Computer and Decision Making
An International Journal

Journal homepage: www.comdem.org
eISSN: 3008-1416

Generalized Explainable AI Framework for Phishing Detection on Heterogeneous Textual Data

Lea Mansour[1], Nour Hilal[1], Nadine Abbas[1,*], Seifedine Kadry[1]

[1] Department of Computer Science and Mathematics, Lebanese American University, Beirut, Lebanon

ARTICLE INFO

Article history:
Received 5 July 2025
Received in revised form 1 October 2025
Accepted 18 October 2025
Available online 3 July 2026

Keywords:
phishing detection; machine learning; deep learning; explainable AI; binary classification; heterogeneous text

ABSTRACT

Phishing remains one of the most pervasive cybersecurity threats, exploiting human and technical vulnerabilities and targeting users through deceptive Emails, URLs, and SMS messages. Artificial Intelligence (AI) and Machine Learning (ML) techniques have been widely used to improve phishing detection accuracy. However, most existing studies have focused on specific data types, thereby limiting the scope of their applicability, and lacking a generalized framework integrating heterogeneous data sources within phishing context. In this study, we propose a generalized phishing detection framework that leverages classical machine learning (Random Forest and Logistic Regression) and deep learning (Convolutional Neural Network) to identify phishing attempts across heterogeneous textual data, such as Emails, URLs, and SMS messages. Moreover, we integrate interpretability into model decisions using Explainable AI, particularly SHapley Additive exPlanations (SHAP), to enhance transparency and trustworthiness. The framework is evaluated based on both predictive performance and inference efficiency. Experimental results show that Random Forest achieves the highest accuracy (93%) and F1-score (85%), highlighting the efficiency of the classifier on tabular data for the binary classification task at hand, while SHAP local and global explanations reveal semantically relevant features influencing model decisions, where words such as “admin” and “login” are identified as strong phishing indicators. These results demonstrate the promise of our unified, interpretable approach in advancing adaptive and trustworthy generalized phishing detection systems.

1. Introduction

As cybersecurity aims to safeguard digital infrastructures through measures such as encryption, firewalls, and threat detection systems [1], researchers have increasingly applied Artificial Intelligence (AI) and Machine Learning (ML) to strengthen these defenses. Many works in the literature have leveraged AI-based defenses such as in [2–4] and machine learning based techniques to improve phishing detection. For instance, the works in [5–9] addressed email phishing detection by applying various machine learning and deep learning (DL) models to preprocess and classify email content, achieving high accuracy with methods like Random Forest, SVM, and neural networks.

[*] *Corresponding author.*
E-mail address: nadine.abbas@lau.edu.lb

https://doi.org/10.59543/comdem.v3i.18329

Some studies also incorporated multilingual detection and adaptive learning to enhance performance across different languages. The authors in [10–13], addressed phishing detection for URLs by extracting features like domain age, URL length, and suspicious characters, and applying models such as Decision Trees, SVM, XGBoost, KNN, and Naive Bayes, with SVM often showing the best detection performance. SMS phishing detection was investigated in [14–17] by analyzing features such as word frequency, suspicious keywords, sender information, and message patterns, with studies comparing machine learning models like SVM, Naive Bayes, Random Forest, KNN, and Logistic Regression, where Random Forest generally achieved the best performance. However, most existing studies typically focus on a single type of input data, and rely on black-box machine learning or deep learning models that offer little to no interpretability into how predictions are made. This limits both their applicability across different real-world scenarios and their trustworthiness in critical domains like cybersecurity. Moreover, in high-stakes domains like cybersecurity, the inability to explain decisions can undermine user trust and hinder adoption. Thus, there is a pressing need for a unified and interpretable phishing detection framework that performs robustly across heterogeneous text data while offering insight into its internal decision-making process, enabling practitioners to implement reliable defenses and researchers to refine methods that better address evolving phishing strategies.

Unlike existing studies that focus on a particular data type or lack scalability and interpretability, we propose a comprehensive phishing detection framework capable of robustly handling diverse textual inputs. Moreover, our framework incorporates interpretable Explainable AI (XAI) mechanisms to provide transparent insights into its decision-making process, addressing both generalization and explainability gaps identified in the literature. As such, this paper makes the following key contributions:

- We propose a generalized phishing detection framework that integrates and evaluates heterogeneous textual data sources (emails, SMS messages, and URLs), extending beyond the limited scope of prior studies.
- We implement and benchmark traditional ML models such as Random Forest and Logistic Regression, against DL CNN model to assess their performance under different splits and balancing techniques.
- We leverage SHAP (SHapley Additive exPlanations) to provide interpretable insights into model predictions, identifying key features contributing to phishing classification, and enhancing transparency and trust.
- We introduce inference time as an evaluation metric, highlighting the trade-off between detection accuracy and latency in real-time phishing detection systems.
- We ensure reproductibility and interactivity through a publicly available Google Colab notebook and a PowerBI dashboard summarizing phishing data insights.

The rest of the paper is organized as follows: Section 2 reviews the related work; Section 3 describes our proposed framework, including the methodology, data processing steps, and model architectures; Section 4 presents the performance results and analysis; Section 5 discusses key research insights and limitations; and Section 6 concludes the paper.

2. Related Work

Many studies in the literature have leveraged AI-based defenses, including machine learning techniques, to enhance phishing detection across different communication channels such as emails, URLs, and SMS. In addition, the integration of explainable AI (XAI) has gained increasing attention to improve transparency and trust in phishing detection systems.

