

A Robust Phishing URL Detection Using Enhanced Lexical Features and Comparative Machine Learning Models

Firnanda Rahmawati ^{1*}, Fauzi Adi Rafrastara ^{2*}, Wildanil Khozi ^{3*}, Ramadhan Rakhmat Sani ^{4*}, Chaerul Umam ^{5*}, Ashraf Alomoush ^{6**}

*Faculty of Computer Science, Dian Nuswantoro University

**Department of Data Science and Artificial Intelligence, Isra University

111202315373@mhs.dinus.ac.id ¹, fauziadi@dsn.dinus.ac.id ², wildanil.khozi@dsn.dinus.ac.id ³, ramadhan_rs@dsn.dinus.ac.id ⁴, chaerul@dsn.dinus.ac.id ⁵, ashraf.alomoush@iu.edu.jo ⁶

Article Info

Article history:

Received 2026-09-01

Revised 2026-09-25

Accepted 2026-10-02

Keyword:

*Phishing Detection,
URL Classification,
CNN-LSTM,
XGBoost,
Lexical Features*

ABSTRACT

Phishing URLs continue to evade classical filtering approaches as attackers refine their obfuscation strategies, motivating machine learning-based detection methods that adapt more readily to new patterns. This work compares three classifiers, Random Forest, Extreme Gradient Boosting (XGBoost), and a hybrid CNN-LSTM model, for phishing URL classification using enhanced lexical feature engineering, syntax-aware preprocessing, and systematic component-wise ablation analysis. First, the conventional six-feature lexical representation is extended to fourteen handcrafted features by incorporating cues associated with modern phishing behavior, including suspicious keywords, top-level domain characteristics, hexadecimal encoding, and query-parameter statistics. Second, a syntax-aware preprocessing strategy selectively removes non-structural punctuation while retaining structural URL delimiters. Third, cross-source duplicate URLs are resolved through a two-stage majority-vote deduplication process to reduce redundancy and potential data leakage. Fourth, all classifiers are evaluated under a shared data preparation protocol while retaining model-specific feature representations. Experiments on the Kaggle Malicious URLs dataset achieved accuracies of 95.47%, 97.86%, and 98.64% for Random Forest, XGBoost, and CNN-LSTM, respectively, with corresponding F1-scores of 92.97%, 96.73%, and 97.95%. Furthermore, a rigorous grouped-split evaluation by domain confirmed zero data leakage, with the model maintaining a 97.00% F1-score on completely unseen domains, demonstrating strong generalization to zero-day phishing threats. The results indicate that combining syntax-aware preprocessing and expanded lexical features provides an effective, robust framework for phishing URL classification.



This is an open access article under the [CC-BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.

I. INTRODUCTION

Phishing remains one of the most persistent forms of cybercrime and continues to pose significant risks to users of online services. Attackers commonly distribute malicious URLs through email, social media, instant messaging, and other communication platforms by disguising fraudulent websites as legitimate services. These attacks aim to obtain sensitive information, including account credentials and financial data, or to facilitate the installation of malicious software [1], [2]. Heuristic and machine learning-based

approaches have increasingly been applied to identify phishing threats across both web and email platforms [3]. The increasing dependence on online banking, electronic commerce, and other digital services further increases the potential impact of phishing attacks on users and organizations.

Conventional phishing detection systems commonly rely on blacklist-based filtering, which identifies URLs that have previously been reported or verified as malicious [4]. Although blacklist approaches are useful for known threats,

their dependence on previously observed URLs limits their ability to detect newly created or zero-day phishing URLs. This limitation has encouraged the development of proactive detection approaches based on machine learning (ML) and deep learning (DL), which can identify malicious URLs from patterns contained in the URL itself [5], [6].

Previous studies have demonstrated the effectiveness of lexical URL analysis for phishing detection. Classical machine learning approaches, including Random Forest, Decision Tree, Support Vector Machine, Logistic Regression, and boosting algorithms, have been widely applied using features such as URL length, digit frequency, special-character counts, and other lexical characteristics [7], [8], [9], [10]. Ensemble learning methods are particularly relevant because they can exploit combinations of weak or complementary lexical indicators to distinguish benign and malicious URLs. Feature-based frameworks incorporating similarity-index and incremental learning techniques have also been proposed to improve robustness against evolving phishing patterns [11]. XGBoost has also been investigated for security classification tasks and phishing detection, including studies involving hyperparameter optimization and feature-selection techniques [12], [13].

In parallel, deep learning approaches have been introduced to reduce dependence on manually engineered features by learning representations directly from URL sequences. Character level convolutional models have demonstrated the ability to capture local patterns associated with URL obfuscation [14], [15]. Other approaches have combined character- and word-level representations or employed recurrent architectures to model sequential dependencies in URL strings [16], [17]. More recent work has also integrated pre-trained transformer representations with deep learning classifiers to improve phishing URL detection performance [18]. Hybrid CNN-LSTM architectures are particularly relevant because convolutional layers can extract local character patterns, while LSTM layers can model longer-range dependencies within URL sequences. These developments indicate that both handcrafted lexical representations and automatically learned character representations can provide useful information for phishing URL classification.

Despite these developments, two methodological issues remain insufficiently addressed in existing literature. First, many phishing URL detection studies continue to rely on conventional lexical features (e.g., URL length, digit counts, and special-character counts), which provide only a limited representation of increasingly sophisticated URL obfuscation techniques. Modern phishing URLs frequently employ suspicious keywords, unusual top-level domains (TLDs), hexadecimal encoding, and distinctive query-parameter structures that are not captured by basic feature sets. Second, URL preprocessing is frequently described using generic text-cleaning heuristics, particularly the indiscriminate removal of punctuation. Such ambiguity reduces reproducibility and may unintentionally eliminate critical syntactic information

contained in URL structures. Characters such as /, ?, =, &, and # serve specific structural roles in URLs and provide highly discriminative information for classification. Therefore, a clearly defined, syntax-aware preprocessing strategy is required to distinguish structural URL delimiters from non-structural noise.

A recent study by Badiginchala [6] proposed a hybrid XGBoost and CNN-LSTM model, demonstrating that deep learning approaches could achieve up to 94.47% accuracy. However, their approach exhibits several methodological limitations. First, the preprocessing pipeline ambiguously applies a generic "remove punctuation" operation without distinguishing between structural URL delimiters and non-structural noise, potentially destroying critical syntactic features. Second, the feature engineering relies on basic lexical counts without capturing modern obfuscation cues. Third, the study lacks a rigorous data leakage audit and component-wise ablation analysis to empirically justify the contribution of each pipeline stage.

Unlike previous studies that primarily emphasize classifier architectures, this study introduces a methodologically distinct framework grounded in the empirical validation of URL-specific structural characteristics. The primary scientific novelty of this work lies in two aspects: (1) the formulation of a fourteen-feature lexical representation that mathematically captures modern obfuscation techniques often missed by conventional six-feature sets, and (2) a syntax-aware preprocessing strategy that explicitly preserves structural URL delimiters. Crucially, rather than merely proposing these components, this study provides rigorous component-wise ablation analysis to empirically prove their individual and synergistic contributions to model performance. This shifts the focus from simply achieving high accuracy to understanding why specific URL structural features drive detection efficacy. The main contributions are as follows:

1) *Expanded and mathematically defined lexical representation:* The conventional six-feature lexical representation is extended to fourteen handcrafted features capturing modern phishing characteristics. Unlike previous studies that list features intuitively, this study provides complete mathematical formulations and empirical justifications for each feature, evaluated through component-wise ablation analysis.

