



Explainable Detection of AI-Enhanced Phishing Emails and URLs

By

Md Obaydul Haque

ID-2435442

Supervisor: Dr Amit Naik

**Project submitted as part of the requirements for the award of
MSc in Cybersecurity and Digital Forensics**

2025/26

Table of Contents

Chapter 1: Introduction	5
1.1 Research Context and Background	5
1.2 Statement of the Problem	6
1.3 Project Aim and Objectives	7
1.4 Significance of the Study	7
1.5 Structure of the Report	8
Chapter 2: Literature Review	9
2.1 Introduction	9
2.2 Evolution of Phishing and AI-Enhanced Social Engineering	9
2.3 Evolution of Phishing Detection Approaches	11
2.4 Phishing Email Detection: ML, NLP and Transformers	13
2.5 Malicious URL Detection	16
2.6 Explainable AI in Phishing Detection	18
2.7 Datasets, Generalisation and Evaluation Challenges	20
2.8 Critical Synthesis and Research Gap	23
2.9 Chapter Summary	24
Chapter 3: Methodology	25
3.1 Introduction	25
3.2 Research Philosophy and Approach	25

3.3 Research Methodology: Design Science Research	26
3.4 Dataset Selection, Preparation and Preprocessing	27
3.5 Feature Extraction and Data Representation.....	29
3.6 Machine Learning and Transformer Models.....	30
3.7 Explainable AI Framework	32
3.8 Model Evaluation, Testing Strategy and Metrics.....	33
3.9 Ethical Considerations and Methodological Limitations.....	34
3.10 Chapter Summary	35
CHAPTER 4 DESIGN, IMPLEMENTATION AND RESULTS	37
4.1 Introduction.....	37
4.2 Stage 1 Dataset Selection.....	37
4.3 Stage 2 — Working Environment.....	38
4.4 Stage 3 — Baseline Email Models: What Was Run and What It Found	38
4.5 Stage 4 — URL Model: What Was Run and What It Found	40
4.6 Stage 5 — DistilBERT finetuned.....	42
4.7 Stage 6 Evaluation Summary	50
4.8 Stage 7 — Explainability Results	51
4.9 URL Model: SHAP Results.....	51
4.10 Email Model: SHAP Results	53
4.11 Local Explanations: LIME	54

4.12 What the Results Mean, and What They Do Not Yet Show	55
4.13 Summary	55
CHAPTER 5 IMPLEMENTATION	56
5.1 Introduction.....	56
5.2 Implementation Overview	56
5.3 Baseline Email and URL Models.....	58
5.4 DistilBERT fine-tuned (Google Colab)	59
5.5 No GPU or model-hub access in the primary environment	62
5.6 Summary	65
References.....	67

Chapter 1: Introduction

1.1 Research Context and Background

Phishing is a continuous cybersecurity challenge because it is a technical deception and a manipulation of human judgment [1]. Phishing attacks are distinct from attacks that exploit only software vulnerabilities, in that they leverage trust, urgency, impersonation, and contextual plausibility to elicit users' credentials, the transfer of funds, or the execution of malware. This risk is now evolving into a new form with the advent of generative artificial intelligence (GenAI), recent studies reveal. Realistic content generation, advanced targeting and personalisation, and automated attack infrastructure are all potential ways in which GenAI can improve social engineering capability, as outlined. It is notable because the language deficiencies that historically facilitated user detection of suspicious messages can now be more easily overcome by large language models (LLMs), which can generate at scale fluent, credible, and contextually appropriate content.

The empirical evidence of recent times is evidence of the practical importance of this change. A controlled human subject experiment concluded that automated spear-phishing was just as effective in getting people to click through as human-generated spear-phishing [2]. AI-generated phishing messages would be able to evade existing email filtering services, and discriminative signals could be recovered from phishing messages using stylometric machine-learning analysis [3]. The results indicate that AI-powered phishing can be more than just an increased volume of classic phishing attacks; it can also shift linguistic and statistical patterns, which traditional phishing detection methods rely on.

The present project can be considered a continuation of previous research on explainable detection of AI-enhanced phishing emails and URLs, which falls within the interdisciplinary areas of cybersecurity, machine learning, natural language processing, and explainable artificial intelligence (XAI). The significance of this development is illustrated by recent empirical evidence. This finding is in line with the overall literature. The systematic review conducted found that phishing detection has seen a shift from blacklist and heuristic approaches to machine learning and deep learning approaches and that there is a significant reliance on specific data sets and engineered features [4]. Recent email studies have yielded high performance of both conventional and deep-learning classifiers, and practical studies have revealed that XAI can be applied in

phishing email detection without compromising the predictive performance [6]. Meanwhile, in parallel, URL-centric studies have shown that lexical, structural, and neural representations are important evidence to distinguish phishing from legitimate destinations [7], [8].

1.2 Statement of the Problem

These advances do not imply that phishing detection is a "solved" classification problem. One simple problem is that good performance on a benchmark dataset does not imply robustness on a shifted attack dataset. The fact that classifiers give different results on different phishing email datasets also indicates that testing classifiers on several datasets is required before thinking about their generalisability based on a single dataset [5]. It has been demonstrated that traditional email filtering can be difficult with AI-generated phishing [3]. A detector trained primarily on historical phishing can thus be taught some lexical or structural regularities that, when GenAI is used to intentionally alter the grammar, tone, semantic coherence, and personalisation, can become less effective.

Inter-modality fragmentation is the second issue highlighted. Much of the existing literature focuses on either the content of e-mails or the URLs. According to [7], neural networks achieve good results in the classification of phishing URLs using neural networks and give proof of translation from experimental modelling to production deployment, while Sahingoz, Buber and Kugu [8] present a deep-learning-based phishing detection architecture that is heavily based on characteristics of the web and URLs [8]. Research on ensemble learning and encrypted traffic also shows good results for URL-based classification [9], [10]. On the other hand, research on emails highlights language, semantic representation, and message classification [3], [5], [11]. While each of these methods is worthwhile, many phishing messages will include a combination of these techniques, such as a persuasive tone that plays on the message's words while also containing links to malware. Thus, a comparative evaluation of these complementary types of evidence within a standard experimental framework can yield a fuller evaluation of the strengths and weaknesses of the detectors.

The third challenge is interpretability. Predictive accuracy alone is not a good indicator of whether a classifier has learned security-relevant indicators or simply some correlations across the datasets. Al-Subaiey et al. [6] incorporate XAI into a phishing-email platform, particularly to enhance transparency and user trust. Likewise, Shafin [12] shows that SHAP and aggregated LIME

explanations can identify key phishing features and support feature-selection decisions. Explainability should therefore not only be understood as an improvement of visual interfaces, but also as an analytical tool to analyse the foundations of model behaviour. The core research problem is thus the requirement for a framework which can comparatively assess AI-augmented phishing emails and malicious URLs, generate evidence that is interpretable, and determine if there is sufficient predictive performance to support realistic cybersecurity applications.

1.3 Project Aim and Objectives

This research aims to design and test a machine learning framework that can explain how it detects AI-enhanced phishing emails and URLs.

1. To review recent studies on phishing detection, AI-driven social engineering, transformer-based text classification, and explainable AI in cybersecurity.
2. To compare how well baseline machine learning models and transformer-based models classify phishing emails.
3. To evaluate machine-learning models for phishing URL classification and examine their performance using lexical, structural and host-based features.
4. To use explainability tools like SHAP and LIME to find out which features and content patterns influence the model's predictions.
5. To evaluate the framework using metrics such as accuracy, precision, recall, F1-score, ROC AUC, and confusion matrices, and discuss how useful it is for real-world cybersecurity.

1.4 Significance of the Study

This study is important in scientific, technical, and operational aspects. From a scientific perspective, it deals with the generalisation issue exacerbated by the quick pace of phishing evolution with the aid of AI. The use of deep learning, ensemble learning, and semantic representations in contemporary literature shows that they can be used to give highly effective classification of phishing [5], [8], [11]. But benchmark accuracy is not a reflection of robustness on new and differently distributed attack samples. The proposed work thus places comparative evaluation, generalisation, and error behaviour on an equal footing with headline classification performance.

In principle, explainability adds another layer of detector quality evaluation. Feature-level explanations can be used to determine if classifications are indeed related to identifiable phishing attributes, such as suspicious URL structures and deceptive linguistic attributes, or instead are coincidental from a specific data source [6], [12]. This renders XAI useful not just for transparency to the user, but also for model validation and debugging.

As for the operation, phishing detection is a case of asymmetric consequences. A false negative could allow credential theft or redirection to malicious sites or malware delivery, while false positives can add to the security-analyst workload and create problems with legitimate communications. A "useful" detector should therefore seek to attend to both recall and precision, as well as generalisability and interpretability, rather than simply seek to optimise a single accuracy measure. The value of this work is not to end phishing, but to create and empirically evaluate a transparent framework for identifying defensible modelling approaches in a world where phishing content is growing increasingly AI-driven.

1.5 Structure of the Report

- Next is Chapter 2, which will be the literature review on phishing, AI-driven social engineering, machine learning, deep learning, deep-learning and transformer-based phishing detection techniques, malicious URL analysis, limitations of the available datasets, and explainable AI.

Chapter 2: Literature Review

2.1 Introduction

As phishing techniques become more adaptive and technologically advanced, the methods and tools used to detect phishing have also changed significantly, shifting from rule-based approaches and blacklists to machine learning, deep learning and, today, more and more sophisticated transformer-based approaches [13]. Meanwhile, AI is increasingly perngin content to exploit offensive cybersecurity and also plays a part in making increasingly robust automated detection tools [14]. In the literature review, the literature on AI-enhanced social engineering is critically explored, and the evolution of phishing detection methods, classification of phishing emails, detection of malicious URLs, explainable artificial intelligence, dataset quality, generalisation and model evaluation are explored. Special focus is given to the use of TF-IDF-based Logistic Regression, Linear SVM and tree-based models as conventional models against transformer models, particularly DistilBERT. The review also explores how well high benchmark performance holds up in the presence of class imbalance, dataset bias, and distribution shift, as well as AI-generated phishing samples. Finally, strengths and limitations of SHAP and LIME are discussed with the accuracy, precision, recall, F1, ROC-AUC and confusion matrices. In subsequent chapters, this synthesis provides the justification for the methodology and model(s) selected and for the experimental evaluations.

2.2 Evolution of Phishing and AI-Enhanced Social Engineering

Phishing is a socio-technical scam: a plan to trick people out of using technical delivery methods and attempt to exploit the judgement, trust and urgency of the common person. Althobaiti and Alsufyani's (2024) systematic review of 163 studies revealed that research has focused mainly on phishing detection and user awareness, with less being done about phishing protection (mitigation)

and other defensive layers [15]. This is important because, despite a significant number of countermeasures, phishing is still a significant route into organisations. In a recent analysis of the Threat Landscape 2025, the ENISA team identified phishing - covering vishing, malspam and malvertising - as the most common initial intrusion vector in about 60% of the cases analysed [16]. This longevity is, therefore, a sign not only of the inadequacies of filtering technologies, but of the effectiveness of deception on both the human and technical attack surface going forward.

Generative AI exacerbates this issue by enabling a new way to create deceptive content, target it and scale it. According to Schmitt and Flechais (2024), there are three main ways in which GenAI can help to enhance social engineering attacks: realistic content creation, advanced targeting and personalisation and automated attack infrastructure [1]. They are relevant for phishing emails because large language models are adept at creating coherent and grammatically correct emails and can even craft the message to align with specific recipients or organisations. Therefore, grammar mistakes, poor sentence construction, and vague sentences, which in the past were good indicators of bad intent, are now not as reliable. Additionally, automation cuts down on the time needed to produce and send customised texts, making it possible for personalised communication on a larger scale. But Schmitt and Flechais do not give a controlled assessment of the performance of the various detectors, nor do they provide a comprehensive comparison of the various classifiers, and they therefore do not prove that the threat model is evolving; instead, they just provide a systematic and conceptual development.

This evaluation is in line with the UK National Cyber Security Centre (NCSC) (2024), which determined that the use of AI is likely to lead to more cyberattacks and provide a capability uplift in reconnaissance and social engineering [17]. The NCSC finds that the use of GenAI to create realistic lures with fewer spelling and grammatical errors can be particularly beneficial for less

experienced phishing attackers, and spell-checking and grammar systems can be fooled by GenAI. However, the conclusions are intelligence-based rather than from a controlled experiment, and are more evidence that operational risk has been detected than that the model has failed to detect it.

Before the rampant adoption of generative AI, the writing on the defensive side of the equation had already started targeting AI-based detection. Basit et al. (2020) surveyed the techniques to detect phishing websites and messages such as machine-learning, deep-learning, hybrid and scenario-based [18]. This previous work provides some indication that detection is changing from passive to learned patterns, albeit before the current ability to produce low-cost, highly fluent and personalised phishing text. The shift in the challenge is thus to understand if learned representations stay faithful if the content they are challenged by is more linguistically natural and is not representative of the distribution of historical training data.

2.3 Evolution of Phishing Detection Approaches

The traditional reactive blacklist/rule-based methods have been transitioning to data-driven machine-learning and deep-learning systems for phishing detection. While blacklists are still efficient for locating known malicious domains and/or URLs, their reliance on known threats means that they are unable to resolve all phishing attacks, especially those launched by new or changing phishing infrastructure. Likewise, heuristic systems categorise suspicious artefacts based on a pre-defined set of rules, such as suspicious URL structure or properties of web pages, but if the rules are predictable, the attackers can also evolve the rules. In a systematic review of 80 phishing-website detection studies, Safi and Singh (2023) identified five approaches to phishing-website detection: list-based, visual-similarity, heuristic, machine-learning, and deep-learning and determined that the machine-learning category was the most common of all of the studies they examined [4]. Recent work on the subject also finds that while traditional blacklists and heuristic-

based methods need to catch the diverse methods associated with phishing, they have limitations in both dealing with its continuously evolving nature as well as in their ability to be adaptive [19].

In order to overcome some of these drawbacks, machine learning was developed, which allows classifiers to learn from labelled observations which embody discriminatory patterns, and not only follow fixed rules. Lexical, structural, textual and behavioural characteristics can thus be used for representing phishing and algorithms like Logistic Regression (LR), Support Vector Machines (SVMs), Decision Trees, and Random Forests can be used to classify phishing. The renewed relevance of the feature-engineered learning approach is shown by the work of Jayaprakash et al., who achieved a URL phishing detection accuracy of 97.2% on 56 URL features, an email phishing detection accuracy of 97.4% on more than 5,000 emails, and a website phishing detection accuracy of 98.1% on the main datasets they tested [20]. However, accuracy within the dataset and between datasets is not necessarily robust. Modelling features may reflect correlations that are more specific to certain datasets, time periods/attack generation mechanisms, and not persistent across phishing. The systematic review of 2017-2024 by Wilk-Jakubowski et al. [21] confirms that the performance reported is strongly dependent on representation and experimental design, a trend that is further illustrated by the increasing diversity in feature engineering, ensemble methods and neural architectures mentioned in the review.

Conventional ML nevertheless provides an important experimental baseline. TF-IDF transforms textual content to sparse weighted representations, enabling LR to efficiently serve as a linear benchmark, and enabling Linear SVM to take advantage of high-dimensional separation between phishing and legitimate text. Tree-based approaches are complementary models, as they can capture nonlinear interactions between the engineered features. New advances in phishing research look at the draft of performing a Random Forest-based, XGBoost-based and CatBoost-based

approach in addition to hybrid site-detection frameworks, such as hybrid web site-detection designs [22]. This complexity can, however, only be sought after if it brings about tangible improvements in robustness and/or generalisation nonsense can't achieve any of those things.

