Iraqi Journal for Computers and Informatics
Information Technology and Communications University
Vol. 51, No. 2, 2025, pp. 252-267

DOI: https://doi.org/10.25195/ijci.v51i2.632
Print ISSN: 2313-190X, Online ISSN: 2520-4912

Research Article

A Dataset-Driven Comparison of Traditional and Advanced Machine Learning Techniques for Phishing Detection in Low-Variance Environments

Konaz Kawa Latif [1]
Department of Software Engineering, Koya University
Koya, Kurdistan Region, Iraq
konaz.kawa@gmail.com

Saman Mirza Abdullah [2]
Department of Software Engineering, Koya University
Koya, Kurdistan Region, Iraq
saman.mirza@koyauniversity.org

ARTICLE INFO
Article History
Received: 02/08/2025
Accepted: 27/08/2025
Published: 12/11/2025

ABSTRACT

Phishing attacks continue to grow rapidly, often using obfuscated and deceptive URL patterns to mimic legitimate websites and evade detection. While traditional Machine Learning (ML) models perform well on benchmark datasets, they often struggle in real-world scenarios where phishing URLs are carefully crafted to resemble authentic domains. This study presents a dataset-driven comparison between traditional ML models—Logistic Regression, K-Nearest Neighbors, Support Vector Machine, and Random Forest—and advanced approaches such as Extreme Gradient Boosting (XGBoost), a tuned XGBoost variant, and a soft voting ensemble**.** Two datasets were used**:** (i) a high-variance global dataset from Mendeley Data, and (ii) a custom-built local dataset with region-specific phishing URLs designed with minimal alterations (e.g., character substitutions and deceptive subdomains) to simulate low-variance attacks**.** Preprocessing included feature engineering, variance analysis, and balancing with the Synthetic Minority Oversampling Technique (SMOTE). Experimental results show that Random Forest outperforms other traditional models but still struggles with low-variance phishing URLs. In contrast, advanced models—particularly tuned XGBoost—achieved significantly higher recall (0.99) and strong precision (0.81), while the voting ensemble further improved robustness by combining multiple classifiers. These findings emphasize the importance of realistic datasets and demonstrate that advanced ML strategies are more effective for detecting phishing attempts based on subtle obfuscation. This work contributes by (i) validating advanced ML models under realistic low-variance conditions, and (ii) highlighting precision and recall as more appropriate evaluation metrics than accuracy in cybersecurity.

Keywords: *Phishing Detection, Machine Learning, Obfuscated URLs, Low-Variance Datasets, Ensemble Learnin*

1. INTRODUCTION

Phishing attacks continue to be one of the most important and fast growing threats in cybersecurity. The Anti-Phishing Working Group reported the global phishing incident volume reached an alarming 5 million attacks in 2023 [1], exceeding the 4.74 million reported in 2022. It is estimated that more than 3.4 billion phishing emails are now sent daily, which is over 1 trillion a year [2]. The attacks aim to capture vital information such as usernames, passwords and finances by use of fraudulent websites or URL links [3]. The data collected in the past decade, as illustrated in Figure 1, demonstrates that phishing continues to rise at an unprecedented rate. This indicates the strong growing need and necessity for more effective detection systems on phishing.

Iraqi Journal for Computers and Informatics
Information Technology and Communications University
Vol. 51, No. 2, 2025, pp. 252-267
DOI: https://doi.org/10.25195/ijci.v51i2.632
Print ISSN: 2313-190X, Online ISSN: 2520-4912

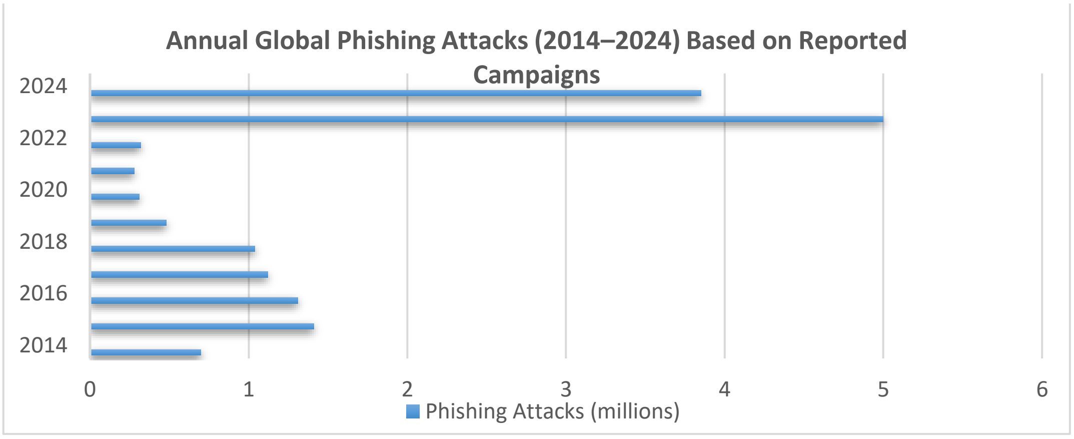


Fig 1: Annual Global Phishing Attacks (2014–2024)

The traditional phishing detection techniques like blacklist filtering and signature-based approaches have started falling behind the attackers' **evasion and anti-detection techniques** [4]. These older techniques have grown less effective with modern phishing attempts. To counter these, modern approaches like Machine Learning (ML) have started being used because the ML techniques have proven to learn complicated patterns and generalize to cases not encountered before [5].