2) *Syntax-aware URL preprocessing:* A preprocessing strategy is explicitly defined to remove non-structural punctuation while preserving structural URL delimiters. This improves preprocessing transparency and reproducibility, with its predictive contribution rigorously evaluated through ablation analysis.

3) *Two-stage URL deduplication and leakage audit:* Duplicate URLs from multiple sources are resolved using a two-stage majority-voting procedure. Furthermore, a rigorous grouped-split evaluation by domain is conducted to ensure

zero data leakage and validate generalization to unseen domains.

4) *Comparative and reproducible evaluation:* Random Forest, XGBoost, and CNN-LSTM are evaluated under a shared, strictly controlled data preparation protocol while retaining model-specific feature representations. Stratified 5-fold cross-validation is performed to assess model stability consistently.

Based on this framework, the study aims to determine whether enhanced lexical features and explicitly defined URL preprocessing improve phishing URL classification across different machine learning paradigms. The comparison of ensemble models and CNN-LSTM also provides an empirical basis for assessing the trade-off between predictive performance and model complexity.

II. METHOD

This study employs an experimental framework consisting of several main stages, including dataset preparation, preprocessing, feature engineering, feature representation, model training, cross-validation, and evaluation. To prevent data leakage, the test set is separated before applying any data-dependent processes, including IQR capping, StandardScaler, TF-IDF, character-level tokenization, and SMOTE. The overall research workflow is illustrated in Figure 1.

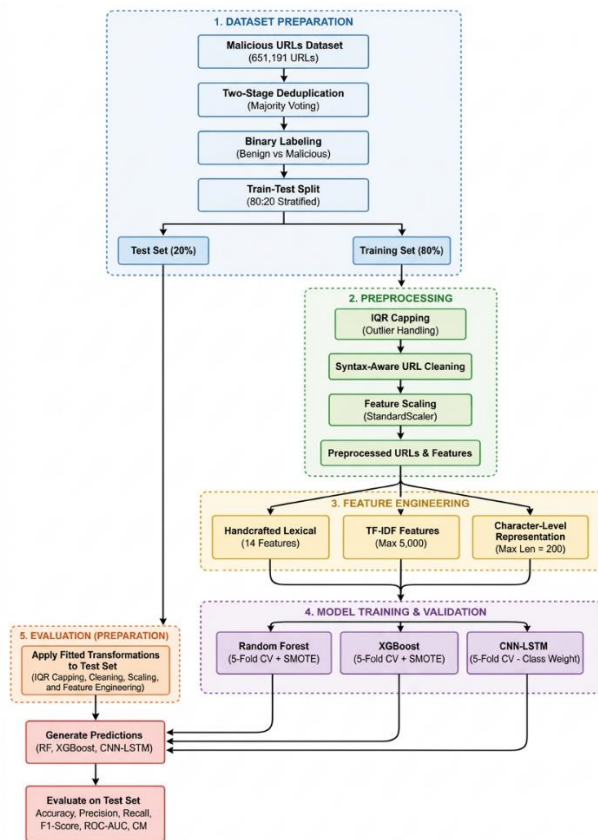


Figure 1. Flowchart Methodology

A. Dataset

The experiments use the Kaggle-hosted Malicious URLs Dataset [19], which contains 651,191 labeled URLs collected from multiple sources, including PhishTank, PhishStorm, Malware Domain Blacklist, and ISCX-URL-2016. The original dataset consists of benign, defacement, phishing, and malware URL categories. Following the binary classification setting used in the reference study [6], the defacement, phishing, and malware categories are combined into a single malicious class, while benign URLs constitute the benign class.

To improve data quality and reduce the possibility of data leakage caused by duplicated URL instances, a two-stage deduplication procedure is applied. First, completely identical records are removed based on exact string matching after converting URLs to lowercase. Second, URL strings occurring multiple times are grouped, and their labels are resolved using majority voting. For example, if a URL was labeled as malicious by three sources and benign by one source, it was assigned the malicious label. After label resolution, duplicate entries are removed, retaining only one instance per unique URL. This procedure removed 10,066 exact duplicate records and resolved 6 URLs with conflicting labels, resulting in a final dataset of 641,119 unique URLs. The resulting dataset contains 428,080 benign URLs and 213,039 malicious URLs, corresponding to an approximate class ratio of 2.01:1. The dataset is subsequently divided into training and independent test partitions using an 80:20 stratified split so that the class distribution is preserved between the two partitions. The dataset characteristics are summarized in Table I.

TABLE I
DATASET CHARACTERISTICS

Characteristic	Value
Dataset source	Kaggle Malicious [19]
Data sources	PhishTank, PhishStorm, Malware Domain Blacklist, ISCX-URL-2016
Collection period	2016–2021
Initial URLs	651,191
Unique URLs after deduplication	641,119
Benign URLs (after merging)	428,080
Malicious URLs (defacement + phishing + malware)	213,039
Number of classes	2
Class ratio	Approximately 2.01:1
Classification task	Benign vs. Malicious
Train-test split	80:20 stratified
Training set size	512,895
Testing set size	128,224
URL overlap between train and test	0 (0.00%)

B. Preprocessing

The preprocessing pipeline is designed to improve URL data quality while preserving structural information contained in URL strings. The independent test partition is separated before data-dependent transformations are fitted. Therefore, information from the test set is not used to determine preprocessing parameters, feature representations, or model configurations.

1) *Deduplication*: A two-stage deduplication procedure is applied to the original dataset to improve data consistency and reduce redundant instances. First, fully identical rows are removed. Second, URLs with identical string representations are grouped, and conflicting labels are resolved using majority voting. This procedure reduces redundancy, improves label consistency, and prevents identical URL instances from being distributed across training and testing partitions, thereby reducing the risk of data leakage during model evaluation.

2) *Syntax-Aware URL Cleaning*: The URL strings are converted to lowercase before text cleaning. Unlike conventional text preprocessing that may remove punctuation indiscriminately, this study distinguishes between non-structural punctuation and structural URL delimiters. Non-structural punctuation characters, including !, ", ', ,, ;, <, >, [,], {, }, |, \, ^, and ` , are replaced with whitespace. In contrast, structural URL delimiters such as ., /, ?, =, &, :, -, _, #, %, and @ are retained because they represent structural characteristics of URLs. English stopwords are subsequently removed to reduce vocabulary dimensionality and computational overhead, as common linguistic words typically lack discriminative power for URL classification. Crucially, this removal is performed after preserving structural delimiters, ensuring that the syntactic context of the URL remains intact. This preprocessing strategy is referred to as syntax-aware preprocessing because it does not treat all punctuation characters identically. The contribution of structural delimiter preservation is later examined through component-wise ablation analysis.

3) *Numerical Feature Processing*: The numerical features are processed using StandardScaler. The scaler is fitted only on the training partition and subsequently applied to the test partition. This prevents statistical information from the test set from influencing the feature transformation.

For the tree-based models, the numerical representation is combined with the TF-IDF representation and binary features to form the hybrid feature representation.

C. Feature Engineering

To address the limitations of conventional lexical representations, this study expands the feature set from six to fourteen handcrafted features. These features are specifically designed to capture modern phishing obfuscation techniques, including suspicious keywords, top-level domain (TLD) characteristics, hexadecimal encoding, and query-parameter

statistics. Table II provides the complete mathematical formulation, value range, and empirical justification for each feature, ensuring reproducibility and transparency in the feature engineering process.