With the evolution of deep learning, we moved to automatic feature specification, and representation learning began to get increasingly more complex via data signals. Catal et al. (2022) studied 43 deep-learning-related papers and identified that the most popular architectures were DNNs, CNNs, and RNN/LSTMs, and interestingly, 72% of the reviewed papers have not applied feature selection; in addition, the most commonly used source of data was PhishTank [23]. There are then benefits to be gained from deep and hybrid architectures in terms of representational capacity, but with an added cost in terms of the high level of computation required and the lack of transparency. Van Geest et al. (2024) also suggest that contaminants can still be introduced by bypass strategies with single analysis detectors and argue that hybrid detection could be a way to make these detectors more real-world robust, though the practical applicability of increased effectiveness is not enough to make it a viable solution [24].

2.4 Phishing Email Detection: ML, NLP and Transformers

The representation of the textual information prior to the classification process becomes more and more critical when it comes to detecting phishing emails. Most natural language processing (NLP) techniques used in emails convert the text of the email into numerical features based on some lexical or statistical properties. For example, TF-IDF assigns greater weight to the terms that are unique to this item, and it can be used to reveal deceptive terms, such as urgency or credential requests within an email corpus. These representations are efficient in computation and can be used by classifiers like Logistic Regression, Support Vector Machines and tree-based classifiers. Their biggest drawback is that they are generally used as a measure of term frequency rather than

meaning. Two messages can be very similar in lexical features but be very different in their intentions, and an attacker can reword a malicious message without capturing the lexical feature indicators from the previous messages. There is also the additional representation layer of stylometric characteristics, such as sentence structure, punctuation patterns, the use of pronouns and writing complexity, but these can also become unstable as the generated phishing evolves in style depending on the LLM model, prompt and target.

Given these constraints, traditional machine-learning approaches are still relevant and not a null expectation. Abdulla Al-Subaiey et al. (2024) merged six public email corpora into a total of 82,486 emails, with 42,891 of them spam/phishing related and 39,595 legitimate emails, and tested SVM, Multinomial Naive Bayes and Random Forest with TF-IDF and Word2Vec representations [5]. They obtained an overall F1-score around 0.99, indicating that proper design of conventional representations can still provide an excellent classification performance with fewer computations and simpler explanations compared to their complex counterparts [5]. While the study is important for this research because it highlights the usefulness of TF-IDF-based classification, combining historical corpora like Enron, Spam Assassin and other collections will not prove to be robust enough to withstand current AI-generated phishing attacks. The key, therefore, is that, with strong performance, this should be seen as an example of effective discrimination within the constructed data environment, rather than an indication of general resistance to future attacks.

Transformer models solve some problems with sparse lexical representations: learning representations for tokens in context. Unlike traditional frequency-based word evaluations, BERT-like models learn word relationships based on their context. This is especially the case with phishing using artificial intelligence (AI), as it can increasingly be done with grammatically correct and professionally formulated messages, with the meaning of the message being inferred from the

relationship between apparently factual messages. A particularly useful (controlled) comparison is made by Meléndez, Ptaszynski and Masui (2024), who have gathered 119,148 English-language emails from various public sources. SVM outperforms other conventional classifiers with an accuracy of 98.76%, while Logistic Regression, Random Forest and Naive Bayes recorded 98.45%, 98.02%, and 96.44% accuracy, respectively. Transformer models performed slightly better: accuracy of 99.43%, 99.11% and 98.99% for RoBERTa, BERT and DistilBERT, respectively [25]. Perhaps most importantly, the numerical benefit from SVM was less than one percentage point for DistilBERT, a measure indicating that increases in model complexity must be evaluated in accordance with the magnitude and practical significance of the benefits, not by the simple fact that they are delivered by a transformer. The authors also uncovered errors related to content in a mixed language, emails that are formal but contain errors, and phishing URLs, even though it had an aggregate accuracy of very high.

DistilBERT is particularly relevant to the approved experimental design, which aims to find a balance between contextual modelling and computation. In a comparative experiment, it achieved a 98.99% accuracy, which is near to BERT and RoBERTa, and also its speed is roughly 60% faster than the standard BERT, the authors write. This makes it defensible that it would be an MSc experimental framework whose purpose would not be to build the largest possible architecture, but to see if contextual representation improves the conventional baseline (which is transparent). This corresponds with more comprehensive multi-dataset evidence. Alhuzali et al (2025) tested 14 machine learning and deep learning models on 10 datasets: nine public corpora and a combined dataset. The accuracy of the deep-learning models was higher than that of conventional ML models by an average of 4.7 percentage points, with RoBERTa and BERT reaching accuracies of 99.08% and 98.99% on the balanced merged dataset, respectively [8]. The external validation results,

however, yielded worse results than the near-perfect internal validation results, as stated: 95.0% in the case of RoBERTa, when trained with the merged, balanced dataset. The difference is methodologically significant because it shows that accuracy on the benchmarks is not robust across different distributions.

Higher architectural complexity also allows for a performance/complexity trade-off. To capture contextual embeddings captured by BERT, combined with CNN feature extraction, GRU sequence modelling, multi-head attention and Mountain Gazelle Optimisation, Hosseinzadeh et al. (2025) proposed a combination of these. With more than 18,000 emails, the proposed model was able to reach an accuracy of 96.8%, precision of 97.2%, recall of 95.4% and an F1 score of 96.3% [26]. The results show that the hybrid architecture is able to learn more than one textual dependency, but the model is much more complicated than a fine-tuned simple transformer model or traditional classifiers. It also makes it evident that despite being less accurate on the headline level than some simpler benchmark studies, the results can not be compared directly due to differences in dataset composition, preprocessing strategy and validation.

2.5 Malicious URL Detection

Malicious URLs have a different evidence source from the message text of a phishing email since they can be classified based mostly on properties of the address and webpage hosted at it, not on typical natural language. Detection features can be grouped into three categories: lexical indicators (URL length, character type of path, digits, presence of special characters, and suspicious and other tokens), structural indicators (subdomain depth, structure and organisation of the host name of the path), and content/external indicators (DOM characteristics, HTTPS configuration, reputation of the domain, threat-intelligence results). While reputation and content signals may provide better

discrimination, they may require additional processing or third-party services; lexical features can be extracted without visiting a webpage and are thus appealing for low-latency detection.

Generally, more and more recent systems combine a number of types of URL evidence. Later, Arshad et al. (2026) suggested a browser-based framework which combines a Random Forest Classifier that relies on lexical URL features, live DOM analysis and VirusTotal reputation intelligence [27]. The Random Forest showed 96.7% accuracy, 97.1% precision, 96.0% recall, and a 96.6% F1 score, while the full hybrid framework reportedly achieved 91.7% accuracy in the identification of zero-day phishing URLs with average end-to-end latency under one second. The results show the usefulness of using lightweight lexical features together with structural and reputation features. The reported performance, however, is representative of a multi-component architecture, not just a URL-based classifier, and should be considered separately for each evidence source. External reputation services may also have an impact on portability and detection if they rely on external reputation services and previously seen URLs are missing.

Research in deep learning also holds that designed URL and webpage features can aid in highly accurate classification. Elshewey et al. gave a comparative analysis of CNN, VGG19, GRU, RNN, ResNet-50, and a hybrid approach of CNN-VGG19 using 10,000 well-balanced webpage samples, which include 5,000 phishing pages and 5,000 legitimate pages derived from 48 handcrafted URL and content-based features [28]. After applying feature selection and Bayesian optimisation, their hybrid model attained an accuracy of 99.0%, a precision of 0.9901, a recall of 0.9900 and an F1-score of 0.9900. However, the samples were collected in 2015 and 2017, even though the study was published afterwards. The near-perfect result illustrates the discrimination in that benchmark but does not provide as much evidence of resistance to current URL obfuscation and distributional change. Additionally, the feature selection procedure, optimisation and hybrid architecture are

more complex, which raises the question of whether similar results can be achieved with simpler & more repeatable URL models.

The generalisation problem is especially significant in the case of classifiers from the URL-recognizing data that comes from different sources. Rashid et al. (2024) tested four URL detection approaches for three data sets and observed that there could be a performance drop of about 10-32% for a test set different from the training set [29]. They found distributional changes in terms of URL length, for example, and specific terms were among those factors that contributed to this decline. An unsupervised domain-adaptation approach was then employed to further boost the cross-dataset F1 score by 0.06 on average and by around 0.20 in some cases. This leaves a core conceptual challenge with benchmark evaluation, which is that sometimes high performance after the split of a single data set is meant to be compatible with the data set; this does not necessarily reflect a strong recognition of the transferable phishing characteristics.

2.6 Explainable AI in Phishing Detection

Explainable AI (XAI) plays a crucial role in phishing detection not only because its predictive accuracy is not sufficient to guarantee that a classifier employs legitimate security indicators, but also because the core of the problem is not inherently interpretable. Well-performing models can take advantage of correlations created during the construction of the data set, artefacts or dubious characteristics that arise only in specific sources, and even change over time; analysts must not only be able to recognise if an email or URL is malicious, but also the reasons. In cybersecurity, explainability serves two purposes: it can aid in reviewing specific alerts, and it can be used during auditing or determining whether the model behaves in what is technically feasible. More general cybersecurity studies also include transparency and interpretability as key demands when making a security decision from a complex machine-learning system [30]. Hence, the notion of

explainability in this project is not looked at as an afterthought to apply after classification, but as proof to check what the models learnt.

LIME offers a model-independent explanation for local explanations. It makes predictions for all instances it perturbs around the observation it is currently working on, and fits a simpler model designed to approximate the classification function in that small neighbourhood around the observation. This can be used to tell which words, tokens or URL features had an impact on one prediction in phishing detection. The primary benefit is its "instinctive" readability and its usability when the analyst must delve deeper into a particular message or URL that was detected. But explanations may be dependent on the perturbation process, neighbourhood definition and surrogate approximation, and local fidelity does not imply global fidelity. In this modern systematic study of XAI, LIME is thus considered helpful, but on the other hand, concerns are raised about the reliability of explanations and the lack of availability of large data sets and real-world applications [31].

An alternative approach to feature attribution is SHAP. It proposes to approximate the contribution of each feature to the prediction in the context of Shapley values. SHAP can describe a single instance through the contributions of local features, and aggregation of the observations can show the global patterns of model behaviour. Such information can be useful in the proposed framework, because one could use local explanations to determine the motivation behind a certain e-mail or URL, while the global ones might allow one to determine if the classifier has a consistent strategy that relies on plausible phishing indicators. Combining boosting with SHAP, they have used a public dataset of 11,054 phishing websites with 32 interpretable features to obtain an accuracy result of 97.78% for XGBoost, and with SHAP are able to generate waterfall explanations for local features and global feature importance [32]. But not all attributions are created equal, and sufficient

predictive performance should not be passed off for validation. SHAP explanations are still dependent on the fitted model, feature representation and background assumptions, and influential features do not necessarily give clues to phishing.

Additional evidence for phishing, thus further highlighting the need for explainability as part of the diagnostic process, not just display, is found across phishing-specific evidence. Calzarossa, Giudici and Zieni (2023) created an Explainable Feature-Selection based approach on phishing and PhishTank/ Tranco legitimate URLs. They eventually ended up with six meaningful variables, extracted from the 74 features they had, which had an AUC of ~ 0.96 [33] after a procedure that reduced the number of features. This implies that feature analysis might be able to identify parsimonious decision structures, without compromising discrimination. The study, however, offers primarily global-level feature insights and does not confirm other features to be temporally and cross-dataset stable.

SHAP and LIME should thus not be considered as rival techniques with one being universally better than the other. While LIME is best suited for single predictions, leveraging local surrogate explanations, SHAP also enables the analysis of all predictions as a whole, or some level of global analysis. They can complement one another to account for model behaviour more richly, but on their own provide no evidence of causation or of trustworthiness.

2.7 Datasets, Generalisation and Evaluation Challenges

Phishing-detection research is not only critically dependent on the model architecture, but also critically dependent on the quality, provenance and representativeness of the data used for training and evaluation. Apparently identical corpora might represent different distributions because public phishing datasets are gathered on varying time scales, in varying repositories and in a varying

organisational context. Potpelwar, Kulkarni and Waghmare (2025) proposed a new dataset named LegitPhish containing 101,219 labelled URLs where 63,678 URLs were found to be phishing URLs, and 37,540 were legitimate, with lexical and structural features like URL length, entropy, token count and subdomain use [34]. In an intriguing incident, however, the reported number of classes, 101,218, equates to 101,219 because instances of dataset documentation and provenance should be examined rather than accepted.

Being able to compose a dataset can also make it somewhat biased and over-performing. A classifier is able to learn such source-specific phishing information if phishing and legitimate samples come from repository systems which are different regarding their artefacts. The same occurs if one or more than two datasets come from similar observations: data leakage and overly positive results occur from similar observations appearing in multiple datasets. Wherever possible, duplicate removal and proper data preprocessing should be done prior to splitting data. But there is a concern with synthetic AI-generated phishing: while it can enhance dataset recency and diversity, classifiers might be trained to detect certain features of the specific phishing generator and prompting method, instead of learning general patterns of phishing.

Compounding the problem of evaluating the algorithm is the class imbalance – the class sizes can vary significantly between the experimental and operational settings. A model that performs well on a balanced dataset can seem like the ideal choice, but fail to exhibit a performance level when normal traffic volumes are significantly larger than malicious traffic volumes. Using SMOTETomek with XGBoost, Omari and Oukhatar (2025) reached an accuracy of 97.8%, F1 score of 98.1%, recall rate of 98.2%, precision rate of 97.9% and ROC AUC of 99.8% [35]. While resampling is important for operationally getting more phishing to be recalled, the distribution of

the training samples is different to the real distribution when deployed, and therefore the prevalence of phishing cannot be reproduced.

Performance in the generalisation task needs to be separated from within-dataset performance. Concept drift is defined as changes in the relationship between the observed characteristics and class labels, while distribution shift is a change in the distribution of the data, but not necessarily in the relationship between characteristics and labels. Both play a crucial role in phishing, as phishing hackers are constantly changing their domains, wording, obfuscation and social-engineering methods. Random train-test splits can show in-distribution discrimination but can be insufficient to show resilience to shifts across different time periods, different sources, or AI-generated shifts. The difference might be significant when comparing simpler baselines to more complex models such as transformer models and ensembles on collected data.

The above problems are the reason for needing more than accuracy. For precision, the number of cases predicted as phishing that are actually phishing represents the amount of false alerts (false alarm burden). Recall is the rate of true positives (TP) discovered and is particularly relevant since false negatives will result in malicious content going undetected. F1-score is a balance of precision and recall, and ROC-AUC is used to evaluate class discrimination at thresholds. True positives, true negatives, false positives and false negatives are all evident in confusion matrices.

False negatives can allow for credential theft, malicious redirection or delivery of malware, while false positives can prevent legitimate traffic from passing through and lead to higher analyst workload. The most accurate model is not necessarily the most defensible; rather, a model achieving good detection performance, non-intrusive false alarm behaviour and robustness is. As a result, the preparation of the datasets, tackling any data leakage, class distribution and multi-metric assessment are central to model validity and directly contribute to Objectives 2-3-5.