2.1 AI-Based Defenses

AI has been integrated within the field of cybersecurity to enhance threat detection, automate response mechanisms, identify vulnerabilities in real time, and adaptively defend against evolving attack patterns through machine learning and natural language processing. [2] proposed a machine learning-based intrusion detection system for Wireless Sensor Networks (WSNs) integrated with IoT, addressing their vulnerability to various attacks due to resource constraints and dense connectivity. It enhances detection accuracy by combining Support Vector Machine (SVM) with Stochastic Gradient Descent (SGD) and introduces context awareness to improve system performance. The detected network risks are categorized using a VG-IDS model, and the proposed WSN-DS algorithm outperforms existing methods, achieving high accuracy, recall, and F1-measure scores. In addition to that, [3] proposed an intrusion detection approach for securing Software-Defined Networking (SDN) environments using a combination of Graph Convolutional Networks (GCN) and Deep Reinforcement Learning (DRL). Aiming to address the vulnerabilities of SDN to unauthorized access and network disruptions, the method leverages the NSL-KDD dataset to train and evaluate its performance. Moreover, authors of [4] focused on evaluating the performance of the Advanced Encryption Standard (AES) to guide network designers in selecting appropriate security solutions. Key performance indicators considered include encryption and decryption time, CPU usage, throughput, latency, and energy consumption across various platforms. The study highlights challenges such as key management, security-performance trade-offs, and platform heterogeneity. In adddition, [18] investigated a hybrid ensemble model for detecting and predicting anomalies in cybersecurity datasets, comparing their performance using metrics such as accuracy, MAE, and MSE.

2.2 Email Phishing Detection

Various papers on the intersection of phishing detection and AI have been explored; one scope under this field focused on machine learning implementations for email input, where raw email content, including subject lines, headers, body text, and attachments, is preprocessed and transformed into structured features that enable the model to distinguish between legitimate and phishing. [5] and [6] explored solely machine learning algorithms for phishing detection, mainly Decision Tree, KNN, Logistic Regression, Naive Bayes, Random Forest, AdaBoost, and SVM. On the other hand, [7] explored both machine learning and deep learning models for the application, such as Neural Network, BART, and CART. [8] delved deeper into deep learning models for phishing detection, as the authors experimented with Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), and tranformers. Considering email text input, Random Forest and SVM models generally showed highest accuracies for phishing detection. [9] also presented an adaptive intelligent learning approach for multilingual spam and phishing email detection, utilizing a visual anti-spam model. The proposed system operates in three phases and employs two Naive Bayes classifiers—one to identify the email's language (Arabic, English, or Chinese) and another to classify the email as phishing or legitimate based on that language.

2.3 URL Phishing Detection

Other work in this field covered phishing detection on URLs, where features such as domain age, URL length, presence of suspicious characters, and redirection patterns are extracted to train models on identifying deceptive or malicious web links. [10] and [11] investigated machine learning techniques for that, comparing their performance for different language preprocessing techniques in [10] and for tuned hyperparameters in [11]. The models include Decision Trees, SVM, XGBoost, KNN, Random

Forest, Naive Bayes, etc. In addition to the numerous machine learning algorithms, [12] experimented with a neural network architecture for detection on URLs, and [13] experimented with a simple neural network-a multi-layer perceptron (MLP). For URL text input, SVM model performed the best in terms of detection phishing attacks.

2.4 SMS Phishing Detection

The remaining input of textual format explored is SMS messages, where concise and often informal text is analyzed for features such as word frequency, presence of suspicious keywords, sender information, and message patterns. Smishing detection has been explored by [14] experimented with machine learning algorithms only, comparing the performance of 4 models (SVM, Naive Bayes, Random Forest, and KNN). Besides the models mentioned, [15] also experimented with Logistic Regression model and compared performance for different vectorization techniques. [16] explored a larger number of machine learning models, comparing a total of 8 models, yielding a best accuracy and F1 score when using Random Forest model. Meanwhile, the authors of [17] only investigated deep learning models for smishing detection, namely CNNs, RNNs, and SNNs, where CNN architecture was found to be perform the best on their chosen dataset. For SMS input data, Random Forest classifier outperformed other ML models, while CNN classifier outperformed other DL models.

2.5 XAI Integration in Phishing Detection

The integration of XAI techniques has been addressed in literature to offer black-box models the advantage of explainability and interpretability, which not restricted to the field of phishing detection, but within this field used mainly under URL phishing. [19] proposed an explainable AI driven approach for identifying phishing websites based on features extracted from URLs. It not only achieves high detection accuracy (for Random Forest and SVM ML models) but also provides interpretable insights into which URL characteristics contribute most to the phishing classification, using LIME and EBM (Explainable Boosting Machine) as XAI techniques. [20] also explored the same topic (URL phishing detection) and proposed a lightweight Gradient Boosting Machine (GBM) model for accurately detecting phishing URLs, incorporating SHAP to provide transparent, feature-level insights into model decisions. Similary, [21] introduced a multi-level stacking ensemble model designed to improve the accuracy and interpretability of phishing URL detection. The ensemble comprises base classifiers—Naive Bayes, SVM, and Decision Tree—whose predictions are aggregated by a Logistic Regression meta-classifier, with an explainable layer using SHAP. [22] also investigated machine and deep learning models for the detection layer of such systems, comparing Naive Bayes, Logistic Regression, MLP, and SVM, then building an interpretable layer on top of LR model using LIME.

As presented in Table 1, existing literature on phishing detection varies based on the type of input data, the use of machine learning or deep learning techniques, and the integration of XAI methods. While prior studies demonstrate high accuracy on specific text types, none provide a unified framework across modalities with integrated explainability. Furthermore, the combination of SHAP with CNNs has not been explored, particularly in real-time phishing detection scenarios. To address these gaps, this work examines both traditional machine learning models such as Random Forest and Logistic Regression, as well as deep learning CNN classification models for phishing detection on generalized heterogeneous textual input, with XAI integration to provide interpretable insights into model decisions.