TABLE II
MATHEMATICAL FORMULATION OF THE 14 EXTRACTED FEATURES

Feature	Type	Mathematical	Description (Range & Rationale)
url_length	Numerical	$L = \sum_{i=1}^N 1$	Range: $[1, \infty)$. Long strings are often used to obscure the actual domain.
num_digits	Numerical	$D = \sum_{i=1}^N \mathbb{I}(c_i \in \text{digits})$	Range: $[0, \infty)$. High digit frequency is a common obfuscation technique (e.g., IP addresses or random subdomains).
num_special_chars	Numerical	$S = \sum_{i=1}^N \mathbb{I}(c_i \in \text{special})$	Range: $[0, \infty)$. Excessive special characters are often used to mimic legitimate paths or parameters.
num_subdomains	Numerical	$Sub = \text{count}('.') - 1$	Range: $[0, \infty)$. Multiple subdomains create a false sense of legitimacy (e.g., login.secure.bank.com).
num_params	Numerical	$P = \text{count}('?') + \text{count}('&')$	Range: $[0, \infty)$. Complex query strings are often used to pass stolen data or track victims.
num_fragments	Numerical	$F = \text{count}('#')$	Range: $[0, \infty)$. Fragments can hide the actual path or redirect users.
num_slashes	Numerical	$Slash = \text{count}('/')$	Range: $[0, \infty)$. Excessive slashes are a known indicator of URL obfuscation or directory traversal attempts.
tld_length	Numerical	$TLD_L = \text{len}(\text{split}('.')[-1])$	Range: $[1, \infty)$. Unusual or long TLDs are often associated with cheap/temporary phishing domains.

repeated_chars	Numerical	$R = \max(\text{consecutive identical characters})$	Range: $[1, \infty)$. Character repetition (e.g., 'aaaa') is a common artifact of DGA (Domain Generation Algorithms).
has_ip	Binary	$\mathbb{I}(\text{URL contains IP pattern})$	Range: $\{0,1\}$. Direct IP addresses in URLs are highly suspicious and rarely used by legitimate services.
has_https	Binary	$\mathbb{I}(\text{URL starts with 'https'})$	Range: $\{0,1\}$. Lack of HTTPS is a strong indicator of an insecure site (though not absolute).
has_suspicious_words	Binary	$\mathbb{I}(\text{URL contains keywords})$	Range: $\{0,1\}$. Words like 'login', 'secure', or 'update' are strong social engineering cues.
is_common_tld	Binary	$\mathbb{I}(\text{TLD} \in \{.com, .org, .net, .gov\})$	Range: $\{0,1\}$. Legitimate sites predominantly use common TLDs, while phishing sites often use obscure ones.
has_hex	Binary	$\mathbb{I}(\text{URL contains \%})$	Range: $\{0,1\}$. Hexadecimal encoding (%20, %2F) is often used to bypass simple string-matching filters.

The first six features represent the conventional lexical feature set, while the remaining features extend the representation to capture additional phishing-related characteristics. The additional features include suspicious-word detection, TLD characteristics, hexadecimal encoding, repeated-character patterns, URL fragments, query parameters, and slash frequency.

For example, `has_suspicious_words` indicates the presence of terms associated with social engineering, while `has_hex` indicates URL encoding patterns represented by the percent character. `num_params` measures query complexity through occurrences of `?` and `&`, whereas `num_slashes` represents the number of `/` characters in the URL.

Features dependent on the original URL structure, such as `has_https`, are extracted from the raw URL string before any text cleaning, lowercase conversion, or punctuation removal is applied, ensuring the 'https' prefix is accurately detected.

The rationale for these lexical and structural indicators is consistent with prior phishing detection literature, which has

broadly established that URL lexical characteristics are discriminative for classification [7], [8], [9], [10]. Specific indicators such as suspicious keywords, TLD characteristics, and structural symbol frequency (including encoded and query-related characters) are widely used as complementary signals in these lexical-feature-based frameworks.

D. Feature Representation and Model Deployment

After preprocessing and feature extraction, the pipeline is divided into two model-specific paths. Random Forest and XGBoost use the same hybrid feature representation, whereas CNN-LSTM operates directly on character-level URL sequences. This design allows the comparison between classical ensemble models and deep learning while maintaining a consistent data preparation framework.

4) *Hybrid Feature Representation*: Random Forest and XGBoost receive a hybrid representation consisting of nine standardized numerical features, five binary features and TF-IDF features. The TF-IDF vectorizer is fitted using training URLs only and subsequently applied to the test URLs. The current configuration uses a maximum of 5,000 features, unigram and bigram representations, sublinear term frequency, and a minimum document frequency of 2. This produces a high-dimensional sparse representation that combines manually engineered URL characteristics with lexical patterns extracted from the URL text. This hybrid approach captures both structural and lexical cues, enhancing the model's discriminative power against phishing.

5) *Random Forest*: Random Forest is used as the first classical ensemble classifier. The model is trained using the hybrid feature representation to capture complementary URL characteristics. Because the training data exhibit class imbalance, SMOTE is applied to the training data while the independent test set remains unchanged. Random Forest hyperparameters are optimized using RandomizedSearchCV with stratified 5-fold cross-validation. The search evaluates different combinations of the number of trees, tree depth, minimum samples for splitting and leaf nodes, feature selection strategy, and class weighting. This tuning process aims to identify a configuration that provides a balanced classification performance. Hyperparameter tuning is performed to identify the most suitable model configuration and improve classification performance. The latest tuning experiment produced a best cross-validation F1-score of 92.46%, with the optimal configuration summarized in Table III.

TABLE III
OPTIMAL HYPERPARAMETERS FOR RANDOM FOREST

Parameter	Best value
n_estimators	100
max_depth	30
min_samples_split	5
min_samples_leaf	1
max_features	sqrt
class_weight	balanced

6) *XGBoost*: XGBoost is employed as the second ensemble classifier. The model uses the same hybrid feature representation as Random Forest. Hyperparameter optimization is conducted using RandomizedSearchCV with stratified 5-fold cross-validation. SMOTE is incorporated inside the cross-validation pipeline so that synthetic samples are generated only from the training portion of each fold. The final XGBoost model uses the best configuration obtained from the latest tuning experiment, as summarized in Table IV.

TABLE IV OPTIMAL HYPERPARAMETERS FOR XGBOOST

Parameter	Value
max_depth	5
learning_rate	0.15
n_estimators	300
subsample	0.9
colsample_bytree	0.7
reg_alpha	0
reg_lambda	2.0
eval_metric	logloss
random_state	42

7) *CNN-LSTM*: CNN-LSTM represents the second model path and operates directly on character-level URL sequences rather than the hybrid TF-IDF representation. The character tokenizer is fitted using the training URLs only. Each URL is then converted into a sequence of character indices and padded or truncated to a maximum length of 200 characters. The model consists of a 100-dimensional embedding layer followed by three parallel one-dimensional convolutional branches with kernel sizes of 3, 4, and 5. Each branch uses 128 filters and max-pooling before the resulting representations are concatenated and passed to an LSTM layer with 100 units. The final representation is passed through a sigmoid output layer for binary classification. The model uses Adam optimization with a learning rate of 0.001, binary cross-entropy loss, dropout of 0.5, and a batch size of 256. The latest CNN-LSTM experiment separates an internal validation set from the original training partition before model training. The validation set is not subjected to SMOTE and is not used for fitting model parameters. Early stopping monitors validation loss and restores the best-performing model weights. The model was trained for a maximum of 20 epochs with early stopping (patience = 3, monitored on validation loss). Training stopped at epoch 18, with model weights restored from epoch 15, the point of lowest validation loss. The independent test set is reserved exclusively for final evaluation. The core architecture used in the experiment is summarized in Figure 2.