The growing use of ML detection systems is still slowed by a number of operational challenges. These challenges include the lack of quality and adequately structured training datasets. An obvious problem is that a majority of the available public datasets are obsolete or have distinguishing features between phishing and legitimate URLs that are too simplistic. While models trained on such datasets perform well in terms of accuracy, their performance degrades drastically in the presence of realistically or subtly altered phishing URLs [3], [6].

Modern phishing tactics increasingly rely on subtle tricks, including single-character substitutions, legitimate-looking subdomains, and the use of HTTPS to gain user trust. These low- variance manipulations closely resemble legitimate websites and reduce the observable differences that ML models typically rely on. Phishing detection models developed on high-variance datasets with high degrees of features may fall prey to overfitting and be incapable of identifying phishing URLs that have been tailored to bypass detection [7]. This work seeks to fill this void with a novel locally constructed dataset, representing Kurdish websites, designed with realistic low-variance manipulations such as character substitution and subdomain spoofing. This dataset is intended to reflect actual phishing practices in the region, offering a more practical evaluation environment than global benchmark datasets.

In addition to dataset design, this study aims to examine the weaknesses of machine learning models by analyzing the sources of misclassifications, especially examining the false positive and false negative rates, to identify the conditions and explanations of failure. It further analyzes the impact of feature variance between phishing URLs and legitimate URLs on the performance of the model. The defined global public dataset as well as the newly created local dataset are used to evaluate the performance of four widely applied classifiers: Logistic Regression (LR), Support Vector Machine (SVM), Random Forest (RF), and K-Nearest Neighbors (KNN).

The study relies on two benchmark evaluation metrics, precision and recall, for fairness and comparability. Accuracy is disregarded in this case due to its irrelevance in the presence of asymmetric cost misclassification prevalent in cybersecurity [8]. For instance, in the case of phishing URLs, not detecting (false negative) can result to major outcomes like information breaches or monetary loss, while the opposite (false positive) is not as significant as simply denying access to a legitimate site. Accuracy fails in this context as it rises to a conclusion treating both errors the same. In addition to this, the presence of phishing datasets often comes with differing feature variance. In datasets with high variance, models are likely to be overfitted to strong patterns, giving the illusion of high accuracy while lacking robustness in the real world. In contrast, low variance datasets like those with obfuscated phishing URLs pose a challenge to a model in detecting subtle differences, and therefore, accuracy cannot expose most failure points. In regards to phishing detection systems, precision and recall have better complement model evaluation, as they reflect a model's ability to reliably identify threats while minimizing unwanted false alarms.

DOI: https://doi.org/10.25195/ijci.v51i2.632
Print ISSN: 2313-190X, Online ISSN: 2520-4912

The main objective of the results is to provide a better understanding of the behavioral patterns of ML models with respect to actual phishing attempts and to establish the performance consistency of different classifiers across various data settings. In the final analysis, the results of the study underscore the need for comprehensive data representation and advanced feature engineering to build real-world applicable and versatile phishing detection systems.

The rest of this paper is organized as follows. Section 2 reviews previous research on phishing detection, focusing on dataset design, feature engineering, and ensemble methods. Section 3 describes our methodology, covering dataset collection, preprocessing, variance analysis, and model development. Section 4 presents and compares the results of both traditional and advanced machine learning models, while Section 5 offers a discussion of the findings, their limitations, and practical implications. Finally, Section 6 concludes the study and highlights possible directions for future work.

2. RELATED WORKS AND BACKGROUND

The development and refinement of new attack methodologies has sparked notable interest from scholars and industries alike, and at the same time, the shortcomings of conventional phishing defense systems have drawn attention to this problem. This particular review analyzes existing phishing detection systems while underlining the importance of the quality of the phishing dataset, feature extraction, and parameter tuning in model performance. *This review pays special attention to studies that address dataset variety and realistic obfuscation, since these aspects play a key role in how well phishing detection methods hold up outside controlled settings.*

2.1 Traditional Phishing Detection Methods

Phishing detection systems relied on static blacklist/whitelist, rule-based, and signature-based filters. While once effective, these systems struggled against novel or obfuscated phishing URLs. For example, blacklist-based methods are unable to identify zero-day phishing or rapidly changing domain attacks. In the same way, content and heuristic-based methods tend to be overly resource-intensive and suffer from high false positive rates [3].

Recent phishing attack techniques, such as the use of HTTPS, domain look-alikes, and URL shortening [9], make many static approaches useless. Phishing websites have always attempted to mimic legitimate sites, but with growing precision and a constant competitive struggle, traditional detection systems become less and less effective. As a result, static defenses alone are not effective once attackers start rotating domains quickly and producing sites that almost perfectly mimic genuine ones.