Table 1: Comparison of the state-of-the-art approaches

	Data types			Techniques		
Ref	Email Phishing	URL Phishing	SMS Phishing	Machine Learning	Deep Learning	XAI Techniques
[2]				✓		
[3]					✓	
[18]				✓	✓	
[5] [6] [9]	✓			✓		
[7] [8]	✓			✓	✓	
[10] [11]		✓		✓		
[12] [13]		✓		✓	✓	
[14] [15] [16]			✓	✓		
[17]			✓		✓	
[19] [20] [21]		✓		✓		✓
[22]		✓		✓	✓	✓
Proposed	✓	✓	✓	✓	✓	✓

3. Methodology

In our work, we propose a generalized phishing detection framework where heterogeneous input data are collected (emails, URLs, SMS messages), preprocessed, and used to train different machine learning and deep learning algorithms. As illustrated in Figure 1, the process begins with collecting and merging datasets from these sources as detailed in Subsection 3.1, followed by comprehensive preprocessing steps such as handling missing values, removing outliers, and addressing class imbalance as described in Subsection 3.2. Labels are then encoded, and the data is split into training and testing sets. Feature engineering is adapted to the model type where traditional machine learning models (Random Forest and Logistic Regression) use TF-IDF vectorization, while the deep learning model (CNN) uses tokenized text sequences enriched with pre-trained word embeddings as presented in Subsection 3.3. Model training is performed using scikit-learn for the machine learning models and Keras for the CNN, with hyperparameter tuning to optimize performance as discussed in Subsection 3.4. To interpret model decisions, SHAP is applied to both traditional and deep models, as detailed in Subsection 3.5, offering transparency through feature attribution. Finally, model performance is evaluated using standard metrics such as accuracy, precision, recall, F1-score, and ROC-AUC, ensuring both effectiveness and interpretability of the classification models in Section 4.

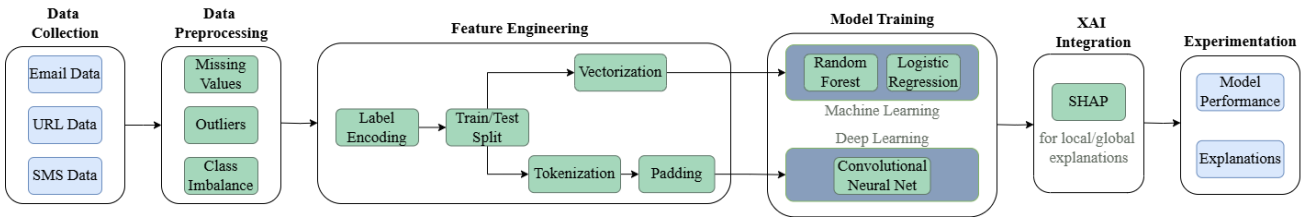

Figure 1: Overall framework for generalized phishing detection

3.1 Data Collection

In order to generate a comprehensive text input dataset with labels, 3 different datasets were retrieved from Kaggle as csv files and combined: a dataset for emails with their labels [23], a dataset for URLs with their labels [24], and a dataset for messages with their labels [25]. Tables 2, 3, and 4

present an overview of each dataset:

Table 2: Sample entries from phishing emails dataset

Email Text	Email Type
Please update your account information immediately to avoid suspension.	Phishing Email
Thank you for your recent payment. Your account is now up to date.	Safe Email
Urgent: Confirm your password to prevent unauthorized access.	Phishing Email

Table 3: Sample entries from phishing URLs dataset

URL	Label
www.dghjdgf.com/paypal.co.uk/cycgi-bin/webscrcmd=_home-customer&nav=1/loading.php	bad
mail.printakid.com/www.online.americanexpress.com/index.html	bad

Table 4: Sample entries from phishing messages dataset

text	category	label
Warning: Unusual login attempt detected on your Amazon. Verify your identity.	Urgency	Phishing
Urgent! Your Google has been compromised. Click here to secure it now!	Urgency	Phishing

The emails dataset [23] is comprised of 18,650 entries of varying lengths (with a mean length of 197 words per email and standard deviation of 175 words), with each entry labeled as either 'Phishing Email' or 'Safe Email'. The number of entries labeled as phishing (7,328 entries) is greater than that for non-phishing (11,322 entries). The URLs dataset [24] is comprised of 549,346 entries of varying lengths (with a mean length of 51 characters per URL and a standard deviation of 45 characters), and each entry is labeled either 'good' (not phishing) or 'bad' (phishing). The number of entries labeled as phishing (156,422 entries) is less than those labeled as not phishing (392,924 entries). Finally, for the messages dataset [25], the total number of entries is 1,000 entries of different lengths (with a mean length of 13 words per message and a standard deviation of words characters) labeled as 'Phishing'. The messages dataset also includes a column on the category of the message, i.e, what tone it uses to lure victims (urgency, authority, or persuasion). The length of each text input for each of the 3 datasets is computed and added as a new column by applying the 'len()' function on the corresponding text column.

3.2 Data Preprocessing

Data preprocessing is mainly divided into 4 steps: handling missing values, outliers, class imbalance, and merging the 3 datasets. First, we inspect if any of the datasets have missing values whereas the other datasets have none; it was found that the emails dataset contains 16 missing text values. Since we are unable to identify what the actual text is, these 16 rows are dropped. Second, outliers were detected when computing the mean of lengths, where abnormally large value was recorded for

the emails dataset. Thus, outliers were handled using IQR (inter-quantile range), where the first quartile (Q1) and third quartile (Q3) are calculated to determine the spread of the middle of the data. The IQR is then computed as the difference between Q3 and Q1. Based on this range, lower and upper bounds are established ($Q1 - 1.5 \cdot IQR$ and $Q3 + 1.5 \cdot IQR$, respectively), beyond which data points are considered outliers. The dataset is then filtered to retain only those entries with length values within these bounds, effectively removing the extreme values. The datasets are then combined to include 4 columns in the final version: ['text', 'type', 'length', 'label']. Finally, class imbalance is handled by upsampling the minority class (phishing class) to have an equal number of entries for both classes (392,238 entries for each).

