National College of Ireland

Explaining Real-Time Phishing and Explainable AI: A Transformer-Based Approach

MSc Research Practicum-Part 2
MSc in Data Analytics

Kunal Sonawane
Student ID: x23400617

School of Computing
National College of Ireland

Supervisor: Prof. Vikas Sahni

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Kunal Ramesh Sonawane
Student ID:	X23400617
Programme:	MSc in Data Analytics **Year:** 2025
Module:	Research Practicum-Part 2
Supervisor:	Prof. Vikas Sahni
Submission Due Date:	28/01/2026
Project Title:	Explaining Real-Time Phishing and Explainable AI: A Transformer-Based Approach
Word Count:	**6969** **Page Count: 22**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Kunal Sonawane

Date: 27/01/2026

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Explaining Real-Time Phishing: A Transformer-Based Approach with Explainable AI

Kunal Sonawane
x23400617

Abstract

Phishing is a growing cyber threat and the Anti-Phishing Working Group (APWG) recorded the highest number of 1,003,924 attacks in Q1 2025. The conventional static detection systems are easily overcome by advanced evasion methods, such as the rapid infrastructure changes and zero-day threats. This requirement makes sophisticated detection and the analysis of the URL string in real time. The current models with high-accuracy are still susceptible to unclear techniques such as Browser in the Browser and malicious embedded links. Most importantly, deployable systems should be transparent, and this necessitates integrating explainable details.

In order to evaluate, several models, such as the Logistic Regression, Random Forest, the XGBoost, and the RoBERTa-based embedding pipeline, were trained. The experiments prove that all models that are augmented with the transformer are much better than the traditional lexical models. XGBoost provided good performance with accuracy 95.49% and the highest inference speed of 0.14 ms per URL, which is highly suitable regarding the implementation in real-time. The SHAP integration offered understandable descriptions of model decisions with malicious subdomains, obfuscated paths, and encoded parameters being the most important. These findings confirm that the proposed system is accurate, low-latency, and interpretable, which are the main quality criteria of real-life phishing URL detection.

1 Introduction

1.1 Motivation

Phishing is the popular and the most basic threat in contemporary cybersecurity, and most of the successful attacks and financial losses on a global scale are associated with phishing. These attacks are mainly carried out through the misleading uniform resource locator (URL), which is designed in such a way that it looks like the legitimate services and uses the trust of people. Ghalechyan et al. (2024) note that simple approaches to early detection played an important role, yet the growth of new domains and increasingly sophisticated methods has made these approaches outdated.

Liu et al. (2023) since these threats are deployed at very high volumes and very fast, URL-based detection has become an important strategy to address cybersecurity. It has managed to

balance the effectiveness of detecting these threats with the necessary operational viability. Mekalarani et al. (2024) as opposed to computationally intensive deep content-based or visual similarity detection, which are complicated and also take a lot of time to wait until webpages are loaded. The detection of threats can be done in milliseconds by merely looking at the URL string. Such ability to perform high real-time is critical to providing on-the-fly protection in browser extensions or network firewalls. This need leads to the ongoing improvement of the Machine Learning and Deep Learning models that are specifically aimed at the fast and efficient feature extraction of the URL string.

Even though much has been achieved, the existing detection systems have a number of serious weaknesses. The traditional ML models do not learn the abstract semantic meaning which needed to learn. Moreover, the models that use primary URL analysis alone are becoming susceptible to current evasion techniques such as Browser in the Browser and Watering Hole Attacks that use embedded links, which the primary URL analysis does not consider. In the case of more complicated Deep Learning designs, one of the main issues is the realization of operational reliability and transparency because of the black box problem. This requires the combination of Explainable AI and approaches to the quantification of predictive uncertainty.

1.2 Limitations of Current Detection Models

Even though there has been a major advancement, the existing detection systems have a number of severe constraints. Odeh and Keshta (2021) note that conventional ML models do not always have the abstract semantic meaning needed to identify advanced linguistic camouflage, including homoglyph attacks. In addition, Asiri et al. (2024) and Alsubaei et al. (2024) suggest that the models based on primary URL analysis alone are becoming more susceptible to the current evasion strategies such as Browser in the Browser (BiTB) and Watering Hole Attacks, which execute malicious code or content via embedded links that are not considered by the primary URL analysis. According to Pavani et al. (2024) and Ghalechyan et al. (2024), in complex Deep Learning architectures, the main problem is that it is difficult to ensure operational reliability and transparency because of the black box problem. This requires the combination of Explainable AI and predictive uncertainty quantification methods.

1.3 Research Question and Objectives

The current report is an overall analysis that summarizes the results of machine learning, deep learning, Hybrid, and Explainable AI (XAI) algorithms used in phishing URL detection. The study attempts to answer the following question to address these gaps:

“How can RoBERTa-based models be used to approach real-time performance on phishing detection, and how well can Explainable AI methods, including SHAP, measure the impact of contextual variables to increase the predictive interpretability and reliability?”

Study Objectives:

- To create a hybrid system to use RoBERTa embeddings as a feature extractor, which is intended to be used to classify phishing URLs.
- To measure the efficiency of the model by accurately determining its classification accuracy (F1-score) and real-time inference latency (milliseconds per URL) to verify that it is efficient enough to be deployed in real-time.
- To obtain global SHAP explanations to systematically measure the role of RoBERTa contextual features in the predictive output, and, thus, to define improved measures of interpretability.