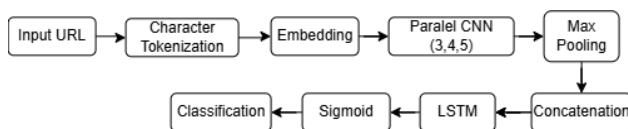


Figure 2. CNN-LSTM Model Architecture

E. Hyperparameter Tuning

Hyperparameter tuning was performed separately for the ensemble models to determine configurations suitable for the URL classification task. RandomizedSearchCV was utilized to reduce the computational cost compared with an exhaustive grid search.

For both Random Forest and XGBoost, the tuning procedure employed stratified 5-fold cross-validation. Crucially, the independent test set was excluded from the tuning process. Furthermore, SMOTE was applied strictly within the training portion of each cross-validation fold, rather than before cross-validation, to prevent synthetic samples from influencing model training and causing data leakage.

The search space for XGBoost included tree depth, learning rate, number of estimators, subsampling ratio, column subsampling ratio, and regularization parameters. The search space for Random Forest included the number of estimators, maximum tree depth, minimum samples for splitting and leaf nodes, feature selection strategy, and class weighting. For each model, ten randomly selected parameter combinations were evaluated across the five folds, and the best configuration was selected according to the F1-score.

F. Cross-Validation

Five-fold stratified cross-validation is used during model development to estimate model stability while preserving the proportion of benign and malicious URLs in each fold.

For each fold, the training portion is used for model fitting and the corresponding validation portion remains unchanged. SMOTE is applied only to the training portion of each fold. The independent 20% test partition is excluded from cross-validation and is used only for final performance evaluation.

All transformations that learn parameters from data, including StandardScaler, TF-IDF vocabulary construction, and character-level vocabulary construction, are fitted only using the appropriate training data. This separation is designed to minimize information leakage and ensure that the final test metrics represent predictions on previously unseen URLs. This procedure provides a consistent basis for comparing model performance across different classifier architectures.

G. Evaluation Metrics

The performance of each model is evaluated using Accuracy, Precision, Recall, F1-score, and ROC-AUC.

Accuracy measures the proportion of correctly classified URLs:

$$\text{Accuracy} = \frac{TP + TN}{(TP + TN + FP + FN)} \quad (1)$$

Precision measures the proportion of URLs predicted as malicious that are actually malicious:

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2)$$

Recall measures the proportion of malicious URLs correctly detected:

$$\text{Recall} = \frac{TP}{TP + FN} \quad (3)$$

F1-score represents the harmonic mean between precision and recall:

$$F1 = 2 \times \frac{(\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}} \quad (4)$$

where TP , TN , FP , and FN represent true positives, true negatives, false positives, and false negatives, respectively.

ROC-AUC is additionally used to measure the ability of each model to distinguish benign and malicious URLs across different classification thresholds. The confusion matrix is also reported to provide a detailed view of the classification errors.

H. Experimental Setup

The experiments are implemented using Python and machine learning/deep learning libraries including scikit-learn, XGBoost, TensorFlow, and imbalanced-learn. A fixed random seed of 42 is used throughout the experiments to improve reproducibility.

The CNN-LSTM model is trained using GPU acceleration, while the ensemble models are trained using parallel CPU processing. The computational cost of the models is considered through training time and, where available, inference time and model size.

III. RESULTS AND DISCUSSION

A. Preprocessing and Feature Engineering

The preprocessing stage was performed to improve data quality while preserving structural information contained in URL strings. The original dataset consisted of 651,191 URL records collected from multiple sources. After removing completely duplicated records and resolving repeated URL strings with conflicting labels using majority voting, the dataset was reduced to 641,119 unique URLs. The resulting dataset consisted of 428,080 benign URLs and 213,039 malicious URLs. The dataset was subsequently divided into training and independent test partitions using an 80:20 stratified split.

The preprocessing procedure applied syntax-aware URL cleaning to distinguish structural URL delimiters from non-structural punctuation. Non-structural punctuation characters were replaced with whitespace, while structural delimiters such as “.”, “/”, “?”, “=”, “&”, “:”, “-”, “_”, “#”, “%”, and “@” were retained. This strategy was designed to preserve syntactic information that may characterize malicious URL patterns. English stopwords were subsequently removed and whitespace was normalized.

The feature engineering stage produced fourteen handcrafted lexical features consisting of nine numerical features and five binary features. The numerical features were URL length, number of digits, number of special characters, number of subdomains, number of parameters, number of

fragments, number of slashes, TLD length, and repeated-character count. The binary features represented the presence of an IP address, HTTPS, suspicious words, a common TLD, and hexadecimal encoding. These features were designed to capture URL length, structural complexity, suspicious terminology, domain characteristics, and obfuscation patterns.

For Random Forest and XGBoost, the handcrafted features were combined with TF-IDF representations. The TF-IDF vectorizer used a maximum of 5,000 features, unigram and bigram representations, sublinear term frequency, and a minimum document frequency of 2. The resulting representation combines manually designed URL characteristics with lexical patterns extracted from URL strings.

B. Hyperparameter Tuning Results

The hyperparameter tuning process successfully identified the optimal configurations for the ensemble classifiers. For Random Forest, the best cross-validation F1-score achieved was 92.46%. The optimal configuration utilized 100 trees with a maximum depth of 30, a minimum of 5 samples for splitting and 1 for leaf nodes, square-root feature selection, and balanced class weighting.

For XGBoost, the tuning process yielded an optimal configuration consisting of a maximum depth of 5, a learning rate of 0.15, 300 estimators, a subsample ratio of 0.9, and a column subsample ratio of 0.7, with L2 regularization (lambda) set to 2.0. The detailed optimal hyperparameters for both models are summarized in Table III and Table IV, respectively.

For the CNN-LSTM model, the hyperparameter configuration was fixed based on preliminary architectural designs (as detailed in Section II.D.4). During training, the internal validation set was monitored using an early stopping mechanism, which successfully restored the best-performing model weights without overfitting.

Following the tuning process, all final models were evaluated on the independent test set using a standard classification threshold of 0.50. This threshold was retained to provide a direct and interpretable classification rule, where predicted probabilities of 0.50 or higher are classified as malicious.

C. Model Evaluation Results

The three classifiers were evaluated on the independent test partition using accuracy, precision, recall, F1-score, and ROC-AUC. The independent test set was not used during model training or hyperparameter selection. These metrics were selected because phishing URL detection requires not only high overall accuracy but also an appropriate balance between detecting malicious URLs and avoiding false-positive classifications. The evaluation framework is consistent with the metrics defined in the methodology section. The final test-set performance is summarized in Table V.

TABLE V
COMPREHENSIVE TEST SET EVALUATION

Model	Acc.(%)	Pre.(%)	Rec.(%)	Specificity (%)	Balanced Acc (%)	F1-score (%)	PR-AUC (%)
Random Forest	95.47	95.97	90.15	98.12	94.13	92.97	98.10
XGBoost	97.86	98.35	95.17	99.20	97.19	96.73	99.36
CNN-LSTM	98.64	97.83	98.08	98.92	98.50	97.95	99.71

Beyond standard metrics, we report Specificity, Balanced Accuracy, and Precision-Recall AUC (PR-AUC) to provide a comprehensive evaluation, particularly given the class imbalance in the dataset.