3.3 *Feature Engineering*

In the first step of feature engineering, 'LabelEncoder' is used to convert categorical labels in the 'label' column into numeric form ('phishing' encoded to 1 and 'not phishing' encoded to 0) and is added as a new column to the dataset called 'label_encoded'. This transformation is necessary because machine and deep learning models require input labels to be in numerical format for training and evaluation. The data is now ready to be model-specific processed and takes the format shown in Table 5:

Table 5: Sample entries from combined dataset

text	type	length	label	label_encoded
hi , paliourg , = = > 78 % . . . (owbsyqlb 9765)	email	16	phishing	1
Special access granted! Login now to receive your premium service before it's too late.	sms	14	phishing	1
youtube.com/watch?v=XTMY_lD_5Ws	url	31	not phishing	0

The second step is splitting the data into training and testing sets, where two different splits are performed: one with 80% of the data used for training and 20% for testing, and another with a 70-30 training-testing ratio, allowing comparison of model performance under different data splits. After that, the data should be prepared to be inputted into two types of models (ML and DL models). Thus, two processing steps are executed (independently):

- ***Vectorization***: Input processing step necessary to convert raw text into numerical features that can be understood by machine learning models. TF-IDF (Term Frequency-Inverse Document Frequency) vectorization is used, which transforms the input into a vector within which each element represents the importance of a word, while also reducing the influence of less informative words. An example can be seen below (Figure 2) where the vectorization indicates the relative importance of each word in the input text. The values (0.467562, 0.592508, and 0.655988) represent the TF-IDF scores for the words "click", "immediately", and "password," showing that "password" has the highest importance, followed by "immediately," and "click," based on their occurrence in this sentence relative to the entire data the vectorizer was fit on.

```
Text input:  Click here to reset your password immediately.

Non-zero values in TF-IDF Vectorization:
      click  immediately  password
0  0.467562     0.592508  0.655988
```

Figure 2: TF-IDF vectorization of sample input text

- ***Tokenization***: Input processing step necessary to convert raw text into numerical data that can be understood by deep learning models. That is done by splitting the input into tokens and assigning each token a unique integer, thus sentences are converted into sequences of integers based on the built vocabulary from the training data the tokenizer is fit on. In addition to that, sequences are padded or truncated to a maximum length since neural networks require inputs of fixed length. Figure 3 presents the result of tokenizing the same input sentence, where each word is converted into an integer based on the tokenizer's learned vocabulary. These numbers are unique IDs assigned to the words, reflecting their positions in the tokenizer's word index.

```
Text input:  Click here to reset your password immediately.

Tokenized Sequence: [[160, 114, 5, 3676, 23, 1928, 1199]]
```

Figure 3: Tokenization of sample input text

3.4 Model Architecture and Training

The machine learning models chosen for investigation are Random Forest and Logistic Regression. Random Forest model builds a collection of decision trees during training and outputs the mode of their predictions for classification. Each tree is trained on a random subset of the data and features, which helps reduce overfitting and improves generalization [26]. Logistic Regression model is used for binary classification tasks, where the goal is to predict one of two possible outcomes. It estimates the probability that a given input belongs to a particular class using the logistic (sigmoid) function, which outputs values between 0 and 1. The model learns by fitting the best set of weights to the input features to minimize the difference between predicted and actual labels [27]. The choice of classical models is limited to the two mentioned since they are the most common classical models in phishing detection research, capturing both linear and non-linear relationships, while having efficient training and testing, as well as avoiding redundancy in terms of comparative analysis.

The deep learning model chosen for investigation is Convolutional Neural Network (CNN). CNNs apply convolutional filters over sequences of word embeddings to capture local features like n-grams or phrase patterns. By using max pooling, CNNs can identify the most important features regardless of their position in the text. Although originally designed for images, CNNs are effective in extracting spatial hierarchies in text, especially for fixed-length input [28]. CNN model is chosen for investigation since LSTM models can be slower to train, and transformers like BERT demand significantly more computational resources and larger datasets to generalize effectively. For all models, the text is transformed either using vectorization (for ML models) or tokenization (for DL models), thus the features inputted to the models are the vectorization scores or the tokenized sequences. ML and DL models parameters are detailed in Table 6.

3.5 XAI Integration

Explainability in AI refers to the ability to understand and interpret how a model makes its predictions or decisions. A model is explainable when its decision-making process can be understood and interpreted by humans in a clear and transparent manner. In the context of phishing detection, LR is highly explainable because its coefficients clearly indicate how each feature influences the likelihood of phishing. RF is partially explainable, as it provides feature importance scores, but its nature can make it complex to interpret. CNNs are less explainable due to their complex, non-linear structures, and are generally considered black-box models in phishing detection.

There are two different scopes for explanations, local and global. A technique offers local explanation if it has a narrow scope (specific to one instance) and explains why a specific decision was made. A

Table 6: ML and DL models parameters

Model	Hyperparameters
Random Forest	n_estimators=50 random_state=42 n_jobs=-1
Logistic Regression	max_iter=1000 random_state=42 solver='lbfgs' penalty='l2' C=1.0
CNN	embedding_dim=128 conv_filters=128 kernel_size=5 pooling='max' dropout_rate=0.5 dense_units=64 activation_output='sigmoid'
	activation_hidden='relu' output_units=2 loss_function='binary_crossentropy' batch_size=64 epochs=5 optimizer='adam' learning_rate=0.001 hidden_layers=2 (Conv1D + Dense layer)

technique offers global explanation if it has a broad scope (applies to all inputs) and explains how the model generally works. For this work, SHAP [29] will be used for explanations: the technique answers the question "how much did each feature contribute to this prediction?". SHAP technique works for both local and global explanations, and is based on cooperative game theory, specifically the concept of Shapley values. As illustrated in Figure 4, SHAP takes a set of features as input to compute their contribution to prediction, the contribution being the Shapley value. It calculates it by averaging its marginal impact across all possible subsets of features. This allows SHAP to provide both local (individual prediction) and global (overall model behavior) interpretability.

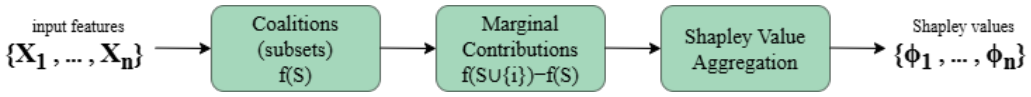

Figure 4: SHAP inputs, outputs, and internal processes for explanations

4. Performance Results and Analysis

We conduct three main experiments to (1) evaluate the performance of our proposed framework for different train-test splits in terms of accuracy (Eq. (1)), precision (Eq. (2)), recall (Eq. (3)), F1-score (Eq. (4)), confusion matrix (Eq. (5)), ROC, statistical significance and inference time to validate

its efficiency and suitability for real-time applications, (2) apply different data balancing techniques, specifically SMOTE and undersampling, to evaluate their impact on model performance and robustness against class imbalance in phishing datasets, and (3) examine model explainability, using SHAP aiming at uncovering the key features influencing predictions, in order to uncover which features and patterns in phishing emails, URLs, and SMS messages most influence the model's predictions, facilitating the identification of phishing tactics.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{4}$$

$$Confusion\ Matrix = \begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix} \tag{5}$$

where TN (True Negatives), TP (True Positives), FN (False Negatives), and FP (False Positives) correspond to the counts of the instances in the confusion matrix, which shows how well the classifier performs in predicting both the positive and negative classes.