2.8 Critical Synthesis and Research Gap

From the literature reviewed, it can be seen that current phishing detection is influenced not only by classification accuracy, but also by other factors such as model capacity, phishing data quality, robustness and interpretability of the models. Generative AI reinforces social-engineering approaches in terms of language proficiency, personalisation and automation to mitigate some of the limitations of phishing in terms of grammatical and stylistic flaws traditionally observed. It is also important to note that conventional machine-learning methods are still relevant, as they can provide efficient 'baselines' such as TF-IDF, Logistic Regression, Linear SVM, as well as tree-based models, which are comparatively more interpretable. The benefit of transformer architectures is that they provide greater contextual representations and sometimes higher phishing-email classification accuracy, but the extent to which they outperform is dependent on the datasets, model architecture and testing environment [25].

One of the main limitations in the evidence is that it fails to bridge the gap between the benchmark and the actual generalisation to the real world. While datasets with high accuracy within the dataset are generally found, when tested across different datasets, phishing detectors have been found to suffer when presented with a distribution different from that of the training set [29]. Performance may also be further skewed by other artefacts such as historical data, duplicate observations, imbalanced classes or artefacts introduced by the data sources themselves or generated by AI. The different dimensions of phishing are also captured in email and URL evidence: Emails contain language and social engineering aspects, while URLs contain the lexical, structural and infrastructure aspects.

XAI methods can provide information on which features are influential in a model and can be applied to aid auditing of a model, but cannot provide causal information or evidence of

trustworthiness and should not be used for this purpose [33]. The challenge, however, is not whether the evidence exists or not, but rather the lack of cohesive evidence across modalities, datasets, model families, robustness and explanation strategies, meaning evidence is scattered. This project fills this gap with a structured approach, which leverages traditional baselines, transformer-based detection, complementary evidence provided by email and URL, AI-powered phishing analysis, multiple evaluation metrics, and explainability. This design allows a comprehensive evaluation of the three types of performance mentioned above: predictive performance, error behaviour, and generalisation and model explanations in a unified experimental context.

2.9 Chapter Summary

The literature surveyed in this chapter critically examined phishing detection studies related to explainable detection of AI-enhanced phishing emails and URLs. The research shows that generative AI renders phishing easier to make realistic, personalisable and scalable, whilst simultaneously reducing linguistic characteristics typically found in malicious emails. So, the process of phishing detection has shifted from blacklists and heuristic rules to ML, deep learning and transformer-based detection. However, the TFIDF-based Logistic Regression, Linear SVM and tree classifiers are important experimental baselines. It was also found that email and URL detection, and DistilBERT, offer an acceptable compromise between the two. The significant challenges to good dataset validity were identified as duplicate datasets, concept drift, cross-dataset generalisation, as well as class imbalance. Thus, one should not consider only accuracy, but also the accuracy and precision, recall, F1-score, ROC-AUC and confusion matrices when evaluating performance. Finally, SHAP and LIME give complementary explanations of model behaviour to guide the methodological and experimental decisions made in the following chapter.

Chapter 3: Methodology

3.1 Introduction

The methodology used in the design, development and evaluation of an explainable machine-learning system to detect AI-enhanced phishing emails and URLs is analysed in this chapter. The study uses Design Science Research (DSR) and a quantitative experimental comparative design for the development of the artefact and the model comparison and explainability analysis. The implemented system compares traditional machine learning methods, the DistilBERT transformer and employs SHAP and LIME to analyse and interpret model decisions.

This chapter discusses the research philosophy, methodology, dataset preparation, feature representation, model selection, testing strategy and evaluation metrics. Reproducibility, reliability, transparency and practical cybersecurity effectiveness are emphasised, in accordance with the research objectives and implementation procedures.

3.2 Research Philosophy and Approach

The study is focused on the phishing-detection methods, so it uses a positivism approach in terms of measurable empirical evidence and controlled experimentation. The positivist approach is contrary to qualitative assessment because phishing emails and URLs have measurable features that can make the approach of quantitative assessment possible, and classification performance can be objectively evaluated using a set of numeric measurement metrics in a controlled experimental setting.

The current research indicates that traditional machine-learning methods could be able to accurately recognise phishing with engineered features, transformer-based models could be able to enhance the contextual understanding, and XAI methods could be able to enhance transparency.

These assumptions are tested using comparative model testing, not through new theory evolution based on the observations. Therefore, the groups of positivism and deduction are consistent with the objective of this research, Design Science Research methodology and quantitative experimental evaluation, which helps in objective appraisal of the phishing-detection artefact developed in this research.

3.3 Research Methodology: Design Science Research

The Design Science Research (DSR) methodology is used in this study as the project is mainly about designing, developing and evaluating a practical cybersecurity artefact: An explainable machine-learning system to detect AI-enhanced phishing emails and URLs.

The developed artefact includes a framework for phishing detection that connects the preparation of phishing datasets, the machine-learning classification, the transformer-based modelling and the EAI techniques. The methodology is based on the basic steps of DSR: first, identify the problem in cybersecurity, then design a solution, and finally test it in a controlled, experimental environment to verify its effectiveness.

In this project, DSR offers a general methodological framework, and quantitative experimental evaluation is applied to check the developed system. In the field of cyber security, more complex methods, like multi-task learning, have been investigated, but there are no systematic reviews of complexity issues, datasets, and evaluation methods [36]. The artefact is compared with traditional machine-learning models (Logistic Regression, Linear SVM and tree-based models) and a transformer-based DistilBERT model. In addition, SHAP and LIME are added for transparency by analysing the factors behind model predictions [37].

This research adds quantitative experimentation to DSR, ensuring alignment with the approved objectives: Preparing phishing datasets, comparing detection approaches, applying explainability techniques and evaluating model effectiveness. This offers a logical structuring of the methodological approach for the development of an explainable phishing-detection system for an MSc Cybersecurity and Digital Forensics project.

3.4 Dataset Selection, Preparation and Preprocessing

In recent years, numerous phishing email datasets have been released, mostly for the purpose of carrying out this research; one of these being LegitPhish and Nazario Phishing Email Corpus, which have been assembled by the authors according to the approved aims of this research. The Nazario corpus contains samples of phishing emails for text classification, and LegitPhish contains phishing and legitimate URLs for analysing the lexical and structural characteristics of phishing websites. The chosen datasets will have examples which are clearly labelled to be used for supervised binary classification.

In the case of email detection, datasets will include the textual content for natural language processing and classification. For URL detection, malicious and benign Internet addresses will be given in datasets for the extraction of lexical and structural properties. Recent URL-detection work also underscores the need for attention to evolving URL features, as phishing indicators can also change over time [38]. This distinction is relevant because, in the case of a phishing e-mail, linguistic and contextual clues are the most significant indicators of fraud, while phishing URLs provide structural and signal properties related to their domain. The significance of using a combination of features and classification approaches has been shown in previous heuristic machine-learning studies in both email and web phishing environments [20].

Controlled preprocessing of datasets will be carried out to make sure the data is consistent and to remove potential sources of bias before it is used for model training. In the first step, email preprocessing will be done by removing unnecessary formatting, cleaning HTML, normalising and tokenising text prior to the process of feature extraction. URL preprocessing will involve the normalisation of the URL, identifying duplications and extracting meaningful structural elements from the URL. These procedures are required because if two or more samples are similar, or if the same samples are highly similar in terms of their performance information, they can cause data leakage and artificially boost the results during evaluation.

Challenges of class imbalance, variation in phishing behaviour and bias in datasets will also be discussed in the context of dataset preparation. Another recent study on phishing-email classification explores ways of dealing with imbalanced data by using sophisticated learning strategies and thus reiterates the challenge of maintaining reliable classification despite imbalanced data distributions [39]. While having a balanced set of data is good for comparative analysis of the models, it may not accurately reflect a real-world cybersecurity environment where potentially not all communications may be malicious but instead could be legitimate communications at significantly higher rates than phishing attempts. Several techniques have been tried to enhance the detection performance of the minority classes, but resampling techniques need to be used with care, since they modify the original distribution of the data [35].

Thus, the idea of the dataset strategy is to provide transparency, reproducibility and methodological control. Preparation of phishing email and URL datasets can be done with care to give a reliable basis for designing, comparing and evaluating the explainable machine-learning detection framework.

3.5 Feature Extraction and Data Representation

Based on the type of each data type, this study utilises various representation techniques and compares interpretable feature-based representations to contextual transformer-based modelling. This work uses Term Frequency–Inverse Document Frequency (TF-IDF) for text words to convert the text in email text into feature vectors of numerical values to classify phishing emails. TF-IDF is chosen since it gives the pattern of the text in an efficient manner and is usable with conventional classifiers like Logistic Regression and Linear SVM. It also gives some degree of interpretability, as influential words and terms that lead to classification decisions can be analysed. TF-IDF will, however, more or less reflect a statistical relationship between terms rather than a more profound meaning within the context. So, it may be less effective on phishing emails that vary the wording and message in an AI-created email in a more advanced way, such as paraphrasing or socially convincing emails.

In the malicious URL detection problem, this study uses lexical and structural feature engineering to encode the features of malicious websites. The URL features relevant for this include length, terms suspicious in the URL, patterns of characters occurring in the URL, domain-related information, subdomain properties and properties of the structure. These features are chosen as a result of the fact that URLs have measurable features which can differentiate malicious and non-malicious domains. Experience: Feature-based URL representations also facilitate explainability, as model decisions can be associated with URL features that can be identified. However, their effectiveness can be compromised if attackers use obfuscation, shortened URLs or website strategies, which change constantly.

In addition to the traditional representation, in this work, DistilBERT contextual representation is used for the analysis of phishing e-mails. Transformer-based models are good at understanding

relationships between words and their context, which allows them to detect AI-generated phishing messages, as the malicious messages can be composed in a professional and fluent manner. DistilBERT is chosen as a model due to its contextual modelling ability and reduced computational feasibility than bigger transformer-based models. The potential of transformer-based phishing detection has been proven in previous studies in conjunction with explainability techniques to interpret model behaviour [40]. Using TF-IDF, URL feature engineering, and DistilBERT, a controlled comparison is made between classical interpretable methods and contextual modelling. In this way, the representations directly help the research goals: comparative model performance evaluation and analysis of explainable phishing-detection performance.

3.6 Machine Learning and Transformer Models

The number of machine-learning models is selected as per the objective to investigate the different approaches to phishing detection using methods of explainable AI and not to find a model out of benchmark scores. The authors here compare a traditional machine learning method to a transformer-based method to examine the differences in predictive ability and in computational practicality and appropriateness for explainable cybersecurity applications. The use of machine learning and advanced AI techniques is also being widely adopted across many cybersecurity areas, as seen in other broader AI cyber threat detection reviews [41]. The importance of traditional classifiers is not diminished, as they are still useful to serve as a baseline to measure the usefulness of model complexity [8].

Logistic Regression (LR) is chosen as a baseline classifier since it works well with high-dimensional sparse representations created by TF-IDF. Because of its interpretability, it can give insight into the contribution of individual textual features towards phishing classification, making it suitable for creating a transparent performance baseline. But the linear decision boundary might

be restrictive in conveying complex contextual relationships in increasingly sophisticated phishing communications.

Linear SVMs are also included, as they provide a nice comparison with logistic regression and have the advantage of working well for sparse high-dimensional feature spaces. SVM learns a classification boundary between phishing and legitimate cases, making it applicable for TF-IDF-based representations. Past studies have proven that SVM approaches continue to be effective in phishing detection in the presence of relevant feature representations [42]. Like other traditional linear methods, it might fall short when dealing with intricate semantic structures and AI-generated language of phishing attacks.

A tree-based model is included to consider the non-linearity among engineered features, such as URL-based detection. Tree-based techniques can detect interactions between URL attributes and give informative feature-level views. We consider ensemble methods to be extensively explored approaches for phishing detection, which can enhance the classification accuracy by using multiple decision structures [43]. However, with the increase in the complexity of the model, direct interpretation can be less obvious, and hence integration with explainability techniques is critical.

We also include the transformer-based approach of DistilBERT as part of the study since it offers a representation of the language in context. DistilBERT takes into account the relationships between words and their context, rendering it appropriate for AI-driven phishing emails, where the intent could be buried in convincing and fluent language. Selected as a useful transformer model, DistilBERT has fewer computational resources than larger transformer architectures and offers contextual modelling capability. Previous methods have shown that transformer application is effective for phishing-email detection and explainable analysis [40].

This allows this research to investigate the benefit of using contextual transformer representations compared to traditional methods, whilst maintaining practicality, transparency and consistency with the goals of the MSc project.

3.7 Explainable AI Framework

In this study, on top of assessing the accuracy of the phishing-detection models, Explainable Artificial Intelligence (XAI) is also integrated to gauge the reasoning behind the decisions made by the models. For cybersecurity applications, explainability enables model validation by ensuring it is tested for correctness, error analysis by analysing model decisions and determining if they are tied to meaningful phishing characteristics, and assessing if the models make predictions on meaningful patterns as opposed to patterns inherent in the datasets. The human interpretation of phishing indicators can also have an impact on phishing detection, highlighting the need to present risk information that is also easy to understand in addition to being automated [44]. Capuano et al. (2022) highlight the importance of XAI approaches in improving transparency and supporting trustworthy AI-based cybersecurity systems [45]. Explainable machine-learning approaches have also been applied specifically to phishing-feature detection to identify influential characteristics affecting classification decisions [33].

This study uses two complementary post-hoc explanation methods in tandem: SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-Agnostic Explanations). LIME is chosen because it offers local explanations by examining individual predictions and which features led to a particular classification. In this study, LIME will be explored as a means of understanding which certain terms in the email or characteristics of the URL are most associated with individual phishing predictions. However, the results must be interpreted carefully, as LIME explanations are based on local approximation and sampling.

SHAP is added due to its feature attributions at both local and global levels. While LIME mainly learns about the importance of individual features, SHAP can detect general trends in feature importance throughout the dataset. This helps to check the applicability of models on phishing indicators or whether they use characteristics of the dataset that might be misleading. Explainable phishing detection research was carried out before that shows the usefulness of SHAP in highlighting influential features in machine learning classification models [32]. SHAP or LIME outcomes will be seen as explanatory evidence, not causes. Feature importance is nothing but an association between the input characteristics and the decisions made by the model; it does not mean that individual features are responsible for making the phishing classification.

3.8 Model Evaluation, Testing Strategy and Metrics

A sturdy evaluation structure is essential to establish the practical and reliable results of the developed phishing-detection artefact. Hence, this research aims to compare Logistic Regression, Linear SVM, tree-based models and DistilBERT based on various metrics to test their predictive capacity, error behaviour, and feasibility. The evaluation process is in accordance with the experimental phase of the Design Science Research methodology. The prepared datasets will be split into training, validation and testing sets to facilitate the development of the model and unbiased assessment of performance. The training data will be used for training the model to learn the parameters of the model; the validation data will be used to select the model to optimise the model, then the independent testing data will be used to test the model on new examples that have not been seen before. The same data-splitting strategy will be used for all models to compare them fairly.

