., Naïve Bayes and Logistic Regression. Naive Bayes became popular mainly because of its simplicity and low computational overhead, making it attractive for large-scale spam filtering systems. Logistic Regression, which is also interpretable, was particularly effective when combined with engineered text features. Nevertheless, these models are highly dependent on feature independence or linear separability and are significantly challenged by the complex language structures in phishing emails. Some of these limitations were addressed using Support Vector Machines, which could learn nonlinear decision boundaries via a kernel function. There is evidence that vector machines perform very well on high-quality features, especially when derived from email headers, URLs, and lexical content. The ensemble methods of Random Forest and XGBoost also enhanced detection accuracy by combining multiple weak learners to reduce variance and overfitting. For instance, XGBoost has been reported to be resilient to an imbalanced phishing dataset because it can assign larger weights to misclassified samples during training [13].

Feature engineering is a characteristic of the classical machine learning methods, which are all feature-dependent. Some of the features include term frequency-inverse document frequency, n-grams, URL properties, HTML structure, and metadata extracted from email headers [14]. Feature engineering is a labour-intensive and highly expertise-demanding process. Worse still, handcrafted features may become fragile as the attackers learn to change their tactics to avoid known detection cues. This results in a constant treadmill of maintenance, as functionality needs to be updated to prevent it from becoming ineffective. The properties of the dataset affect the performance of machine learning-based phishing detection systems. Many datasets used for study purposes are small or curated and cannot capture the full diversity of real phishing campaigns. Models trained on these datasets can often achieve high accuracy in offline evaluations but are prone to failing when deployed into dynamic environments. Class imbalance is also an enduring issue, given the vast majority of legitimate emails versus the relatively few phishing emails. While resampling and cost-sensitive learning can partially alleviate imbalance, they can introduce bias or

deteriorate generalisation if not used appropriately. Sensitive to dataset drift, classical machine learning models are not invincible either. Phishing attacks keep changing, leading the statistical features of arriving emails to drift over time. Historical models get outdated; their performance degrades unless they're continuously retrained.

Retraining is expensive and may also be infeasible in terms of the time required, if not the resources, for such learning strategies to take place. In mobile and real-time environments, these limitations are exacerbated. Feature extraction pipelines for traditional machine learning techniques often include a series of preprocessing stages that incur end-to-end latency and overhead. This is a barrier to reliable online detection, especially on resource-constrained mobile phones that lack sufficient processing or battery power. In addition, most machine learning algorithms are black boxes to the user; they do not readily explain why an email was classified as a phishing attack [15]. This opacity may erode user confidence and undermine the effectiveness of phishing prevention measures. While classical machine learning methods are a definite advancement over manually created rule-based systems, the need for handcrafted feature engineering, their sensitivity to data quality issues, and deployment difficulties limit their applicability to current-day phishing detection requirements. These issues have led to the move of deep learning and transformer-based models to automate feature learning and achieve better robustness, generalisation, and adaptivity. This shift is further investigated in the subsequent sections, with a focus on systems designed for real-time phishing detection for users.

2.4. Deep Learning Techniques for Phishing Detection

Email phishing detection has been substantially improved by a deep learning approach that enables models to learn intricate patterns directly from raw text data. Classic machine learning-based approaches are heavily dependent on handcrafted features, whereas deep learning models automatically extract features and capture semantic, syntactic, and content information in email texts [16]. This change improved detection performance, particularly against advanced and new phishing attacks. Convolutional Neural Networks (CNNs) are commonly used for phishing detection tasks because they can capture local patterns in text, including character sequences, keywords, and structural information. CNNs have been widely applied to email phishing detection, where they take character-level or word-level representations of email bodies and URLs as input. Previous work shows that CNN-based models can learn discriminative features that cannot be manually encoded and are efficient at detecting phishing emails. However, CNNs essentially capture local feature patterns and suffer from strong inter-feature dependencies in long email sequences. In this context, Recurrent Neural Networks (RNNs), specifically Long Short-Term Memory (LSTM) networks, have also been proposed. LSTMs are suitable for modelling sequential data because they can maintain context over long sequences. It is well-suited for analysing content from emails, where the interdependence among distant words or sentences may influence the meaning of a message [17]. It has been shown that models based on the long short-term memory (LSTM) architecture achieve better phishing detection performance than several other conventional classifiers, due to their ability to handle context-based cues and the writing styles used in the tactics of social engineering attacks (Figure 3).

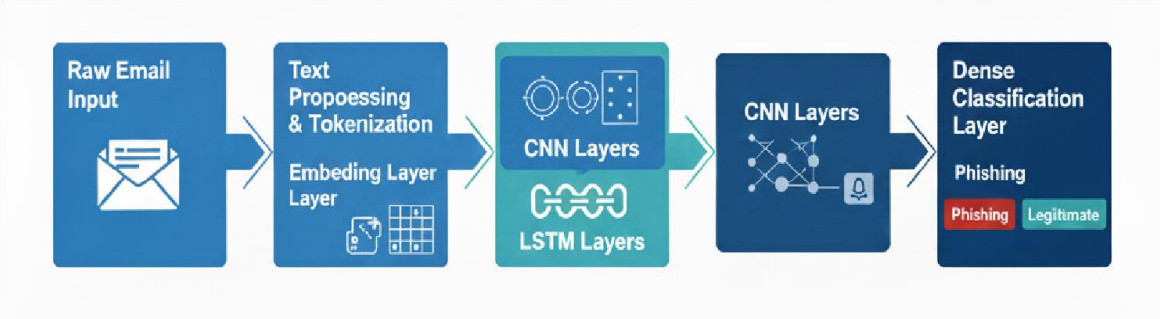


Figure 3: Phishing detection using machine learning

Despite their powerful performance, LSTMs are slow to train and not easily parallelised, which could potentially limit scalability. The performance can be enhanced by using hybrid deep learning approaches that integrate. CNNs and LTSM. In such architectures, CNN layers are employed to extract features, and LSTM layers are utilised for sequence modelling [18]. These hybrid techniques have demonstrated the best detection performance when combining both architectures. However, the added architectural complexity results in higher computational cost and a higher risk of overfitting, and becomes more problematic with small or imbalanced training data. Despite the potential of deep learning models to obviate certain handcrafted feature requirements, they face challenges in terms of training costs and ease of deployment. Such models need to be trained on large, well-labelled datasets to achieve generalizability, and their performance may deteriorate when trained

on outdated or biased data. Overfitting remains challenging, especially when phishing datasets are not comprehensive enough to keep pace with evolving attack patterns.

Regularisation methods and data augmentation can help, but they do add more complexity to the training. Deep learning models are computationally intensive during both training and runtime [19]. However, they consume a lot of memory and are slower to infer, making them unsuitable for real-time or mobile applications. Deep learning methods can be considered an alternative to classical machine learning models and typically achieve greater accuracy, but at the cost of lower interpretability and greater system complexity. This trade-off is particularly important for security applications where immediate reaction and user confidence are crucial. In general, deep learning methods represent a significant step forward in phishing identification by enabling automatic feature learning and improving generalisation. Nevertheless, their computational cost, interpretability, and real-time applicability highlight the need for lighter architectures. Such difficulties are drivers of migration to transformer-based models, which aim to strike a balance between performance and deployability [20]. This evolution is presented in the following section, dedicated to transformer architectures and their use for real-time email phishing detection.

2.5. Transformer-Based Models for Phishing Detection

Transformer-based models have emerged as the predominant method for a wide range of NLP tasks, including email phishing detection. In contrast to conventional machine learning approaches and early deep learning models, transformers feature self-attention mechanisms that enable them to learn the dependency structure between words across the entire sequence [21]. This is very important in many cases, such as phishing emails, where the malicious intent can be conveyed only through subtle semantics or social engineering clues that require context, not just individual keywords. One of the most renowned transformer models is the Bidirectional Encoder Representations from Transformers (BERT). BERT is pre-trained on masked language modelling and next-sentence prediction tasks using a large-scale text corpus, thereby learning deep bidirectional representations of the text. Fine-tuned for phishing classification, BERT has been shown to perform well by capturing the context and semantics of the email body. Examples from Songailaitė et al. [10] show that BERT-based classifiers outperform traditional machine learning methods and most CNN- or LSTM-based models, especially at detecting advanced phishing emails that evade rule-based or feature-driven systems. Despite such impressive detection accuracy, BERT's large model size and high computational resource requirements raise concerns about practical adoption. The BERT base model has 110M parameters and is quite memory-intensive with a longer inference time. These can make it unsuitable for most real-time detection systems and resource-limited platforms, including mobile devices.

As described in subsequent sections of this review, user-facing phishing detection applications require strict criteria for real-time response and low computational demand. To overcome these drawbacks, lighter transformer alternatives have been proposed. DistilBERT is a small, fast, cheap, and light Transformer model distilled from BERT base. DistilBERT achieves performance on par with BERT using only half as many parameters and less than half the memory footprint, enabling easy use even on low-resource machines. Damatie et al. [11] show that DistilBERT models are the state of the art for phishing detection, offering reduced latency and high accuracy, enabling deployment on mobile devices and in real-time scenarios. Other transformer-style models have also been investigated for phishing detection. Tawil et al. [7] note that RoBERTa and ALBERT models can achieve even better detection when trained on large, diverse data. Yet, increasing performance inevitably increases training complexity and computational cost, which may not always be practical for deployment in all settings. One of the main benefits of transformer models is their ability to learn context embeddings. In contrast to static word embeddings or handcrafted features, contextual vectors capture word sense by considering the context window. This enables transformers to model better phishing cues, such as urgency levels, impersonation, and deceitful language structures.

As a result, the transformer model is also more tolerant of expression variations and less reliant on a handcrafted feature set, which addresses some disadvantages of classical machine learning methods. However, the transformer-based models do have drawbacks [22]. A considerable amount of backend computational resources and the labelling of large datasets are needed for training and fine-tuning models to prevent overfitting. Further, transformer models can have complex architectures that are less interpretable than simpler models, making it more difficult for an end user to understand why the model made a particular prediction. It's a significant concern for security use cases where the user's trust and awareness are in question. There is an open trade-off between model performance and deployment productivity. Although recent very large transformer models can achieve the best performance across a wide range of tasks, they are difficult to deploy in real time or on mobile devices, as they typically require substantial computational resources and memory. Lightweight models like DistilBERT provide a feasible compromise that offers strong detection performance while potentially embedding into a refuse-to-say mobile-based phishing detection system. Transformer-based models are thus state-of-the-art for email phishing detection, and current work builds on them, providing better context awareness and robustness than previous methods. But in practice, the computational costs of these methodologies must be weighed against deployment constraints and user-facing requirements. Choosing a suitable transformer structure is thus a compromise between detection accuracy and practical utility.

2.6. Datasets and Feature Representation

A key to the system's performance is the quality, size, and representativeness of the datasets utilised. Early research had to work with small, curated sets of emails from a few sources, limiting how well models could generalise. Recent work involves larger, public datasets collected from phishing repositories, user reports, and threat intelligence feeds [23]. Examples include the dataset from Machine Learning Approach for Email Phishing Detection by Fares et al. [13], the large Kaggle phishing dataset by Alam [16], and datasets cited in emerging research, such as those by Zhou et al. [17]. Despite more data, public datasets have issues. Small sets can yield inflated accuracy when models memorise patterns rather than learn general phishing traits. Class imbalance is another problem, because legitimate emails often outnumber phishing, or datasets are balanced in ways that don't reflect real traffic. This can skew the model and degrade generalisation to new data. The quality of feature representation influences the model performance when learning from datasets. Generally, the handcrafted features such as TF-IDF vectors, n-grams, and URL lexical features are extracted. These are quick but also fragile to attacker modification. Word embedding techniques such as Word2Vec assign continuous vectors to words, modelling semantic relationships. More recent transformer models create contextual embeddings that reflect how words are used. These richer representations generally improve detection but still depend on dataset diversity and size.

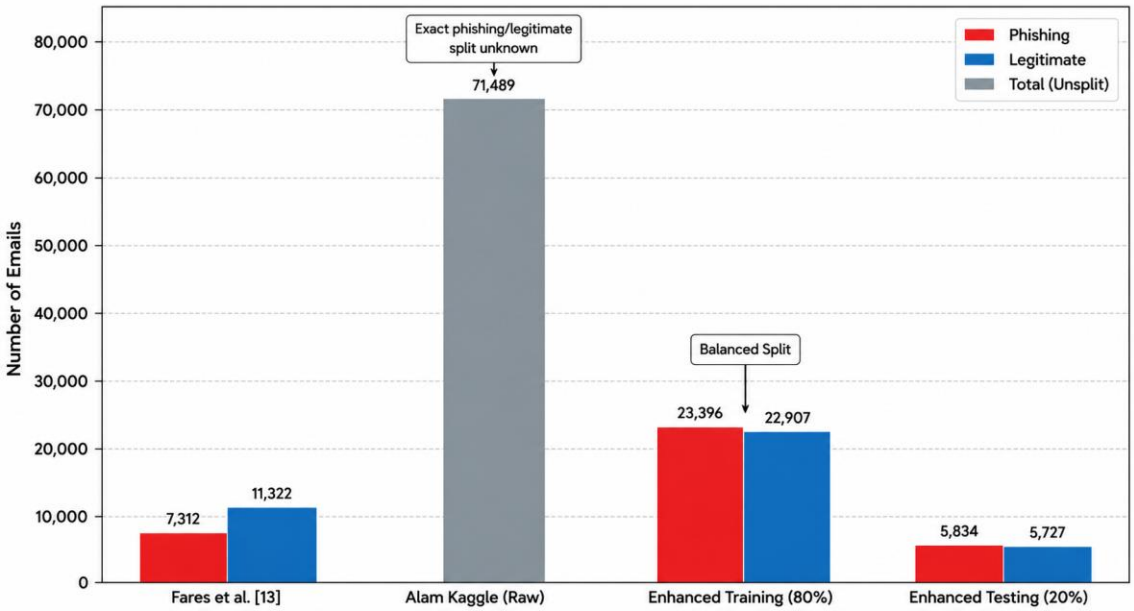


Figure 4: Phishing and legitimate email statistics in both the raw and enhanced datasets

Poor linguistic variety or outdated attack types in datasets can bias learned features and hurt real-world performance. To improve robustness, this dissertation uses an enhanced dataset strategy. The dataset from Machine Learning Approach for Email Phishing Detection by Fares et al. [13] contains 18,634 emails, of which 7,312 are phishing, and 11,322 are legitimate. The Alam [16] Kaggle phishing dataset originally contained 71,489 emails. After cleaning and combining these sources, the final dataset was split 80/20 for training and testing. The training set includes 23,396 phishing and 22,907 legitimate emails, while the testing set includes 5,834 phishing and 5,727 legitimate emails. This balanced and larger dataset supports reliable model evaluation. Figure 4 compares the sizes and class distributions of the target paper dataset, the Alam [16] Kaggle dataset, a typical prior study dataset, and the enhanced training dataset used in this dissertation. Figure 4 illustrates the significant growth of PIN, both phishing and legitimate samples, indicating better coverage for a robust solution.

2.7. Evaluation Metrics and Experimental Practices

Assessment metrics are the main tool for evaluating the performance of email Phishing detection systems. Model performance in most studies is reported using baseline classification metrics such as accuracy, precision, recall, and F1-score. Accuracy is the ratio of correctly classified emails to all samples and is a simple, well-defined metric. Phishing datasets, however, tend to be heavily biased, with a large proportion of legitimate emails included alongside phishing samples, leading to overestimated accuracy scores that do not reflect real-world detection rates. Precision and recall are more meaningful for phishing detection approaches. Precision refers to the fraction of detected emails that are phishing attempts and are actually malicious. It is very important for minimising the volume of false positives and the disruption of legitimate communication. Recall measures a system's ability to correctly identify phishing emails, a crucial security characteristic because false negatives can be extremely

harmful [24]. The F1-score is an average metric that balances precision and recall, which are often in opposition, especially with imbalanced classes.

Although various evaluation metrics exist, previous work shows heterogeneity in evaluation choices. Some works report only accuracy or focus on a subset of metrics, making it difficult to compare with baselines. Furthermore, dataset sizes are inconsistent across works and processing techniques, as well as in train-test splitting, further complicating benchmarking. In some cases, superhuman performance may be due to small or otherwise overly controlled datasets rather than true model generalisation. Especially in the case of phishing detection, accuracy by itself is very deceptive. Even on imbalanced datasets, it cannot achieve high accuracy if it classifies everything containing email as legitimate, while the overall goal is to detect phishing. This overhead is particularly troublesome for real-time and mobile settings, which provide little tolerance for the cost of false negatives. As reported by Fares et al. [13], sound evaluation involves the use of precision, recall, and F1-score jointly with properly defined experimental settings. As a result, in recent studies, more and more focus is being set on large-scale data processing that represents the actual deployment scenario. This involves reporting metrics across classes, using large, diverse datasets and ensuring transparency in the experimental setup. These activities are necessary to determine whether phishing detection models can function effectively in the wild, beyond the controlled laboratory setting.