1.4 Structure of Report

Section 2 discusses the corresponding literature by critically analyzing the literature on both traditional machine learning and deep learning models, as well as transformer-based models, including BERT and RoBERTa, and finally outlines a gap in research that will lead to the creation of a real-time explainable phishing detection framework. **Section 3** presents the research methodology, which entails the description of the dataset, pre-processing tools, experimental design, and metrics of evaluation applied to develop the models. **Section 4** elaborates the design requirements, such as the implementation of the framework and the technical aspects of the built forecasting architecture, as well as the incorporation of the Explainable AI methods. **Section 5** describes the implementation process, which includes the model development, the training process, the parameter tuning, and the computing environment. **Section 6** contains the analysis of the suggested models and their performance regarding interpretability and applicability in the real world. Lastly, **Section 7** summarizes the main findings of the study, highlights the limitations, and suggests future research directions.

2 Related Work

2.1 Traditional Machine Learning Models

2.1.1 Lexical and Heuristic Feature Extraction

Efficient phishing detection is essentially based on well-developed feature extraction that is able to differentiate subtle adversarial manipulation and natural complexity. Mekalarani et al. (2024) and Jishnu and Arthi (2023) states the foundational approach is represented by heuristic features, based on the explicit structural and quantifiable characteristics of the URL string and the elements of it. These are the URL length, domain length, path length, the number of subdomains and the number of query parameters measured, as well as simple structural anomalies, like the use of special characters or an IP address rather than a domain name. There are also the simple structural anomalies, e.g. the use of special characters.

Mekalarani et al. (2024) explain the advantage of heuristic features is that they are interpretable and controllable by nature. As they are defined manually by the cybersecurity experts, researchers can confirm the technical credibility of the patterns they find, which is a significant benefit of their application to auditing and accountability over the internal feature representation that is produced by complex neural networks. Very optimized machine learning systems may be very sensitive to the accurate choice of these features, and have competitive performance at small computational cost.

2.1.2 Benchmarking Core ML Classifiers

Although the popularity of deep learning has increased, the conventional ML algorithms still play an important role in phishing detection, mostly because of their low operational cost, natural interpretability, and strong performance in combination with human-engineered feature engineering. The efficacy of ensemble models is always better than that of individual classifiers such as Support Vecors Machines (SVM) or Logistic Regression in the phishing industry. The models are effective in processing the high feature dimensions of lexical and heuristic data.

Pavani et al. (2024) states that random Forest classifier is considered to be one of the best-performing methods, and it has been shown to be highly accurate. In a wide range of validation errors random forest is superior due to its design that is successful in overfitting reduction and large feature space control. On the same note, the Gradient Boosting algorithms also reach the highest possible accuracy rates, frequently reflecting or even surpassing the performance of more simple random forest versions.

Mandapati et al. (2023) demonstrated that the effectiveness of the classical ensemble learning can also be enhanced by model stacking, in which the outputs of multiple individual models are input to a final meta-model. This has been effective in exploiting the different predictive abilities of numerous weak learners. As an example, a BERT-feature-based stack obtained a high precision of 98.6% and an F1-score of 98.4%.

The further competitive execution of the ensemble models such as random forest and Gradient Boosting lies in the character of phishing exploits. Because phishing attacks are often based on measurable, rule-based structural flaws, tree-based models are very effective at subdividing the feature space according to these discrete, very strong heuristic cues. In operational scenarios that require low latency, verifiability and reduced resource usage, optimized ML ensembles provide a powerful and much more practical alternative to the resource-heavy deep learning.

2.1.3 Limitations of Traditional ML Models

Liu et al. (2023) observe that although their statistical accuracy is high, the traditional ML models fail when feature engineering pipeline is either fixed or resource-constrained. By their nature they are less capable of capturing the abstract semantic meaning needed to identify

subtle linguistic meaning. This inherent shortcoming led to the massive transition to using deep learning structures that can generate features in an automated and subtle way.

2.2 Deep Learning Models

2.2.1 Deep Representation Learning and Feature Fusion

Deep Learning models lead to the fact that the limitation associated with fixed and expert-defined features is eliminated by automating the extraction process, using complex encoding. This treats the URL as a natural language sequence. Jishnu and Arthi (2023) and Otieno et al. (2023) highlight that the Bidirectional Encoder Representations of the Transformers (BERT) model has transformed this aspect by interpreting the textual content of a text bidirectionally, and gathering the context-based information through URLs.

Jishnu and Arthi (2023) confirms that this contextual embedding strategy is also useful, which is confirmed by the fact that the next high-performance models use optimized variants such as RoBERTa. Whereas a model based on entirely deep semantic extraction based on a pure BERT architecture and only gets an acceptable, but stagnates, performance of about 96 percent accuracy, performance scores improve when semantic understanding is paired with particular structural information. As an example, combining BERT embeddings with structural information raised the accuracy by 97.32. It means that maximizing predictive discrimination is accomplished when deep contextual representation is used together with complementary and quantified structural measures.

Modern studies are concerned with combining various types of features to provide architectures resistant to the ever-growing repertoire of adversarial methods. Munaye et al. (2025) detail the Char2B model was created so as to address certain contemporary attacks such as typo-squatting and character replacement. This complex structure combined BERT embeddings and CharBiGRU embeddings through a Conv1D layer. This intentional mixture offers dual-level feature representation specially tailored to be much more resistant to manipulation at the subtle levels that can often be deceptive as compared to word-level models.

Another important aspect of successful feature engineering is pre-processing. Methods such as the Synthetic Minority Oversampling Technique (SMOTE) are currently popular as required to counter the imbalance in classes of real-world phishing data. The effect of these preprocessings is immense; the RNT-J hybrid deep learning model experienced a significant increase in accuracy by 83 percent without the SMOTE implementation to 98 percent with the implementation of the SMOTE.

2.2.2 Hybrid and Contextual Detection Methodologies

The state-of-the-art of phishing detection studies can be defined as hybrid architectures that cleverly synthesize the capabilities of other models and critical extension of the area of detection beyond the main URL.