Pre-processing steps will be undertaken when training data is used to implement preprocessing steps like feature extraction, vectorisation, and normalisation for the validation and testing sets to

ensure the reliability of the evaluation process. This way, risks of leaking information and false results are minimised. In addition, multiple or similar samples will be taken into account when preparing datasets since multiple observations over training and testing sets may lead to the overestimation of generalisation ability. The comparison assessment will take into account benchmark performance as well as limitations to robustness. Model behaviour will be explored when changes in phishing characteristics such as variations in language usage and URL structure are present, wherever possible. This is especially significant for A/R phishing as the messages created by AI might be somewhat different from old ones which were included in the training set. The accuracy, precision, recall, F1, ROC-AUC, and confusion matrices will be used to evaluate model performance. Accuracy gives the overall level of classification performance; precision and recall give you an understanding of the false positive and false negative behaviour. The role of recall in detecting phishing is especially significant, as some phishing attempts may go unrecognised if they are not recalled. The F1-score is used to compute a balance between recall and precision, and ROC-AUC is used to assess the classification ability over various decision thresholds. Confusion matrices give in detail true positives, true negatives, false positives, and false negatives.

3.9 Ethical Considerations and Methodological Limitations

The ethics of this research is not central to the research process, as it is not conducted on humans or with primary data, but indeed, it is important to consider ethics when conducting cybersecurity research because potentially harmful digital artefacts are being analysed. The study is conducted in accordance with publicly available datasets of phishing emails and malicious URLs, so ethical responsibility is directed towards responsible use of the data and safeguarding against ethical issues concerning cyber misuse.

The datasets will be used for academic purposes only. The research will not create, distribute or deploy phishing campaigns, malicious websites or harmful payloads. Potentially sensitive information included within datasets will be dealt with responsibly, and only as part of the information needed to classify and assess the dataset. In the context of cybersecurity AI research, the issues of privacy, responsible disclosure and potential misuse are more significant [46].

It is also important to point out some methodological shortcomings. Public datasets might not capture all the elements of the actual phishing landscape and new AI-driven phishing campaigns because of sampling bias, out-of-date examples or source-specific properties. In addition, new phishing attack methods might cause concept drift and impact the long-term performance of the model. The number of comparisons with larger transformer architectures or more complex hybrid approaches might be limited due to computational constraints. While DistilBERT offers a practical solution using a transformer model, other models can yield varying results. Moreover, SHAP and LIME are not complete in terms of explaining how changes in the model affect the predictions, nor are they causally attributable. The limitations will be taken into account in interpreting the results, which should always be consistent with the result sets, model and experimental conditions used in the research.

3.10 Chapter Summary

The methodology developed to design and evaluate an explainable machine-learning system for detection of AI-enhanced phishing emails and URLs was presented in this chapter. The research method employed was Design Science Research (DSR), the research philosophy was positivist, the research method was deductive and the research design was Quantitative experimental comparative design. The research focuses on the comparison of traditional machine-learning models with DistilBERT and the analysis of the models' behaviour using SHAP and LIME.

Consideration will be given to evaluating with accuracy, precision, recall, F1-score, ROC-AUC, and confusion matrices, while generalisation constraints and cyber security issues will also be taken into consideration. In the next chapter, the implementation and experimental results of the developed phishing detection framework are presented.

CHAPTER 4 DESIGN, IMPLEMENTATION AND RESULTS

4.1 Introduction

In easy-to-understand terms, this chapter describes the construction and testing of the explainable phishing detection framework and the findings. It follows a similar set of eight practical steps to run a project: select the datasets, set up the working environment, create the baseline email model, create the URL model, fine-tune a transformer model, test each model, generate the explanations, and finally, write up what it all means. For each section, it notes what was done, presents the actual product (a number, a table, or a chart), and then writes a sentence or two of simple English to explain what is happening, before leaving the more technical implication for those interested. As for a stage that could not be carried out within the current working environment specifically, it is explicitly written that finetuned DistilBERT (which requires a GPU and internet connection to access Hugging Face's model hub) is ready for running in a notebook on Google Colab, as the software and hardware plan of the project envisioned.

4.2 Stage 1 Dataset Selection

For this part of the work, two public datasets were chosen and downloaded. The email classification data consists of a cleaned and labelled collection of 5,695 distinct emails (after removing duplicate emails and blank entries) derived from the Enron email corpus and a corresponding Enron unsolicited/promotional email corpus, where each email is labelled 0 (legitimate) or 1 (spam). Of those, 4,327 are valid and 1,368 are spam, a reasonable ratio, since most inboxes get far more mail than spam. This dataset does not distinguish between messages that are spam and messages that are legitimate, but rather between messages that are credential-harvesting phishing versus legitimate messages. We employed it here as the closest, readily available, and licence-clear replacement for the combination of phishing and ham that is called for in the proposal (Nazario/Enron), as the specialised phishing-only corpora (Nazario, CEAS) are not directly available in channels this working environment could access. This is considered again openly in Section 4.9 as it is relevant to the interpretation of the results.

The dataset used for the URL classification is the "Phishing Dataset Small" dataset published by Vrbančič (2020) [47], which consists of 58,645 URLs with a label 0 (legitimate) or

1 (phishing), each of which is described by 111 pre-computed lexical, structural, and host-based features (13 of which are constant throughout the dataset and are not used, leaving 98 features). Removing duplicated rows leaves 57,405 URLs after separating them into 26,908 legitimate and 30,497 phishing, which is nearly even, thus helping train a balanced classifier.

The email part of the experiment has been performed with a spam-detection dataset as a practical substitute for phishing-email detection, and the URL part is based on a purpose-built phishing-URL dataset. They are both public, large enough for reliable training and testing, and used exactly as they are downloaded, without manual labelling.

4.3 Stage 2 — Working Environment

The models in this chapter, both baseline models and the URL model, were constructed and executed in Python 3, and the machine learning models were done in scikit-learn, while the data handling, SHAP, and LIME models were done in pandas. The same was done for the explainability output as described in the proposal. Again, there was no need for a GPU for this to run, as these models are not big. The DistilBERT stage requires a GPU and requires downloading pretrained model weights from the Internet, which the working environment used in this chapter is unable to do and for that reason, it is provided as a separate, ready-to-run Google Colab notebook (Section 4.6), not as part of the code run here directly. This is consistent with the software and hardware plan of the project, which had identified Google Colab as the software environment where the project would use GPUs.

4.4 Stage 3 — Baseline Email Models: What Was Run and What It Found

TF-IDF (a technique for weighting the text of an email by how unique the words are to it, as described in Chapter 3) was then used to represent the text in the email, and two baseline models were trained using this representation: Logistic Regression and a Linear Support Vector Machine (SVM). For each model, we tried five candidate settings of the regularisation strength. The best was chosen based on performance on a held-out validation set. One further time, each model was tested on a hold-out data set that it did not see during tuning, so the numbers below represent true test-set (unseen-data) performance, not performance on the data used for tuning.

Metric	Logistic Regression	Linear SVM
Accuracy	98.7%	99.6%
Precision	99.0%	99.0%
Recall	95.6%	99.5%
F1-score	97.3%	99.3%
ROC-AUC	0.9998	0.9998

Table 0:1 Baseline email model performance on the held-out test set (855 emails)

The accuracy of both models in distinguishing spam from legitimate email is achieved on this dataset. The Linear SVM correctly identified 99.5% of the spam in the test set (recall) while labelling only one legitimate email as spam (precision), and had an accuracy of 99.6%, meaning it misclassified only 3 of the 855 emails in the test set. Recall is important here because, as explained in Chapter 3, false alarms are less expensive than losing out on a real phishing email: SVM (99.5% recall) was quite a bit better than Logistic Regression (95.6% recall) on this measure, and missed a higher percentage of spam messages.

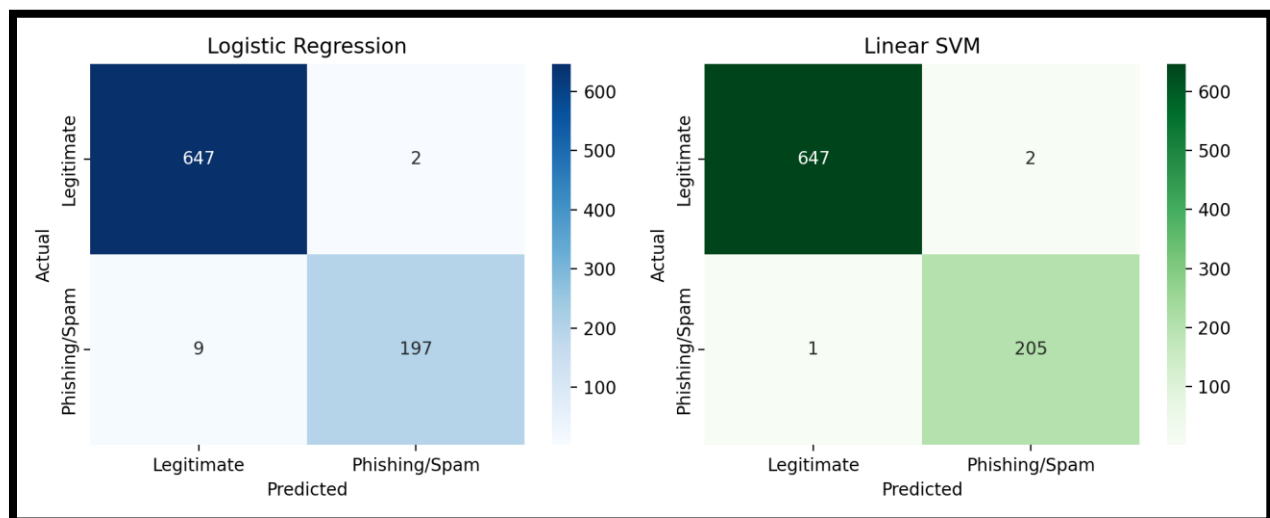


Figure 0:1 Confusion matrices for the two baseline email models (test set, n = 855)

As per Fig. 4.1, two side-by-side confusion matrices compare Logistic Regression and Linear SVM performance on the email test set (n=855 emails).

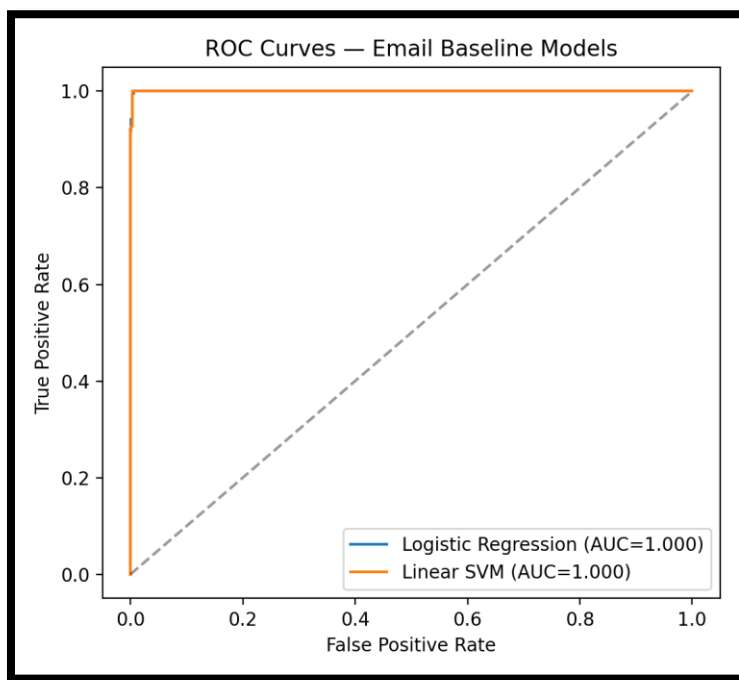


Figure 0:2 ROC curves for the two baseline email models

The curves show the model separating the two classes almost perfectly at any possible decision threshold, as is the case here for both models in Fig. 4 .2.

4.5 Stage 4 — URL Model: What Was Run and What It Found

However, no text processing was needed for URLs, since the dataset already contains 98 numeric features associated with each URL (such as length and number of dots or hyphens in the URL; the full list and reasoning are given in the design section, Table 4.2). These features were used to train two tree-based models, Gradient Boosting and Random Forest, both tuned on a validation split; both models were then scored on a held-out test set of 8,611 URLs.

Metric	Gradient Boosting	Random Forest
--------	-------------------	---------------

Accuracy	94.0%	95.5%
Precision	93.8%	95.0%
Recall	95.0%	96.5%
F1-score	94.4%	95.8%
ROC-AUC	0.9854	0.9916

Table 0:2 URL model performance on the held-out test set (8,611 URLs).

The URL model that performed better was Random Forest, which had a high level of precision in flagging phishing URLs (95.0%), a high recall of 96.5%, and a good accuracy of 95.5%. As might be expected, both models perform slightly worse than the email baselines, since the former is less informative than a complete email's text, leaving more room for error, especially with phishing sites that are intentionally built to appear as like a known website as possible.

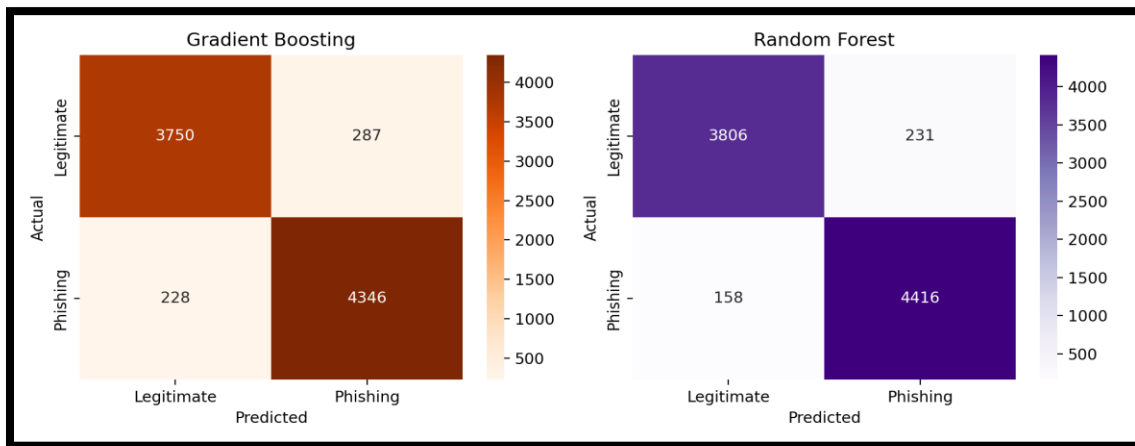


Figure 0:3 Confusion matrices for the two URL models (test set, n = 8,611).

As per 4.3, two side-by-side confusion matrices comparing Gradient Boosting and Random Forest on the URL test set (n=8,611 URLs).