2.8. Real-Time Phishing Detection Systems

Phishing detection techniques have been categorised into offline and real-time methods. Offline systems work on collected data and analyse without time limitations. These products are frequently used in experimental research to validate model performance and compare algorithms under ideal conditions. While offline evaluation is valuable for understanding model performance, it does not capture the cognitive workload of real-world phishing detection, where decisions are made on the fly as emails arrive and users encounter them. Real-time phishing detection systems try to analyse emails as soon as they are received and provide an immediate classification decision. This entails low-latency, efficient feature extraction and fast model inference. In deployments, this may apply especially on mobile or client-side platforms, where seconds of delay are a long time and bad for usability and trust. Rao et al. [5] note that detecting botnets in real time is challenging due to the use of complex feature extraction pipelines, external blocklists, or computationally intensive models. The deployment restrictions make real-time systems different from offline systems. Real-time object detection often occurs in harsh environments with limited computational resources, constrained memory, and inconsistent network connectivity.

Especially in mobile-based phishing-detection applications, these restrictions are important, as models need to trade off detection accuracy for promptness and energy consumption. While offline, some models perform well; however, they may not be suitable for real-time applications due to their high inference time or reliance on servers. One major limitation is the general focus on offline evaluation metrics, primarily based on accuracy on static test datasets, rather than a real-time operational perspective. High detection rates are often presented without discussing inference time, scalability, or user-facing latency. It follows that there is a shortfall between experimental success and deployable systems. Real-time phishing is inherently a live scenario, and systems not architected to support real-time attestation may not adequately protect users. Addressing these issues will require detection models that are computationally efficient and resilient to evolving phishing methods. In the design of a real-time phishing detection system, lightweight preprocessing and model construction, along with a fast decision-making strategy, are necessary. This transition from mainly offline evaluation to deployment-aware design is critical to producing phishing detection solutions that are practical for both including mobile and interactive applications.

2.9. User Awareness and Human-Centred Detection

User education is already known to be an important part of phishing fighting. Nevertheless, mounting evidence suggests that mere knowledge is inadequate in the face of current phishing threats. Today's phishing emails are often crafted with relevance and psychology in mind, making it hard to distinguish the real from the fake, even for experienced users. As a result, studies have highlighted the importance of systems that not only provide accurate phishing detection but also communicate. Although they enhance transparency, explainability methods to date tend to be too technical, presuming that users can interpret explanations based solely on machine models. This assumption severely constrains the practicality and usability of these systems, especially for non-expert users. One such solution to close the gap between user expectation and algorithmic decision-making is Explainable Artificial Intelligence (XAI). For phishing detection, XAI approaches aim to emphasise suspicious words, unusual sender information, or risky URLs that influence a model's decision. Govindaraaj [19] argues that justifications should help users bridge the gap between model output and real-world indicators of attacks. However, many XAI systems do not live up to this ideal in practice. Explanations are typically presented as abstract feature importance scores or confidence values, which fail to tell the user what to do next.

Consequently, users inadvertently choose to ignore the alerts or over-trust automated tools without fully understanding the associated risks. Another gap in previous studies is the unification of explainability and real-time user interaction. Most

existing phishing detection systems produce explanations after classification, without accounting for how users consume the feature at decision time. In everyday settings, emails are read and responded to rapidly under cognitive load. Practicality may be lost if explanations are slow, unclear, or poorly integrated with the interface. This breakdown limits the system's power to nudge users toward better decisions at the time of decision-making. Human-centred phishing-detection studies are increasingly emphasising the need to balance technical performance with usability and trust. Systems that can provide reliable detection, coupled with clear, actionable explanations, could help users in the short term while teaching them skills for long-term self-protection. Yet that integration is barely tackled in mobile and real-time. This gap specifically informs the research focus of this work: to simultaneously provide real-time, informed phishing detection and use explanation in a user-facing manner for quick decision-making, while raising awareness.

2.10. Mobile-Based Phishing Detection Solutions

Phishing protection tools are frequently used as browser add-ons or plugins, or device-specific applications. For example, typical browser-based solutions work best in desktop environments by observing web pages or emails to suppress malicious content on the fly. Although these methods work well, they are not well-suited to the mobile world, where emails are handled through dedicated apps, and users have other means of communication. The existence of mobile platforms limits the detection of phishing due to constrained computational resources, diverse operating systems, and varied user interfaces. Mobile-based phishing detection systems that must work efficiently on smartphones have received considerable attention in recent years. Rao et al. [20] introduced a mobile-centric phishing detection method that employs a hybrid super learner ensemble and demonstrated that high detection accuracy can be achieved while keeping costs low. Their work shows that it is possible to run phishing-detection models directly on mobile devices, overcoming the latency and performance challenges in practical settings. But the work primarily focuses on URL-based phishing detection and does not fully address email-specific issues, such as message content analysis and integration with mobile email clients.

However, despite these advancements, mobile platforms remain an underserved area in phishing detection research. The vast majority of studies report classification performance without addressing issues relevant to real-time applications, such as interface design, alert presentation, and user engagement. Anyhow, due to factors such as poorly designed warnings on mobile phones or long, intricate explanations, users may ignore the alerts, which weakens the system. Furthermore, energy and memory are seldom considered despite their direct impact on user adoption and long-term sustainability. Time is of the essence even more in on-the-go scenarios, where users frequently read and act on emails before external checks are run. It is therefore necessary for a good mobile phishing detection system to achieve a balance between accuracy, speed and usability. But in practice, delivering short, clear warnings and providing simple, educational feedback might increase users' awareness, helping them learn from phishing incidents and detect patterns over time. The system proposed in this dissertation addresses those gaps by providing real-time phishing email detection through a mobile application, a user-friendly interface, and explanations that help users understand the threat. Although the proposed solution targets mobile availability, instant feedback and user engagement, it definitely does not address the wider campaign or deployment workshop.

2.11. Critical Summary and Research Gaps

The literature reviewed in this paper shows significant advancement in phishing detection, especially with the use of machine learning and deep learning methods. Yet, a methodical cross-examination of datasets, evaluation procedures, and system construction and deployment platforms would identify some long-standing knowledge gaps that hinder real-world uptake. The first major gap is a technical, experimental one. Many works on phishing detection are evaluated almost exclusively offline using a static dataset and report high accuracy without discussing generalisation or deployment conditions. Performance metrics may be inflated by small or well-curated datasets when random train-test splits are used. Furthermore, while a few methods use computationally intensive models that achieve good performance on challenging data with no temporal constraints, these methods are not suitable for online operation. These restricting factors included scalability, latency, and deployment robustness in the face of new phishing tactics encountered online. Second, when it comes to application platforms. Most phishing detection systems are designed for browser-based or desktop environments, and little attention is paid to mobile systems. Mobile usership is now the dominant platform for email use, but is often overlooked in phishing detection literature. Here are the challenges of limited computing power, memory usage, and battery life, as mentioned in prior work. This gulf makes many proposed systems uncontentious and irrelevant as they do not correspond to the behaviour of the real user [25].

The third gap is the interaction between humans and AI. Such models are black boxes that provide only binary classification, not an interpretable justification. It has this effect of placing a ceiling on user trust and educational value. Users frequently receive word that an email they received is malicious, with little more information on why, which does little to build long-term awareness or make people less vulnerable to future attacks. A lack of user-friendly explanations is a clear downside of the existing system. These gaps directly inspire the design decisions taken in this paper. To fill the technical gap, a lightweight DistilBERT-based model is leveraged to trade off detection performance for computational efficiency. The model is tested

with a balanced and augmented dataset, increasing reliability and reducing the risk of overfitting. To bridge the platform gap, the system is deployed as a mobile-tailored application with real-time detection to mimic how users read email in their daily interactions. Lastly, to help narrow the human-AI interaction gap, our system also provides explanations alongside predictions, helping users understand phishing cues rather than relying solely on automation. By focusing on these three gaps, the current study transcends offline performance benchmarks and delivers a user-friendly phishing-detection system with practical relevance. Our approach balances technical strength with practical deployment requirements, making the system a more realistic and viable solution for current email phishing threats.

3. Methodology

3.1. Research Process

The study employs an iterative development approach to address the technological and practical challenges of email phishing detection. This starts with an understanding of the constraints of previous methods, including reliance on offline evaluation, small or imbalanced datasets, computationally intensive models, and a lack of focus on mobile deployment and user interpretability. These restrictions motivate the development of a lightweight, real-time phishing-detection system suitable for mobile devices. The literature review provides the study framework and highlights shortcomings in dataset quality, model evaluation, deployment methodologies and user engagement. Researchers created a complete collection by merging several publicly available phishing email databases. The merged dataset was cleaned, deduplicated, balanced, and preprocessed. It was then split into training and test sets of 46,243 and 11,561 emails, respectively. When required during iterative development, pretreatment and dataset partitioning were repeated to ensure data consistency and reliable model evaluation.

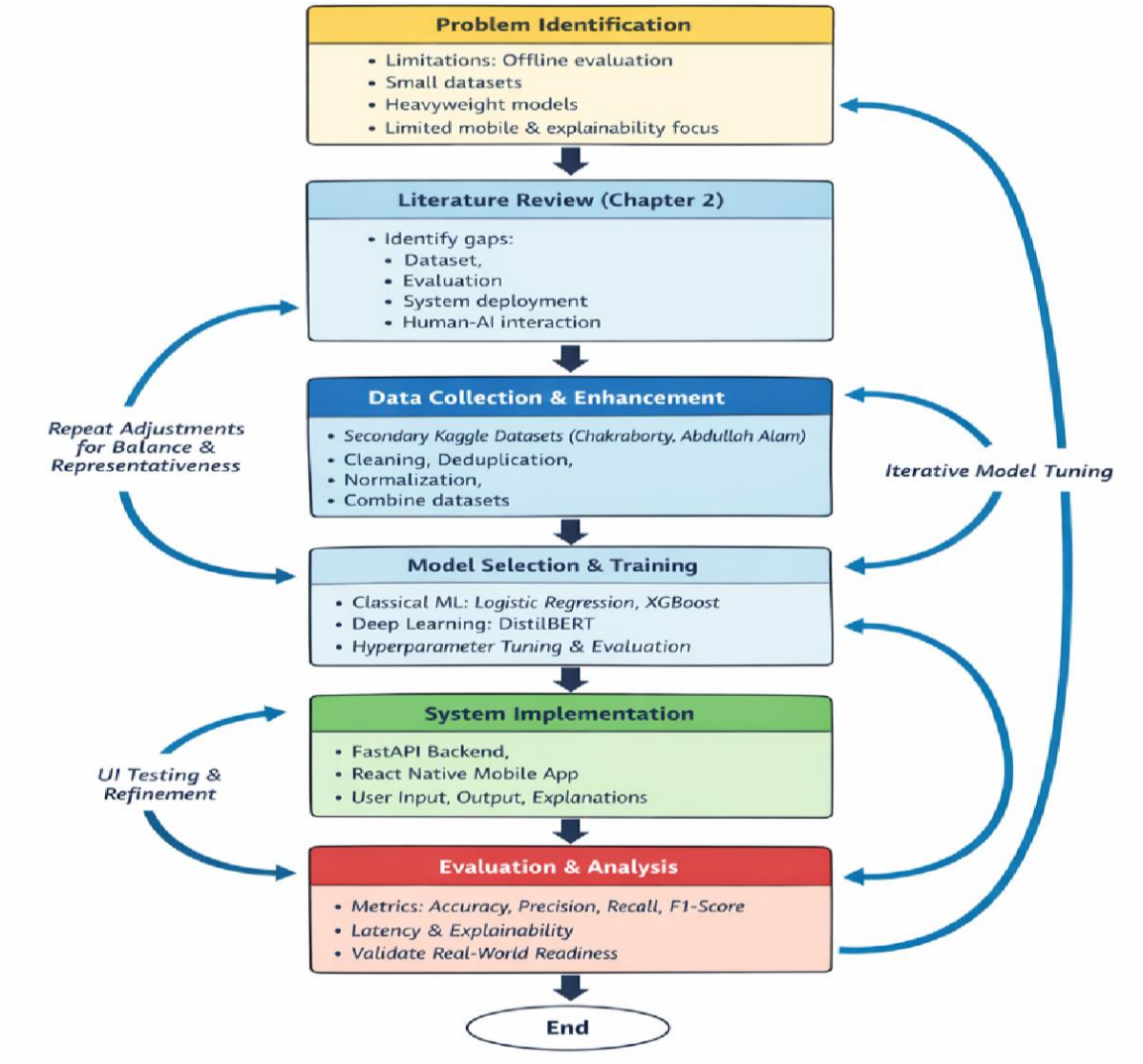


Figure 5: Research process workflow

As part of the model construction, the lightweight transformer model DistilBERT was evaluated alongside classic machine learning techniques such as Logistic Regression and XGBoost. Researchers performed hyperparameter optimisation and performance evaluation in Python to find the best trade-off between classification accuracy, computational efficiency, and deployment adaptability. Based on comparative performance studies, Researchers decided to perform real-time inference on the lightweight DistilBERT architecture, which runs efficiently on mobile systems. The deployment architecture combines a FastAPI backend for online inference and a React Native mobile application that allows users to enter email content and receive phishing predictions, confidence scores and explanatory commentary. The design and usability of the user interface were iteratively modified to improve clarity, accessibility and overall user experience. Finally, the system is evaluated using real operational conditions to measure detection accuracy, inference speed, deployment on mobile devices, and the quality of the explainability. The overall methodology is a continuous iterative cycle, as illustrated in Figure 5. It starts with problem identification, followed by data preparation, model development, deployment and evaluation, with each step leading to further refinements to improve the technical performance and practical usability.

3.2. Research Methodology

The study takes a pragmatic experimental approach to designing, implementing, and evaluating a mobile-friendly real-time email phishing detection system. An experimental methodology for direct training, validation and comparison of different machine learning models under controlled but realistic situations. This differs from survey-based or qualitative research, which focuses on reviewing existing methodologies, as it allows for the systematic evaluation of categorisation models and their capacity to be deployed in real time. Machine learning is particularly well-suited for phishing detection, as supervised learning algorithms can learn complex linguistic and contextual patterns that are difficult to capture with traditional rule-based methods. The performance of the constructed models is evaluated using conventional classification metrics, including accuracy, precision, recall, and F1-score. Accuracy is a general measure of classification performance, but it can yield overly optimistic results for imbalanced datasets when the majority class dominates the predictions. Therefore, more focus is given to recall and F1-score, as these are better measures of performance in detecting phishing emails and of the balance between false negatives and false positives. This evaluation strategy provides a more comprehensive assessment of model robustness and reliability in real-world phishing detection scenarios, where phishing emails typically constitute a small proportion of overall email traffic but must be accurately identified to minimise security risks. To confirm that the chosen model is adequate for real-world mobile-based phishing detection, the experimental setup in Figure 6 includes quantitative performance assessment, dataset characteristics, and deployment feasibility.

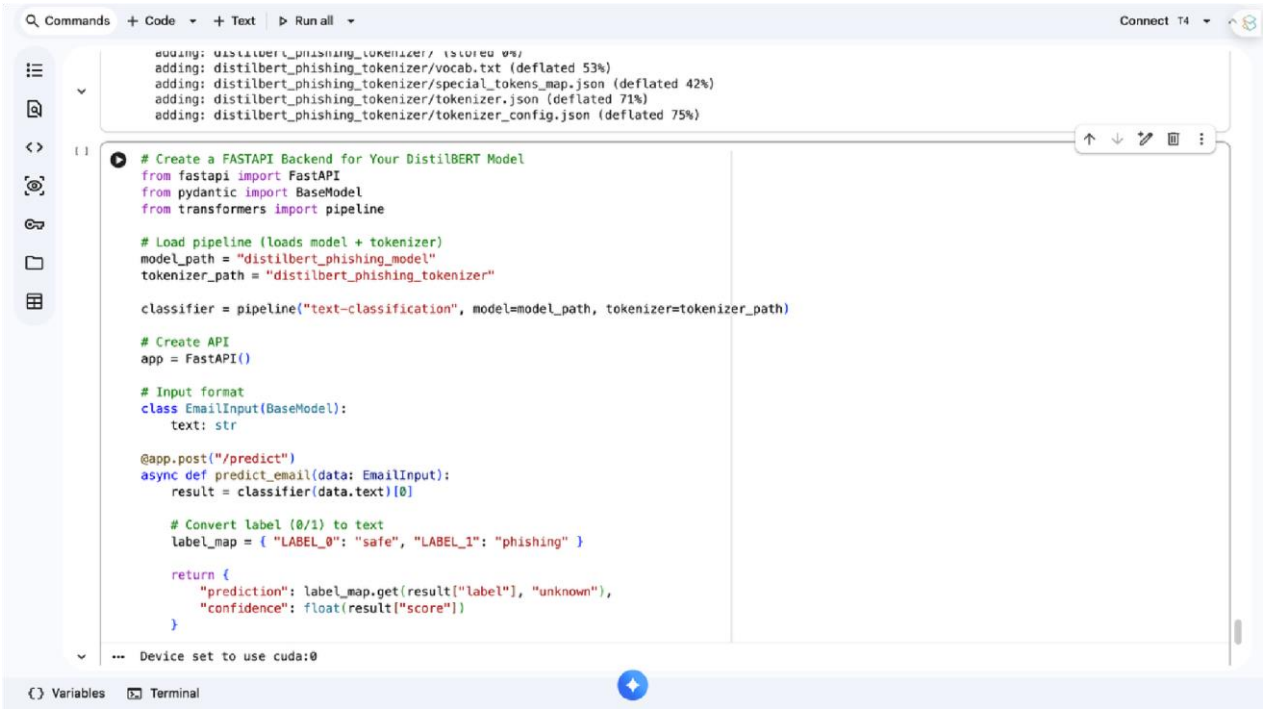


Figure 6: FASTAPI backend

Researchers implemented a comparative model strategy comprising Logistic Regression, XGBoost, and a lightweight DistilBERT transformer. Classical methods have often been adopted due to their interpretability of model features and low

computational load. In contrast, DistilBERT embeds richer contextual representations, enabling it to better understand complex phishing patterns in email bodies. Prediction performance and deployment efficiency were optimised through hyperparameter tuning and iterative evaluation in Python. Lastly, a real-time application was fundamental to the methodology. The FastAPI backend guarantees low-latency inference, while the React Native mobile interface provides explicit feedback to end users. By integrating model predictions, confidence scores, and an explanation summary, our system delivers not just accurate detection but, more importantly, interpretable explanations that bridge interactions between users and systems and address black-box problems identified in previous work.