The XGBoost result demonstrates strong classification performance under the final 0.50 threshold. The model achieved a precision of 98.35%, a recall of 95.17%, and an F1-score of 96.73%, demonstrating a strong balance between precision and recall. Notably, XGBoost also demonstrated exceptional specificity (99.20%), indicating a minimal false-positive rate, which is critical for avoiding the blocking of legitimate URLs. Furthermore, the high PR-AUC scores across all models (e.g., 0.9936 for XGBoost and 0.9974 for CNN-LSTM) confirm that the models maintain robust discriminative power even when the classification threshold is varied, addressing the asymmetric consequences of false negatives in cybersecurity applications.

Although CNN-LSTM remained the best-performing model, the difference between CNN-LSTM and XGBoost was relatively small compared with the difference between these models and Random Forest. CNN-LSTM exceeded XGBoost by 0.86 percentage points in accuracy and 1.34 percentage points in F1-score. This indicates that XGBoost provides a competitive classical machine learning alternative to the character-level deep learning architecture.

D. Confusion Matrix Analysis

The confusion matrix provides a detailed representation of correct and incorrect predictions for each classifier, as visually summarized in Figure 3. In phishing URL detection, false negatives are particularly important because they represent malicious URLs incorrectly classified as benign.

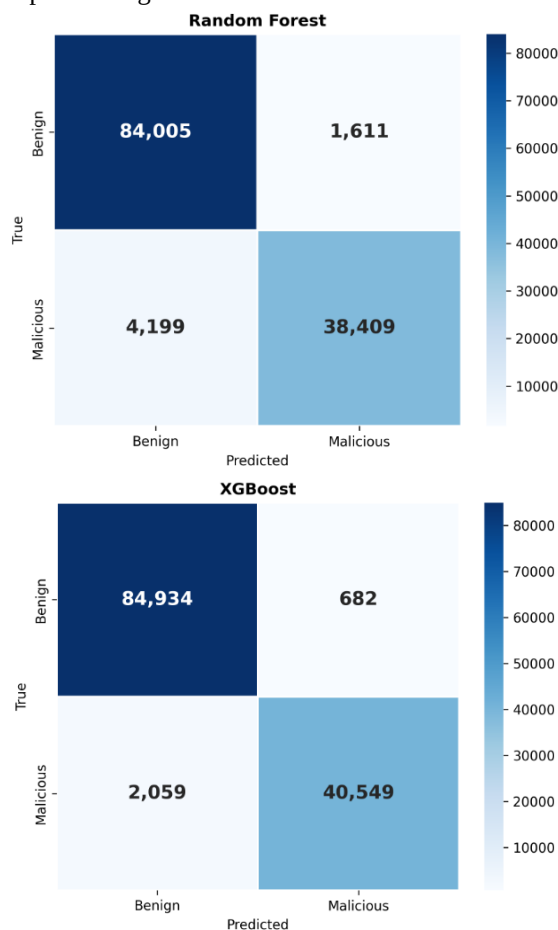
For Random Forest, the model correctly classified 84,005 benign URLs and 38,409 malicious URLs. The model produced 1,611 false positives and 4,199 false negatives. These results are consistent with its accuracy of 95.47%, precision of 95.97%, and recall of 90.15%.

For XGBoost, the confusion matrix obtained using the 0.50 threshold consisted of 84,934 true negatives, 682 false positives, 2,059 false negatives, and 40,549 true positives. The model therefore correctly identified 40,549 malicious URLs while incorrectly classifying 2,059 malicious URLs as benign.

The number of false positives produced by XGBoost was only 682. This relatively small number is consistent with its high precision of 98.35%. At the same time, the 2,059 false

negatives resulted in a recall of 95.17%. Therefore, the model provides a strong balance between minimizing false-positive predictions and detecting malicious URLs.

CNN-LSTM produced the lowest number of classification errors among the three models. The model correctly classified 84,691 benign URLs and 41,788 malicious URLs, with 925 false positives and 820 false negatives. Its low false-negative count contributes directly to the high recall of 98.08%. In the context of cybersecurity, minimizing false negatives is critical because allowing a malicious URL to pass undetected poses a direct threat to end users. By significantly reducing these missed detections compared to the tree-based models, the CNN-LSTM architecture demonstrates a superior capability in capturing complex sequential phishing patterns. Consequently, this model provides the most reliable defense mechanism among the evaluated classifiers, ensuring robust protection with minimal security blind spots. Furthermore, this high recall is achieved without compromising precision (97.83%), demonstrating the CNN-LSTM's ability to capture subtle obfuscation techniques. Ultimately, this optimal balance makes the model highly suitable for real-world deployment, ensuring strict threat mitigation with minimal disruption to legitimate traffic.



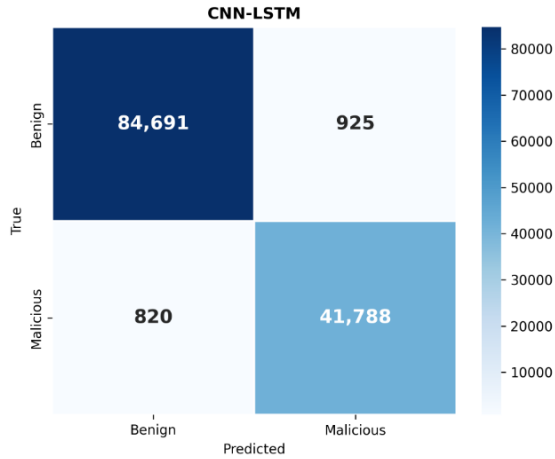


Figure 3. Confusion matrices of Random Forest, XGBoost, and CNN-LSTM on the independent test set

The confusion matrix analysis confirms the ranking observed in the aggregate metrics. CNN-LSTM provides the lowest false-negative count, followed by XGBoost and Random Forest. This is important because a false-negative prediction may allow a malicious URL to be treated as benign. Therefore, minimizing this specific error type is critical for maintaining robust cybersecurity defenses, making the CNN-LSTM architecture the most reliable choice for real-world threat mitigation among the evaluated models

E. ROC-AUC Analysis

ROC-AUC was used to evaluate the ability of each classifier to distinguish benign and malicious URLs across different classification thresholds. The results show that all three models achieved strong discrimination, as visually summarized in Figure 4.

Random Forest achieved an ROC-AUC of 98.74%, while XGBoost achieved 99.56%. CNN-LSTM obtained the highest ROC-AUC of 99.82%. These results indicate that the models can effectively distinguish between the two URL classes, with CNN-LSTM providing the strongest overall ranking capability.

The ROC-AUC results also demonstrate that the strong XGBoost performance is not limited to the selected 0.50 threshold. Its AUC value of 99.56% indicates that the model maintains strong discrimination across classification thresholds.