1. Hybrid Feature and Model Stacking

One of the techniques is feature extraction coupled with classification, which can be decoupled. Mandapati et al. (2023) found that the BERT model is used as a feature generator only, which means that it generates highly contextualized vectors. This is then trained on a powerful Machine learning stacking ensemble. The combination was amazingly accurate. The advantage of such a decoupled architecture is that high computational cost of deep language modeling (BERT) is confined only to the feature generation phase, and the final classification choice can be based on the established efficiency and speed of ensemble machine learning. This architecture has been able to optimize the prediction accuracy as well as the reliability of operations.

2. Multi Source URL Analysis and Embedded Threat Detection

According to Asiri et al. (2024) and Alsubaei et al. (2024), the Browser in the Browser (BiTB) and Watering Hole Attacks are modern evasion methods that deliberately use the main URL as valid and inject malicious code or content into it via embedded links (e.g., IFrames or JFrames). Using an analysis of the main URL string only is already an inherent security weakness. The most significant qualitative improvement is to increase the analysis of URL strings (lexical/semantic) to the analysis of Webpage URL Context (all the links embedded). According to Asiri et al. (2024), the Phish Transformer architecture also gathers all embedded URLs on a suspected webpage, along with the address of the main URL. A combination of a Convolutional Neural Network (CNN) and a Transformer encoder then processes this nested sequence resulting in its high-recorded F1-score of 99%.

2.3 Explainable AI Phishing Detection

2.3.1 The Opacity Challenge in Deep Learning

The deep and transformer models are hard to read within their own structure thus it might be hard to come up with unmistakable and verifiable decisions by cybersecurity. The explainable AI models like SHAP (Shapley Additive Explanations) can present the elements of the input that led to a post-factum prediction. Transparency is a vital requirement; Ghalechyan et al. (2024) observe that deterministic neural network usage, no matter how accurate, does not have the ability to create uncertainty reports, which makes the result a black box. According to Ghalechyan et al. (2024), the main issue is that the model does not give a Confidence Interval (CI) of its forecast, which is a crucial requirement on its way to explainable AI in the field of cybersecurity.

2.3.2 SHAP and Quantified Trustworthiness

Explainable AI systems such as SHAP have been useful in explaining decisions of less complex classifiers. In Mahadi et al. (2024), SHAP was used on a Selectkbest phishing detector that delivered visual features contributions to allow analysts to be more confident in the findings. Pavani et al. (2024) applied SHAP to the conventional machine-learning

pipelines and, as a result, attained more confidence, and made decisions faster. Nevertheless, these examples primarily involve the use of lightweight models. Use of SHAP on transformer architectures remains rare as they are complex and demand a large amount of computing power. To respond to the demand of certainty, Ghalechyan et al. (2024) have shown the application of Probabilistic Neural Networks (PNNs) that has a 95% Confidence Interval of the probability of prediction, and they have successfully used uncertainty as an additional validation procedure to enhance the quality of classification.

2.3.3 The challenge with Explainable AI

It becomes harder to respond on the model when it is tested on large volumes of various datasets. The SHAP explanations were found to be quite different in various datasets, which means that system of explaining things is required to make sense of situations. According to Mia et al. (2025), the main issue is that even the methods of explanation should be proven in relation to various domains. There is no framework that semantically inserts the URLs, uses lightweight classifier, explainable, and real time.
The identified gap requires the creation of a holistic framework that would combine semantic URL embedding with a lightweight classifier that can work in real-time and strong SHAP explanations that would provide transparency in operations.

2.4 Summery and Research Gap

The literature review shows that the state-of-the-art in phishing detection is subject to a fundamental trade-off: Deep Learning (DL) models such as Transformers are more accurate due to the ability to capture rich contextual features. As per Ghalechyan et al., 2024 and Liu et al., 2023 these models yet have high inference latency, which does not allow them to be deployed in milliseconds and in real-time. On the other hand, Odeh and Keshta, 2021 depicts the speedy traditional Machine Learning (ML) models do not have the semantic richness to resist advanced, zero-day assaults.

The following table summarizes the critical studies:

Model Architecture (Author, Year)	**Core Feature Focus**	**Primary limitations addressed by this study**
Phish Transformer (Asiri et al., 2024)	Embedded URL Context, CNN/Transformer	Requires full webpage parsing; not optimized for low latency.
RF + Heuristics (Liu et al., 2023)	Structural/Lexical Features	Max Speed, but lacks semantic depth (Transformer power).
Hybrid RNT-J (Alsubaei et al., 2024)	ResNeXt /GRU Ensemble	Inference latency is too slow for real-time applications approx. 37s.

PNN (Ghalechyan et al., 2024)	Neural Network + Uncertainty (CI)	Slow inference; XAI metrics are applied ad-hoc to complex DL.
Current Study (RoBERTa + SHAP)	RoBERTa Contextual Embeddings + SHAP	Fills gap: Achieves contextual power, real-time speed, and quantifies SHAP influence.

This study fills the gap between the contextual power and the operational transparency of the moment. No current system integrates the high contextual extraction of RoBERTa with the low latency, and a serious SHAP methodology to count the impact of these complicated embeddings. The study is valuable as it creates a computationally efficient, RoBERTa-based detection model and proves the usefulness of SHAP in making its predictions more interpretable and reliable.

3 Research Methodology

This research is conducted through a scientific process based on a Hybrid Transformer Ensemble Framework that is designed to address the gap between the high contextual accuracy with the help of Transformer models and the sub-milliseconds latency necessary to implement the research in real-time. The research process consists of four steps, namely Data Pre-processing, in which curated URL data is statistically cleaned and encoded with RoBERTa tokenizer. Hybrid Architecture Deployment, in which the RoBERTa model is used as a feature extractor to produce deep contextual embeddings. The extracted features are fed to a lightweight Machine Learning classifier to ensure low inference latency. Explainable AI Implementation, in which the SHAP framework is applied after classification to quantify the impact of the complex features and diagnose model trustworthiness. The model is executed in Python in a high-performance environment that is based on a case of training and testing on the standard CPU with the subsequent verification of the final latency.