3.3. System Architecture Overview

The architecture of the system in this paper aims to deliver real-time phishing attack prevention with usability and low latency. It consists of three major parts: a mobile app, a FastAPI backend, and a machine learning inference layer. The React Native mobile application allows users to input email content and receive detection results, confidence scores, and actionable explanations. The FastAPI backend efficiently handles the request-response loop, facilitating communication between the mobile app and the inference layer. The inference layer runs the trained models, Logistic Regression, XGBoost and the lightweight DistilBERT transformer. DistilBERT is deployed in real time, and other models were benchmarked for comparison (Figure 7).

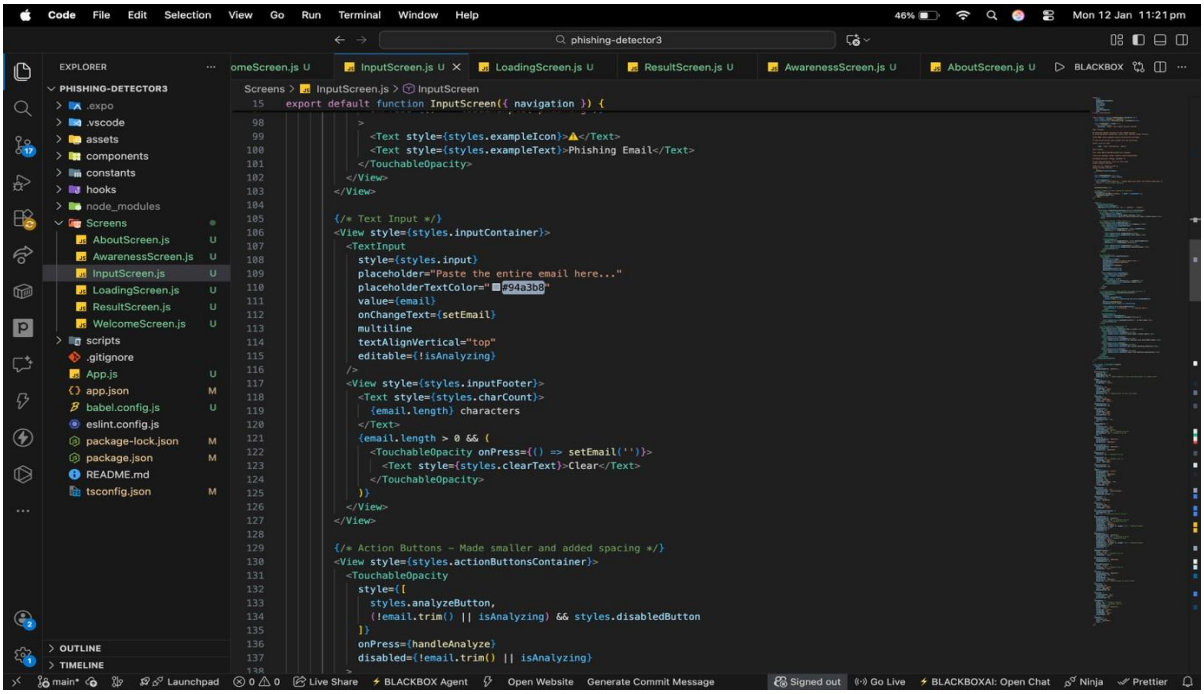


Figure 7: App email input code

Data flow begins when a user sends an email via the mobile application. Raw text is preprocessed, including tokenisation, lowercasing and padding or truncation. The preprocessed input is then fed to the ML inference layer, which outputs a prediction, confidence scores, and a textual explanation. This information is sent to the mobile app and is displayed there. Architectural choices were based on latency, ease of use and maintainability.

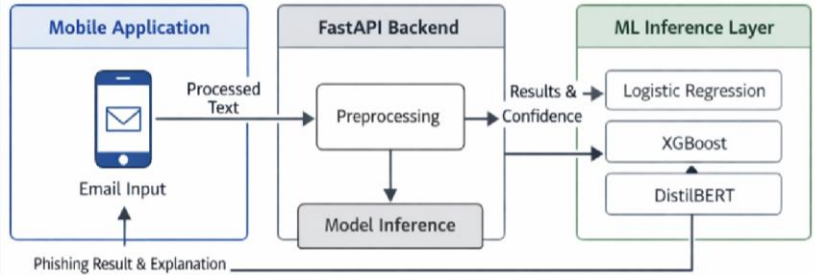


Figure 8: System architecture diagram

Back-end inference was selected to offload computation from mobile devices, ensure consistent performance, and facilitate model updates. The decoupling of the mobile interface and the inference layer also provides room for scaling and for injecting more models in the future. This architecture combines technical efficiency and user accessibility through a hybrid approach; related work addresses concerns about offline learning systems. Most critically, prior works that are based on offline evaluation do not explicate deployment concerns e.g., network delays and model loading times) which is crucial to a real-time mobile system. The real-time implementation indicates the phishing detection system may be deployed on a mobile platform. The proposed technique aims to improve user engagement in online inference as compared to traditional methods, which are mostly based on offline model performance. The system provides confidence rankings and explanatory commentary on predictions, helping users better understand the reasoning behind each prediction and make informed decisions. This explainable technique increases user trust in automated phishing detection and encourages higher security awareness. Researchers chose the DistilBERT model for real-time inference due to its lightweight architecture and computational efficiency, ideal for mobile deployment. This allows us to get accurate predictions with little latency. Figure 8 presents the entire system architecture, demonstrating the workflow from user input to phishing prediction, confidence estimation, and explainable feedback. This results in a responsive, interpretable, and mobile-ready phishing-detection system. Figure 8 summarises the real-time application implementation environment, the backend framework, the deployable model and the frontend technological stack.

3.4. Data Collection and Dataset Preparation

This study analyses two publicly accessible phishing email datasets from Kaggle, Chakraborty [15] datasets. These datasets comprise phishing and legitimate emails from various sources, including user-submitted reports, open phishing feeds, and online email libraries. The use of secondary data sets also provides an ethical approach by avoiding the potential for collecting sensitive personal information and enabling reproducibility and comparability with prior studies. Importantly, using well-documented, diverse datasets redresses the limitation of small-scale, Gaussian-like datasets used in many earlier studies to detect phishing attacks (Figure 9).

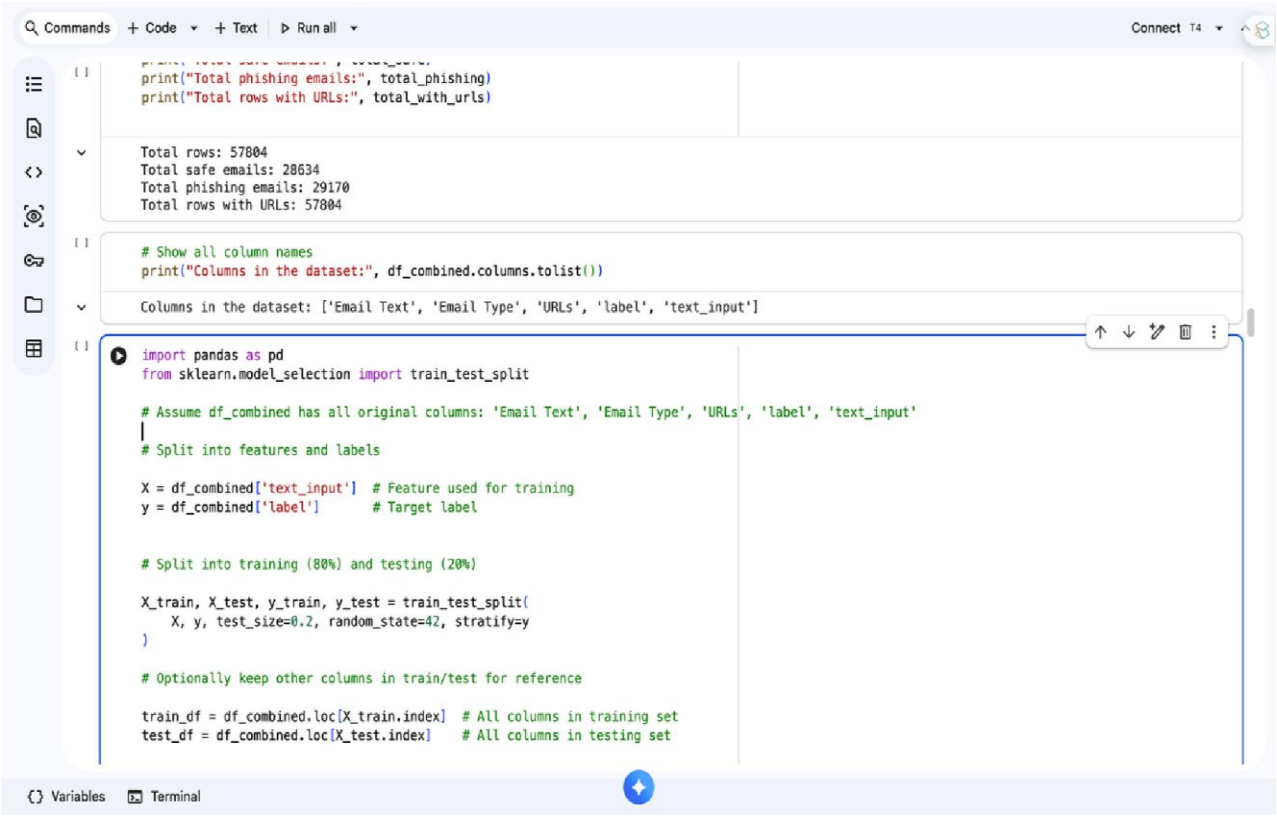


Figure 9: Combining the dataset and splitting it into features and labels

Raw Datasets were normalised and denoised to improve quality. This included removing duplicate emails, ensuring label consistency between phishing and legitimate classes, and standardising text by reducing it, breaking it into words (tokenisation), and eliminating noisy entities. These preprocessing rundowns are crucial, as discrepancies may well introduce bias into the model. This, of course, applies to both classical machine learning and deep learning (Figure 10).

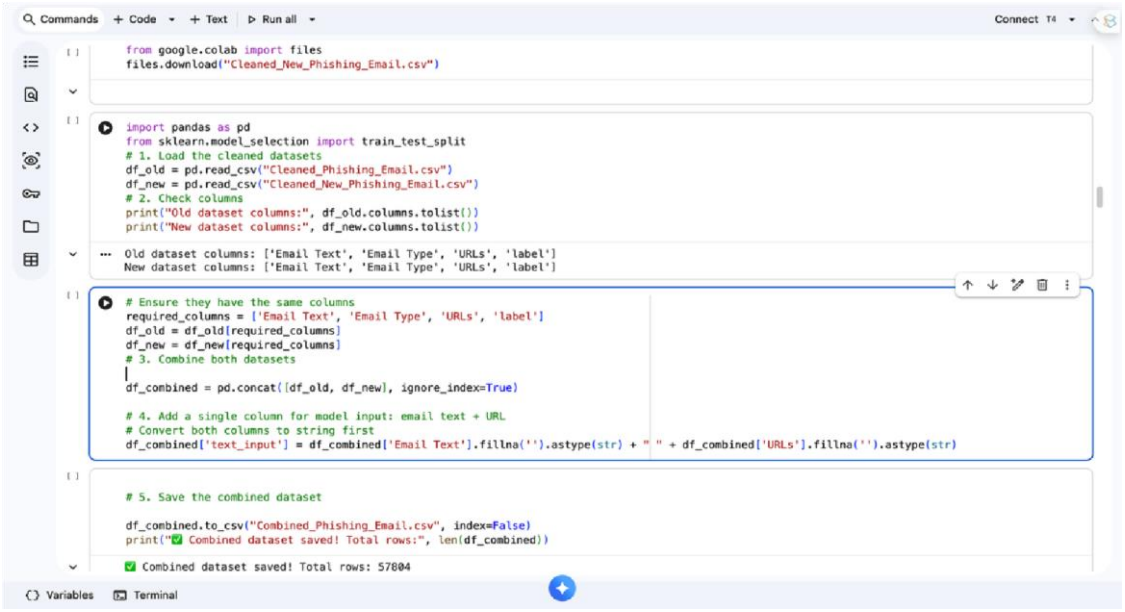


Figure 10: Loading the cleaned dataset

After the data cleaning process, a larger email dataset was created with 46,243 training emails and 11,561 testing emails, with a near balance between the phishing and legitimate classes. To obtain sufficient labelled data for model training and validation, the dataset was split into 80/20 train-test sets. Although high accuracy indicates good overall performance, it can mask poor phishing email detection, especially in imbalanced datasets. This underscores the need for well-balanced, carefully constructed datasets for reliable model performance. This approach to preparing the dataset enables models such as DistilBERT to learn important patterns from both phishing and valid emails without overfitting to the majority class.

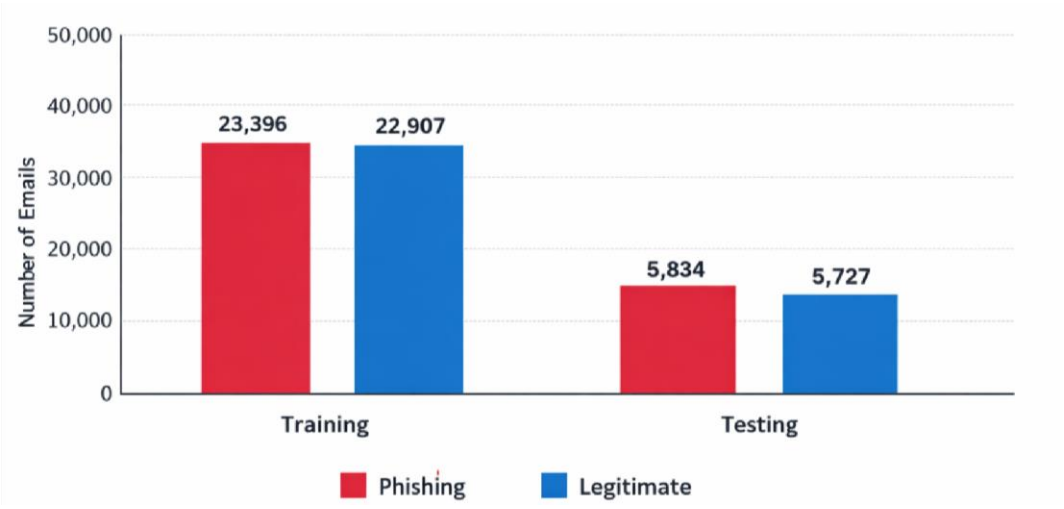


Figure 11: Dataset composition and class distribution

The final dataset compilation and class distributions are shown in Figure 11, indicating improved training and testing splits and a balanced set of phishing spam and legitimate emails. This diagram visualises the thoughtful design of the dataset to enable consistent training and evaluation of both classic and transformer-based models, while addressing biases arising from small or unbalanced inherent datasets.