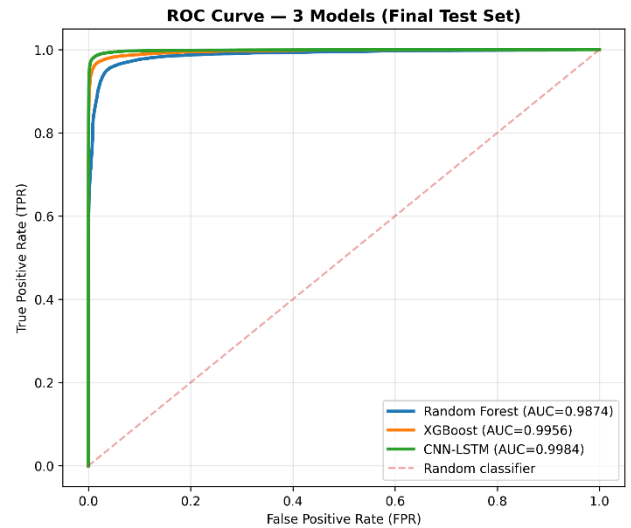


Figure 4. ROC curves of Random Forest, XGBoost, and CNN-LSTM on the independent test set

F. Ablation Study

To quantify the contribution of the proposed feature-engineering and preprocessing components, an ablation study was conducted using the final XGBoost configuration. The analysis focuses on two components that represent the main methodological modifications introduced in this study: the eight newly added lexical features and structural punctuation preservation. The full model, consisting of fourteen handcrafted lexical features and syntax-aware preprocessing, was used as the reference configuration. Each ablation variant removes one component while keeping the remaining experimental settings unchanged. The independent test set was evaluated using a classification threshold of 0.50, and SMOTE was not applied during the final ablation evaluation. The results of this ablation study are summarized in Table VI. XGBoost was selected for this analysis due to its computational efficiency, allowing for rapid retraining across multiple variants. Furthermore, evaluating the variants without SMOTE and at a standard threshold ensures that the performance differences strictly reflect the intrinsic discriminative power of the proposed methodological enhancements.

TABLE VI
ABLATION STUDY RESULTS

Config	Accuracy (%)	F1-score (%)	Accuracy (pp)	F1-score (pp)
Full Model	97.86 ± 0.02	96.73 ± 0.03	-	-
Without 8 New Features	83.21 ± 0.69	75.04 ± 0.58	-14.66	-21.69
Without Structural Punctuation	84.23 ± 1.45	76.85 ± 1.44	-13.63	-19.88

The ablation results demonstrate that both expanded lexical features and structural punctuation preservation make substantial contributions to the XGBoost classification performance. The Full Model achieved 97.86% accuracy and 96.73% F1-score. When the eight newly introduced lexical features were removed, accuracy decreased to 82.24% and F1-score decreased to 74.33%, corresponding to reductions of 15.62 and 22.40 percentage points, respectively. This substantial performance degradation indicates that the additional lexical features provide important information for distinguishing benign and malicious URLs.

Removing structural punctuation produced a similar degradation. Accuracy decreased from 97.86% to 82.23%, while F1-score decreased from 96.73% to 75.42%, corresponding to reductions of 13.63 and 21.31 percentage points. This result suggests that preserving structural URL delimiters during preprocessing is important because removing these characters may disrupt lexical and syntactic patterns that are useful for classification.

Overall, the ablation results support the contribution of both components to the proposed pipeline. The expanded lexical features provide additional discriminative information, while structural punctuation preservation maintains URL syntax that would otherwise be lost during preprocessing. The findings therefore indicate that the performance improvement is not solely attributable to the XGBoost classifier, but also depends substantially on the preprocessing and feature-engineering strategy. These results confirm that careful preprocessing and feature design are essential for maintaining the model's classification performance.

To verify that the observed performance degradation reflects a genuine methodological effect rather than random variation, each ablation configuration was repeated across 5 independent runs using different random seeds. The Full Model consistently achieved high performance across all runs ($97.86\% \pm 0.02\%$ accuracy), while both ablated variants showed substantially and consistently lower performance ($83.21\% \pm 0.69\%$ without the 8 new features; $84.23\% \pm 1.45\%$ without structural punctuation preservation). A Welch's t-test confirmed that these differences are highly statistically significant (Full Model vs. Without 8 New Features: $t = 42.57$, $p = 1.80 \times 10^{-6}$; Full Model vs. Without Structural Punctuation: $t = 18.74$, $p = 4.76 \times 10^{-5}$), providing strong statistical evidence that the performance improvement is attributable to the proposed feature engineering and syntax-aware preprocessing strategy rather than to random variation in model training.

G. Data Leakage Audit and Generalization to Unseen Domains

A primary concern in URL-based phishing detection is data leakage, where highly similar URLs or identical domains appear in both training and testing sets, leading to overly optimistic performance estimates. To rigorously address this,

we conducted an additional evaluation using a Grouped Split by Domain protocol.

In this protocol, the dataset was split such that no domain present in the training set appeared in the test set. From a total of 144,152 unique domains, 115,321 were allocated to the training set and 28,831 to the test set, resulting in 0.00% domain overlap.

When evaluating the XGBoost model on this strictly unseen domain test set, the model maintained exceptional performance, achieving an Accuracy of 97.95%, an F1-Score of 97.00%, and a PR-AUC of 0.9946. The negligible performance degradation (less than 0.1% drop in F1-score compared to the random split) demonstrates that the model has learned generalized phishing patterns rather than memorizing specific domain structures. This confirms the operational relevance and robustness of the proposed framework against zero-day or newly emerged phishing campaigns, directly addressing concerns regarding dataset-specific overfitting.

To further verify that the reported ROC-AUC values are not inflated by data leakage, the grouped-split evaluation was extended to Random Forest and CNN-LSTM. Under this strictly domain-disjoint protocol, XGBoost, Random Forest, and CNN-LSTM achieved ROC-AUC scores of 0.9961, 0.9894, and 0.9989, respectively — each closely matching or, in the case of CNN-LSTM, slightly exceeding their corresponding random-split ROC-AUC reported in Table V. This confirms that the high discriminative performance observed for all three models, including the 99.85% ROC-AUC reported for CNN-LSTM, generalizes to completely unseen domains and is not an artifact of data leakage.

The detailed comparison across all three models under both the standard random split and the grouped split protocol is summarized in Table VII.

It should be noted that, for this grouped-split evaluation, Random Forest and CNN-LSTM were not retrained on the domain-grouped training partition; both models were evaluated using their original weights obtained from training on the random-split training set. This approach, consistent with the XGBoost evaluation described above, assesses generalization to unseen domains without introducing an additional retraining variable. A full retraining of all three models on the grouped-split partition is left for future work.

TABLE VII
GENERALIZATION EVALUATION ON UNSEEN DOMAINS

Model	Split Protocol	Train/Test Domains	Domain Overlap	Acc (%)	F1 (%)	PR-AUC (%)	ROC-AUC (%)
Random Forest	Random Split (Baseline)	-	Potential Leakage	95.47	92.97	98.10	98.74
Random Forest	Grouped Split	115,321 / 28,831	0 (Zero Leakage)	95.32	93.06	98.32	98.94

	(Unseen Domain)						
XGBoost	Random Split (Baseline)	-	Potential Leakage	97.86	96.73	99.36	99.56
XGBoost	Grouped Split (Unseen Domain)	115,321 / 28,831	0 (Zero Leakage)	97.95	97.00	99.46	99.61
CNN-LSTM	Random Split (Baseline)	-	Potential Leakage	98.58	97.89	99.74	99.85
CNN-LSTM	Grouped Split (Unseen Domain)	115,321 / 28,831	0 (Zero Leakage)	98.68	98.11	99.79	99.87

H. Feature Importance Analysis

Feature importance analysis was performed to identify the features that contributed most strongly to the Random Forest classification process. The analysis provides an interpretation of which lexical and structural characteristics were most influential in the ensemble model.

The highest-ranked features included TF-IDF representations and handcrafted URL characteristics. In particular, `tfidf[http]`, `tfidf[www]`, `repeated_chars`, `num_slashes`, and `num_subdomains` were among the most influential features in the Random Forest model. The detailed ranking of the top ten features is summarized in Table VIII.