4.1 Performance Evaluation

For the first experiment, we compare the performance of the three models (RF, LR, and CNN) for different train-test splits (80-20% and 70-30%) in terms of accuracy, precision, recall, F1-score, ROC, statistical significance and inference time. Tables 7 and 8 show that in both cases, Random Forest consistently outperformed the other models, achieving the highest accuracy (93%) and demonstrating strong precision, recall, and F1 scores for both classes. Logistic Regression followed closely behind, with a stable accuracy of 91% across both splits. However, its recall for class 1 (the minority class) remained lower than that of Random Forest, indicating slightly weaker performance in detecting positive cases. The CNN model lagged behind, especially in classifying the minority class, where it achieved lower recall values (0.56 for 80-20% and 0.58 for 70-30%), resulting in overall lower F1 scores and accuracy (81-82%). This could be due to CNN's inability to effectively capture feature interactions in tabular data, as it relies on spatial hierarchies that are absent in such datasets, leading to poorer generalization and reduced performance. While the metrics remained largely consistent across the two splits, the 70-30 split, having more data in the test set, did not cause significant degradation in performance. The results highlight that Random Forest is overall the most reliable for phishing detection among the considered models in terms of accuracy, with Logistic Regression demonstrating slightly less yet competitive accuracy values, and CNN's comparatively poor results emphasize challenges in applying DL to structured data without further optimization.

Table 7: Performance metrics for different models and train-test splits

Train-Test Split	Model	Accuracy	Precision	Recall	F1 Score
80-20	Random Forest	0.93	0.93	0.79	0.85
	Logistic Regression	0.91	0.91	0.73	0.81
	CNN	0.81	0.67	0.56	0.61
70-30	Random Forest	0.93	0.92	0.79	0.85
	Logistic Regression	0.91	0.91	0.73	0.81
	CNN	0.82	0.68	0.58	0.63

Table 8: Statistical significance of F1 score differences between models

Comparison	t-statistic	95% CI F1 Difference
RF vs LR	1455.88	[0.0351, 0.0403]
RF vs CNN	4188.19	[0.2313, 0.2422]
LR vs CNN	3354.09	[0.1930, 0.2048]

The ROC curves shown in Figure 5 plot the true positive rate (TPR) against the false positive rate (FPR) at various classification thresholds. A curve closer to the top-left corner indicates better model performance, as it shows high TPR and low FPR. The area under the curve (AUC) quantifies this performance, with values closer to 1 indicating better classification ability, while an AUC of 0.5 suggests random guessing. We can see that the AUC values for RF and LR are consistent for both data splits (0.97 and 0.96, respectively) whereas that for CNN is slightly less (0.83 for 80-20 split compared to 0.82 for 70-30 split). These results demonstrate that classical ML models like Random Forest and Logistic Regression maintain robust and stable performance across different training-test splits, highlighting their reliability for phishing detection on structured data. On the other hand, the AUC values for the CNN model suggest that DL architectures may require more extensive tuning or larger datasets to achieve comparable stability and accuracy in this domain.

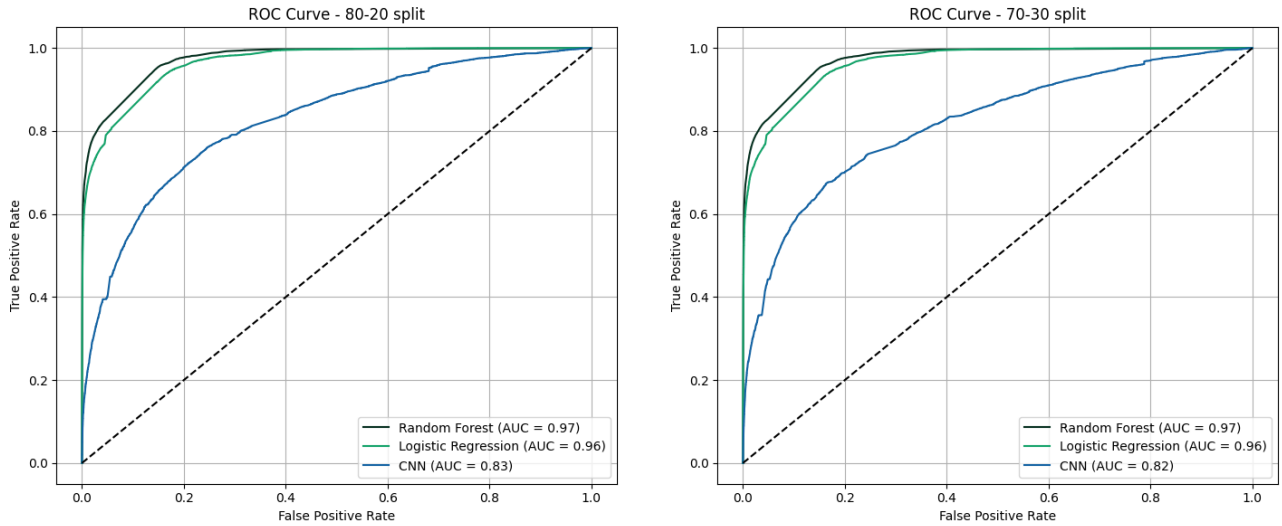

Figure 5: ROC curves for different models and train-test splits

In terms of time taken for inference, since phishing detection is a time-critical application and requires real-time results, model inference results must be generated in least time possible. Figure 6 shows the average inference time for the three models over 10 samples for 10 iterations; it can be seen

that Logistic Regression performs the best in terms of inference (fraction of a second) given its simple computations, followed by Random Forest, and finally CNN (0.1456 seconds) due to its complex neural architecture. These results highlight the trade off between model complexity and real-time applicability, with Logistic Regression offering a favorable balance for latency-sensitive phishing detection scenarios. In general, the results show that Random Forest achieved the best performance, with higher accuracy and F1-scores than both Logistic Regression and CNN. Logistic Regression, however, proved fastest in inference, making it suitable for real-time use, while CNN underperformed on both accuracy and speed. These results highlight the trade-off between accuracy and latency, with Random Forest providing strong overall balance and Logistic Regression being more practical for latency-sensitive phishing detection.