3.5. Feature Representation and Text Processing

Pre-processing email text is an important step before feeding it to machine learning models for better interpretation. At first, tokenisation split the email text into words; then all words were made lowercase to normalise the text and reduce vocabulary. Padding and truncation were used to ensure equal sequence length in the DistilBERT architecture, which requires inputs to

have a constant size. These processes are performed to help mitigate differences in sequence length, which could lead to performance differences. To resolve these problems, various padding truncation strategies were tested, enabling both short and long emails to be adequately represented whilst avoiding loss of salient content. This allowed a fixed input size for the transformer, which is important for real-time inference on mobile devices. For comparison, classical models such as Logistic regression or XGBoost used TF-IDF vectors to represent term importance. The principle of TF-IDF is that it captures relevance based on frequency, but without considering context, and hence is unable to discriminate between subtle phishing follow-up cues and benign language patterns. This may be more challenging, specifically when emails use slightly modified forms of phishing to make them harder to detect, as has already been shown. To remedy this, experiments were conducted with varying n-gram ranges and stop-word treatments to improve TF-IDF performance (Figure 12).

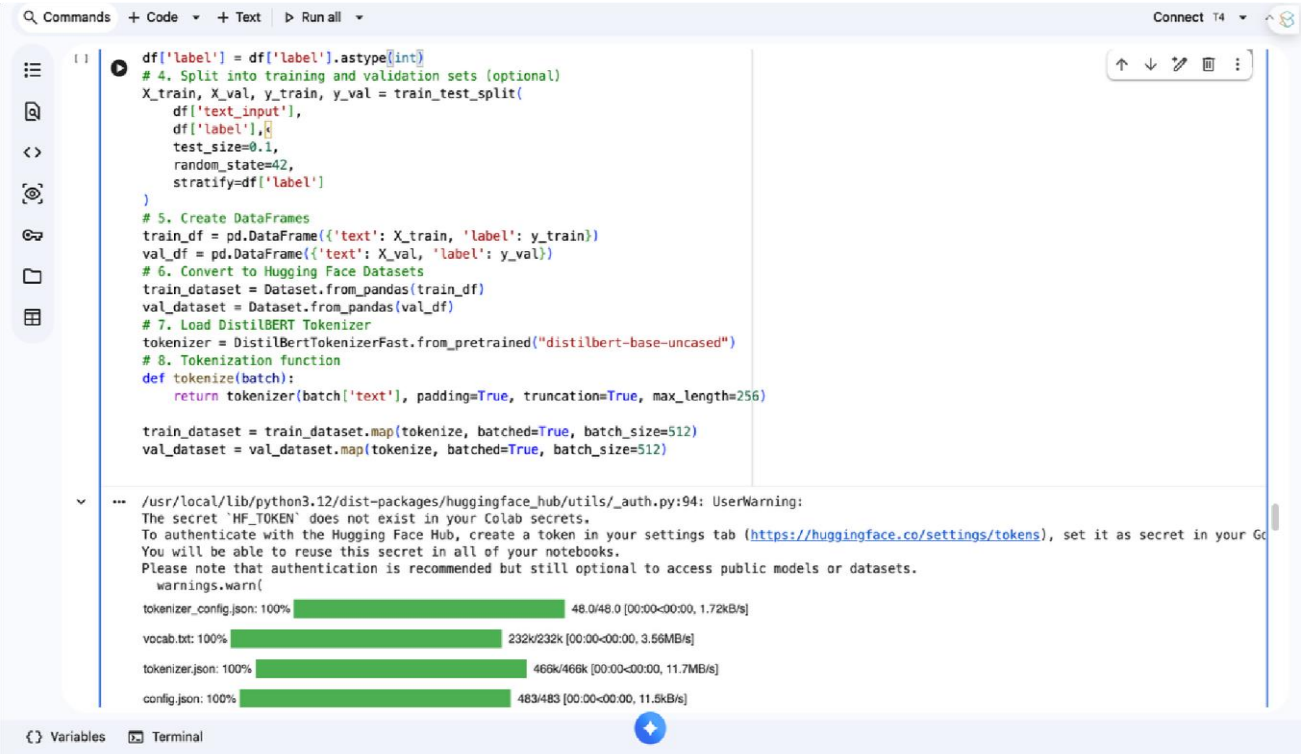


Figure 12: Loading DistilBERT tokenisation

Even after these modifications, it was found that TF-IDF alone could not consistently infer phishing intent when context and semantics played crucial roles, validating the necessity of contextual embeddings in contemporary phishing detection. The DistilBERT approach, on the other hand, leverages contextual embeddings from a transformer to learn semantic linkages and word dependencies within an email. This reduces the need for manual feature engineering, as the model learns an appropriate representation directly from the raw text. Contextual embeddings are especially useful for phishing detection, since the same words may have multiple meanings based on the context, such as expressions of urgency or incorrect sender information. To obtain good classification performance with minimal inference latency for real-time mobile deployment, Researchers experimented with different sequence lengths, batch sizes and attention mask setups during training. This allowed the model to rapidly evaluate various email content and give actionable insights through the mobile application. Researchers compare classic TF-IDF-based features with transformer-based contextual embeddings to assess the merits and limits of each representation. While TF-IDF features are computationally more efficient, contextual embeddings better simulate phishing-related patterns and serve as the foundation of the real-time system based on DistilBERT.

3.6. Machine Learning Models and Training

In this paper, three machine learning models were employed for email phishing detection: Logistic Regression, XGBoost, and the Lightweight DistilBERT model. Logistic Regression and XGBoost are maintained as representatives for classical and ensemble-based ML approaches, respectively. DistilBERT was used as a transformer-based model that captures the semantics of parsing threats, gist, and context information from emails (Figure 13).

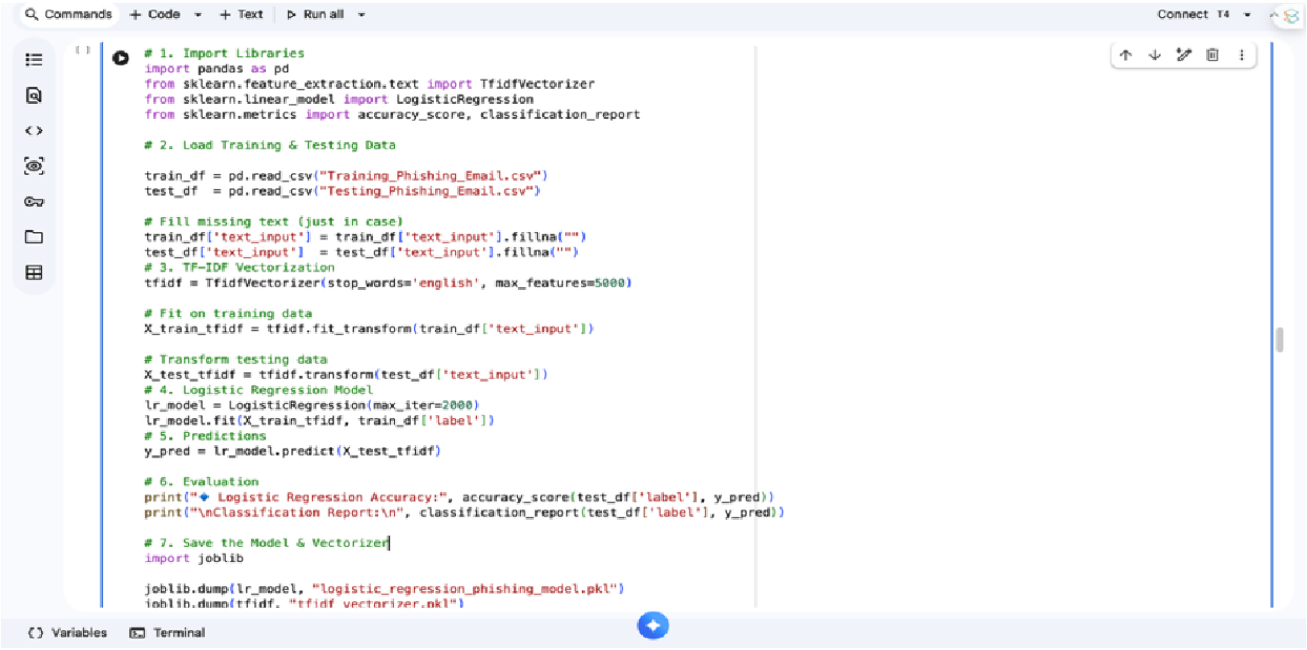


Figure 13: Logistic regression model

The training configuration involved carefully tuning hyperparameters for each model. Logistic Regression used regularisation with a tuning parameter optimised via cross-validation. Training with XGBoost involved fine-tuning the learning rate, max tree depth, and number of estimators to ensure stable convergence. DistilBERT was trained over epochs with a small batch size due to hardware limitations. There was no particular reason beyond deployment that led us to choose DistilBERT over full BERT. DistilBERT has been reported to increase inference speed by 40% while consuming fewer resources, while maintaining comparable accuracy, enabling its use in real-time mobile applications. This speed is fast enough to enable timely predictions for users without sacrificing model performance. It also points out the trade-off between accuracy and deployment cost. In contrast, classic models such as XGBoost or Logistic Regression are lightweight but might be inefficient for complex phishing language patterns; full BERT is highly accurate but computationally expensive (Figure 14).

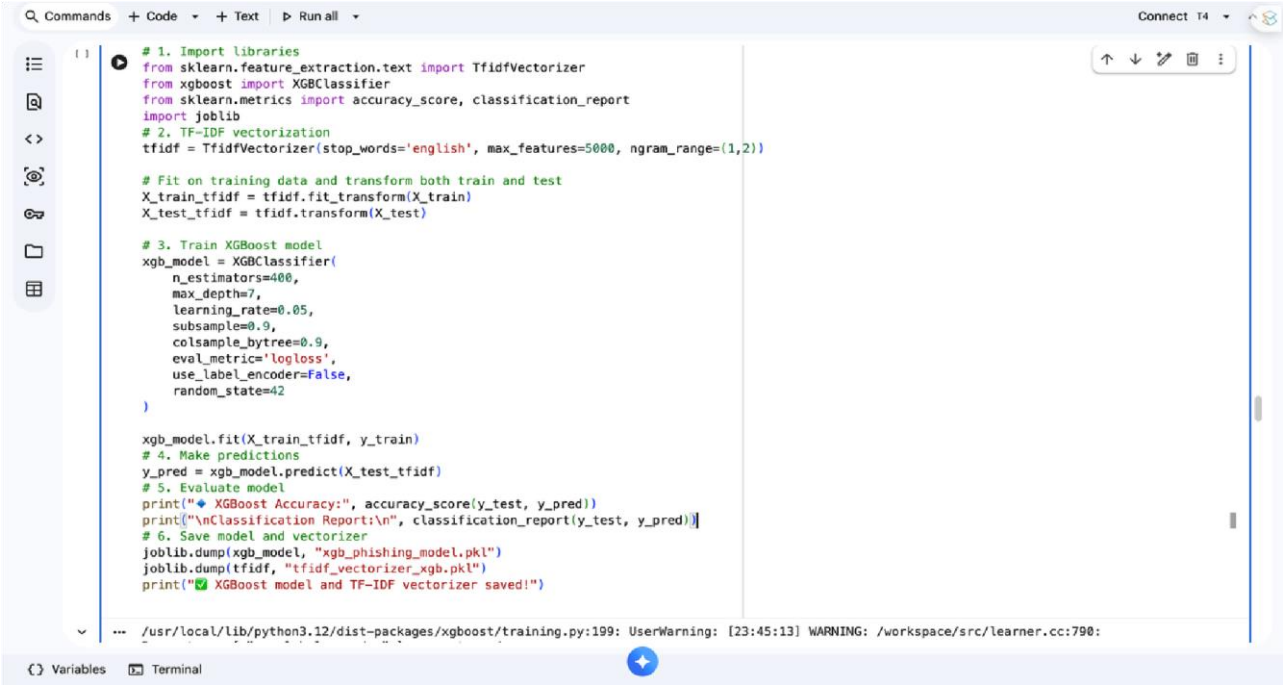


Figure 14: XGBoost model

DistilBERT offered a good trade-off, delivering strong predictive performance with minimal loss of time sensitivity, an important property for a real-time detection system on mobile devices. With these models integrated into the FastAPI back-end, the system runs inference and returns the prediction and explanatory feedback to the application. This strategy ensures timely alerts to users and, at the same time, helps them understand why they are vulnerable to phishing attacks. Furthermore, the modular custom inference layer enables adding new models easily in the future, if required, with minimal time latency and minimal degradation in user experience.

3.7. Real-Time Detection and Backend Implementation

The real-time detection part of this work is embedded as a FastAPI backend with REST endpoints that receive the text from email content in a mobile application and return classification results and actionable explanations. Request-Response Process: Between the mobile app and our back end, there is a request-response flow that minimises latency. The mobile app posts email text, etc. via HTTP; the backend preprocesses the input, reuses the DDistilBERT-trained model, and returns the results in JSON. Real-time detection does not run batch tests offline; instead, it processes individual emails as input, prioritising responsiveness and the end-user experience. Latency was a primary concern in the architectural design. Time of inference reduced thanks to the use of a lightweight model, as DistilBERT is also optimised for batch size and sequence length, which are limited to reduce computational cost. Only the required information is transmitted to minimise payload size, thereby reducing network transfer times for mobile clients (Figure 15).

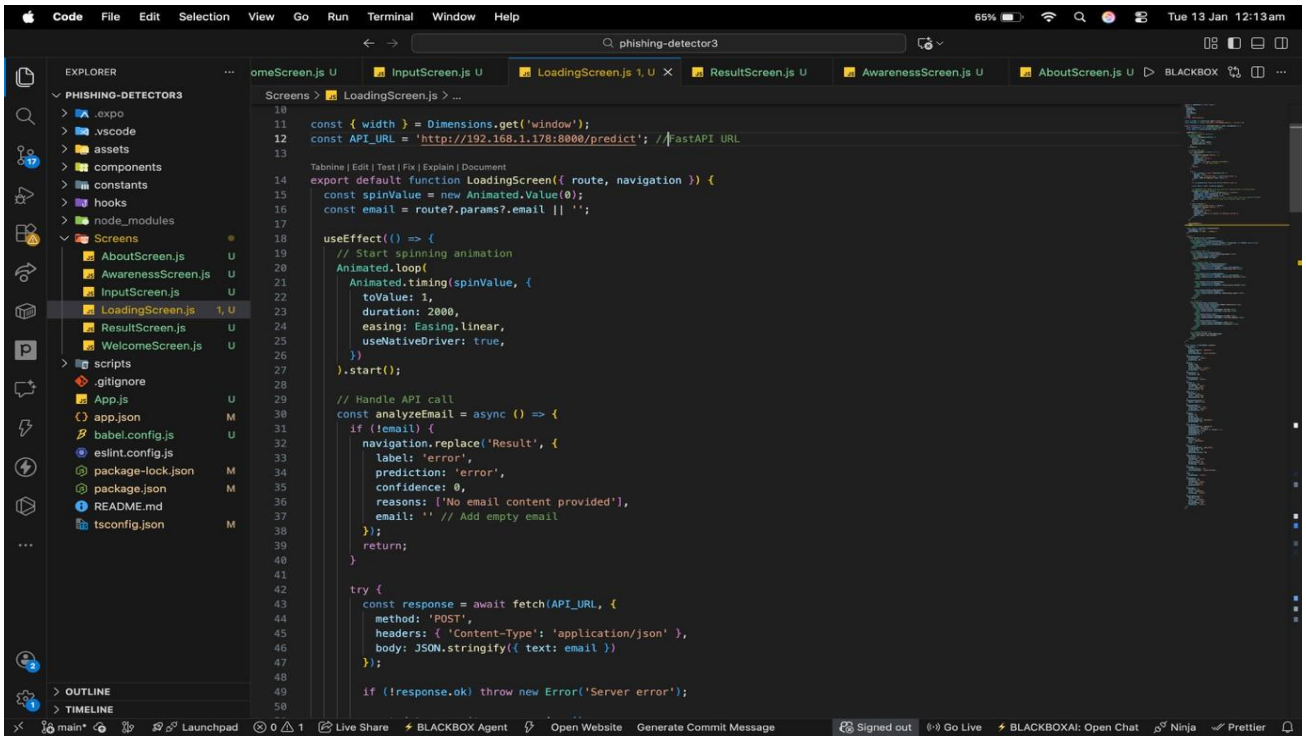


Figure 15: API call code in react native

These optimisations are crucial, as high latency may reduce user trust in the detection system and its usability. Many prior works focus on offline evaluation; they demonstrate high precision on curated datasets with no real-world constraints. These methods do not account for the need for real-time response, as required for detecting mobile or web-based phishing. Practical deployment concerns include scaling the model under real-time load, processing a variety of email content quickly, and staying responsive while providing understandable answers. The latter accounts for model performance in terms of real-world applicability, ensuring users receive timely, actionable advice on potential phishing threats. This demonstrates that high accuracy is not enough; both speed and quality of feedback are also important if researchers want to develop a user-centric phishing detection system.