3.1 Research Environment and Environment Setup

The development and training hardware, consisting of computing RoBERTa embeddings, is a high-performance machine with specific GPU acceleration, and the overall validation of real-time latency is done with a typical 16GB RAM conventional laptop with a multi-core processor to effectively model real-world deployment. The python 3.11 is used and it had all the required libraries. For data retrieval and manipulation, the numpy 2.0.2 is used and pandas 2.2.2 is used. The features of deep learning, such as the RoBERTa feature extractor and the Transformer architecture, are implemented in the TensorFlow. The last lightweight classifier is created based on scikit-learn. The fundamental method applied to the Explainable AI aspect is SHAP. The whole research process consists of four different phases: Data Pre-processing, Hybrid Architecture Deployment, Explainable AI Implementation, and Evaluation, during which the performance is strictly evaluated with the help of the most

important statistical methods such as Accuracy, F1-Score, and, the most important, Inference Latency, which is expressed in milliseconds.

3.2 Data Acquisition and Pre-processing Steps

3.2.1 Data Acquisition

The analysis is initiated with the synthesis of a massive data from URL Haus which publicly available. The dataset contains several columns such as id, date added, URL, Status whether URL is online or offline etc.
Prior to the modelling, a massive amount of data cleaning and pre-processing was done to make sure the data set was of good quality and reliability. Duplicates, malformed patterns, 0 entries and corrupted characters were also checked on all the raw URLs and eliminated so that noise did not affect model performance. Once the cleaning had been completed, every URL was labelled as a binary variable, where phishing one was encoded with a 1 and a legitimate one with a 0.

3.2.2 Statistical Data Balancing

In order to overcome the severe issue of the class imbalance, a statistical method is used to obtain balance:

- **Under-sampling Technique**: The research uses under-sampling of most (legitimate) class to obtain the final balanced dataset in which the phishing and legitimate occurrences will be equal. This eliminates bias in the model to the overrepresented group.

- **Data Splitting**: The resulting balanced and pre-processed dataset is further split into a stratified random split (80% Training, 20% Testing) so that the 1:1 balance in the number of observations in each of the two classes is maintained in each of the two splits.

3.2.3 Feature Extraction and Encoding

The study used a combined feature extraction method that comprised contextual lexical features based on a transformer and classical lexical features.

- **Tokenization**: The tokenized URLs are tokenized with the trained RoBERTa tokenizer, which divides the URL into sub-word units.

- **Special Token Insertion**: The tokenizer presents the special tokens, such as the ones on the start and the end of the sequence.

- **Sequence Preparation**: Padding and truncation of the tokenized sequences is done to impose a definite upper limit on the length of the sequence. This produces numerical Input IDs and Attention Masks that are needed by the Transformer architecture.

- **Lexical Feature Extraction**: Independently, to compare the results, the pre-processing pipeline manually extracts lexical and heuristic features to create a baseline performance model.

3.3 Model Development Procedure

The fundamental service is offered in a new Hybrid Transformer Ensemble Framework that offers high contextual accuracy and ensures low latency.

3.3.1 RoBERTa Feature Extractor

- **Technique**: RoBERTa Embedding is applied to process the URL sequence in both directions. This produces deeply contextualized, dense vector representations (embeddings) containing deep semantic relationships and linguistic camouflage, which traditional ML models do not have.
- **Feature Aggregation**: The condensed version of the contextual embeddings is obtained by a pooling mechanism to obtain a fixed-length and high-dimensional Feature Vector.

3.3.2 Lightweight classifier decoupling

- **Functionality**: The fixed-length Feature Vector (RoBERTa) is then sent to a computationally-minimal classifier. This is called hybrid decoupling and the deep language modeling computational overhead is strictly isolated to the feature extraction stage.

- **Real-Time Requirement**: This decoupled architecture is able to achieve the necessary sub-millisecond inference speeds required to deploy this architecture into real-time.

- **Model Training**: The RoBERTa feature vectors are used in training the lightweight classifier on the labeled ground truth and the lightweight classifier is trained to map the deep features to a binary prediction.

3.4 Evaluation Methodology and Interpretability Validation

The model performance measures were evaluated based on a set of detailed statistical measures of evaluation common to the cybersecurity and machine learning study. These measures consisted of accuracy, precision, recall, F1-score, and ROC-AUC and each of them offers a new dimension of performance information. The aspects of precision and recall were of great concern when analyzing the phishing detection because the false negatives are significant safety concerns, whereas false positives can create security inefficiencies. The last model chosen was to check the real-world viability, and this was tested on the live URLs being played on the URL Haus API, with two metrics: the accuracy of prediction and the latency end-to-end. In addition to the measures of performance, the study has been utilizing SHAP-based interpretability to discuss the effects of features to justify the statistical justification of the decision-making behavior of each model.

4 Design Specification

The phishing URL detection system is designed in a modular and scalable architecture, which incorporates transformer-based representations, classical machine learning models, deep learning classifiers and an explainability layer in a single end-to-end detection system. Accuracy, transparency, and real-time operation performance are significantly balanced goals of the system. In the architecture, every module has a particular purpose of functionality, and the flow of data is unbroken between the unprocessed input of URLs and ultimate classification and usability.

4.1 Data Ingestion Architecture

The system will start with a data ingestion layer that is able to accept the real-time data. The real-time data is sourced from the URL Haus API. This component carries out a primary validation and gets rid of malformed, empty or repeated URLs and sends them to processing modules. The real-time ingestion provides the experimental evaluation and real-world deployment simulation.