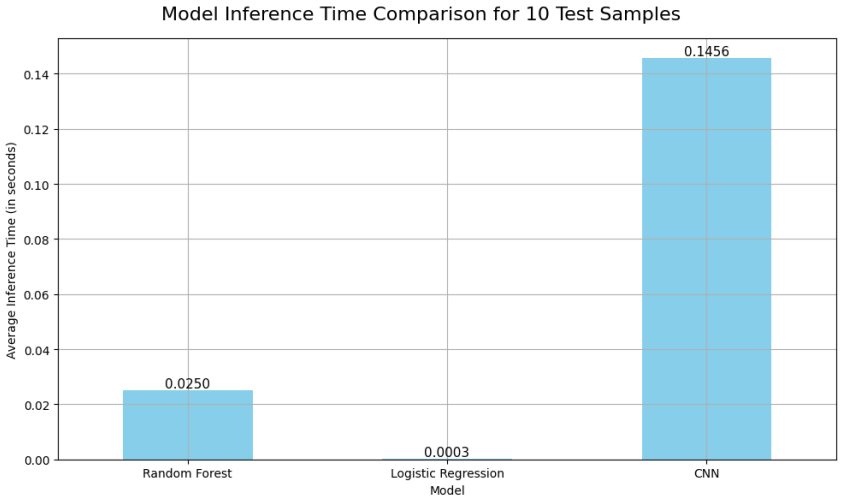

Figure 6: Average inference time using different models over 10 iterations

For the second experiment, we compare the performance of the machine learning models (RF and LR) for data balancing techniques. Two different data balancing techniques in addition to no balancing were compared through confusion matrices. The first technique is SMOTE (Synthetic Minority Over-sampling Technique) which is an over-sampling technique used to address class imbalance by generating synthetic examples for the minority class. It works by selecting minority class instances and creating new instances by interpolating between them and their neighbors. The second technique is undersampling which reduces the size of the majority class to balance the dataset. This is achieved by randomly removing samples from the majority class to match the number of instances in the minority class.

As presented in Table 9, both models significantly improve their ability to detect the minority class after applying SMOTE and undersampling, as evidenced by the higher number of true positives. However, this improvement came with a slight increase in false positives. Random Forest consistently outperformed Logistic Regression across all methods, achieving a better balance between sensitivity and specificity, particularly when undersampling was applied. The comparison of these data balancing techniques highlights their impact on model performance and underscores the importance of addressing class imbalance in phishing detection. SMOTE generally enhanced recall by enabling models to better identify minority class instances, though it occasionally increased false positives. Conversely, undersampling effectively balanced the dataset but sometimes led to the loss of valuable information from the majority class, which negatively affected overall accuracy.

Table 9: Confusion matrices for different data balancing techniques

Technique	Model	TP	FP	TN	FN
None	Random Forest	26548	1597	79553	6098
	Logistic Regression	25548	2042	79108	7098
SMOTE	Random Forest	30884	9855	71295	1762
	Logistic Regression	30357	11029	70121	2289
Undersampling	Random Forest	31191	10948	70202	1455
	Logistic Regression	2341	69892	69882	2341

4.2 XAI Results

In order to generate local explanations, a text sample is selected randomly from the test set. The sample is selected to be of type SMS message and labeled as phishing. The sample text is pre-processed (vectorized and tokenized) to be fed into the three models to generate and explain the predictions on this sample (Figure 7). The prediction probabilities reflect the classification of this sample as phishing by all models. Using the Python built-in SHAP [29] kernel explainer, model predictions are explained, and the 3 features with highest contribution to the prediction being phishing are presented in Figure 8. It can be seen that words like 'click', 'secure', and 'urgent' contribute highly to a text being classified as phishing, which is semantically logical and aligns with natural human language. This local explanation not only validates the model's ability to focus on meaningful and relevant features but also enhances trust and transparency in the detection process. By highlighting interpretable keywords that align with human intuition, the approach facilitates better understanding and validation of the model's decisions by field experts.

```
Sample Text:  Urgent! Your Facebook has been compromised. Click here to secure it now!

Vectorized:
       click  facebook   secure    urgent
0  0.464145  0.382075  0.47063  0.645829

Tokenized:
 [[2264, 23, 68, 110, 162, 1, 160, 114, 5, 246, 24, 130]]
```

Figure 7: XAI test sample-vectorized and tokenized

```
RF Explanation
Feature names: ['secure', 'click', 'wrote']
Top 3 Feature Contributions:  [0.1716139  0.15910992 0.04184985]

LR Explanation
Feature names: ['click', 'secure', 'facebook']
Top 3 Feature Contributions:  [0.34748352 0.2124324  0.14640958]

CNN Explanation
Feature names: ('your', 'secure', 'urgent')
Top 3 Feature Contributions:  (np.float64(0.20182090951397247), np.float64(0.17994541205846096), np.float64(0.12589781900200447))
```

Figure 8: SHAP local explanations on selected sample

In addition to local explanations on a sample input, we also generate global explanations to examine which features (words) influence the model decision into classifying an input as phishing. Figure 9 compares the top 6 features for each of the three models; each bar represents the average absolute SHAP value of a given word, indicating how much that word contributes to the model's output on average. Across all models, the word "login" emerged as the most influential indicator of phishing, followed by "admin", which also showed high importance across the three classifiers. Other words like "update", "info", and "auth" also contributed significantly, especially for CNN and Logistic Regression.