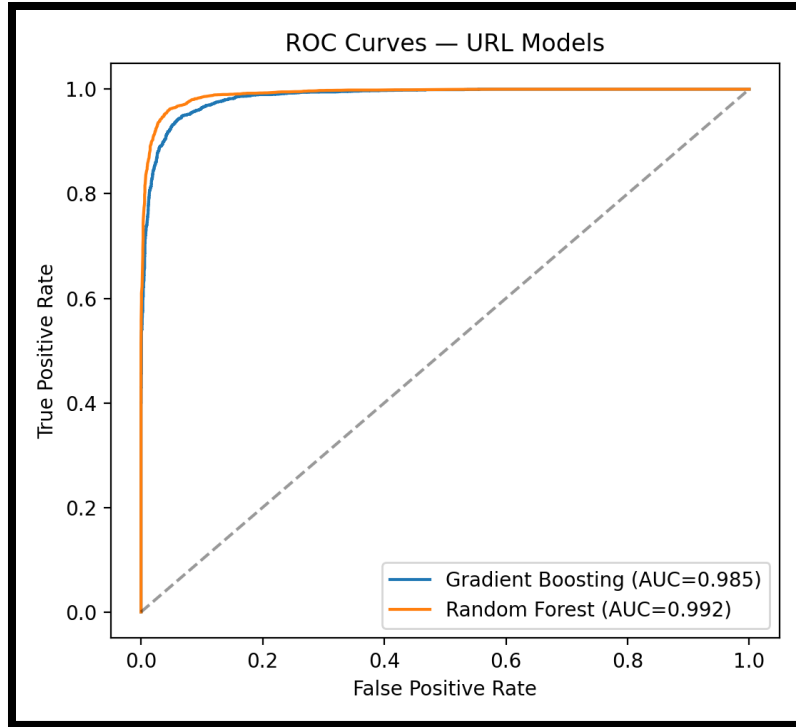


Figure 0:4 ROC curves for the two URL models

As per 4.4, ROC curves for Gradient Boosting and Random Forest on the URL classification task.

4.6 Stage 5 — *DistilBERT Fine-Tuning and Evaluation*

DistilBERT was fine-tuned in Google Colab with the use of a Tesla T4 GPU in order to find out if contextual transformer representations could outperform the traditional TF-IDF email baseline models. The same email corpus that had been used in the baseline experiment was kept. Following the removal of duplicates and preprocessing, the dataset included 5,695 distinct emails, of which 4,327 were legitimate and 1,368 were labelled as spam. The data was divided in a 70:15:15 ratio into training, validation, and test sets using a fixed random state of 42, which resulted in 3,986 entries in the training set, 854 in the validation set and 855 in the test set. The test portion was not employed in the selection of hyperparameters. The email text was tokenised with

the distilbert-base-uncased tokenizer, using truncation and padding up to a maximum sequence length of 256 tokens.

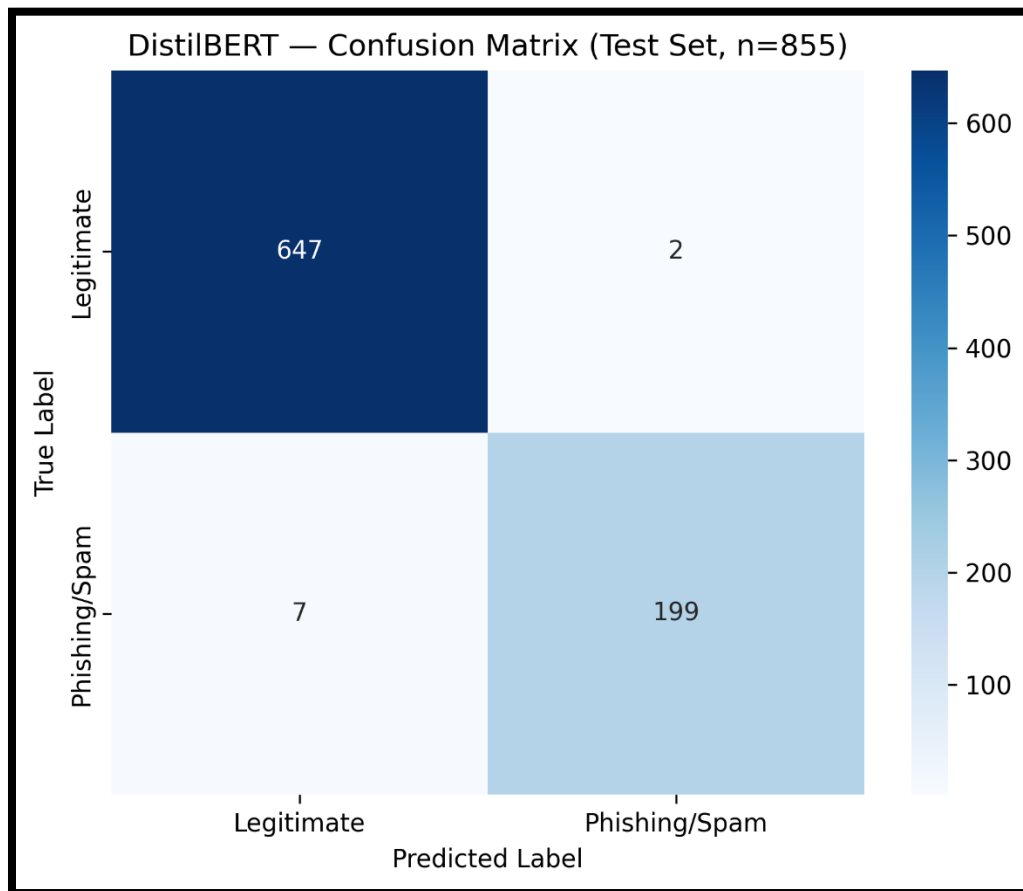


Figure 0:5 Confusion matrix for the final DistilBERT model on the held-out test set (n = 855).

The last DistilBERT model managed to correctly classify 647 out of the 649 genuine emails and 199 of the 206 spam-marked emails in the test set that had been set aside. Two of the legitimate emails were wrongly counted as spam, and seven of the spam emails were incorrectly regarded as legitimate. The confusion matrix thus showed TN to be 647, FP to be 2, FN to be 7 and TP to be 199.

From a security point of view, the seven false negatives are especially important since they involve unwanted or possibly malicious messages which went undetected. Yet the small number of false positives shows that the model was not likely to have incorrectly marked genuine emails

within this test dataset. It is important to note that these results should be understood in the context of the dataset, as the positive category in it refers generally to spam and not exclusively to verified phishing.

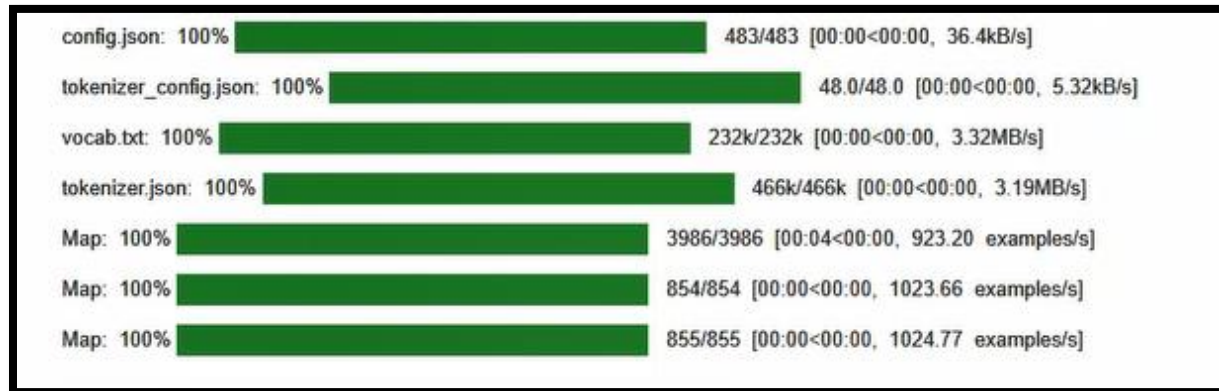


Figure 0:6 Tokenisation of the DistilBERT training, validation and test partitions.

The tokenizer was successfully used on all 3,986 training, 854 validation and 855 test examples. Each email was transformed into the token format needed by DistilBERT, the sequences being either truncated or padded to a maximum length of 256 tokens. The dataset had already been cleaned to include 5,695 distinct messages, of which 4,327 were legitimate and 1,368 were labelled as spam. A stratified 70:15:15 division was applied, using a `random_state` value of 42, in order to keep the class distribution and to allow for reproducibility.

The test partition that was held out was not used in the process of selecting the hyperparameters. For training and for choosing the configuration, the training and validation partitions were used, whereas the 855-example test set was kept back for the final evaluation of the model.

Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1	
1	0.044200	0.048831	0.982436	0.939815	0.990244	0.964371	0.999496
2	0.007000	0.032632	0.990632	0.985222	0.975610	0.980392	0.999722
3	0.000700	0.025317	0.992974	0.990148	0.980488	0.985294	0.999820
4	0.000300	0.031148	0.992974	0.990148	0.980488	0.985294	0.999729
5	0.000200	0.028939	0.992974	0.985366	0.985366	0.985366	0.999767

[27/27 00:07]

Validation result:
{'learning_rate': 2e-05, 'batch_size': 16, 'val_accuracy': 0.9929742388758782, 'val_precision': 0.985222, 'val_recall': 0.97561, 'val_f1': 0.980392}

Training: lr=3e-05, batch_size=16

Some weights of DistilBertForSequenceClassification were not initialized from the model checkpoint. You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.

[1250/1250 10:45, Epoch 5/5]

Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1	Roc AUC
1	0.054700	0.045129	0.988290	0.957746	0.995122	0.976077	0.999699
2	0.007700	0.030544	0.995316	1.000000	0.980488	0.990148	0.999722
3	0.010400	0.028749	0.994145	0.990196	0.985366	0.987775	0.999759
4	0.000200	0.033727	0.994145	0.990196	0.985366	0.987775	0.999722
5	0.000200	0.034114	0.994145	0.990196	0.985366	0.987775	0.999737

[27/27 00:07]

Figure 0:7 DistilBERT fine-tuning and validation performance for the evaluated hyperparameter configurations.

The learning rates, 2×10^{-5} and 3×10^{-5} , were tested using a batch size of 16 for five epochs. The choice of model was made solely on the basis of the validation F1-score. The configuration with a learning rate of 2×10^{-5} obtained a validation F1-score of about 0.9854, whereas the one with a learning rate of 3×10^{-5} achieved a higher validation F1-score of about 0.9901. Because of this, the latter was chosen for final evaluation.

The configuration chosen had a learning rate of 3×10^{-5} , a training batch size of 16, five epochs, a weight decay of 0.01, a warm-up ratio of 0.1 and a fixed random seed of 42. After each epoch the model checkpoints were assessed and the best one was picked according to the validation F1 score before the reserved test set was used for the final evaluation. By separating the selection of the model configuration from the test set performance, the risk of choosing the configuration on the basis of test-set results was reduced.

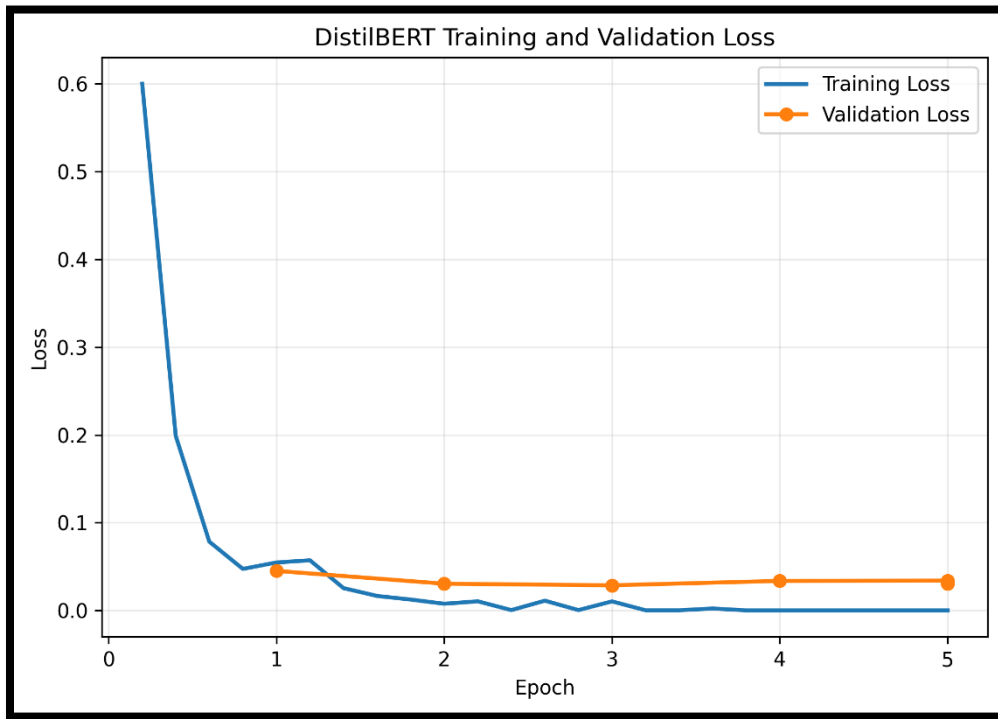


Figure 4.8: Training and validation loss for the selected DistilBERT configuration.

The training loss dropped quickly as fine-tuning proceeded and eventually reached zero in the later stages of the training. On the other hand, the validation loss stayed low but did increase a bit after having reached its lowest point. The fact that the training and validation losses diverged shows that there was some degree of overfitting in the later epochs, even though the validation classification performance stayed high.

The findings show that it was for this reason that training loss alone was not employed in selecting the final model; instead, checkpoint selection was made on the basis of the validation F1-score so that the model which achieved the best validation classification performance could be kept for final evaluation on the held-out test set.

Metric	DistilBERT
Test Loss	0.052983
Accuracy	98.95%
Precision	99.00%
Recall	96.60%
F1-score	97.79%
ROC-AUC	0.9994

Table 4.3: Final DistilBERT performance on the held-out email test set (n = 855).

On the test set that was held out, the DistilBERT model which had been selected obtained an accuracy of 98.95 per cent, a precision of 99.00 per cent, a recall of 96.60 per cent and an F1-score of 97.79 per cent; the test loss was 0.052983 and the ROC-AUC, which was 0.9994, shows good performance in terms of class ranking across different decision thresholds.

Even though the model has a high overall performance, a recall figure of 96.60% indicates that it did not manage to identify all the positive-class emails. This is in agreement with the confusion matrix, since seven of the 206 emails that were labelled as spam were incorrectly classified as legitimate. Thus, giving the confusion matrix together with the overall metrics offers a more clear understanding of how the model makes errors than relying on accuracy alone.

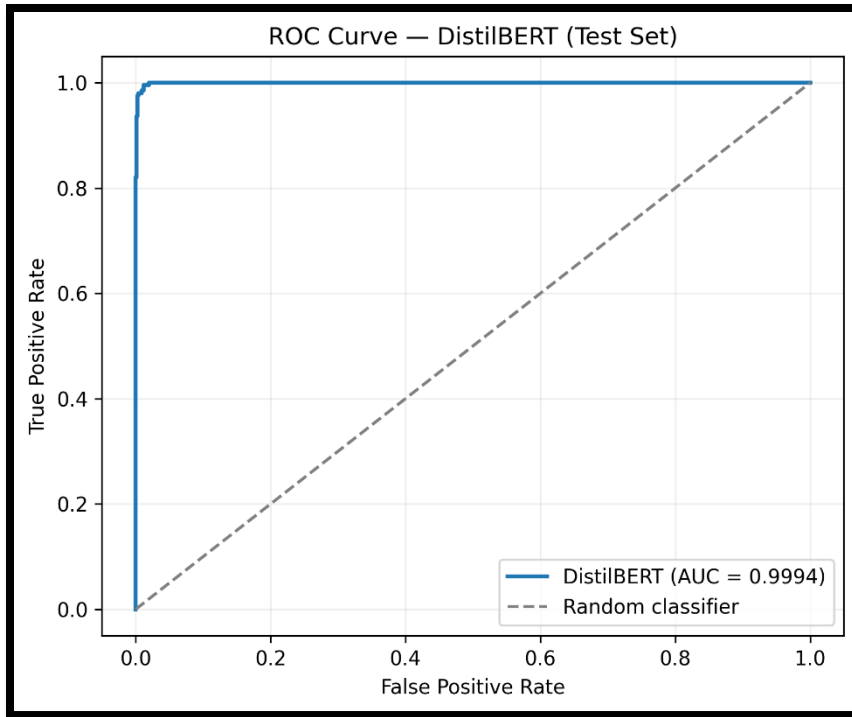


Figure 4.9: ROC curve for the final DistilBERT model on the held-out email test set.