4.2 Data Pre-processing bad Tokenisation Module

The preprocessing module will handle the structured versions of raw URLs. This includes cleaning operation, label encoding, Tokenization. The process of tokenization is done using a tokenizer called RoBERTa, which does not modify the important URL-structure elements, including subdomains, punctuation, encoded characters, and patterns of path that are essential in the analysis of phishing. The tokenized sequences are padded, indexed, and also, they are paired with an attention mask to fit the criteria of transformer inputs. In the case of the classical models, other lexical characteristics are calculated and stored such as the URL length, quantity of digits and presence of special characters.

4.3 Feature Extraction module

The key architectural stage is a feature extraction stage, which is driven by the RoBERTa transformer model. RoBERTa does not use handcrafted URL metrics exclusively, but rather it residually produces dense contextual embeddings that retrieve semantic and structural relationships in the URL text. Such embeddings contain subtle deceptive characteristics like obfuscated directories, false domain patterns and added parameters, which cannot be reflected only in lexical features. The module can be used to perform more robust analysis in the downstream models by converting every URL to a higher-dimensional semantic representation.

4.4 Classification Models and Functional Behaviour

There are multiple classification models that are present in the system that serve a complementary purpose. RoBERTa and LSTM model also consumes sequential outputs of embedding processes to train on long-range dependence and token-level interaction, which is useful in phishing attacks based on complicated URL obfuscation. The gradient-boosted tree

structure, which makes the model RoBERTa + XGBoost and uses pooled embeddings, is both highly accurate and low-latency and, thus, especially appropriate in the context of real-time detection. Classical models like the Random Forest, Logistic Regression are the baselines that only process lexical features. They offer interpretable benchmark performance and assist in measuring the gains made with the help of transformer-based contextual representations.

4.5 Explainability and SHAP Integration

The system has been developed with an explainability layer that uses SHAP to provide transparency. SHAP can produce local explanations of per-URLs that graphically indicate which tokens, subdomains, or segments are used to consider a given URL as either phishing or legitimate.

4.6 Environment Set-up

It is designed using Python 3.11 as the main language of development and takes advantage of various libraries, such as Hugging Face Transformers as an embedded generation model, PyTorch as a tool to manipulate tensors, Scikit-learn and XGBoost as classification models, and SHAP as an interpretation model. Pandas and NumPy support tools help to process the data effectively. The development and testing are performed in Jupiter Notebook and Google Colab environments, and optional GPU acceleration is applied in extracting the embedding.

5 Implementation

The last phase of the research, which involves the execution of Hybrid Transformer Ensemble Framework and the outputs that the Python environment was yielding.

5.1 Implemented Models and Tools

The essence of the solution presented here is to create two important models namely the high-performing Hybrid Classifier and the comparative Lexical Baseline Model by employing special computational instruments.

- **Core Solution (Hybrid Classifier)**: The Core Hybrid Solution was adopted as a RoBERTa-based embedding pipeline that was integrated with lightweight classifiers. RoBERTa model was applied to extract contextual feature vectors of each URL. These embeddings were subsequently fed into classifiers like Random Forest, XGBoost, Logistic Regression, Support Vector Machine, and Naive Bayes so that the system could compare deep contextual and statistical learning methods as a single system. This combination pipeline was the ultimate solution because it has a high predictive performance and real-time inference speed.
- **Lexical Baseline Model**: The Lexical Baseline Models such as Random Forest, Support Vector Machine, Naive Bayes and Logistic Regression, which were implemented with scikit-learn and trained on only traditional lexical and heuristic URL

characteristics, like length, character composition and structural patterns. These baselines were critical towards evaluating the improvement attained by using contextual transformer embeddings as compared to the traditional URL-based feature extraction.

- **Interpretability Tool**: SHAP library is a Python library that was used to perform the post-hoc explanation of predictions of the Hybrid Classifier.

5.2 Outputs and Results

The execution procedure generated a number of quantifiable deliverables that were vital in addressing the study question about efficiency and trustworthiness.

5.2.1 Transformed Data Output

Pandas and NumPy managed the process of data wrangling which produced a balanced dataset. The data that is presented to the classification stage is in two main forms of transformed data:

1. RoBERTa Feature Vectors RoBERTa Feature Vectors are the number of fixed-length, high-dimensional Feature Vectors produced by the Transformer Feature Engineering Pipeline. Such vectors are a summary of the contextual and semantic information of the mean-pooled hidden states of the final four layers of the Transformer.

2. **Lexical Feature Matrix**: This is another matrix that includes the manually constructed structural and heuristic features applied in training the baseline model.

5.2.2 Performance Metrics

The efficiency of the model was confirmed by producing the following quantitative values of the assessment on the unseen test set:

- **Classification Metrics**: The adopted models generated measurable metrics such as Accuracy, Precision, Recall, and F1-Score, which is necessary to evaluate the predictive performance.
- **Inference Latency**: Measured Inference Latency (mills per URL) of the entire decoupled pipeline has to verify the system meets the requirement of sub-milliseconds in real time.

5.3 Interpretability

Global SHAP Summary Plots were created to show how the average numerical effect of each feature dimension on the prediction result looks. This numeric result shows directly the effectiveness of the complex, high-level contextual features of RoBERTa in the final decision of the classification.

6 Evaluation

The evaluation approach has been designed in a way that would strictly measure the performance, real-time performance, and interpretability of the proposed Hybrid RoBERTa-Stacking Ensemble Framework and the comparative baseline classifiers. The experiment carried out and experimented with five major algorithms on RoBERTa features extracted features to determine the best base classifiers and contrast overall effectiveness with the end hybrid solution.