Some features were emphasized more by certain models—for instance, ”administrator” and ”verify” had noticeable influence in Random Forest and CNN, respectively, while ”browse” was highlighted mainly by Logistic Regression. This global explanation highlights the key features driving model decisions, providing valuable insight into the common patterns recognized across different classifiers. The consistency of terms like ”login” and ”admin” as strong phishing indicators reinforces their critical role in identifying malicious content. Additionally, the variation in feature importance across models suggests that each model captures different nuances in the data, which could be leveraged for ensemble approaches to improve detection robustness. In general the third experiment revealed both local and global insights into model behavior, where local explanations show words that strongly influenced predictions across all models, while global explanation show terms like “login” and “admin” consistently emerging as the most influential phishing indicators. This consistency reinforces the critical role of certain features in detecting malicious content.

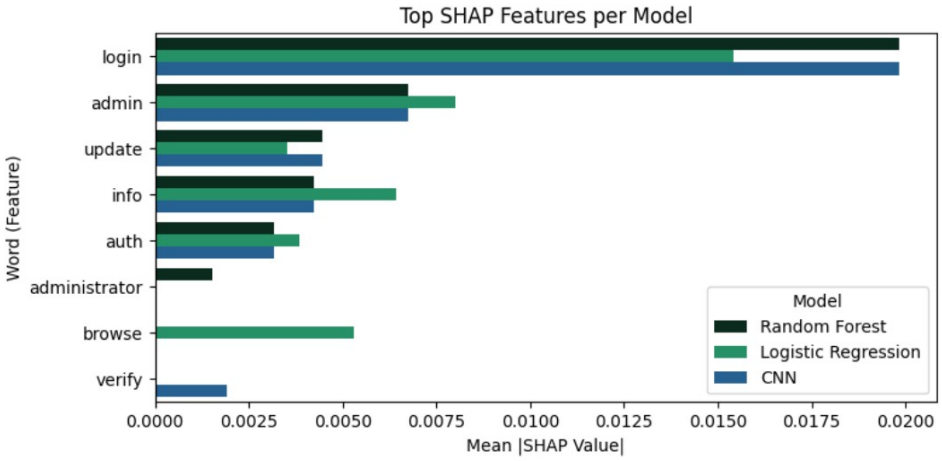

Figure 9: SHAP global explanations feature comparison for different models

5. Research Insights and Limitations

This work presents several key contributions to the field of phishing detection in cybersecurity. The result shows that Random Forest consistently outperforms Logistic Regression and CNN, achieving the highest accuracy (93%) and F1 scores, particularly in detecting phishing cases. Techniques like SMOTE and undersampling further improved detection of the minority class, with Random Forest maintaining top performance across all balancing strategies. The integration of SHAP-based explainability provides both local and global feature attributions, with keywords such as “click,” “login,” and “admin” emerging as strong phishing indicators. These findings offer practical insights into model effectiveness across data types and emphasize the value of interpretability, adaptability, and robustness in phishing detection systems.

Nevertheless, despite its strong performance, our approach has certain limitations that pave the way for future research. One limitation is the focus on English language datasets which restricts generalization in multilingual settings, as such future work should explore cross-lingual transfer and low resource language settings where phishing attacks exploit specific regional linguistic cues. Another limitation is the scope of detection model architectures, as recent advances in deep learning architectures such as transformer-based and multi-modal models would further enhance robustness against sophisticated attacks. Additionally, real-time deployment considerations remain underexplored, including the challenges of integration with live email gateways or messaging platforms, ensuring low-latency performance, and maintaining privacy in distributed environments through techniques such

as federated learning. Beyond the mentioned technical aspects, strengthening the explainability layer is also crucial, not only by comparing multiple XAI methods but also by conducting user-centered evaluations to examine how explainability influences trust, decision-making, and adoption by end-users and security analysts.

6. Conclusions

This study presents a generalized phishing detection framework that explores ML, DL, and XAI techniques to effectively identify phishing content across diverse text sources, including emails, SMS messages, and URLs. Through systematic experimentation, Random Forest emerged as the most reliable model, achieving the highest accuracy and balanced performance across precision and recall, while Logistic Regression offered superior inference speed. The integration of SHAP provided both local and global interpretability, highlighting linguistically meaningful features such as urgency-related terms that strongly influence model predictions.This research advances the field by demonstrating the feasibility and benefits of integrating heterogeneous data types into a single, robust detection pipeline, moving beyond models that typically focus on a single input type. Our work also contributes to the growing body of literature emphasizing the importance of interpretability in cybersecurity, showing how SHAP-based explanations can provide both local and global insights into model decisions, thereby enhancing trust and accountability in automated phishing detection.

While our approach shows strong performance, there are several limitations, including the language of the dataset and the diversity and scale of real-world phishing attacks, potentially limiting generalizability, in addition to SHAP's computational overhead which may impact real-time application, especially for deep learning models. Lastly, the CNN model's relatively lower performance suggests a need for further exploration of architectures better suited to structured tabular data or alternative embedding strategies. Future research should explore expanding the dataset to incorporate more languages and emerging phishing vectors, such as social media messages and voice phishing transcripts, to improve the framework's robustness. Investigating hybrid or ensemble models that combine the strengths of machine learning and deep learning could enhance both accuracy and efficiency.

7. Dataset, Code, and Dashboard

Our work is fully reproducible via and open-source notebook and an interactive PowerBI dashboard:

- Dataset and Notebook link
- Dashboard link

Acknowledgement

This research was not funded by any grant.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Mijwil, M., Unogwu, O. J., Filali, Y., Bala, I., & Al-Shahwani, H. (2023). Exploring the top five evolving threats in cybersecurity: An in-depth overview. *Mesopotamian Journal of CyberSecurity*, *2023*, 57–63. https://doi.org/10.58496/MJCS/2023/010

[2] Ahmed, O. (2024). Enhancing intrusion detection in wireless sensor networks through machine learning techniques and context awareness integration. *International Journal of Mathematics, Statistics, and Computer Science*, *2*, 244–258. https://doi.org/10.59543/ijmscs.v2i.10377

[3] Alibrahimi, F., Hazzaa, F., Jabbar, M., Fadhil, J., Sekhar, R., Shah, P., & Parihar, S. (2024). Intrusion detection in software-defined networks: Leveraging deep reinforcement learning with graph convolutional networks for resilient infrastructure. *Fusion Practice and Applications*, *15*, 78–87. https://doi.org/10.54216/FPA.150107

[4] Hazzaa, F., Qashou, A., Barazanchi, I., Sekhar, R., Shah, P., Bachute, M., & Shawkat, A. (2024). Performance analysis of advanced encryption standards for voice cryptography with multiple patterns. *International Journal of Safety and Security Engineering*, *14*, 1439–1446. https://doi.org/10.18280/ijsse.140511

[5] N B, H., Ravi, V., & Kp, S. (2018). A machine learning approach towards phishing email detection cen-security@iwspa 2018.