3.8. User Interaction and Explainability Design

PhishGuard focuses on explainability and ensures that outputs are provided in an interpretable, actionable manner. Instead of simply displaying a single message, whether it's phishing or safe, the system uses confidence scores, risk indicators, and

reason summaries. Confidence scores estimate the likelihood that an email is a phishing attack, enabling users to focus on high-risk emails. Risk indicators provide a rapid, high-level threat assessment, while reason summaries enumerate particular suspicious characteristics to help describe why a message is considered suspect. Reasons are highlighted based on key phishing indicators, such as urgent or intimidating message text, strange sender addresses, and content inconsistencies. For example, a reason summary may identify an apparent sender domain that does not match the claimed organisation, or use language that urges immediate action. These attributes of models lend transparency to their outputs, enabling decision-makers to avoid being confined by automatic predictions (Figure 16).

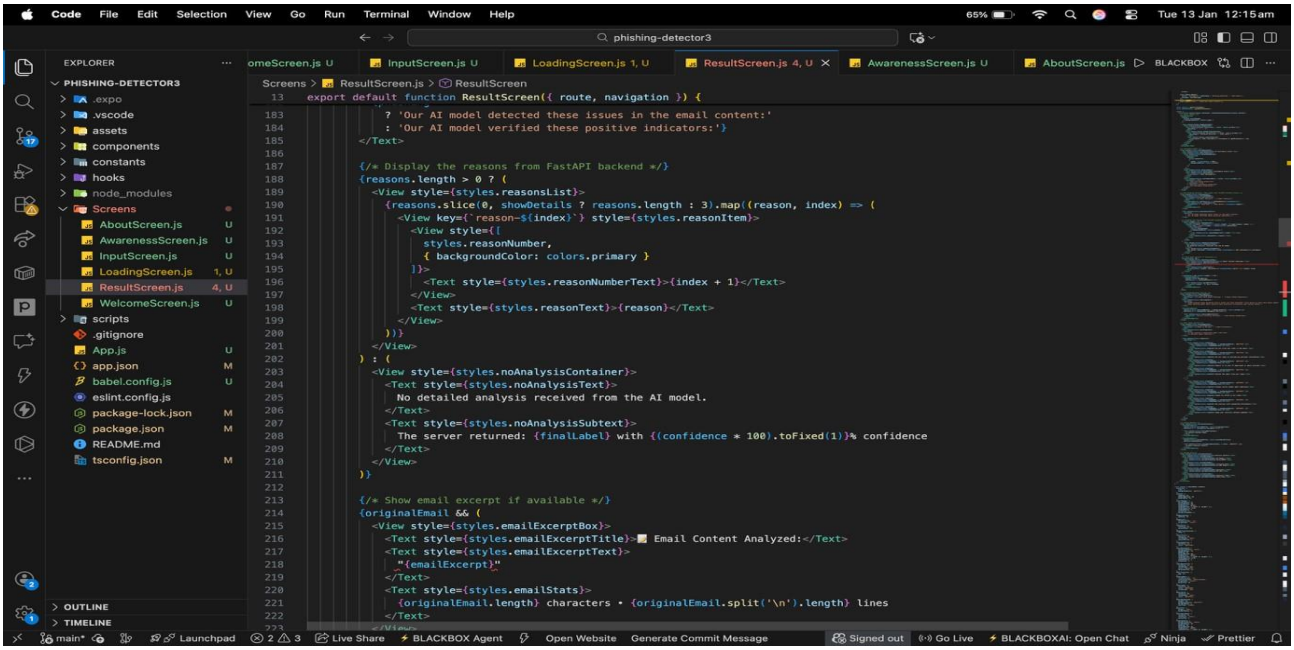


Figure 16: Email scanning result code

Explainable explanations are important for user trust and security. While the predictions are mapped onto observable properties of an email, the system encourages safer behaviour and alertness to learn from phishing attacks. This design allows non-expert users to access state-of-the-art detection capabilities, thereby closing the loop between automated detection and human readability. Leveraging these results, our solution customises the reason summaries and risk indicators to support real-time user decision-making, ensuring results are accurate and actionable.

3.9. Evaluation Metrics and Experimental Setup

Model performance was assessed using Accuracy, Precision, Recall, and F1-score. Accuracy is general correctness; Precision is the fraction of predicted phishing emails that are correct; Recall is the ratio of true phishing emails discovered; and F1-score combines both performance measures, Precision and Recall (Figure 17).

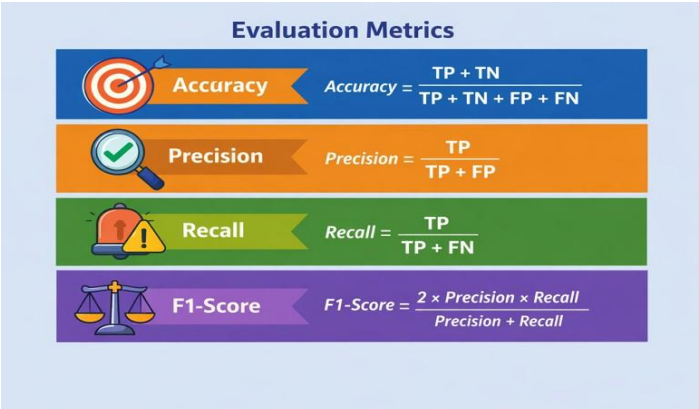


Figure 17: Evaluation metrics used for phishing email classification

These scores are based on true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). Model performance was assessed using Accuracy, Precision, Recall and F1-score. Accuracy indicates how many emails are classified correctly; Precision represents the percentage of predicted phishing emails that are correct; Recall reflects the number of observed phishing-rated emails; and F1-score is a measure of precision. These statistics are derived from true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

Table 1: Comparative performance of phishing detection models

Modal	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.9787	0.9744	0.9837	0.9790
XG Boost	0.96.76	0.9517	0.9858	0.9684
DistilBERT	0.991	0.9886	0.9935	0.991

The same fixed train–test split was used for all models to ensure comparability. An execution-time analysis complemented the predictive performance to assess the feasibility of practical deployment.

Table 2: Training and prediction time for evaluated models

Modal	Training Time (s)	Prediction Time (s)
Logistic Regression	0.31	0.01
XG Boost	130.65	0.09
DistilBERT	1000.55	41.76

However, classical models do not fit real-time mobile use cases, and accuracy is always traded off against latency. Although training and inference take a long time, DistilBERT achieves higher recall and F1 scores; therefore, researchers choose to use it as a back-end model in the mobile application. In addition to the above, scalable transformers also enable actionable confidence scores and risk indication, as well as reason summaries. Further enhancements might adopt ensemble learning and interpretability approaches, such as LIME or SHAP, to boost the model's robustness and interpretability while maintaining real-time prediction.

3.10. Legal and Ethical Considerations

This work analysed only public phishing email datasets, meaning no personal or sensitive user data was collected or stored in the process. To the extent that no private data is included in the original datasets, this study respects privacy and complies with data protection regulations. Biases were solicited by dataset balancing and a careful train-test split. Balancing the number of phishing and legitimate emails is important to prevent models from being dominated by the majority class, thereby avoiding biased predictions and ensuring fairness in real-world applications. The responsible AI principles guided the entire paper. The model's outputs are interpretable and actionable, with confidence values and risk indicators provided to end users. This provides a safeguard against unquestioningly trusting the machine's predictions and encourages consideration before action. Last but not least, the study is in line with academic research ethics, including transparency and reproducibility of methodology, as well as proper data and literature referencing. Privacy-aware data use, bias mitigation, and responsible deployment make the system potentially both effective and ethical for real-world phishing detection.

4. Experiments and Results

4.1. Experimental Environment and Setup

All tests were run in both cloud and local environments for model training and system deployment. Researchers have trained and evaluated the model using Google Colab, which offers free access to a GPU. The environment was chosen because it is readily available, reproducible, and can be used to train deep learning models on systems with limited local hardware resources. Traditional machine learning models were trained on a CPU, and DistilBERT training and inference were performed on a GPU to minimise training time, as shown in Figure 18. Python and popular machine learning libraries comprised the software stack. Logistic Regression and XGBoost were developed with common Python machine learning libraries, while DistilBERT was trained in PyTorch and the Hugging Face Transformers library. This facilitated compatibility with the latest research routine and reproducibility. Real-time prediction. Real-time inference was served by a backend system built on top of FastAPI. The mobile frontend was implemented with React Native and tested in Expo Go, enabling cross-platform development throughout the development phases. The dataset for the analysis comprises 57,804 labelled emails obtained from publicly available sources. The dataset was split into training and test sets at 80/20, yielding 46,243 training samples and

11,561 test samples. To make a fair comparison, the same train-test split is used consistently across all models. Three classification models were tested, including Logistic Regression, XGBoost, and DistilBERT. This ensures fair comparison of classical and transformer-based models under the same experimental conditions.

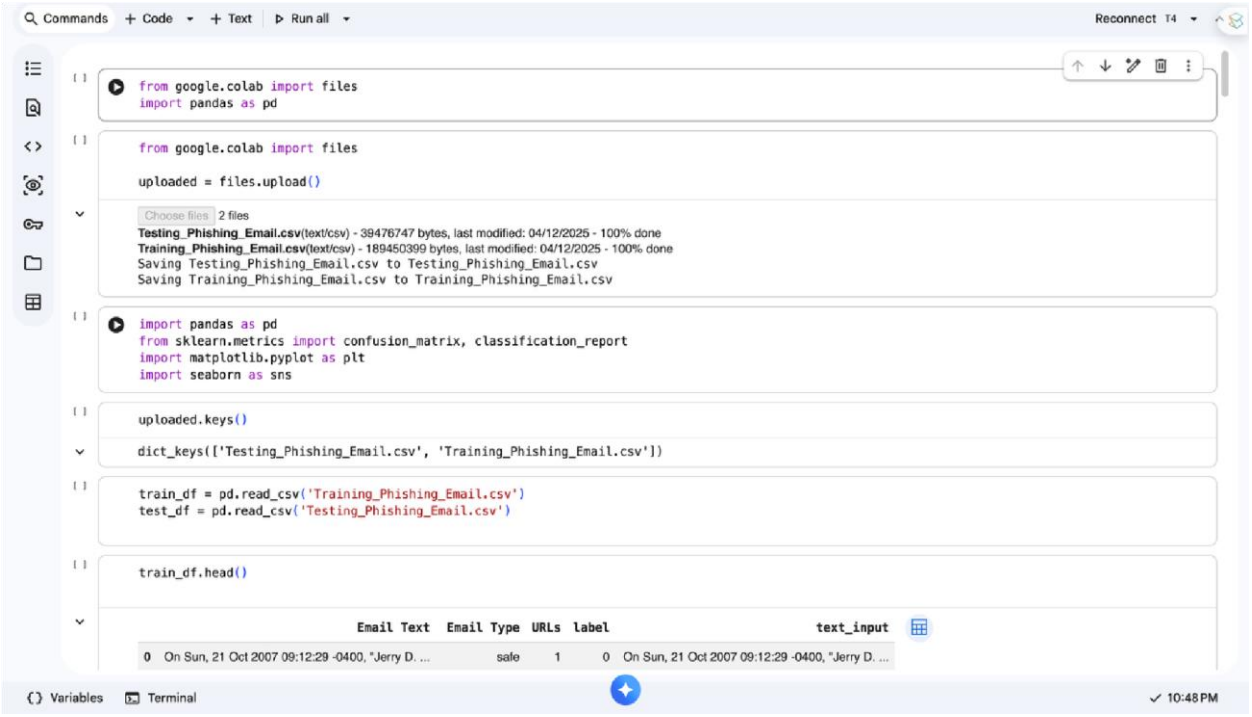


Figure 18: Setting up Google Colab

4.2. Evaluation Metrics

Four classical classification metrics, accuracy, precision, recall and F1-score, were used to assess the phishing detection models, whose formulations are illustrated in Figure 19. These measures give a fuller understanding of model performance, especially in imbalanced datasets, as with email phishing detection.

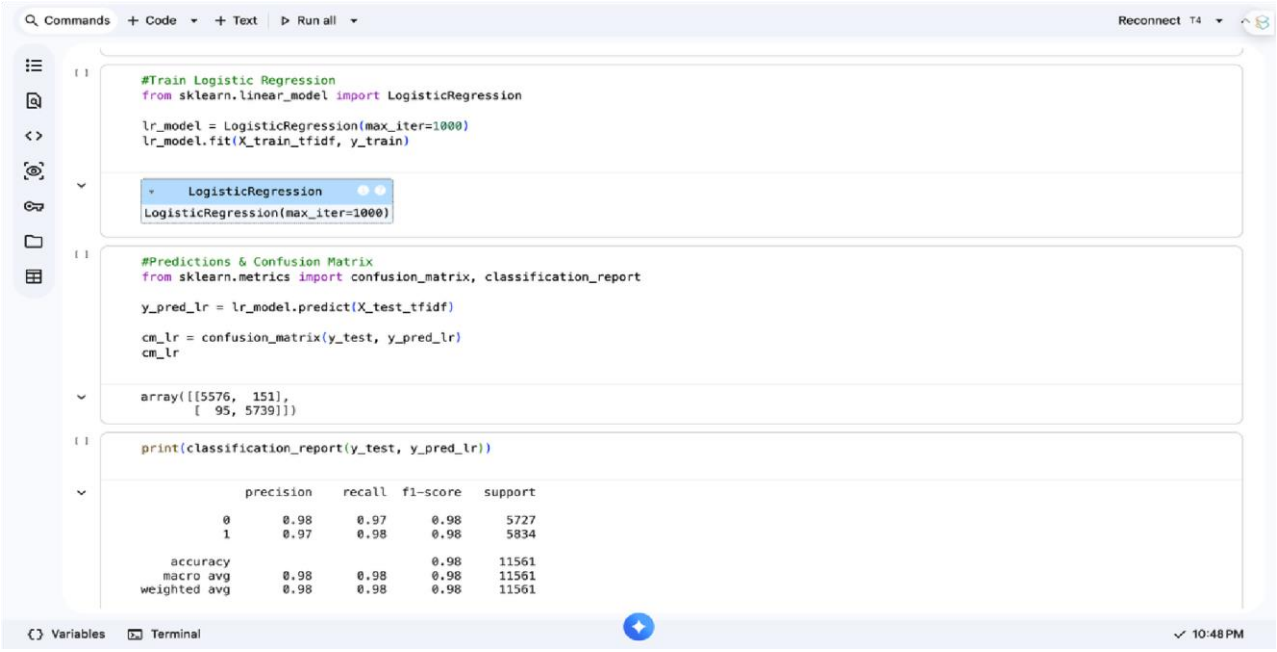


Figure 19: Logistic regression classification report and confusion matrix

Accuracy is the percentage of emails correctly categorised into both the phishing and non-phishing classes. Although effective at providing an overall performance summary, accuracy is misleading for phishing detection, with legitimate emails dominating the majority class (Figure 20).

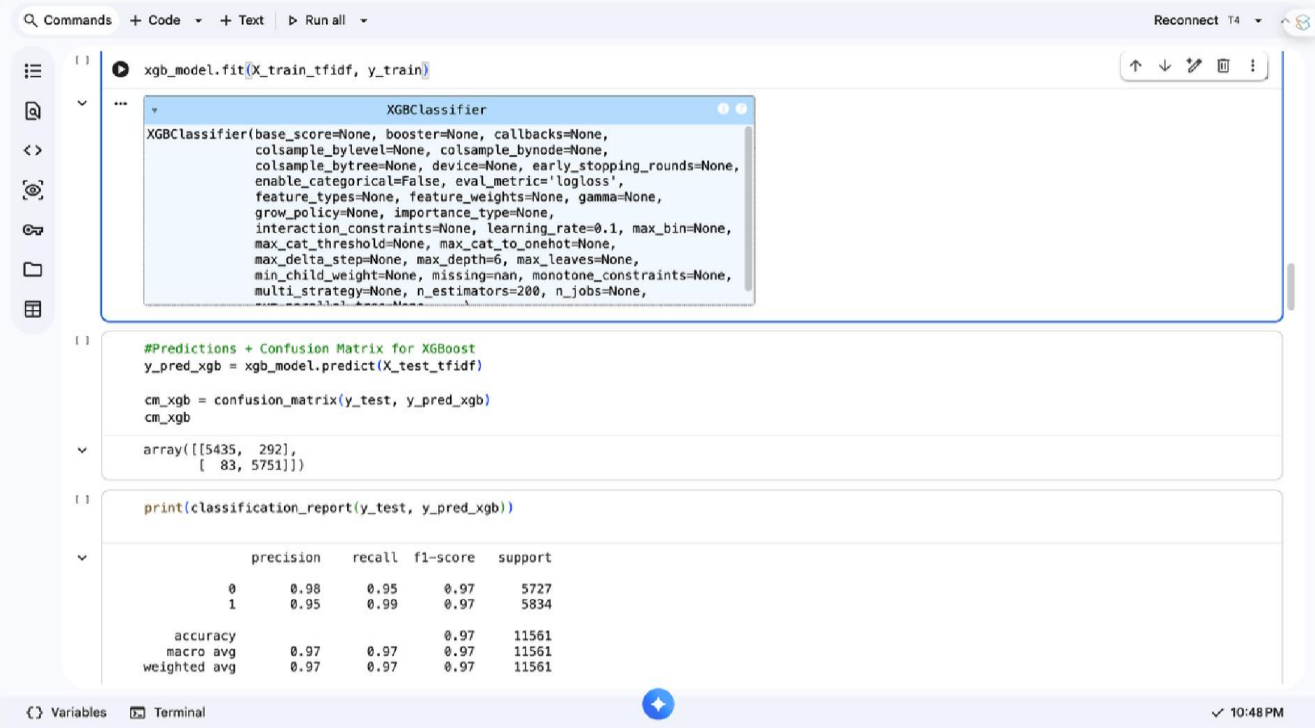


Figure 20: XGBoost classification report and confusion matrix

Phishing Email Identifying models can also work extremely well, but still perform poorly, finding only a few phishing emails. Precision is the percentage of actual phishing emails our system identified as such. High precision means fewer false positives and prevents real emails from getting labelled as spam. But precision does not indicate how many phishing emails fall through the cracks. Recall is the ratio of real phishing emails that are accurately identified. This is especially critical for mitigating phishing, as false negatives directly expose users to fraud, malware, or credential compromise. As a result, recall is emphasised in the analysis. The F1-score is the harmonic mean of precision and recall, balancing false positives and false negatives. It is particularly useful for model comparison with an imbalanced design. Jointly, these measures establish a security-centric, robust assessment of the phishing system, ensuring that both detection accuracy and risk mitigation are well covered.