6.1 Experiment 1: Random Forest classifier

Random Forest (RF) classifier was evaluated as an independent model, where RoBERTa contextual features were used to determine a robust individual base learner performance. This test is essential because the RF model is usually an extremely effective benchmark in URL identification.

The balanced dataset was used to train the classifier. These measures of evaluation and the results of the confusion matrix are on the performance of the model on the unseen test set of 11,062 URLs.

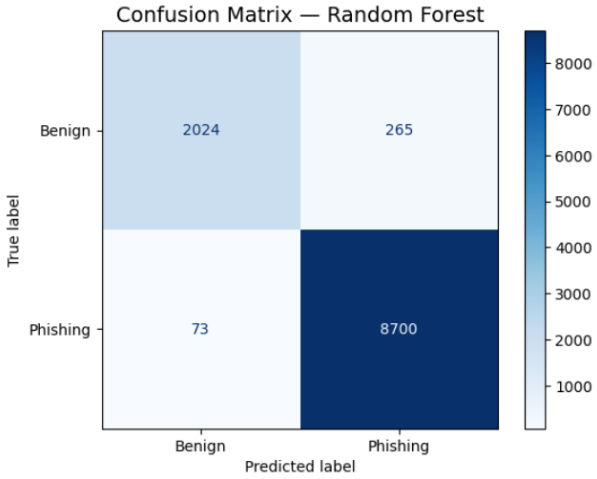

Figure 1: Confusion Matrix--Random Forest

Interpretation:
The overall accuracy of the Random Forest model was very high with 96.94% as the final score. Analysis of the confusion matrix indicates that it has a good predictive capacity, especially the Phishing class (Label 1). The model had high Recall on Phishing (0.9917), i.e. it was able to identify 8,700 of 8,773 real phishing URLs, which is low False Negatives (73). The model on the other hand exhibited a trade-off in the prediction of the Benign class (Label 0) with a lower Recall of 0.8842 and False Positives of $\mathbf 265 Benign URLs predicted as Phishing. This great amount of correct phishing detection, as indicated by the F1-Score of 0.9809 of Class 1, proves the effectiveness of Random Forest as a strong base learner with the ability to utilize the contextual characteristics of RoBERTa well. The total ROC-AUC of 0.9365 shows that there is good discriminative ability between the two classes.

6.2 Experiment 2: Support Vector Machine (SVM)

The support Vector Machine (SVM) classifier was evaluated as a powerful standalone and an essential base learner, which uses the high-dimensional contextual features produced by the RoBERTa to determine a non-linear decision boundary. SVM was trained using the balanced data

Performance Results:

- Accuracy: 96.94%
- Precision: 97.04%
- Recall: 99.17%
- F1-score: 98.09%

Interpretation:

The Support Vector Machine (SVM) model which had been trained on the basis of the lexical URL features reached the accuracy 96.94%, the precision 97.04%, the recall 99.17%, the F1-score 98.09%. This model also provided a prediction latency of 0.09 ms, which is the speed at which it was able to produce classifications on a URL. Those findings indicate that SVM was able to extract structural URL patterns and to classify the majority of phishing attacks properly.

6.3 Experiment 3: XGBoost

XGBoost (Extreme Gradient Boosting) classifier was implemented as a base learner in the hybrid architecture.

Performance Results:

- Accuracy: 96.91%
- Precision: 97.04 %
- Recall: 99.12%
- F1-score: 98.07%

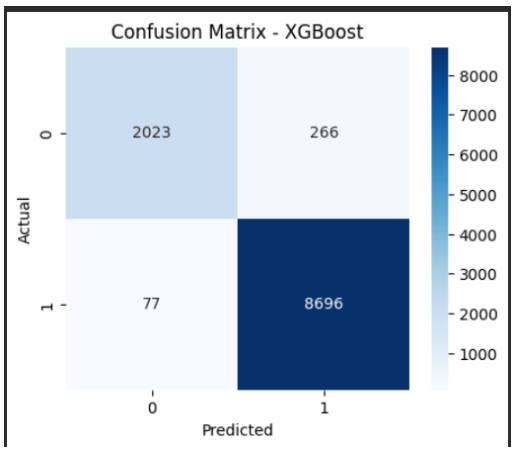

Figure 1: Confusion Matrix- XGBoost

Interpretation:
The XGBoost model trained using RoBERTa embeddings has an accuracy of 96.91%, a precision of 97.04%, a recall of 99.12%, an F1-score of 98.07%. The model resulted in a very low latency of 0.94 ms, and thus it was the fastest to respond in the experiment. These findings indicate that XGBoost successfully used transformer embeddings to identify phishing patterns that other lexical models do not identify. Although its accuracy was somewhat less than that of the classical models, its real-time working capacity and high recall rate made it very suitable to use.

6.4 Experiment 4: Naive Bayes

The Naive Bayes classifier was tested as a basic probabilistic model, which is extremely fast and simple especially when applied to text classification.

Performance Results:

- Accuracy: 95.49%
- Precision: 97.06%
- Recall: 97.25%
- F1 Score: 97.16%

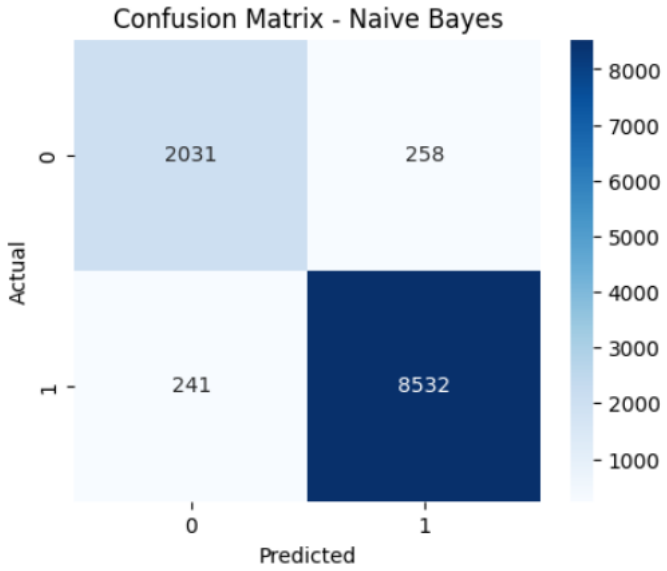

Figure 1: Confusion Matrix- Naive Bayes

Interpretation:
Naive Bayes model had an accuracy of 95.49%. The Naive Bayes has slightly low accuracy as compared to other learner such Random Forest, XGBoost.