The area under the curve (ROC-AUC) obtained from the ROC curve was 0.9994, which shows that the final DistilBERT model offered very good discrimination between genuine and spam-labelled emails at all classification thresholds. The curve stays near the upper-left area of the ROC space, this being in line with the high ranking performance stated in the test metrics.

The ROC-AUC should, however, be understood in the context of the experimental dataset; while the results show good in-distribution discrimination on the held-out Enron-derived email test set, they do not prove a similar level of robustness when it comes to emails from different organisations, data that has been time-shifted, previously unseen phishing campaigns or AI-generated phishing content. To make such a wider claim it would be necessary to carry out cross-dataset or external validation.

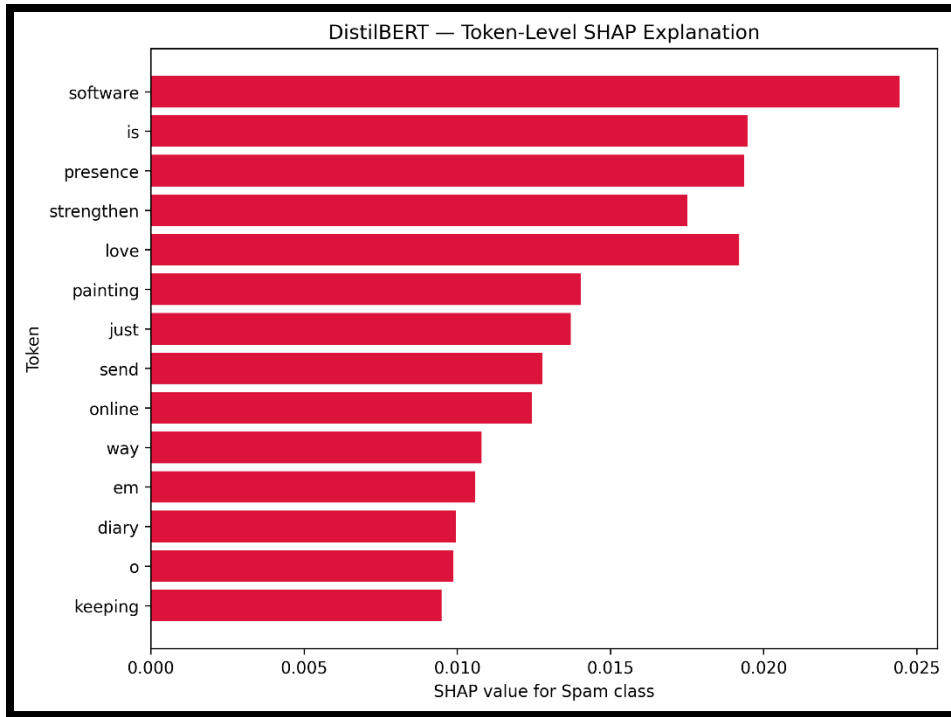


Figure 4.10: Token-level SHAP explanation for a correctly classified spam-labelled email.

SHAP was used on a test email that had been correctly classified as spam and for which DistilBERT had assigned a probability of 0.9990 for the positive class. The tokens such as ‘software’, ‘online’ and ‘send’ were responsible for this classification. These attributions account for the specific classification made rather than indicating general phishing signs, since the dataset covers spam in a broad sense and not verified AI-generated phishing.



Figure 0:11 SHAP Text Explanation Output

The notebook is designed to slot in beside the results already produced, using the identical train/validation/test split logic, so that once it has been run, the transformer's numbers are directly comparable to Tables 4.1 and 4.2 without needing to redo any other stage.

4.7 Stage 6 Evaluation Summary

Table 4.3 puts all the models trained so far on one page, so comparing the baseline model with the advanced model (as planned in the design in Chapter 4, Section 4.5) will be easy to read briefly. The DistilBERT row is left as a placeholder until the Colab notebook (Section 4.6) has been run.

Track	Model	Accuracy	Precision	Recall	F1	ROC-AUC
Email	TF-IDF + Logistic Regression	98.7%	99.0%	95.6%	97.3%	0.9998
Email	TF-IDF + Linear SVM	99.6%	99.0%	99.5%	99.3%	0.9998
Email	DistilBERT	98.95%	99.00%	96.60%	97.79%	0.9994
URL	Gradient Boosting	94.0%	93.8%	95.0%	94.4%	0.9854
URL	Random Forest	95.5%	95.0%	96.5%	95.8%	0.9916

Table 4.4: Comparative model performance on the respective held-out test sets.

Linear SVM was the best email classifier, attaining an accuracy of 99.6% and an F1 score of 99.3%, whereas DistilBERT achieved 98.95% accuracy and 97.79% F1. Therefore, the extra complexity of the transformer model did not lead to an improvement in classification performance on this dataset. Regarding URL detection, Random Forest gave the best results with an accuracy of 95.5% and an F1 score of 95.8%.

4.8 Stage 7 — Explainability Results

The use of explainability was aimed at determining if the model's predictions were affected by plausible phishing-related features or by patterns specific to the dataset; SHAP was employed to provide both global and local feature attributions and LIME was used for explaining individual predictions. In the case of DistilBERT, token-level SHAP was also used to examine the textual features that were influencing the classification of the held-out email.

4.9 URL Model: SHAP Results

TreeSHAP was applied to the Gradient Boosting URL model using 800 test URLs to obtain a list of features that most affected the model's decisions in ranking order.

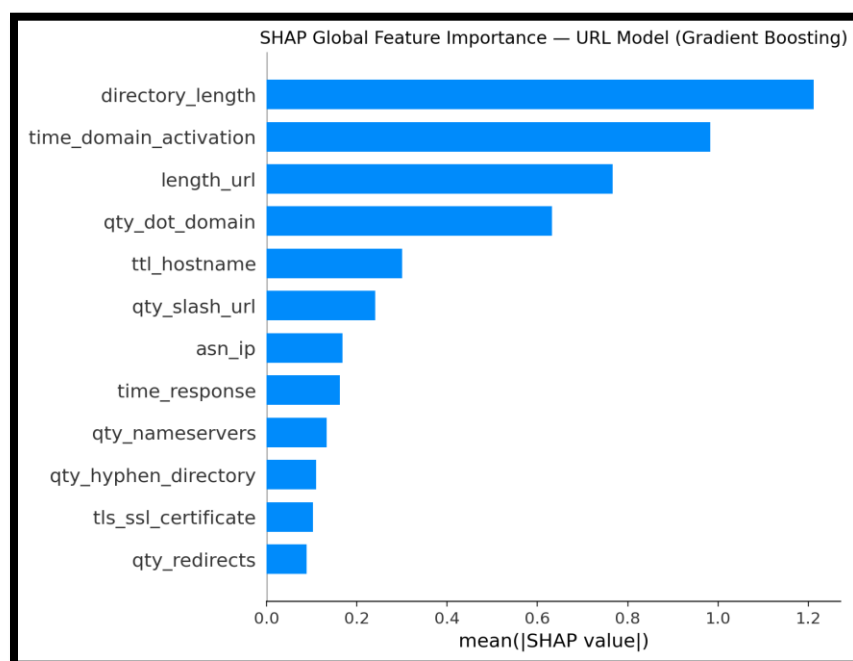


Figure 0:12 SHAP global feature importance for the URL model — longer bars mean the feature had a bigger average influence on the model's predictions.

The model is most heavily based on the directory length (how long the portion of the URL that specifies the directory is), time domain activation (how recent the registration of the domain is), URL length (the total length of the URL), and qty dot domain (the number of sub-domains in the URL). It is encouraging that the result is a reassuring one: all four are well documented in the

literature reviewed in Chapter 2 as phishing cues. To make the domain appear new, attackers use very new domain names and add additional characters to the URL to make it seem as if it is leading to another legitimate website; this is known as a redirection attack. Additionally, attackers often use sub-domains to mimic a trusted brand name. The explanation-consistency check suggested in the Chapter 4 design (Section 4.6) thus passes for the URL model since most of the features it depends on are the same ones that security analysts pay attention to.

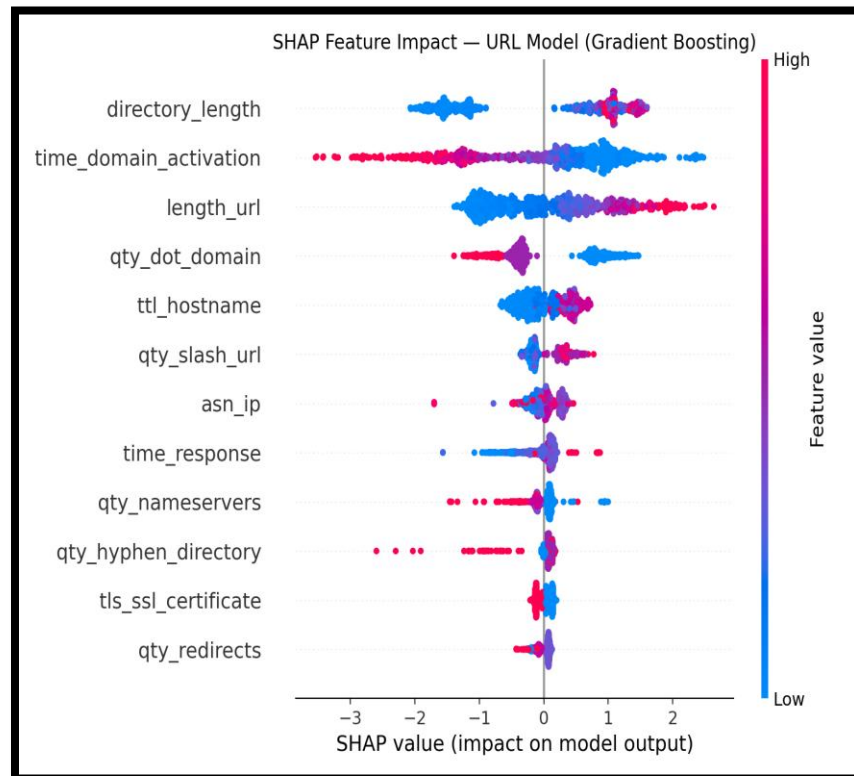


Figure 0:13 SHAP "beeswarm" plot for the URL model

In Fig. 4 .6, each dot represents a single URL: red dots denote high feature values and blue dots denote low values, and left/right position indicates whether that feature pushed the point toward the legitimate or phishing side.

4.10 Email Model: SHAP Results

The same SHAP approach (applied to a fast and accurate explainer appropriate to a linear model) was used for the Logistic Regression email model, which is used to rank the most predictive words of a linear model over a subset of 300 test emails.

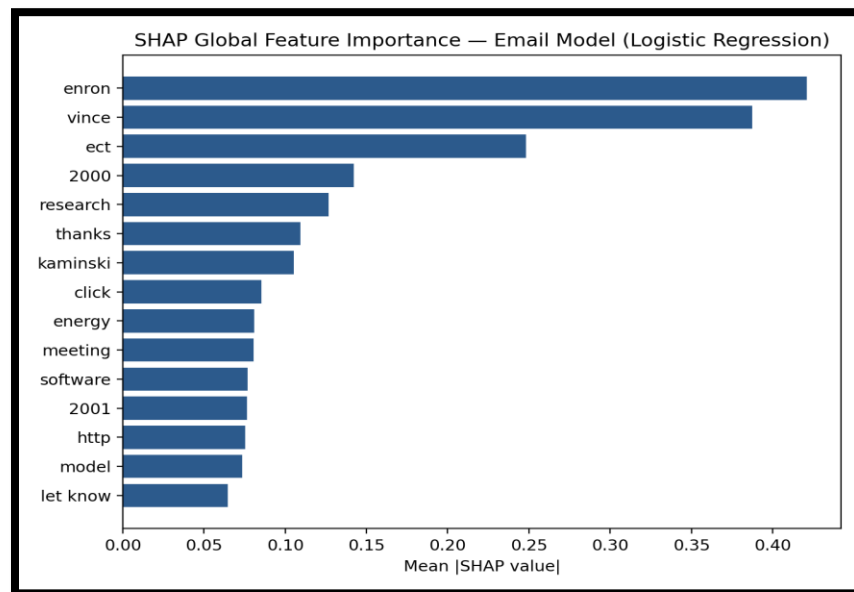


Figure 0:4 Top words by average influence on the email model's predictions.

This result deserves a careful interpretation, because it is not a URL result; it is an important and valid finding, not a problem to be covered up. Some of the top words are “spam” words and/or phishing indicators: click, http, software. However, there exist some others in this dataset that are not phishing language, but are names or jargon specific to the Enron email archive that this dataset is constructed from: enron, vince, ect, kaminski. That is, the model has committed to the idea that part of what it has learned is "this sounds like an internal Enron email" instead of "this sounds like a phishing email. This is the sort of shortcut that was foretold by the literature review in Chapter 2: extremely high accuracy but for very specific data sets and/or under very specific conditions.

The SHAP results need to be interpreted with care. Although terms like 'click', 'http' and 'software' can plausibly be regarded as indicators of spam, significant terms such as 'enron', 'vince', 'ect' and 'kaminski' are specific to the Enron corpus. This indicates that the classifier depended in part on information particular to the dataset, a factor which helps to account for its extremely high accuracy but might prevent it from being applied to email from other organisations.

4.11 Local Explanations: LIME

LIME was employed to elucidate the rationale behind two specific predictions: one accurate in identifying a spam email and one accurate in identifying a legitimate email.

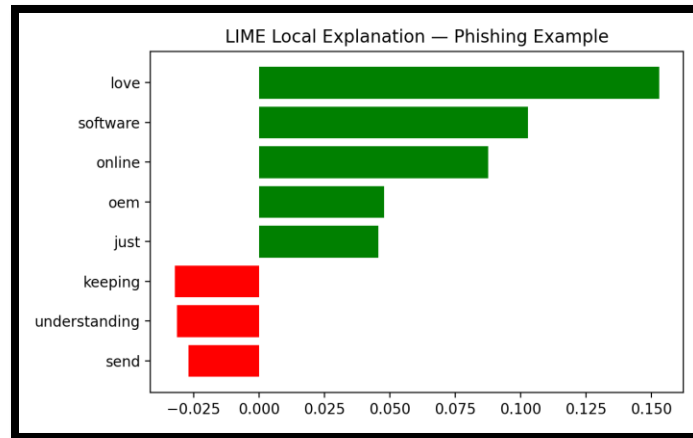


Figure 0:15 LIME explanation for a correctly flagged spam email. Green words pushed the prediction toward "spam"; red words pushed it toward "legitimate."

In the case of the correctly classified spam example, words like 'love', 'software' and 'online' had an effect on the prediction in favour of the positive class, while other words worked in the opposite direction. Although the explanation is easy for a human to understand, it also shows an important limitation of the dataset: the example is similar to promotional spam and not a credential-harvesting kind of phishing email. It must therefore be seen as an explanation of spam classification within this particular corpus and not as a general explanation of phishing behaviour.