4.2.1. Reasons to Choose Evaluation Metrics

These evaluation metrics are chosen due to the operational needs of the phishing detection system. First of all, accuracy, as it gives a mean performance, should be interpreted cautiously, as accuracy alone can hide differences in phishing vs safe email detection. Then, Precision measures system trustworthiness. High precision means the flagged emails are probably real threats. Recall is chosen because it measures security effectiveness. High recall is necessary to catch phishing emails and reduce the risk of security breaches. In cybersecurity, false negatives tend to be costlier than false positives. F1-score, as it balances precision and recall. Selected as the primary score optimisation because it considers the tradeoff between false positives and false negatives. The confusion matrix is chosen because it indicates the exact error types, allowing risks to be assessed (false positives vs false negatives). And the reason why Receiver Operating Characteristic (ROC) and Area Under the Curve (AUC) are not chosen is that a phishing email must receive either a yes or a no decision. Also, ROC-AUC measures ranking by probability and is less useful for real-time deployment in email security.

4.2.2. Cross-Validation Methodology

For a fair comparison, stratified 5-fold cross-validation was performed to evaluate the classical models, including Logistic Regression and XGBoost, using 80% of the training data. Five folds were created while preserving the class distribution to maintain balanced proportions of phishing and legitimate emails in each fold. The evaluation reports the mean performance and standard deviation across all folds, as illustrated in Figure 21.

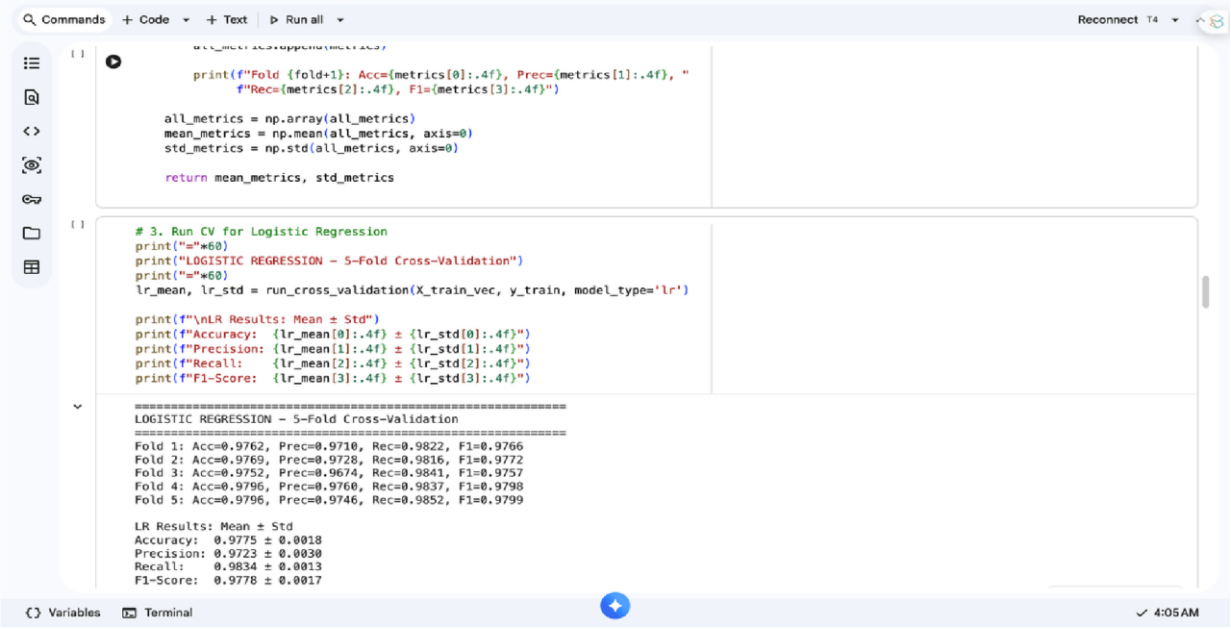


Figure 21: Logistic regression cross-validation

For DistilBERT, it was evaluated using an 80-20 train-test split. A reason not to use cross-validation is that, as training transformers, it takes approximately 45 minutes per run. This procedure is in accordance with the general principles of deep learning practices found in recent research (Table 3).

Table 3: Cross-validation results (5-fold, Mean ± SD)

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.9775 ± 0.0018	0.9723 ± 0.0030	0.9834 ± 0.0013	0.9778 ± 0.0017
XGBoost	0.9768 ± 0.0014	0.9688 ± 0.0020	0.9858 ± 0.0025	0.9772 ± 0.0014

DistilBERT was evaluated using an 80-20 train-test split. DistilBERT would take 225 minutes for 5 folds. So, it is impractical within the paper constraints (Figure 22).

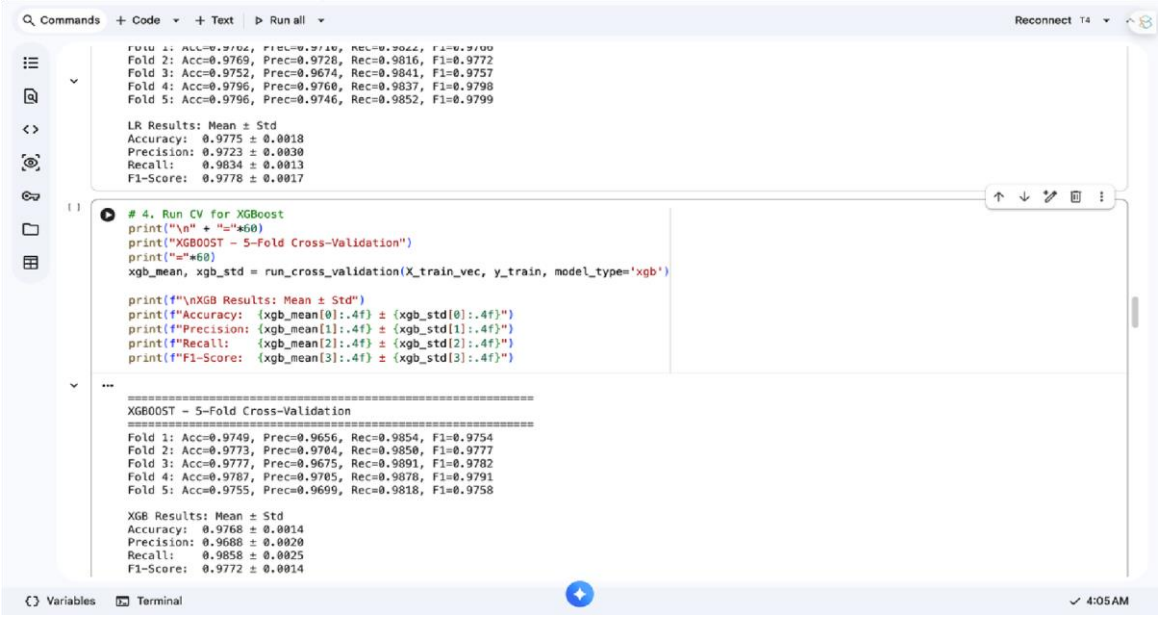


Figure 22: XGBoost cross-validation

4.3. Comparative Model Performance

The three phishing detection models, Logistic Regression, XGBoost and DistilBERT, are compared using common classification measures such as accuracy, precision, recall, and F1-score. The results are reported in Table 1. The respective training and prediction times are shown in Table 2. Together, prediction performance and computational efficiency provide a thorough basis for choosing a viable model for real-time mobile deployment. DistilBERT was the most accurate model, with 99.10% accuracy, compared to Logistic Regression (97.87%) and XGBoost (96.76%). Also, DistilBERT showed a superior performance in precision, recall and F1-score, all of which were close to 0.99, demonstrating a robust detection of phishing emails with fewer false positives and false negatives. Recall is crucial in phishing detection, since phishing emails are usually a minority in the dataset and missing them can be costly for consumers, leading to fraud, stolen credentials or malware downloads. Therefore, DistilBERT's high recall performance supports its applicability to security-critical applications.

Although it performs less well, especially in recall and F1-score, Logistic Regression is less suitable when high detection confidence is needed, yet it is computationally efficient. However, it provides a basic architecture and highly fast inference, taking only 0.01 seconds to classify a new e-mail. These properties are desirable for scenarios in which computation is minimal for at most n steps, but they are less expressive than those of more complex models. XGBoost can balance predictive performance and efficiency. It has a recall of 0.9858 and an F1 score of 0.9684, and it performs well on both classes, with reasonable training and inference times, as will be evident from the experiments section. XGBoost's prediction of a single email in 0.09 seconds is much faster than running DistilBERT, making it a good choice when researchers need to balance the model's complexity with fast prediction. As a transformer-based deep learning model, DistilBERT also has a much longer training time (1000.55 seconds) and inference time (41.76 seconds per email). Meanwhile, its linguistic pattern knowledge is the strongest among all in detection. This is especially suited for backend deployment in the PhishGuard system, where the model doesn't need to be self-contained and can use server-side resources; computation can be done on the servers, and anglers can receive results from their mobile app without delay in model responses (Figure 23).

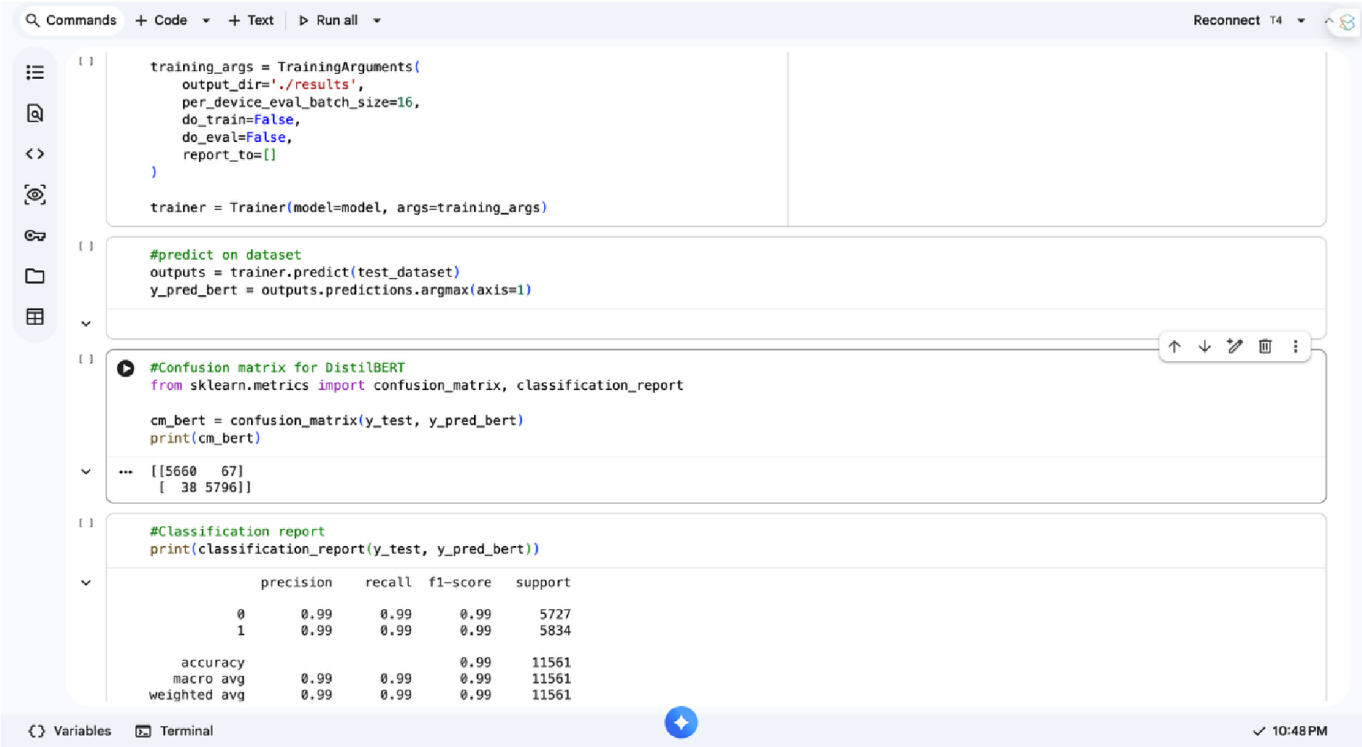


Figure 23: Classification report and confusion matrix for DistilBERT

Typical results reveal the natural trade-off between speed and prediction quality. Faster models like Logistic Regression and XGBoost achieve higher speed but lower recall and overall trustworthiness. A more robust phishing detection. Unlike Universal Sentence Encoders, DistilBERT brings strong protection that can be used across a variety of advanced app functionalities, including AI confidence scoring and detailed reasoning summary. Even though DistilBERT has higher training and inference latency, it achieves stronger predictive performance in terms of recall and F1-score; therefore, it is well-suited for the backend inference stage of mobile applications.

4.4. Confusion Matrix Analysis

The confusion matrices of all the examined models are shown in Figure 24 (a–c). These matrices show the accurate and incorrect predictions and provide a detailed assessment of how each model performs in categorising both authentic and phishing emails.