[6] Rawal, S., Rawal, B., Shaheen, A., & Malik, S. (2017). Phishing detection in e-mails using machine learning. *International Journal of Applied Information Systems*, *12*, 21–24. https://doi.org/10.5120/ijais2017451713

[7] Abu-Nimeh, S., Nappa, D., Wang, X., & Nair, S. (2007). A comparison of machine learning techniques for phishing detection. *Proceedings of the Anti-Phishing Working Groups 2nd Annual ECrime Researchers Summit*, 60–69. https://doi.org/10.1145/1299015.1299021

[8] Chien, A., & Khethavath, P. (2023). Email feature classification and analysis of phishing email detection using machine learning techniques. *2023 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*, 1–8. https://doi.org/10.1109/CSDE59766.2023.10487729

[9] Mohammed, M., Ibrahim, D., & Salman, A. (2021). Adaptive intelligent learning approach based on visual anti-spam email model for multi-natural language. *Journal of Intelligent Systems*, *30*, 774–792. https://doi.org/10.1515/jisys-2021-0045

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

[11] Ferdaws, R., & Majd, N. E. (2024). Phishing url detection using machine learning and deep learning. *2024 IEEE World AI IoT Congress (AIIoT)*, 0485–0490. https://doi.org/10.1109/AIIoT61789.2024.10579005

[12] BOUIJIJ, H., & BERQIA, A. (2021). Machine learning algorithms evaluation for phishing urls classification. *2021 4th International Symposium on Advanced Electrical and Communication Technologies (ISAECT)*, 01–05. https://doi.org/10.1109/ISAECT53699.2021.9668489

[13] Charan, A. N. S., Chen, Y.-H., & Chen, J.-L. (2022). Phishing websites detection using machine learning with url analysis. *2022 IEEE World Conference on Applied Intelligence and Computing (AIC)*, 808–812. https://doi.org/10.1109/AIC55036.2022.9848895

[14] Uplenchwar, S., Sawant, V., Surve, P., Deshpande, S., & Kelkar, S. (2022). Phishing attack detection on text messages using machine learning techniques. *2022 IEEE Pune Section International Conference (PuneCon)*, 1–5. https://doi.org/10.1109/PuneCon55413.2022.10014876

[15] Verma, S., Ayala-Rivera, V., & Portillo-Dominguez, A. O. (2023). Detection of phishing in mobile instant messaging using natural language processing and machine learning. *2023 11th International Conference in Software Engineering Research and Innovation (CONISOFT)*, 159–168. https://doi.org/10.1109/CONISOFT58849.2023.00029

[16] Abdul Samad, S. R., Ganesan, P., S, T., Balasubramaniyan, S., Rajiakodi, S., & Rashid Al Kaabi, A. S. (2024). Sms-shield: A lightweight approach for smishing detection using machine learning. *2024 1st International Conference on Innovative Engineering Sciences and Technological Research (ICIESTR)*, 1–6. https://doi.org/10.1109/ICIESTR60916.2024.10798213

[17] Kohilan, R., Warakagoda, H. E., Kitulgoda, T. T., Skandhakumar, N., & Kuruwitaarachchi, N. (2023). A machine learning-based approach for detecting smishing attacks at end-user level. *2023 IEEE International Conference on e-Business Engineering (ICEBE)*, 149–154. https://doi.org/10.1109/ICEBE59045.2023.00042

[18] Salman, A. M., Al-Nuaimi, B. T., Ali Subhi, A., Alkattan, H., & Alfilh, R. H. C. (2025). Enhancing cybersecurity with machine learning: A hybrid approach for anomaly detection and threat prediction. *Mesopotamian Journal of CyberSecurity*, *5*(1), 202–215. https://doi.org/10.58496/MJCS/2025/014

[19] Galego Hernandes, P. R., Floret, C. P., Cardozo De Almeida, K. F., Da Silva, V. C., Papa, J. P., & Pontara Da Costa, K. A. (2021). Phishing detection using url-based xai techniques. *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, 01–06. https://doi.org/10.1109/SSCI50451.2021.9659981

[20] Pavani, B. V., Mahitha, D., & Maheswari, B. U. (2024). Enhancing online safety: Phishing url detection using machine learning and explainable ai. *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 1–6. https://doi.org/10.1109/ICCCNT61001.2024.10723976

[21] Alam, R., Khune, A., Kalal, T. V., & Nautiyal, A. (2024). E2phish: Explainable ensemble machine learning model for enhanced phishing url detection. *2024 IEEE 8th International Conference on Information and Communication Technology (CICT)*, 1–6. https://doi.org/10.1109/CICT64037.2024.10899721

[22] Choutapally, R., & Das, T. (2024). Paciphish: Intelligent and interpretable phishing url detection framework. *2024 Cyber Awareness and Research Symposium (CARS)*, 1–7. https://doi.org/10.1109/CARS61786.2024.10778820

[23] Tiwari, T. (2022). Phishing site urls.

[24] Journal, S. (2022). Phishing emails dataset.

[25] Tijjani, A. (2023). Phishing - urgency, authority, persuasion dataset.

[26] Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

[27] Starbuck, C. (2023). Logistic regression. In *The fundamentals of people analytics: With applications in r* (pp. 223–238). Springer International Publishing. https://doi.org/10.1007/978-3-031-28674-2_12

[28] Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: An overview and application in radiology. *Insights into Imaging*, *9*(4), 611–629. https://doi.org/10.1007/s13244-018-0639-9

[29] Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions [SHAP Python library. Available at: https://shap.readthedocs.io/]. *Advances in Neural Information Processing Systems*, *30*. https://github.com/shap/shap