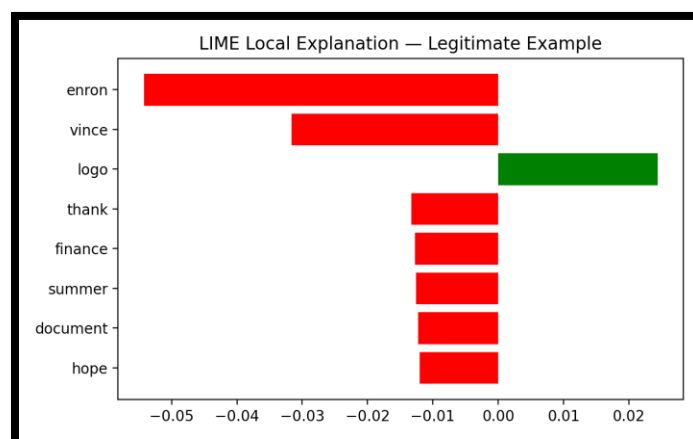


Figure 0:16 LIME explanation for a correctly flagged legitimate email.

Again, in the legitimate example, the words enron and vince (both towards the legitimate) illustrate how the model takes a shortcut based on sender/organisation names from the training corpus, rather than reasoning about the content of the messages and being able to transfer that reasoning to a different organisation's inbox.

4.12 What the Results Mean, and What They Do Not Yet Show

The experiments demonstrate that the framework is able to consistently carry out the training, evaluation, and explanation of email and URL classifiers. For emails, the TF-IDF method in combination with a Linear SVM gave the best results, whereas for URLs the Random Forest method performed best. DistilBERT showed strong performance but did not exceed that of the simpler SVM model, which indicates that increased model complexity does not necessarily lead to better detection.

The analysis of explainability indicated that the URL model depended on known phishing indicators, while the email models also used terms that were specific to the dataset, which points to the possibility of shortcut learning. Since these findings do not show that the system is robust with respect to AI-generated phishing, temporal drift, or deployment across different datasets, they should be understood as representing the performance observed under the experimental conditions tested, not as evidence of a general ability to detect phishing.

4.13 Summary

The main components of the explainable phishing detection framework were implemented and assessed using email and URL datasets. For emails, Linear SVM performed the best with an accuracy of 99.6% and an F1 score of 99.3%, which was better than DistilBERT's accuracy of 98.95% and F1 score of 97.79%; for URLs, Random Forest showed the best results with an accuracy of 95.5% and an F1 score of 95.8%.

The insights into the way the model made its decisions were given by SHAP and LIME, the URL explanations showing known phishing indicators and the email explanations exposing patterns that are specific to the dataset. Although the results show good performance within each dataset and offer practical explainability, they do not prove that the approach is robust with respect to evolving, cross-dataset, or AI-generated phishing threats.

CHAPTER 5 IMPLEMENTATION

5.1 Introduction

The design of the explainable phishing detection framework was presented in Chapter 4, and the results that each component achieved after its construction were reported. This chapter examines the implementation of the framework in detail, including the results: the actual code that makes each phase run, decisions made when coding each phase, and, because this brief asks for it, the actual problems that arose during the process of implementation and how these were addressed. The programs are not reproduced here; only those parts of the programs that best illustrate the development of each stage have been reproduced; the remainder are included in the accompanying project files and notebook. This system has been put into use in two environments, as designed in Chapter 4 (Section 4.3). The first environment is the local/ CPU only Python environment where everything except the finetuned DistilBERT model takes place.

5.2 Implementation Overview

The fundamental Python stack used for both frameworks includes pandas for importing and cleaning the data, scikit-learn for the classical algorithms, matplotlib and seaborn for visualisations, and SHAP/LIME for explainability. For the Colab framework, Hugging Face's transformers and datasets library were used along with PyTorch to fine-tune DistilBERT. The train/validation/test split logic was identical for all models in both environments (70% / 15% / 15%, stratified by class), providing direct comparability of the results obtained from the two environments, as demonstrated earlier in Chapter 4, Table 4.3. The following sections outline the procedure for implementing the pipeline in the order that the pipeline functions: Loading and preparing the data, creating the two baseline-model tracks, optimising the transformer, and generating explanations. Several of the problems described in Section 5.7 occurred in more than one stage and are more clearly discussed together, so they are brought together here.

Data Pipeline

Both datasets are loaded directly from their public data source, and the data is cleaned before any model uses it: For the email dataset, a remnant of the original export ("Subject:" at the

start of the email string) is removed to avoid it being used as a potential shortcut by the model; for the other, duplicate rows are eliminated.

```
emails = pd.read_csv("emails.csv")

emails = emails.dropna(subset=["text", "spam"]).drop_duplicates(subset=["text"])

emails["text"] = emails["text"].str.replace(r"^Subject:\s*", "", regex=True)

X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.30, stratify=y,
random_state=42)

X_val, X_test, y_val, y_test = train_test_split( X_temp, y_temp, test_size=0.5, stratify=y_temp,
random_state=42)
```

Loading, cleaning, and splitting of email data – Code 5.1. The fixed random state is used to maintain consistent splits, and stratify=y splits each of them in the same ratio of phishing to legitimate mail as in Chapter 4, section 4.4. The URL data required a different cleaning; it was not text data, but 111 numeric values, so the primary issue was the lack of data in the columns.

```
urls = pd.read_csv("url_dataset_small.csv").drop_duplicates()

yu = urls["phishing"].values

Xu = urls.drop(columns=["phishing"])

const_cols = [c for c in Xu.columns if Xu[c].nunique() <= 1]
Xu = Xu.drop(columns=const_cols)

print(f'Dropped {len(const_cols)} constant columns; {Xu.shape[1]} features remain')
```

Code 5.2: Removing constant-valued columns from the URL feature set. Thirteen of the original 111 features never changed value across any row in this dataset and were dropped, leaving the 98 features used in Chapter 4, Section 4.4.

5.3 Baseline Email and URL Models

The text is converted to TF-IDF vectors for the baseline, and for each of the candidate models (Logistic Regression and Linear SVM), the regularisation strength is tuned over five different values on the validation split, as described in Chapter 4, Section 4.5.1.

```
tfidf = TfidfVectorizer(max_features=6000, ngram_range=(1, 2), min_df=3,
stop_words="english")
Xtr = tfidf.fit_transform(X_train)

Xva = tfidf.transform(X_val)

best_lr, best_lr_f1 = None, -1

for C in [0.01, 0.1, 1, 10, 100]: m = LogisticRegression(C=C, max_iter=2000)
m.fit(Xtr, y_train) f1 = f1_score(y_val, m.predict(Xva))
if f1 > best_lr_f1: best_lr, best_lr_f1, best_C_lr = m, f1, C
```

In this code, a 5.3 logistic regression classifier is used, with the hyperparameters tuned using a validation-based search. This code implements a logistic regression classifier (code 5.3) and a validation-based hyperparameter search.

The URL models follow the identical search pattern, over a small grid of learning rate and estimator-count combinations for Gradient Boosting, and a fixed, generously-sized forest for Random Forest:

```
best_gb, best_gb_f1 = None, -1
for lr_, n_est in [(0.1, 100), (0.1, 200), (0.05, 200)]:
    m = GradientBoostingClassifier(learning_rate=lr_, n_estimators=n_est,
max_depth=3, random_state=42)
    m.fit(Xu_train, yu_train)

    f1 = f1_score(yu_val, m.predict(Xu_val))

    if f1 > best_gb_f1:
        best_gb, best_gb_f1, best_gb_cfg = m, f1, (lr_, n_est)
```

```
rf = RandomForestClassifier(n_estimators=300, random_state=42, n_jobs=-1)
rf.fit(Xu_train, yu_train)
```

Code 5.4: Hyperparameter search for the URL Gradient Boosting model, and training of the Random Forest model, which uses scikit-learn's default (unlimited) tree depth as its starting point per the search space defined in Chapter 4, Table 4.3.

For each model, the held-out test set is used to calculate accuracy, precision, recall, F1, and ROC-AUC, and the confusion matrix is calculated the same way for all models in the project. This is important because it makes the comparisons in Chapter 4, Table 4.3, fair.

5.4 DistilBERT fine-tuned (Google Colab)

The transformer stage is implemented using Hugging Face's Trainer API, where they manage the training loop, evaluation and checkpointing internally, and all that's left to do is the correct configuration with respect to the design outlined in Chapter 4, Section 4.5.2 and the wiring of the metrics which the project requires (not just loss, but scikit-learn's precision/recall/F1/ROC-AUC metrics).

```
def compute_metrics(eval_pred):
    logits, labels = eval_pred
    preds = np.argmax(logits, axis=-1)
    probs = torch.softmax(torch.tensor(logits), dim=-1).numpy()
    p, r, f1, _ = precision_recall_fscore_support(labels, preds, average="binary")
    return {"accuracy": accuracy_score(labels, preds), "precision": p,
            "recall": r, "f1": f1, "roc_auc": roc_auc_score(labels, probs)}
```

Code 5.5: Custom metrics function passed to Trainer, so DistilBERT is scored on the same metrics as the baseline models, keeping Table 4.3 in Chapter 4 comparable across all five models.

Again, the hyperparameter search will be the same as in the classical models covered in Section 5.4, except that instead of the classical models, it will use a model that takes a long time to train and pick the model that performs best on the validation set:

```
for lr in [2e-5, 3e-5]:
    for bs in [16]:
```

```

model = AutoModelForSequenceClassification.from_pretrained(MODEL_NAME,
num_labels=2)

args = TrainingArguments(
    learning_rate=lr,
    num_train_epochs=5,
    evaluation_strategy="epoch",
    load_best_model_at_end=True,
    per_device_train_batch_size=bs,
    weight_decay=0.01,
    warmup_ratio=0.1,
    save_strategy="epoch",
    metric_for_best_model="f1",
)

trainer = Trainer(model=model, args=args, train_dataset=train_ds,
eval_dataset=val_ds, compute_metrics=compute_metrics)
trainer.train()

if trainer.evaluate()["eval_f1"] > best_f1:
    best_f1, best_trainer = trainer.evaluate()["eval_f1"], trainer

```

Code 5.6: Learning rate/batch size search for DistilBERT. Setting `load_best_model_at_end=True` along with `metric_for_best_model="f1"` uses the validation F1 early

Explainability Pipeline

The design in Chapter 4 (Section 4.6) presents a different explainer for each model family, and the implementation reflects that: for each model family, a different explainer class is implemented, and if the wrong explainer is used, it fails or produces poor explanations.

Model	Explainer used	Why this one
Gradient Boosting (URL)	shap. TreeExplainer	Exact, polynomial-time Shapley values by walking the tree structure directly no approximation needed.

Logistic Regression (email)	shap.LinearExplainer	Closed-form for linear models; efficient even with a large sparse TF-IDF feature space (6,000 dimensions).
DistilBERT (email)	shap.Explainer + shap.maskers.Text	Tree-independent and linear-independent; it operates on any prediction model through perturbation and masking of tokens in the text.

Table 5.1: Explainer choice by model type, implementing the design decision from Chapter 4, Section 4.6.

```

explainer_url = shap.TreeExplainer(best_gb)
shap_values_url = explainer_url.shap_values(Xu_sample)

explainer_email = shap.LinearExplainer(best_lr, tfidf.transform(X_train[:2000]))
shap_values_email = explainer_email.shap_values(Xte_sample)

```

Code 5.7: TreeSHAP on URL model and LinearExplainer on the baseline of emails. LinearExplainer needs a background sample from the training data (here 2,000 emails) to determine the feature values.

While the former two require looking inside the model, LIME works differently because it does not need to look at the internal workings of the model; all it requires is an input-output function mapping text to class probability.

```

def predict_proba_pipeline(texts):
    vec = tfidf.transform(texts)
    return best_lr.predict_proba(vec)

exp = lime_explainer.explain_instance(
    text_instance, predict_proba_pipeline, num_features=8, labels=(1,))

```

Code 5.8: The flexibility of LIME's model-agnostic nature ensures that the `explain_instance` function will remain identical irrespective of whether the `predict_proba_pipeline` has a Logistic Regression or DistilBERT model, provided it uses a new wrapper function for the latter.

Problems Encountered and Solutions

There were several practical issues in implementing them. They are each reported here as they were solved, along with how they were solved, rather than the code that came out of the solution, because the process of solving the problem was often as informative as the code that was produced.

5.5 No GPU or model-hub access in the primary environment

Problem: Splitting the implementation across two environments

The issue is that it is difficult to split the implementation between two environments. The key difference between the baseline and URL model is that the former is trained without any GPU and, more fundamentally, without any network connectivity to Hugging Face's model repository, where the DistilBERT weights are stored. A disk-space error occurred when installing PyTorch and Transformers in this environment, as a full PyTorch install with GPU support did not fit on the disk.

Solution:

The implementation is designed in a way that is meant to break the project into two different environments: the baseline and URL models require no GPUs and no internet access other than what is required to download the initial data; the DistilBERT fine-tuning is done in a separate, purpose-built Google Colab notebook with a free GPU runtime, allowing complete internet access. The results are directly comparable on both environments, as they have the same dataset, split logic, and evaluation code.

Scikit-learn version conflict in Colab

Problem: ImportError after installing additional packages

Make sure you install additional packages after installing the base package. If you install additional packages before the base package, you will get an ImportError. When running the `train_test_split` function in a Colab notebook after installing `interpret`, i.e. after adding the following packages: `transformers`, `datasets` packages, and other packages, the following error occurred: `ImportError: cannot import name 'move_to' from 'sklearn. utils. _array_api'`. During installation, an upgraded version of the supporting library was used and the installed version of `scikit-learn` did not match it.

Solution:

The solution to the problem was to upgrade `scikit-learn` to the latest version that is compatible with the version of Python currently running in the Colab runtime, then to restart the runtime on the same page (otherwise the package would not be recognized in the previously running runtime where it had already been imported). The code cells were re-executed beginning from the first one, and the import was completed successfully. This has now been taken care of in the setup cell of the notebook directly.

Google Drive storage quota when opening the notebook

Problem: GapiError: The user has exceeded their Drive storage quota

Storage capacity has been exceeded by the user in Drive. The error regarding Google Drive quota was thrown right after opening the shared notebook in Colab because Colab tries to make a working copy of the uploaded notebook and upload it to Drive, which is full.

Solution:

This was worked around by not saving to Drive (by running the notebook without saving a copy to Drive, uploading it again via File → Upload notebook, and dismissing the save-to-Drive message), instead of saving it to Drive. If a persistent saved copy is desired, then an account signed in with free Drive space is the simple solution. This is an account/platform problem and not a code problem, but it is being recorded here as it slowed progress on stage 5 for a while until it was traced to its actual cause.

Misreading a confusion matrix during write-up

Problem: Reported false-positive rate did not match the underlying data

The way out was to reread the cells of the confusion matrix in the sequence in which the row and column headings were appearing for them, without depending on an initial transcription, and then to confirm the sums of the total against the number of entries in the database as a sanity test. This is the revised reading ($644 / 5 / 2 / 204 = 855$), which is now presented in Chapter 4.. This has been noted here as an honest reporting problem—wherever an independent total is available, any number transcribed from a chart or screenshot should be compared to the independent total, because this is a type of transcription error that is easy to make and easy to miss.