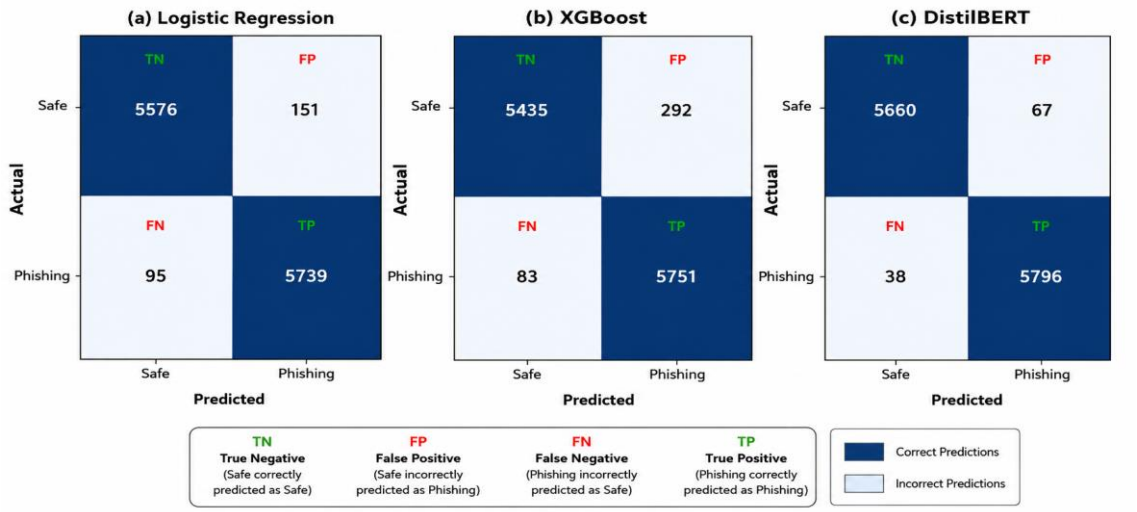
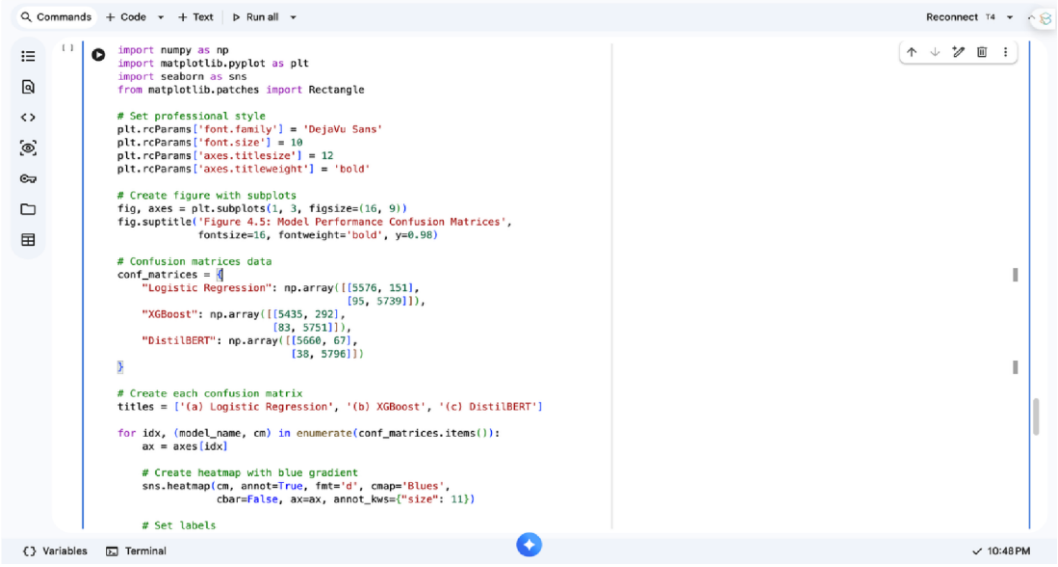


Figure 24: Confusion matrices for modals (a-c)

The Logistic Regression model correctly classified 5,739 phishing emails out of 5,834, resulting in 95 false negatives. And it correctly flagged 151 legitimate emails as phishing (false positives). Even if Logistic Regression offers lower computational complexity and faster inference, which are ideal for low-resource settings, the 95 phishing emails that were not detected represent substantial security vulnerabilities. XGBoost performs better than Logistic Regression; it accurately predicted 5,751 phished emails and had only 83 false negatives. But it has 292 false positives. Although it achieves faster inference times than deep learning models, XGBoost achieves superior classification performance than the linear model across all measures. The transformer-based model, DistilBERT, classified 5,796 phishing emails correctly at the cost of only 38 false negatives. It also reduces the number of false positives to 67. DistilBERT missed 60% fewer phishing threats and drop 56% fewer safe emails, making it the best tradeoff between security and usability. This high-level performance is crucial for phishing detection; as false negatives pose a significant security threat by allowing malicious emails to reach users. Since DistilBERT performs well in terms of sensitivity and exhibits strong contextual comprehension, it could be the optimal model for backend service deployment in the PhishGuard system (Figure 25).



```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.patches import Rectangle

# Set professional style
plt.rcParams['font.family'] = 'DejaVu Sans'
plt.rcParams['font.size'] = 10
plt.rcParams['axes.titlesize'] = 12
plt.rcParams['axes.titleweight'] = 'bold'

# Create figure with subplots
fig, axes = plt.subplots(1, 3, figsize=(16, 9))
fig.suptitle('Figure 4.5: Model Performance Confusion Matrices',
             fontsize=16, fontweight='bold', y=0.98)

# Confusion matrices data
conf_matrices = {
    "Logistic Regression": np.array([[5576, 151],
                                     [95, 5739]]),
    "XGBoost": np.array([[5435, 292],
                         [83, 5751]]),
    "DistilBERT": np.array([[5660, 67],
                            [38, 5796]])
}

# Create each confusion matrix
titles = ['(a) Logistic Regression', '(b) XGBoost', '(c) DistilBERT']

for idx, (model_name, cm) in enumerate(conf_matrices.items()):
    ax = axes[idx]

    # Create heatmap with blue gradient
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                cbar=False, ax=ax, annot_kws={"size": 11})

    # Set labels
```

Figure 25: Confusion matrices code

Overall, the confusion matrix analysis supports the choice of DistilBERT for real-world applications. On the one hand, smaller models are faster to predict but have a higher risk of false negatives. DistilBERT's strong performance ensures that phishing emails are correctly identified, providing features such as confidence scoring and actionable user information (Figure 26).

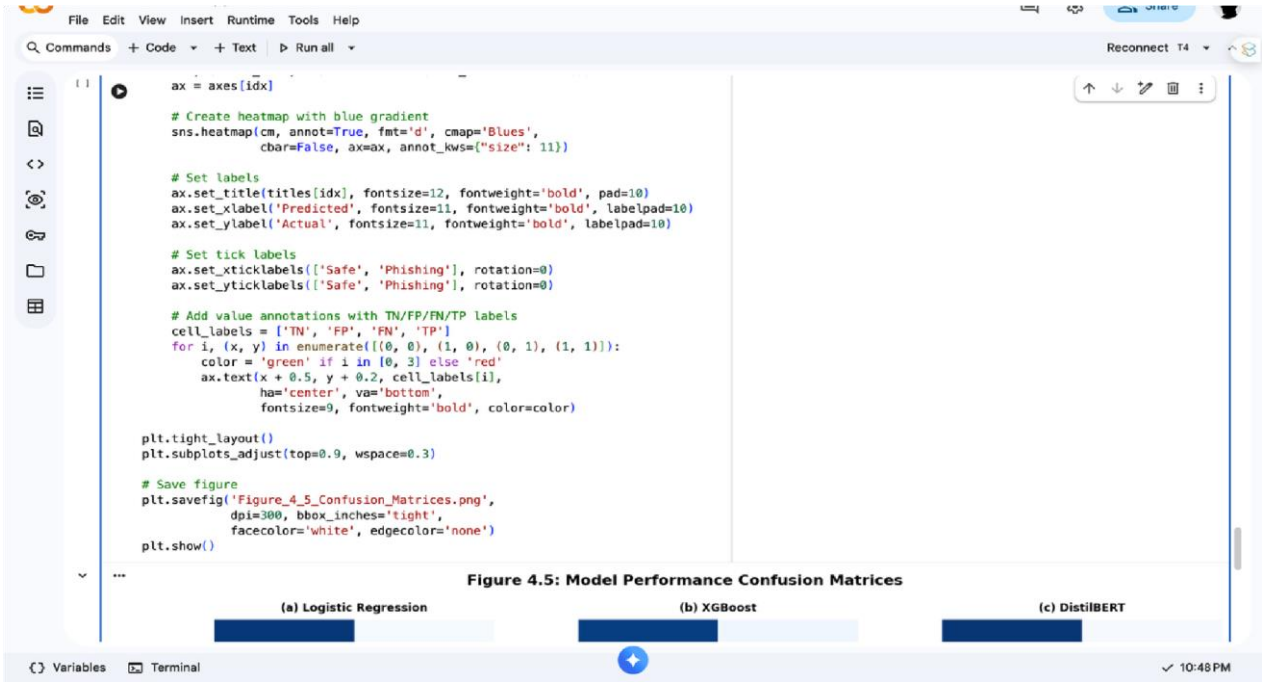


Figure 26: Setting value annotations for confusion matrices

4.5. Real-Time System Performance

The real-time status of the PhishGuard system is a critical component of its effectiveness. The workflow starts with a welcome screen that indicates the application is ready to analyse emails and displays the model used, an accuracy score, and a few statistics. From there, users move to the input screen, where they can paste an email for inspection or choose a phishing or non-phishing sample email. When the user clicks submit on the email, tokenisation, feature extraction, and inference with the DistilBERT model take place in the background during the loading screen. And last but not least, the result screen provides a definitive judgment on whether this email is safe or not, the level of confidence of the AI consumer discrimination model, with detailed explanation pointing to indications or warning signs.

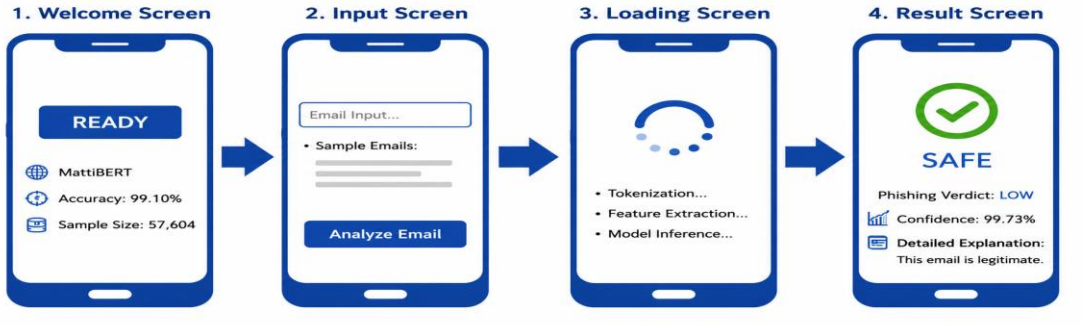


Figure 27: Mobile application workflow

Figure 27 shows a workflow visualisation of user interactions across application screens and the backend inference for energy analysis. The system has been proposed for server-side use rather than on-device testing, relying on a server's computational capacity to run a transformer-based model such as DistilBERT effectively. In this manner, the mobile application maintains a smooth, responsive user experience while still providing high-quality predictions without taxing the device's CPU or battery. On average, the API response time for email analysis is in a few seconds, so the user does not have to wait too long. This

backend-centric approach enables complex computations, such as tokenisation and feature extraction, to be executed efficiently while keeping the mobile application responsive. In addition to presenting the classification results, the report screen includes actionable information, such as a list of flagged suspicious material detected. It suggests next steps, promoting both real-time detection and user education. In a word, this design strikes a fine balance between robust detection and practical applicability. Users receive ultra-fast, ultra-reliable phishing detection capabilities, combined with plain-language explanations of why content is not safe and clear direction for reporting unsafe Web resources, all running on slim hardware inside the user's device, the source says.

4.6. Critical Discussion

The PhishGuard application not only offers high predictive power but also incorporates interpretability and user feedback, thereby mitigating a limitation of previous work in phishing detection (Figure 28).

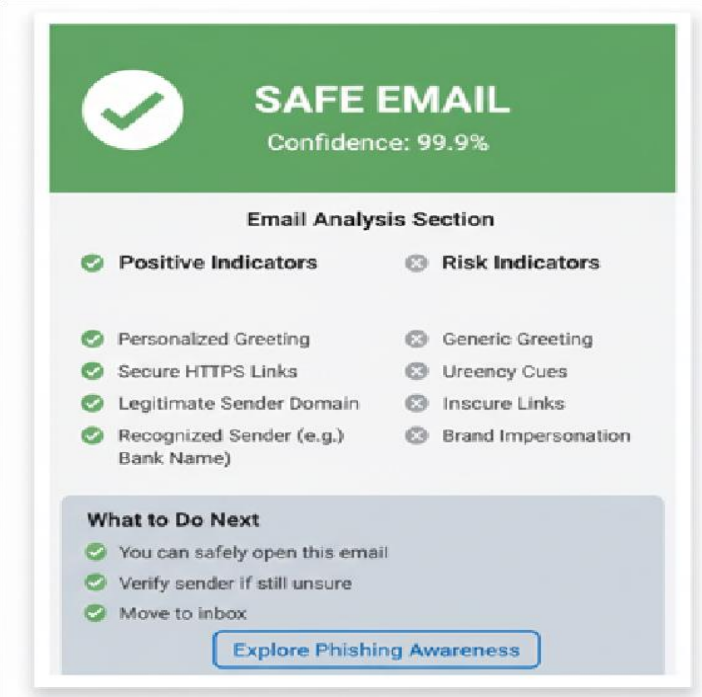


Figure 28: Positive indicators highlighted in the result screen for a safe email

While the former work, Fares et al. [13], exhibits good classification performance, it is more of an evaluation of metrics and lacks user insights or a decision report. PhishGuard displays positive points and risk factors for each email, along with actionable feedback in its UI and a confidence score, which, in this case, is high. This directly reduces user disorientation and suspicion, helping bring model predictions closer to user understanding. For safe emails, the model looks for positive signals, including personalised content, valid, secure HTTPS links, and proper sender identification. In phishing emails, it highlights generic greetings, urgent language, insecure links and brand impersonation. These are presented on separate screens, demonstrating the system’s potential to communicate the justification for classifications. By providing such cues, the model ensures that users can act in an informed manner, e.g., refrain from clicking a suspicious email or harmful messages solely on predictions.

PhishGuard represents a significant improvement over the past work as follows. Fares et al. [13] achieve high classification accuracy with machine learning models, but they do not explain their results meaningfully to the user. In comparison, incorporating explainability into PhishGuard enhances its operational usability, especially in real-world settings where user behaviour is the focus of security efforts. Moreover, the system achieves a high recall (0.99) and F1-score (0.99), indicating that almost all the phishing emails are detected with few false positives. This is crucial because of the imbalance between phishing and legitimate email examples in datasets. The explicit coalescence of high-performance detection and user-oriented reasoning warrants selecting DistilBERT as our final model. While computationally more expensive than Logistic Regression or XGBoost, this contextual awareness it provides improves detection and helps incorporate explainability features, which is handy for server-side deployment in the mobile app. Overall, this approach demonstrates technical and practical improvements over existing work, resulting in a solid, easy-to-use, and actionable phishing detector (Figure 29).

Figure 29: Risk indicators highlighted in the result screen for a phishing email

4.7. User Awareness

PhishGuard offers a feature-rich toolbar with a dedicated user education module that provides early-warning tools and helps users learn about phishing dangers. The module is divided into 4 interactive tabs, each with a specific educational goal (Figure 30).

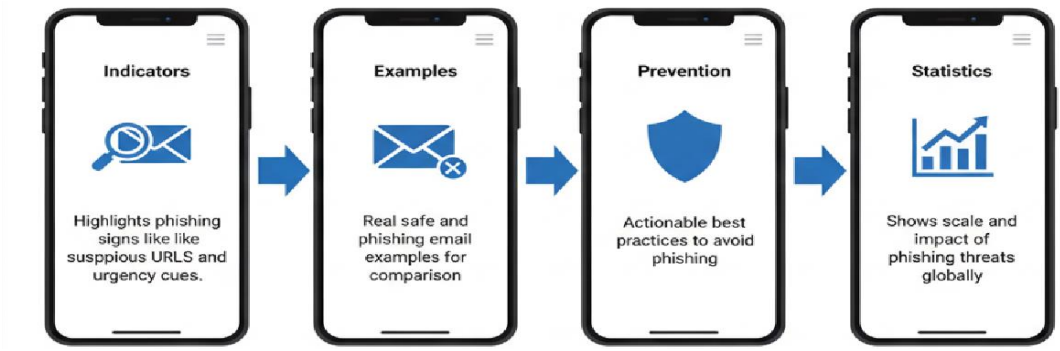


Figure 30: PhishGuard security awareness module and its four major tabs

The indicators tab features several standard indicators that cybercriminals and other threat actors use when sending phishing emails, e.g., an unfamiliar URL, generic addressing, excessive pressure to complete requested actions, improper language or spelling and fraudulent sender information requests. This allows the user to identify threats before engaging with suspicious emails. To see examples of actual emails, including phishing and legitimate messages. Through these comparisons, users learn the commonalities and nuances that distinguish legitimate domains from phishing sites, equipping them to spot phishing attacks in the wild:

- **Prevention:** This section presents tips for when action is required, such as verifying the sender, hovering over links rather than clicking them, setting up two-factor authentication, keeping your software up to date, and reporting suspicious emails, along with ongoing education. These steps help users be proactive in reducing their risk of being phished.

The Statistics tab provides an overview of phishing worldwide, including how many phishing attacks occur each day, what percentage of breaches are caused by phishing, and the financial implications of a successful phishing. In this context, information empowers such vigilance and promotes good habits. Browsing within the app is especially impactful, exceeding that of externally recommended training, as learning is contextualised during a live detection event. The knowledge a user gains through the awareness module can be directly applied to the analysed emails, resulting in a continuous feedback loop of

detection and education. And this unity boosts engagement, retention, and, in turn, the effectiveness of the phishing prevention program. Briefly, the experiments confirm that DistilBERT outperforms Logistic Regression and XGBoost for phishing email detection, achieving the highest accuracy and F1 score. The detection process was developed as a real-time analysis workflow that can be easily embedded in the PhishGuard mobile application to analyse user-submitted phishing URLs and provide actionable explanations, along with confidence scores, to users. The explainability features, e.g., positive and risk indicators, and the forward steps increase users' trust and confidence. Furthermore, the built-in security education module helps people apply what they learn and practice safe email habits. Collectively, these findings validate the system's efficacy as a unified counter-phishing and education solution with pervasive mobile accessibility, thereby satisfying the research objectives.