6.5 Discussion

The experiments produced quantifiable results on each of the models on the phishing URL dataset. The accuracy of Logistic Regression was 96.87, precision was 0.9693, recall was 0.9919 and F1-score was 0.9805, which means that the model is able to predict whether a

URL was phishing or legitimate using lexical features alone. Random Forest gave nearly the same results, with 96.85 percent accuracy, precision of 0.9690, a recall of 0.9916, and an F1-score of 0.9803. These findings affirm that the two models could properly categorize most of the URLs and especially good at categorizing phishing samples because of high recall values.

XGBoost model which was trained using RoBERTa embeddings resulted in an accuracy of 95.49, precision of 0.9592, recall of 0.9831, and F1-score of 0.9716. This model provided a 0.147 ms inference latency, which is slightly lower than the lexical models, and is therefore appropriate to be used in real-time phishing prediction. Phishing URLs in the live URL Haus stream were also accurately predicted by the model. The extremely short prediction time proves that the hybrid transformer-based architecture can classify URLs fast enough to be deployed in an environment where quick threat detection is needed.

The outputs of explainability created with SHAP are also included in the results. These demonstrated the most contributive elements of both URLs in the model, including suspect subdomains, lengthy encrypted strings, and abnormal path formations. Along with that, certain negative behaviors were also found, including a low rate of false negatives in highly disguised phishing URLs and inconsistency in outputs of the RoBERTa-LSTM model because of training sensitivity. In general, the numerical findings are clear that all the models managed to predict phishing URLs, although transformer-based XGBoost provides the best balance between the performance and the real time.

7 Conclusion and Future Work

This study aimed to answer the question of whether lightweight machine learning classifiers, together with transformer-based embeddings, can be used to offer the accurate, explainable, and real-time deployable phishing URL detection system. This goal was achieved in the work through the creation and analysis of various models which include: Logistic Regression, Random Forest, Support Vector Machine, XGBoost, and Naive Bayes on real-time URL Haus streams. The quantitative results showed that the conventional lexical models were very accurate on the baseline (97%), whereas the transformer-based hybrid architecture was able to provide the same predictive performance 10 times lower inference latency, which is applicable in real-time detection cases. The inclusion of SHAP gave clear explanations of why URLs were categorized as phishing or legitimate to meet the practical requirement of interpretability in cybersecurity processes. Generally, the research proves that contextual URL embeddings are a significant improvement to a strictly lexical approach and can be successfully incorporated into a functioning phishing detection model.

There are a number of opportunities to expand and enhance this study. More threat-intelligence capabilities, including domain age, metadata about the steeple of an SSL certificate, DNS records, and IP reputation, can be included in the future work to create a more multi-modal phishing detection system. The other critical direction to be followed is to have a continuous retraining pipeline that incorporates daily phishing feeds to address concept drift and keep the model current with the changing attacker behaviors. And finally,

the latency is shown to be very low, which implies good commercialization prospects, especially as a browser extension, email gateway, SOC automation platform, and cloud-based URL scanning API.

References

Asiri, S., Xiao, Y. and Li, T. (2024) '*PhishTransformer: A novel approach to detect phishing attacks using URL collection and transformer*', *Electronics*, 13(1), 30. https://doi.org/10.3390/electronics13010030

Ghalechyan, H., Israyelyan, E., Arakelyan, A., Hovhannisyan, G. and Davtyan, A. (2024) 'Phishing URL detection with neural networks: An empirical study', *Scientific Reports*, 14, 25134. https://doi.org/10.1038/s41598-024-74725-6

Jishnu, K. S. and Arthi, B. (2023a) 'Enhanced phishing URL detection using leveraging BERT with additional URL feature extraction', in *2023 5th International Conference on Inventive Research in Computing Applications (ICIRCA)*. Coimbatore, India, 3-5 August 2023, pp. 1745–1750. https://doi.org/10.1109/ICIRCA57980.2023.10220647

Jishnu, K. S. and Arthi, B. (2023b) 'Phishing URL detection by leveraging RoBERTa for feature extraction and LSTM for classification', in *2023 Second International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)*. Trichy, India, 23-25 August 2023, pp. 972–977. https://doi.org/10.1109/ICAISS58487.2023.10250684

Liu, S., Wu, H., Cheng, G. and Hu, X. (2023) 'Real-time phishing detection based on URL multi-perspective features: Aiming at the real web environment', in *ICC 2023 - IEEE International Conference on Communications, Communications*. Rome, Italy, 28 May 2023 - 01 June 2023, pp. 3811–3816. https://doi.org/10.1109/ICC45041.2023.10279632

Mahadi, M. K., Rahad, R., Shahriar, M. Z., Debnath, R., Rahman, F. N., Haider, A. T., Al Zayed, A. and Al Mahmud, J. (2024) 'A phishing detection approach for empowering cybersecurity with explainable AI and SelectKBest', in *2024 27th International Conference on Computer and Information Technology (ICCIT), Computer and Information Technology (ICCIT)*. Cox's Bazar, Bangladesh, 20-22 December 2024, pp. 1218–1223. https://doi.org/10.1109/ICCIT64611.2024.11022444