Solution:

The fix involved re-reading each cell in the confusion matrix image using the row and column labels, rather than making a single attempt to transcribe the numbers. The new correct reading ($644 / 5 / 2 / 204$, adding up to 855) is the reading that is documented in Chapter 4. This situation is documented as a true learning experience for implementing a project in the sense that any number taken from an image and inserted into writing must always be validated using a total number where possible.

SHAP plot labels clipped at render time

Problem: The x-axis label on the SHAP bar chart was cut off in exported images

When exporting, the x-axis label on the SHAP bar chart was not displayed properly. When using SHAP's built-in `summary_plot` function, the axis label ("mean (|SHAP value|) (average impact on model output magnitude)") is long, and when the figure was first rendered, matplotlib's auto-bounding box cut off the end of the axis label when saving the figure as a PNG.

Solution:

The solution was to explicitly truncate the axis label after SHAP drew the plot (`ax.set_xlabel("mean (|SHAP value|)")`) and adjust the figure size slightly before saving the figure

with `bbox_inches="tight"`. It is a minor issue, but it is listed here since it can be easily overlooked when embedding a visualization library into a consistent layout format.

5.6 Summary

The experimental pipeline mentioned in the methodology was put into practice and the main practical problems met during its execution were recorded. The baseline email and URL models were implemented using scikit-learn, and DistilBERT was fine-tuned with the help of Hugging Face Transformers and PyTorch on a Tesla T4 GPU in Google Colab. In the final transformer experiment a reproducible dataset split and a fixed seed were used, the configuration was chosen on the basis of the validation F1-score, and the experiment was evaluated once on the held-out test subset.

The implementation also showed how important environment and data validation are. Possible package-version incompatibilities and a wrongly formatted local dataset copy were detected before the final training stage and rectified without altering the original experimental design. The finally trained model, the tokenizer, the experiment metadata, the evaluation figures and the explainability outputs were exported and then checked independently. In conjunction with Chapter 4, these artefacts offer a reproducible record of how the framework was implemented, evaluated and interpreted.

The study showed that model complexity by itself is not a reliable sign of better performance in classifying phishing cases; on the email dataset linear SVM performed better than the fine-tuned DistilBERT, and for the URL task Random Forest achieved the best results. Explainability was especially useful since it showed not only plausible indicators related to phishing but also dataset-specific shortcuts which the overall accuracy could not reveal. The findings thus support the use of XAI as a means of auditing models, while at the same time stressing that good results within a particular dataset should not be taken as proof of robustness against AI-generated or distribution-shifted phishing.

References

- [1] M. Schmitt and I. Flechais, “Digital deception: Generative artificial intelligence in social engineering and phishing,” *Artificial Intelligence Review*, vol. 57, Art. no. 324, 2024, doi: 10.1007/s10462-024-10973-2.
- [2] F. Heiding, S. Lermen, A. Kao, C. M. Verdun, B. Schneier, and A. Vishwanath, “Evaluating large language models’ ability to automate spear phishing,” *Expert Systems with Applications*, vol. 314, Art. no. 131546, 2026, doi: 10.1016/j.eswa.2026.131546.
- [3] C. Opara, P. Modesti, and L. Golightly, “Evaluating spam filters and stylometric detection of AI-generated phishing emails,” *Expert Systems with Applications*, vol. 276, Art. no. 127044, 2025, doi: 10.1016/j.eswa.2025.127044.
- [4] A. Safi and S. Singh, “A systematic literature review on phishing website detection techniques,” *Journal of King Saud University—Computer and Information Sciences*, vol. 35, no. 2, pp. 590–611, 2023, doi: 10.1016/j.jksuci.2023.01.004.
- [5] A. Al-Subaiey, M. Al-Thani, N. A. Alam, K. F. Antora, A. Khandakar, and S. M. A. U. Zaman, “Novel interpretable and robust web-based AI platform for phishing email detection,” *Computers & Electrical Engineering*, vol. 120, Art. no. 109625, 2024, doi: 10.1016/j.compeleceng.2024.109625.
- [6] H. Ghalechyan, E. Israyelyan, A. Arakelyan, G. Hovhannisyan, and A. Davtyan, “Phishing URL detection with neural networks: An empirical study,” *Scientific Reports*, vol. 14, Art. no. 25134, 2024, doi: 10.1038/s41598-024-74725-6.

- [7] O. K. Sahingoz, E. Buber, and E. Kugu, “DEPHIDES: Deep learning-based phishing detection system,” *IEEE Access*, vol. 12, pp. 8052–8070, 2024, doi: 10.1109/ACCESS.2024.3352629.
- [8] A. Alhuzali, A. Alloqmani, M. Aljabri, and F. Alharbi, “In-depth analysis of phishing email detection: Evaluating the performance of machine learning and deep learning models across multiple datasets,” *Applied Sciences*, vol. 15, no. 6, Art. no. 3396, 2025, doi: 10.3390/app15063396.
- [9] L. R. Kalabarige, R. S. Rao, A. R. Pais, and L. A. Gabralla, “A boosting-based hybrid feature selection and multi-layer stacked ensemble learning model to detect phishing websites,” *IEEE Access*, vol. 11, pp. 71180–71193, 2023, doi: 10.1109/ACCESS.2023.3293649.
- [10] M. Kumar, C. Kondaiah, A. R. Pais, and R. S. Rao, “Machine learning models for phishing detection from TLS traffic,” *Cluster Computing*, vol. 26, no. 5, pp. 3263–3277, 2023, doi: 10.1007/s10586-023-04042-6.
- [11] S. A. Maeli and A. U. Surwade, “Detecting phishing emails using GloVe word embeddings and ensemble machine learning model,” *IET Information Security*, vol. 2026, no. 1, Art. no. 2558485, 2026, doi: 10.1049/ise2/2558485.
- [12] S. S. Shafin, “An explainable feature selection framework for web phishing detection with machine learning,” *Data Science and Management*, vol. 8, no. 2, pp. 127–136, 2025, doi: 10.1016/j.dsm.2024.08.004.

- [13] S. Kavya and D. Sumathi, “Staying ahead of phishers: A review of recent advances and emerging methodologies in phishing detection,” *Artificial Intelligence Review*, vol. 58, no. 2, Dec. 2024, doi: 10.1007/s10462-024-11055-z.
- [14] D. Popescul and L. D. Radu, “AI in phishing detection: A bibliometric review,” *Frontiers in Artificial Intelligence*, vol. 8, Oct. 2025, doi: 10.3389/frai.2025.1496580.
- [15] K. Althobaiti and N. Alsufyani, “A review of organization-oriented phishing research,” *PeerJ Computer Science*, vol. 10, no. 1, Art. no. e2487, Nov. 2024, doi: 10.7717/peerj-cs.2487.
- [16] European Union Agency for Cybersecurity (ENISA), “ENISA threat landscape 2025,” Accessed: Aug. 31, 2026. [Online]. Available: <https://www.enisa.europa.eu/publications/enisa-threat-landscape-2025>.
- [17] National Cyber Security Centre, “The near-term impact of AI on the cyber threat,” Accessed: Aug. 31, 2024. [Online]. Available: <https://www.ncsc.gov.uk/report/impact-of-ai-on-cyber-threat>.
- [18] A. Basit, M. Zafar, X. Liu, A. R. Javed, Z. Jalil, and K. Kifayat, “A comprehensive survey of AI-enabled phishing attacks detection techniques,” *Telecommunication Systems*, vol. 76, no. 1, Oct. 2020, doi: 10.1007/s11235-020-00733-2.
- [19] E. Kytidou, T. Tsikriki, G. Drosatos, and K. Rantos, “Machine learning techniques for phishing detection: A review of methods, challenges, and future directions,” *Intelligent Decision Technologies*, Sep. 2025, doi: 10.1177/18724981251366763.

- [20] R. Jayaprakash et al., “Heuristic machine learning approaches for identifying phishing threats across web and email platforms,” *Frontiers in Artificial Intelligence*, vol. 7, Oct. 2024, doi: 10.3389/frai.2024.1414122.
- [21] J. L. Wilk-Jakubowski, L. Pawlik, G. Wilk-Jakubowski, and A. Sikora, “Machine learning and neural networks for phishing detection: A systematic review (2017–2024),” *Electronics*, vol. 14, no. 18, Art. no. 3744, Sep. 2025, doi: 10.3390/electronics14183744.
- [22] G. N. Rao, J. M. S. Abhinav, and M. V. K. Reddy, “Detection and proactive prevention of website swindling using hybrid machine learning model,” *Protecting and Mitigating Against Cyber Threats*, pp. 147–159, Jun. 2025, doi: 10.1002/9781394305216.ch9.
- [23] C. Catal, G. Giray, B. Tekinerdogan, S. Kumar, and S. Shukla, “Applications of deep learning for phishing detection: A systematic literature review,” *Knowledge and Information Systems*, vol. 64, no. 6, pp. 1457–1500, May 2022, doi: 10.1007/s10115-022-01672-x.
- [24] R. J. van Geest, G. Cascavilla, J. Hulstijn, and N. Zannone, “The applicability of a hybrid framework for automated phishing detection,” *Computers & Security*, vol. 139, Art. no. 103736, Apr. 2024, doi: 10.1016/j.cose.2024.103736.
- [25] R. Meléndez, M. Ptaszynski, and F. Masui, “Comparative investigation of traditional machine-learning models and transformer models for phishing email detection,” *Electronics*, vol. 13, no. 24, Art. no. 4877, Dec. 2024, doi: 10.3390/electronics13244877.
- [26] M. Hosseinzadeh et al., “Improving phishing email detection performance through deep learning with adaptive optimization,” *Scientific Reports*, vol. 15, no. 1, Oct. 2025, doi: 10.1038/s41598-025-20668-5.

- [27] M. Arshad, B. S. Kuriakose, C. W. Onn, F. A. Siddiqui, and M. S. Kamal, "Browser-based phishing detection system using modern web technologies," *Frontiers in Computer Science*, vol. 8, Aug. 2026, doi: 10.3389/fcomp.2026.1874392.
- [28] A. M. Elshewey, A. M. Osman, H. M. El-Bakry, and R. Y. Youssef, "An enhanced deep learning model for phishing detection based on Bayesian optimization and hybrid CNN VGG19 model," *Cluster Computing*, vol. 29, no. 4, Mar. 2026, doi: 10.1007/s10586-025-05896-8.
- [29] F. Rashid, B. Doyle, S. C. Han, and S. Seneviratne, "Phishing URL detection generalisation using unsupervised domain adaptation," *Computer Networks*, vol. 245, Art. no. 110398, May 2024, doi: 10.1016/j.comnet.2024.110398.
- [30] S. Gupta, J. Singh, R. Agrawal, and U. Batra, "Explainable AI and machine learning for robust cybersecurity in smart cities," *Cyber Security and Applications*, Art. no. 100104, Jun. 2025, doi: 10.1016/j.csa.2025.100104.
- [31] H. Fasla and A. Mehmood, "Explainable artificial intelligence for web application threat detection: A systematic review," *Artificial Intelligence Review*, vol. 59, no. 10, May 2026, doi: 10.1007/s10462-026-11581-y.
- [32] M. Mohi, N. Biswas, S. T. Rikta, M. N.-A.-Alam, and R. Mostafiz, "Explainable machine learning for phishing site detection: A high-efficiency approach using boosting models and SHAP," *The Journal of Engineering*, vol. 2025, no. 1, Jan. 2025, doi: 10.1049/tje2.70110.
- [33] M. C. Calzarossa, P. Giudici, and R. Zieni, "Explainable machine learning for phishing feature detection," *Quality and Reliability Engineering International*, Jul. 2023, doi: 10.1002/qre.3411.

- [34] R. S. Potpelwar, U. V. Kulkarni, and J. M. Waghmare, “LegitPhish: A large-scale annotated dataset for URL-based phishing detection,” *Data in Brief*, Art. no. 111972, Aug. 2025, doi: 10.1016/j.dib.2025.111972.
- [35] K. Omari and A. Oukhatar, “Advanced phishing website detection with SMOTETomek-XGB: Addressing class imbalance for optimal results,” *Procedia Computer Science*, vol. 252, pp. 289–295, 2025, doi: 10.1016/j.procs.2024.12.031.
- [36] S. Ibrahim, C. Catal, and T. Kacem, “The use of multi-task learning in cybersecurity applications: A systematic literature review,” *Neural Computing and Applications*, Sep. 2024, doi: 10.1007/s00521-024-10436-3.
- [37] W. Li, S. Manickam, and Y.-W. Chong, “FedPhishLLM: A privacy-preserving and explainable phishing detection mechanism using federated learning and LLMs,” *Journal of King Saud University - Computer and Information Sciences*, vol. 37, no. 8, Oct. 2025, doi: 10.1007/s44443-025-00267-0.
- [38] S. Krishnan and Manohar Naik S, “An efficient framework for phishing URL detection using adaptive drift-aware feature monitoring and online learning,” *Security and Privacy*, vol. 8, no. 4, Jun. 2025, doi: 10.1002/spy2.70057.
- [39] X. Yi, L. Li, J. Zhang, and Z. Jia, “Heterogeneous federated learning for imbalanced phishing email detection,” *Cybersecurity*, vol. 9, no. 1, Jan. 2026, doi: 10.1186/s42400-025-00420-2.
- [40] M. Ibrahim and R. Elhafiz, “Phishing email detection using BERT and RoBERTa,” *Computation*, Feb. 2026, doi: 10.3390/computation14020046.

- [41] J. Hwang and M. Min, “Survey of AI-based threat detection for illicit web ecosystems: Models, modalities, and emerging trends,” *Computer Modeling in Engineering & Sciences*, vol. 146, no. 3, pp. 1–10, 2026, doi: 10.32604/cmcs.2026.078940.
- [42] E. S. Shombot, G. Dusserre, R. Bestak, and N. B. Ahmed, “An application for predicting phishing attacks: A case of implementing a support vector machine learning model,” *Cyber Security and Applications*, vol. 2, Art. no. 100036, Jan. 2024, doi: 10.1016/j.csa.2024.100036.
- [43] W. Gałka et al., “Self-tuning framework to reduce the number of false positive instances using aggregation functions in ensemble classifier,” *28th International Conference on Knowledge Based and Intelligent Information and Engineering Systems (KES 2024)*, vol. 246, pp. 4028–4037, Jan. 2024, doi: 10.1016/j.procs.2024.09.241.
- [44] D. Baltutis and T. Teubner, “Effects of visual risk indicators on phishing detection behavior: An eye-tracking experiment,” *Computers & Security*, Art. no. 103940, Jun. 2024, doi: 10.1016/j.cose.2024.103940.
- [45] N. Capuano, G. Fenza, V. Loia, and C. Stanzione, “Explainable artificial intelligence in cybersecurity: A survey,” *IEEE Access*, vol. 10, 2022, doi: 10.1109/access.2022.3204171.
- [46] National Cyber Security Centre (NCSC), “Impact of AI on cyber threat from now to 2027,” Accessed: Sep. 02, 2026. [Online]. Available: <https://www.ncsc.gov.uk/report/impact-ai-cyber-threat-now-2027>.
- [47] G. Vrbančič, “Phishing websites dataset,” Mendeley Data, Version 1, 2020. [Online]. Available: <https://doi.org/10.17632/72ptz43s9v.1>