5. Discussion, Limitations and Challenges

5.1. Discussion of Results

The experimental results show that the designed phishing detection system delivers high-performing models, with clear trade-offs in prediction quality, computational cost, and deployment applicability. In contrast to simply presenting raw performance figures, the results are presented in relation to phishing risk and system design limitations. Taken together, these results show that all three models are likely to identify phishing messages successfully. But those methods have limitations based on the sophistication of phishing content and the operational support required for deployment. Logistic Regression provides very fast predictions with low computation cost. This advantage surely improves its robustness against advanced phishing attacks. However, it lacks strong context and semantic capture, so against well-crafted phishing that involves subtle social engineering and language manipulation, robustness might not be assured. XGBoost is an upgrade over Logistic Regression by capturing feature interactions, leading to better detection accuracy without a significant loss in inference speed. Its balance makes it a convenient compromise for systems of modest resources.

However, this makes it difficult to quantify resistance to novel phishing approaches, as adversaries adapt their wording and structure to bypass such features. Overall, DistilBERT detects more reliably than all other models. Its transformer-based architecture can model deeper language patterns, helping it better detect persuasive phrasing, urgency signals, and brand impersonation. This strongly mitigates false negatives, which is very important for phishing detection, because a missed phishing email may expose users to potential credential loss, financial loss, or malware infection. An intuitive conclusion from the results is that there is a trade-off between detection accuracy and latency. Although Logistic Regression and XGBoost are very fast at prediction, their limited expressive power is detrimental to robust detection. DistilBERT increases training and inference overhead; such a trade-off is reasonable for safety-critical tasks prioritising consistency over minimal response time. In the PhishGuard approach, this problem was solved by deploying to the backend, meaning inference is performed by a FastAPI service rather than running it on the device, thereby achieving headless interaction while still preserving detection quality.

5.1.1. Study Model Used Best Real-Time Explainability Reported Deployment Accuracy

Table 4. Comparison of the suggested system with previous studies on Phishing detection. In contrast to prior work, which mainly focuses on classification accuracy with traditional machine learning models, the proposed system provides high detection performance, real-time deployment, and explainable outputs for end users.

Table 4: Comparison of the proposed system with a recent phishing detection study

Fares et al. [13]	SVM	97.6%	No	No
This study	DistilBERT	99.1%	Yes (FastAPI backend)	Yes

This connection offers additional practical value by increasing user awareness and enabling more informed decisions. Overall, the obtained data show that DistilBERT is a good model for the PhishGuard system. Its robust detection performance, quick backend processing and explainable outputs make it a feasible solution for real-time detection of phishing attacks while improving user safety.

5.2. Limitations

Despite the good performance and practical applicability of the proposed phishing detection system, several limitations remain to be addressed. It is important to be aware of these restrictions when interpreting results and planning for the future. One primary restriction pertains to the dataset on which this study is based. Models were trained on publicly available datasets. Despite their popularity in phishing research and their ability to provide a good baseline for comparison with previous work, these datasets do not reflect the dynamic nature of real-world phishing campaigns, which continually evolve. Phishing tactics

evolve all too quickly, and perpetrators are constantly refining language, content, and payload URLs to stay a step ahead. Therefore, models trained on historical records are prone to performance degradation when exposed to new attack types. While the class imbalance of 0 and 1 is much reduced in this study, and a large dataset was used, the absence of live or regularly updated emails prevents assessment of long-term robustness. Model-specific limitations are also evident. Traditional machine learning models, e.g., logistic regression and XGBoost, are based on human-engineered feature designs. Unfortunately, their generalisability is limited, since they rely on predefined features and must be trained manually. Such models enable fast inference and require little computation, though they are less expressive than deep learning approaches. DistilBERT addresses this by learning contextual language, but it also presents new challenges. Training transformer-based models is significantly resource- and time-consuming, particularly during fine-tuning. This increases development costs and could limit adoption in resource-limited settings.

Table 5: Limitations of evaluated phishing detection models

Modal	Key Strengths	Key Limitations
Logistic Regression	Fast inference, low resource usage	Limited feature expressiveness
XG Boost	Balanced speed and accuracy	Dependence on handcrafted features
DistilBERT	High accuracy and recall	High training and inference cost

Deployment-related constraints also impact system design. DistilBERT is too computationally intensive to perform on-device inference on mobile platforms. Therefore, the system relies on the backend inference using a FastAPI server. Although this setup guarantees a pleasant user experience, it creates dependence on network connectivity. Real-time detection is difficult or not possible in the context of low or no internet access. Furthermore, it is known that the backend deployment has scalability and server-maintenance issues, but these are out of scope for this work. Table 5 presents the main strengths and weaknesses of all the models considered, emphasising the trade-offs involved in selecting a proper solution. Nevertheless, the system shows that integrating backend inference and explainability can yield an efficient, practical phishing-detection system.

5.3. Challenges

In this section, researchers discuss the key technical and engineering obstacles faced during the design, development and evaluation of the phishing detection system and the strategies taken to overcome each challenge. These issues stem from system-level engineering and machine learning decisions in a real-time mobile application. The initial challenge was to decide between traditional feature extraction methods and contextual embeddings. Classic models like Logistic Regression and XGBoost depend on hand-crafted features that must be carefully selected and optimised. This strategy results in faster inference but lacks scalability and the ability to adapt to new phishing tactics. Transformer-based embeddings, on the other hand, can automatically capture semantic and contextual linkages in email content but are more computationally expensive. To find a middle ground, Researchers chose the lightweight DistilBERT model. It offers contextual comprehension but requires less memory than the full-scale transformer models. Another major problem was dealing with class disparity. The majority of phishing datasets contain fewer phishing emails than legal emails, which can lead to misleading accuracy findings. A model could achieve high accuracy, but still not detect phishing attempts efficiently. To solve this problem, in addition to accuracy, Researchers examined recall and F1-score. This evaluation method primarily aims to detect phishing emails and minimise false negatives, which directly affect user security (Table 6).

Table 6: Key technical challenges and mitigation strategies

Challenge	Impact	Mitigation Report
Class imbalance	Biased evaluation results	Recall- and F1-focused evaluation
Real-time latency	Degraded user experience	Backend inference via FastAPI
Explainability	Reduced user trust	Indicator-based explanations
Feature engineering	Limited scalability	Transformer-based embeddings

Additionally, the latency constraints of real-time prediction were challenging to meet, especially with a transformer-based model. Such models are not very realistic to run directly on a mobile phone due to memory and computational limitations. This was solved by migrating the model inference to a FastAPI backend and using the mobile app as a thin client. This implies network dependency but improves user interaction by avoiding heavy computation on the device. This is in contrast to the common practice in actual security systems. The trade-off between explainability and model complexity was another concern considered. Although deep learning models are often highly accurate, they are commonly criticised for being black boxes. To mitigate this challenge, the system incorporates atomic interpretations to showcase positive signals for safe email and suspicious signs for phishing email. This strategy enhances user trust without imposing a substantial burden on the system infrastructure or causing latency. Finally, the combination of a mobile frontend with a backend inference service brought

engineering trade-offs. The applications were tested in a development environment using Expo Go and a locally running FastAPI server on the local network. While no such setup can be considered a full production deployment, it is sufficient for analysing architectural choices with respect to latencies and user interaction. These limitations and trade-offs are insightful for understanding the implementation constraints when deploying ML-based anti-phishing systems.

6. Conclusion and Recommendations

6.1. Conclusion

The purpose of this research was to develop a realistic, explainable phishing detection system for mobile phones. The suggested solution combines machine learning approaches with human-centred design to balance between detection accuracy and usability. Researchers have examined DistilBERT, Logistic Regression and XGBoost on a big email dataset. Among the models, DistilBERT offered the best trade-off between predictive power and mobile applicability, with good recall and F1 scores while maintaining reasonable latency for real-time inference. The results show that lightweight transformer models can achieve high accuracy and computational efficiency, enabling the deployment of complex models in resource-constrained contexts. Besides predicting performance, the study highlights the importance of explainability in helping users make decisions. The mobile software integrated confidence rankings, risk indicators, and explanatory summaries to help users understand why an email was classed as phishing. This solution mitigates the black-box constraints of existing techniques by converting model predictions into interpretable insights that users can act on, hence increasing trust and knowledge of phishing dangers.

The results show that identifying a practical trade-off between robust detection performance and interpretable outputs is not only an important research goal but also helps prevent overly complex system designs. The research questions of this paper have been answered. Firstly, the results reveal that ML models achieve good performance in phishing detection on an enlarged, balanced dataset. Second, researchers showed that explainable outputs can be combined with a mobile display to assist end users in their decision-making. Third, the work illustrated trade-offs among detection accuracy, real-time performance, and computation cost during mobile deployment, underscoring the choice of DistilBERT for mobile deployment, even though some classic models had slightly higher accuracy. In summary, the work presents a full-blown phishing detection capability, from model training and evaluation, down to mobile deployment with user-reactive explanations. It further verifies that integrating explanationable design into advanced machine learning can enhance detection reliability and user perception. This work lays the groundwork for additional improvements, such as ensemble methods and state-of-the-art explainability techniques,r to keep email Phishing detection systems both accurate and applicable in real-world scenarios.

6.2. Recommendations

Based on the results of this work, several suggestions are presented to improve the quality, interpretability, and real-world applicability of phishing detection solutions. First, the prediction power could be further enriched by using ensemble methods. A multi-model, such as DistilBERT, combined with classical machine learning classifiers like XGBoost or Logistic Regression, could mitigate the weaknesses of a single model and improve robustness even against diverse phishing approaches. Ensembles can help trade off precision and recall, address false negatives and preserve overall accuracy. Second, the system's explainability could be enhanced with methods such as LIME (Local Interpretable Model-agnostic Explanations) or SHAP (Shapley Additive Explanations). Such techniques offer fine-grained, instance-level explanations of model predictions, helping to understand the most important features in making a phishing prediction. Such tools, if integrated, would enhance the existing confidence scores, risk indicators, and reason summaries by allowing moderate drilling without compromising usability. Practical deployment and online updating should also be discussed in future work. Putting the system into a real email could measure its effectiveness against current phishing methods and user responses.

Integrating lifelong learning mechanisms would enable the model to stay up to date by incrementally updating it with new email samples, so it remains effective in the face of new threats. Moreover, user studies could examine how end users interact with confidence scores, risk indicators, and reason summaries, thereby improving interface design and educational impact. Lastly, additional investigation can be considered to more effectively deploy our approach across other platforms, e.g., web clients and desktop applications, so that phishing protection can be provided across multiple environments while maintaining the same level of detection and explainability, targeting local applications rather than cross-platform ones. Integration with cybersecurity could also be possible via corporate email platforms, opening up enterprise security for this very large market. In general, these recommendations aim to complement the existing system by strengthening detection, improving interpretability, and ensuring practical utility in real-world settings. Taking all these actions would not only improve PhishGuard's performance but also help us build strong email habits and combine them to make that leap from a well-built machine learning model to critical thinking by users.

Acknowledgement: The authors sincerely thank Northumbria University for providing academic support and resources for this research. Researchers also appreciate the guidance and encouragement received from the faculty and staff throughout the study.

Data Availability Statement: Data relevant to this study will be provided by the corresponding author upon reasonable request, where permissible.

Funding Statement: This study was conducted without any external financial support.

Conflicts of Interest Statement: The authors report no conflicts of interest, and all sources used in this original work have been duly acknowledged.

Ethics and Consent Statement: The study followed ethical guidelines, with informed consent obtained from all participants.

References

1. A. Aljofey, Q. Jiang, Q. Qu, M. Huang, and J. P. Niyigena, "An effective phishing detection model based on character level convolutional neural network from URL," *Electronics*, vol. 9, no. 9, p. 1514, 2020.
2. Anti-Phishing Working Group (APWG), "Phishing Activity Trends Report, 4th Quarter 2024," *APWG*, 2025. [Accessed by 12/04/2026].
3. Anti-Phishing Working Group (APWG), "Phishing Activity Trends Report, 1st Quarter 2024," *APWG*, 2024. [Accessed by 13/02/2025].
4. G. A. Thomopoulos, D. P. Lyras, and C. A. Fidas, "A systematic review and research challenges on phishing cyberattacks from an electroencephalography and gaze-based perspective," *Personal and Ubiquitous Computing*, vol. 28, no. 3, pp. 449–470, 2024.
5. R. S. Rao, T. Vaishnavi, and A. R. Pais, "CatchPhish: Detection of phishing websites by inspecting URLs," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 5, pp. 813–825, 2020.
6. T. Ige, C. Kiekintveld, A. Piplai, A. Waggler, O. Kolade, and B. H. Matti, "An investigation into the performances of the Current state-of-the-art Naive Bayes, Non-Bayesian and Deep Learning Based Classifier for Phishing Detection: A Survey," *arXiv preprint arXiv:2411.16751*, 2024. [Accessed by 16/02/2025].
7. A. A. Tawil, L. Almazaydeh, D. Qawasmeh, B. Qawasmeh, M. Alshinwan, and K. Elleithy, "Comparative analysis of machine learning algorithms for email phishing detection using TF-IDF, Word2Vec and BERT," *Computers, Materials & Continua*, vol. 81, no. 2, pp. 3395–3412, 2024.
8. Z. Alshingiti, R. Alaqel, J. Al-Muhtadi, Q. E. Ul Haq, K. Saleem, and M. H. Faheem, "A Deep Learning-Based Phishing Detection System Using CNN, LSTM, and LSTM-CNN," *Electronics*, vol. 12, no. 1, p. 232, 2023.
9. S. Atawneh and H. Aljehani, "Phishing Email Detection Model Using Deep Learning," *Electronics*, vol. 12, no. 20, p. 4261, 2023.
10. M. Songailaitė, E. Kankevičiūtė, B. Zhyhun, and J. Mandravickaitė, "BERT-based models for phishing detection," in *proc. 8th Conf. Information Society and University Studies (IVUS'2023)*, Kaunas, Lithuania, 2023.
11. E. M. Damatie, A. Eleyan, and T. Bejaoui, "Real-time email phishing detection using a lightweight DistilBERT model," in *Proc. International Symposium on Networks, Computers and Communications (ISNCC)*, Washington District of Columbia, United States of America, 2024.
12. U. A. Butt, R. Amin, H. Aldabbas, S. Mohan, B. Alouffi, and A. Ahmadian, "Cloud-based email phishing attack using machine and deep learning algorithm," *Complex & Intelligent Systems*, vol. 9, no. 6, pp. 3043–3070, 2023.
13. H. Fares, J. Kilani, F. E. Fagroud, H. Toumi, F. Lakrami, Y. Baddi, and N. Aknin, "Machine learning approach for email phishing detection," *Procedia Computer Science*, vol. 251, no. 12, pp. 746–751, 2024.
14. S. D. Guptta, K. T. Shahriar, H. Alqahtani, D. Alsalman, and I. H. Sarker, "Modeling hybrid feature-based phishing websites detection using machine learning techniques," *Annals of Data Science*, vol. 11, no. 3, pp. 217–242, 2024.
15. S. Chakraborty, "Phishing email detection dataset," *Kaggle*, 2023. [Accessed by 15/02/2025].
16. N. A. Alam, "Phishing email dataset," *Kaggle*, 2024. [Accessed by 16/02/2025].
17. L. Zhou, A. Gaurav, W. Alhalabi, V. Arya, and E. Alharbi, "Integrating AI and semantic web technologies for robust phishing detection in virtual realities," *International Journal on Semantic Web and Information Systems*, vol. 21, no. 1, pp. 1–19, 2025.
18. R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM Computing Surveys*, vol. 51, no. 5, pp. 1–42, 2018.
19. J. Govindaraaj, "The role of explainable AI in understanding phishing susceptibility," *Journal of Recent Trends in Computer Science and Engineering*, vol. 12, no. 1, pp. 1–6, 2024.
20. R. S. Rao, C. Kondaiah, A. R. Pais, and B. Lee, "A hybrid super learner ensemble for phishing detection on mobile devices," *Scientific Reports*, vol. 15, no. 5, p. 16839, 2025.

21. H. B. Pandya and K. Zalawadia, "A comprehensive review of phishing detection techniques based on machine learning," *Metallurgical and Materials Engineering*, vol. 31, no. 5, pp. 302-310, 2025.
22. H. Abed Farhan, "Robust and accurate phishing detection using enhanced DistilBERT: a transformer-based approach," *Journal of Al-Qadisiyah for Computer Science and Mathematics*, vol. 17, no. 3, pp. 230–246, 2025.
23. N. Altwaijry, I. Al-Turaiki, R. Alotaibi, and F Alakeel, "Advancing phishing email detection: a comparative study of deep learning models," *Sensors*, vol. 24, no. 7, p. 2077, 2024.
24. J. Madake, S. Kadam, S. Kadam, and A. Khandelwal, "An experimental analysis of transformer-based models for phishing detection," *in the proceedings of the International Conference on Smart Systems and Social Management (ICSSSM 2025)*, Assam, India, 2025.
25. M. Hosseinzadeh, U. Ali, S. Ali, R. Abbaszadi, F. S. Gharehchopogh, P. Khoshvaght, T. Porntaveetus, and J. Lansky, "Improving phishing email detection performance through deep learning with adaptive optimization," *Scientific Reports*, vol. 15, no. 10, p. 36724, 2025.