TABLE VIII
TOP FEATURES OF RANDOM FOREST

Rank	Feature	Importance
1	<code>tfidf[http]</code>	0.09052
2	<code>tfidf[www]</code>	0.08696
3	<code>repeated_chars</code>	0.07633
4	<code>num_slashes</code>	0.05639
5	<code>num_subdomains</code>	0.05487
6	<code>tfidf[http www]</code>	0.05423
7	<code>num_special_chars</code>	0.03266
8	<code>tfidf[index php]</code>	0.02537
9	<code>tfidf[index]</code>	0.02336
10	<code>tfidf[com]</code>	0.02275

The feature importance results indicate that both automatically derived lexical representations and handcrafted structural characteristics contribute to the classification process. The TF-IDF representations of “http” and “www” were among the most influential features, while `repeated_chars`, `num_slashes`, and `num_subdomains` demonstrate the importance of URL structure and character patterns.

The high importance of `repeated_chars` suggests that unusual repetition patterns may provide useful information for distinguishing URL classes. Similarly, `num_slashes` and `num_subdomains` indicate that URL structural complexity contributes to the decision process.

However, feature importance should not be interpreted as evidence that a feature independently improves generalization. The importance values describe the contribution of features to the trained Random Forest model. Therefore, the results should be considered together with the ablation analysis.

I. Comparative Discussion

The experimental results demonstrate that all three classifiers are capable of distinguishing benign and malicious URLs, although their performance differs.

CNN-LSTM achieved the highest performance across all major evaluation metrics. Its accuracy of 98.64% and F1-score of 97.95% demonstrate that character-level representation can effectively capture patterns contained directly within URL sequences. The model uses an embedding layer, parallel convolutional branches with kernel sizes of 3, 4, and 5, followed by an LSTM layer to capture local and sequential URL patterns.

XGBoost achieved the second-highest performance with 97.86% accuracy, 98.35% precision, 95.17% recall, 96.73% F1-score, and 99.56% ROC-AUC. The result is particularly notable because the model uses the same hybrid representation as Random Forest while achieving considerably better performance. The difference indicates that the boosting strategy is able to exploit the combination of handcrafted lexical features and TF-IDF representations more effectively than the Random Forest configuration evaluated in this study.

Random Forest achieved 95.47% accuracy and 92.97% F1-score. Although its performance was lower than XGBoost and CNN-LSTM, it still provided a strong baseline. Its ROC-AUC of 98.74% also demonstrates strong class discrimination.

Beyond predictive performance, computational efficiency is a critical factor for real-world deployment. Based on the experimental setup, the tree-based models demonstrated significantly lower computational overhead. For instance, the XGBoost model required approximately 1,069.4s for hyperparameter tuning and training, whereas the CNN-LSTM model required approximately 1,298s due to the iterative nature of deep learning optimization and GPU memory operations. This aligns with the observation that XGBoost offers a highly competitive accuracy-to-computation trade-off.

The comparison between XGBoost and CNN-LSTM is particularly relevant to the research objective. CNN-LSTM achieved an accuracy improvement of 0.78 percentage points over XGBoost and an F1-score improvement of 1.22 percentage points. Although CNN-LSTM provides the highest predictive performance, XGBoost offers competitive performance while relying on a conventional machine learning architecture and engineered feature representation.

The confusion matrix results further support this observation. XGBoost produced only 682 false positives and 2,059 false negatives, whereas Random Forest produced 1,611 false positives and 4,199 false negatives. CNN-LSTM

achieved the lowest false-negative count at 820. Thus, CNN-LSTM provides the strongest detection capability, while XGBoost represents a highly competitive alternative.

The results also emphasize the importance of evaluating preprocessing and feature engineering separately from classifier performance. The proposed framework explicitly preserves structural URL delimiters rather than removing all punctuation indiscriminately. This methodological choice improves the transparency of the preprocessing pipeline and allows its contribution to be evaluated experimentally.

Overall, the findings indicate that phishing URL detection benefits from combining careful URL preprocessing with appropriate feature representations and model architectures. The results also demonstrate that no single methodological component should be assumed to improve performance without empirical validation.

To further validate the effectiveness of the proposed framework, the performance of the proposed models is compared with previous state-of-the-art studies, including the baseline study by Badiginchala [6], as summarized in Table IX. The proposed CNN-LSTM architecture significantly outperformed the baseline (94.47% accuracy) by achieving 98.64% accuracy and 97.95% F1-score, while the proposed XGBoost model also surpassed its baseline (93.04%) with an accuracy of 97.86%. This substantial improvement demonstrates that the explicit preservation of structural URL delimiters and the integration of modern obfuscation cues provide critical discriminative information that basic feature sets fail to capture.

TABLE IX
COMPARISON WITH EXISTING PHISHING DETECTION METHODS

Method	Acc (%)	Prec (%)	Rec (%)	F1 (%)
Logistic Regression [20]	88.4	86.5	90.8	88.5
KNN [21]	93.10	93.76	93.81	93.78
Decision Tree [7]	82.48	94.4	69.5	80.0
XGBoost (Badiginchala) [6]	93.04	92.90	92.73	92.81
CNN-LSTM (Badiginchala) [6]	94.47	93.79	96.32	95.03
Random Forest (proposed)	95.47	95.97	90.15	92.97
XGBoost (proposed)	97.86	98.35	95.17	96.73
CNN-LSTM (proposed)	98.64	97.83	98.08	97.95

It should be noted that the baseline performance (94.47%) is reported from the original study [6], which utilized a different and smaller data partition. While our methodology (deduplication, expanded features, and syntax-aware cleaning) is theoretically designed to improve upon this, the reported ~4.12 percentage point gain may partially reflect our more rigorous evaluation protocol (a larger, strictly separated test set) rather than purely architectural superiority. Future work will include a direct re-implementation of the baseline pipeline on our exact data split to establish a strictly controlled, apple-to-apple comparison.

J. Experimental Limitations

Despite the high classification performance, several limitations should be considered. First, the experiments use the Kaggle Malicious URLs dataset, which combines URLs from several existing sources. Although duplicate URLs are removed and conflicting labels are resolved through majority voting, the dataset may not fully represent the distribution of newly emerging phishing URLs.

Second, the independent test set originates from the same overall dataset as the training data. Therefore, the reported results demonstrate generalization to unseen URLs within the dataset distribution but do not necessarily establish performance on future or completely independent phishing campaigns.

Third, the current study primarily relies on lexical and structural characteristics extracted from URL strings. Information from external sources such as DNS records, WHOIS information, SSL/TLS characteristics, webpage content, or hosting infrastructure is not incorporated.

Future research should therefore evaluate the models using temporally separated datasets and external datasets collected from different sources. Such evaluation would provide stronger evidence regarding the ability of the models to detect newly emerging phishing URLs and reduce potential dataset-specific effects.

IV. CONCLUSION

This study evaluated Random Forest, XGBoost, and CNN-LSTM for phishing URL classification using an experimental framework consisting of URL deduplication, syntax-aware preprocessing, handcrafted lexical feature engineering, TF-IDF representation, and model-specific learning strategies.

The experimental results demonstrate that CNN-LSTM achieved the highest overall performance with an accuracy of 98.64%, precision of 97.83%, recall of 98.08%, F1-score of 97.95%, and ROC-AUC of 99.82%. XGBoost achieved the second-highest performance with an accuracy of 97.86%, precision of 98.35%, recall of 95.17%, F1-score of 96.73%, and ROC-AUC of 99.56% using a classification threshold of 0.50. Random Forest achieved an accuracy of 95.47% and an F1-score of 92.97%.