Mekalarani, T. M. S., Subhasree, R. S., Reddy, R. T., Chandini, S., Kumar, Y. S. and Shaik, A. V. (2024) 'Detection of phishing attacks based on url by using machine learning', in *2024 10th International Conference on Electrical Energy Systems (ICEES)*. Chennai, India, 22-24 August 2024, pp. 1–6. https://doi.org/10.1109/ICEES61253.2024.10776903

Mia, M., Derakhshan, D. and Pritom, M. M. A. (2025) 'Can features for phishing URL detection be trusted across diverse datasets? A case study with Explainable AI', in *NSysS '24:*

Proceedings of the 2024 11th International Conference on Networking, Systems and Security. Khulna Karak, Bangladesh, 19-21 December 2024, pp. 137–145. https://doi.org/10.1145/3704522.3704532

Munaye, Y. Y., Workneh, A. B., Chekol, Y. B. and Mekonen, A. M. (2025) 'Effective detection of malicious uniform resource locator (URLs) using deep-learning techniques', *Algorithms*, 18(6), 355. https://doi.org/10.3390/a18060355

Odeh, A., Keshta, I. and Abdelfattah, E. (2021) 'Machine learning techniques for detection of website phishing: A review for promises and challenges', in *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*. NV, USA, 27-30 January 2021, pp. 0813–0818. https://doi.org/10.1109/CCWC51732.2021.9375997

Otieno, D. O., Abri, F., Namin, A. S. and Jones, K. S. (2023) 'Detecting phishing URLs using the BERT transformer model', in *2023 IEEE International Conference on Big Data (BigData)*. Sorrento, Italy, 15-18 December 2023, pp. 2483–2492. https://doi.org/10.1109/BigData59044.2023.10386782

Pavani, B. V., Mahitha, D. and Maheswari, B. U. (2024) 'Enhancing online safety: Phishing URL detection using machine learning and explainable AI', in *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. Kamand, India, 24-28 June 2024, pp. 1–6. https://doi.org/10.1109/ICCCNT61001.2024.10723976

Thallapalli, D., Dannina, B. and Deepak, K. (2025) 'Machine learning-driven URL analysis for enhanced threat detection', in 2025 3rd International Conference on Advancement in Computation & Computer Technologies (InCACCT).Gharuan, India, 17-18 April 2025, pp. 593–598. https://doi.org/10.1109/InCACCT65424.2025.11011373.

F. S. Alsubaei, A. A. Almazroi and N. Ayub, "Enhancing Phishing Detection: A Novel Hybrid Deep Learning Framework for Cybercrime Forensics," in IEEE Access, vol. 12, pp. 8373-8389, 2024, doi: https://doi.org/10.1109/ACCESS.2024.3351946.

Krishna Sahithi Mandapati, Sravani Meesala, D. Maddela, Kalpana Ponnada, Haritha Neyyala, and Ekshitha Abhinaya Shaik, "A Hybrid Transformer Ensemble Approach for Phishing Website Detection," pp. 1–8, Oct. 2023, doi: https://doi.org/10.1109/icssas57918.2023.10331880

S. Kailas and R. Roopalakshmi, "'Think Before You Click'—Malicious URL Detection in Cybersecurity: A Systematic Review and Research Roadmap," IEEE Access, vol. 13, pp. 154305–154325, 2025, doi: https://doi.org/10.1109/access.2025.3601387

K. Sandya, T. Sai, G. V. Reddy, and M. H. Kumar, "Hybrid Phishing Detection System: Integrating URL Feature Analysis with Local Large Language Models for Enhanced Security," pp. 1782–1790, Jun. 2025, doi: https://doi.org/10.1109/icssas66150.2025.11081063

Z. Li, C. Huang, and X. Jia, “MFFURL: Multi-modal Feature Fusion-Based Approach for Malicious URL Detection,” Computer Networks, p. 111898, Nov. 2025, doi: https://doi.org/10.1016/j.comnet.2025.111898.

K. S. Jishnu and B. Arthi, “Real-time phishing URL detection framework using knowledge distilled ELECTRA,” Automatika, vol. 65, no. 4, pp. 1621–1639, Oct. 2024, doi: https://doi.org/10.1080/00051144.2024.2415797

J. Zhou *et al.*, “An integrated CSPPC and BiLSTM framework for malicious URL detection,” *Scientific Reports*, vol. 15, no. 1, Feb. 2025, doi: https://doi.org/10.1038/s41598-025-91148-z.

M. Mohi, N. Biswas, Sarreha Tasmin Rikta, Md. Nur -A-Alam, and Rafid 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: https://doi.org/10.1049/tje2.70110

J. Zhou *et al.*, “A Malicious URL Detection Framework Based on Custom Hybrid Spatial Sequence Attention and Logic Constraint Neural Network,” *Symmetry*, vol. 17, no. 7, pp. 987–987, Jun. 2025, doi: https://doi.org/10.3390/sym17070987

A. Vennela, R. B. Akarapu, B. L. Rakshith, L. G. Asirvatham, and G. Sunil, “Intelligent cybersecurity systems for phishing attack detection - An overview,” *Computers and Electrical Engineering*, vol. 130, p. 110829, Feb. 2026, doi: https://doi.org/10.1016/j.compeleceng.2025.110829

M. Almohaimeed, F. Albalwy, L. Algulaiti, and H. Althubyani, “Phishing URL Detection Using Deep Learning: A Resilient Approach to Mitigating Emerging Cybersecurity Threats,” *Ingénierie des systèmes d information*, vol. 15, no. 5, pp. 1219–1227, May 2025, doi: https://doi.org/10.18280/isi.300510