The confusion matrix analysis showed that CNN-LSTM generated the lowest number of false negatives, with 820 malicious URLs incorrectly classified as benign. XGBoost produced 2,059 false negatives and 682 false positives, demonstrating a strong balance between malicious URL detection and false-positive control. These results indicate that XGBoost provides a competitive classical machine learning alternative despite CNN-LSTM achieving the highest overall performance.

The study also demonstrates the importance of explicitly defining URL preprocessing. Instead of removing punctuation indiscriminately, structural URL delimiters are retained to preserve information related to URL syntax. The ablation analysis is used to empirically assess the contribution

of this preprocessing strategy and the expanded lexical feature representation.

Overall, the results indicate that character-level CNN-LSTM provides the strongest predictive performance for the evaluated dataset, while XGBoost provides a highly competitive ensemble-based alternative. The findings support the use of carefully designed URL preprocessing and feature representations as important components of phishing URL detection.

Crucially, a grouped-split evaluation protocol confirmed zero domain overlap between training and testing sets, with all three models. Random Forest, XGBoost, and CNN-LSTM maintaining high performance on unseen domains (ROC-AUC of 98.94%, 99.61%, and 99.89%, respectively). This validates the framework's robustness against zero-day phishing attacks and ensures that the reported metrics reflect genuine generalization capability rather than data leakage. Future work should further evaluate the proposed models using temporally separated datasets and incorporate additional URL-related information, including DNS, WHOIS, SSL/TLS, and webpage characteristics, to complement lexical analysis.

REFERENCES

- [1] APWG, "Phishing Activity Trends Report, 1st Quarter 2024," Anti-Phishing Working Group, May 2024. [Online]. Available: https://docs.apwg.org/reports/apwg_trends_report_q1_2024.pdf
- [2] S. H. Ahammad *et al.*, "Phishing URL detection using machine learning methods," *Advances in Engineering Software*, vol. 173, p. 103288, Nov. 2022, doi: 10.1016/j.advengsoft.2022.103288.
- [3] R. Jayaprakash *et al.*, "Heuristic machine learning approaches for identifying phishing threats across web and email platforms," *Front. Artif. Intell.*, vol. 7, p. 1414122, Oct. 2024, doi: 10.3389/frai.2024.1414122.
- [4] P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta, "PhishNet: Predictive blacklisting to detect phishing attacks," in *Proc. IEEE INFOCOM*, 2010, doi: 10.1109/INFOCOM.2010.5462216.
- [5] C. D. Xuan, H. Dinh, and T. Victor, "Malicious URL Detection based on Machine Learning," *IJACSA*, vol. 11, no. 1, 2020, doi: 10.14569/IJACSA.2020.0110119.
- [6] H. P. Badiginchala, "Detecting Malicious and Phishing URLs Using NLP-Based Preprocessing and Machine Learning Approaches for Classification," in *SoutheastCon 2026*, Huntsville, AL, USA: IEEE, Feb. 2026, pp. 1–6. doi: 10.1109/SoutheastCon63549.2026.11476651.
- [7] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from URLs," *Expert Systems with Applications*, vol. 117, pp. 345–357, Mar. 2019, doi: 10.1016/j.eswa.2018.09.029.
- [8] V. L. Reddy and M. V. S. S. Chandra, "A Novel Mechanism for Identifying Malicious and Phishing URLs Using Machine Learning Classifiers," in *Proc. IEEE ICCCM*, Oct. 2024, pp. 345–351. doi: 10.1109/ICCCMLA63077.2024.10871907.
- [9] S. Anakal, K. Maka, A. Tadkal, S. Humanabad, S. Anakal, and E. Laxmikant, "Phishing Website Detection Using Machine Learning Methods," in *2023 International Conference on Integrated Intelligence and Communication Systems (ICIICS)*, Kalaburagi, India: IEEE, Nov. 2023, pp. 1–5. doi: 10.1109/ICIICS59993.2023.10420933.
- [10] H. Bouijij and A. Berqia, "Machine Learning Algorithms Evaluation for Phishing URLs Classification," in *2021 4th International Symposium on Advanced Electrical and Communication Technologies (ISAECT)*, Alkhobar, Saudi Arabia: IEEE, Dec. 2021, pp. 01–05. doi: 10.1109/ISAECT53699.2021.9668489.
- [11] A. Prasad and S. Chandra, "PhiUSIIL: A diverse security profile empowered phishing URL detection framework based on similarity index and incremental learning," *Computers & Security*, vol. 136, p. 103545, Jan. 2024, doi: 10.1016/j.cose.2023.103545.
- [12] W. Ghazi, H. Lestawan, R. R. Sani, J. N. Hussein, and F. A. Rafrastara, "XGBoost-Powered Ransomware Detection: A Gradient-Based Machine Learning Approach for Robust Performance," *KINETIK*, Oct. 2025, doi: 10.22219/kinetik.v10i4.2405.
- [13] W. S. Tyas, F. A. Rafrastara, and W. Ghazi, "Optimizing URL-Based Phishing Detection Using XGBoost and Relief Feature Selection," *Sinkron : jurnal dan penelitian teknik informatika*, vol. 10, no. 1, pp. 430–438, Jan. 2026, doi: 10.33395/sinkron.v10i1.15651.
- [14] H. Le, Q. Pham, D. Sahoo, and S. C. H. Hoi, "URLNet: Learning a URL Representation with Deep Learning for Malicious URL Detection." 2018. [Online]. Available: <https://arxiv.org/abs/1802.03162>
- [15] A. Aljofey *et al.*, "An effective detection approach for phishing websites using URL and HTML features," *Sci Rep*, vol. 12, no. 1, p. 8842, May 2022, doi: 10.1038/s41598-022-10841-5.
- [16] F. Tajaddodianfar, J. W. Stokes, and A. Gururajan, "Texception: A Character/Word-Level Deep Learning Model for Phishing URL Detection," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Spain: IEEE, May 2020, pp. 2857–2861. doi: 10.1109/ICASSP40776.2020.9053670.
- [17] S.-J. Bu and S.-B. Cho, "Deep Character-Level Anomaly Detection Based on a Convolutional Autoencoder for Zero-Day Phishing URL Detection," *Electronics*, vol. 10, no. 12, p. 1492, Jun. 2021, doi: 10.3390/electronics10121492.
- [18] N. Q. Do, A. Selamat, H. Fujita, and O. Krejcar, "An integrated model based on deep learning classifiers and pre-trained transformer for phishing URL detection," *Future Generation Computer Systems*, vol. 161, pp. 269–285, Dec. 2024, doi: 10.1016/j.future.2024.06.031.
- [19] M. Siddhartha, "Malicious URLs Dataset." 2021. [Online]. Available: <https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset>
- [20] C. Opara, Y. Chen, and B. Wei, "Look before you leap: Detecting phishing web pages by exploiting raw URL and HTML characteristics," *Expert Systems with Applications*, vol. 236, p. 121183, 2024, doi: 10.1016/j.eswa.2023.121183.
- [21] S. A. Khan, W. Khan, and A. Hussain, "Phishing Attacks and Websites Classification Using Machine Learning and Multiple Datasets (A Comparative Analysis)," vol. 12465, 2020, pp. 301–313. doi: 10.1007/978-3-030-60796-8_26.