2.2 Machine Learning-Based Detection Approaches

The advancement of ML phishing detection systems has accelerated due to ML's capability to learn using new data. Various models have in fact been employed in this domain. Some of them include Logistic Regression, Decision Trees, Random Forest, Support Vector Machine, K-Nearest Neighbors and Naive Bayes [6]. These models work best with a combination of lexical and structural features of URLs, such as, domain length, presence of digits, number of dots in the domain, and usage of special characters. More recently, deep learning models such as CNNs, LSTMs, and hybrids have shown strong results in analyzing raw URLs and web content [7], [10], but much of their success comes from training on high-variance public datasets, which means their performance may drop when applied to low-variance, real-world data.

ML classifiers are increasingly being utilized in practical applications, as practitioners are reporting their successful deployment on benchmark datasets. However, performance issues arise when classifiers are exposed to more practical, realistic, or adversarial datasets, such as those where phishing URLs are only slightly skewed to appear like legitimate URLs [11]. This has highlighted a greater concern regarding a dependence on scarce, outdated datasets that do not capture modern-day phishing tactics [12].

2.3 Dataset Design and Variance Impact

Several scholars and researchers have focused on the importance of dataset construction in phishing detection. A dataset available in PhishTank [13] contains rather obvious phishing URLs. Such datasets with high variance allow models to differentiate between classes with ease. This often leads to overly inflated evaluation scores such as accuracy and F1-score [5]. Noticeable disparity between model performance in controlled environments and in the real world can also be attributed to variance. *Models built on overly uniform data often struggle to adapt when the characteristics of phishing attacks shift or change in practice*. Attempting to apply the models to low variance real world scenarios where phishing URLs are designed very carefully to look like legitimate URLs leads to performance decline [14].

The problem of variance underscores the importance of having tailored or local datasets which incorporate URLs produced through realistic spoofing techniques, brand masquerading, character substitution, and subdomain spoofing. Training models on such datasets increase the chances of general exactness and reliability in real-world scenarios [15]. *This makes it crucial to test detection methods on localized, low-variance datasets to better predict how they will perform in real deployment situations.*

2.4 Feature Engineering and Informative Patterns

Feature engineering is essential to increase the performance of ML classifiers in phishing detection. The majority of models rely on hand-crafted features, for example, the ratio of digits over the total length of the URL, the length of the URL, the inclusion of some ‘suspicious’ words, and the number of subdomains. Some researchers have gone further and devised composite features such as midpoint metrics (length × number of dots), some entropy metrics, and some lexical indicators, for example, use of ‘-‘, or ‘//’ in the words forming the URLs [16].

These features, especially composite features, improve the model’s performance in distinguishing between encrypted and non-encrypted phishing URLs or URLs that have been obfuscated. In the previous work, combination of traditional lexical features with more sophisticated composite and statistical indicators was expected to enhance the robustness of the models against obfuscated and manipulated deceptive URL patterns [17]. *Alongside these techniques, choosing features with an awareness of dataset variance helps models preserve useful distinctions when phishing and legitimate URLs appear almost identical.*

2.5 Ensemble Learning and Deep Models

The application of ensemble learning methods, especially boosting and voting, offers marked improvement in the robustness of phishing detection systems. Ensemble methods, as opposed to single classifiers, combine the predictions of several models which makes ensemble methods more effective in improving generalization, and managing complex feature spaces, including those in low-variance datasets, such as the dataset of phishing URLs that closely resembles legitimate URLs.

Among these methods, Extreme Gradient Boosting XGBoost has emerged as one of the most effective **for structured URL features.** *XGBoost constructs its decision trees one after another, with each step correcting errors from the previous ones, while its built-in regularization helps maintain reliability even with noisy or imbalanced data* [7].

Voting classifiers, both hard and soft, also have real-world benefits because they combine predictions from different base learners, like Logistic Regression, SVM, and Random Forest, to make a final prediction that is more stable and accurate [18]. This diversity in model decision boundaries helps improve precision and recall, which are critical in cybersecurity applications.

This study investigates and compares the performance of XGBoost and voting ensembles to assess their effectiveness in detecting obfuscated phishing URLs within a low-variance, realistic dataset. *To conclude, much of the earlier research has focused on high-variance datasets, which creates a gap in understanding how models behave in more realistic, low-variance contexts. This study tackles that gap by introducing a region-specific dataset that reflects subtle obfuscation strategies, offering a more practical way to evaluate both traditional and advanced models.*

3. METHODOLOGY

3.1 Research Design

This study employs a machine learning approach to identify phishing websites by examining the structural and lexical features of URLs. This framework comprises a theory-based component as well as a field component which employs modern techniques and real data. It attempts to evaluate the model’s performance and generalization in relation to the scope of data variability, notably in the case of global versus local data**.**

As shown in figure 2, the methodology encompasses the collection and preprocessing of two distinct datasets: a global dataset and a region-specific local dataset reflecting phishing activity in the Kurdistan Region. Each dataset underwent data balancing (SMOTE) and then feature engineering processes. Finally, several traditional and advanced machine learning models were trained and evaluated using two distinct metrics, which are precision and recall.

DOI: https://doi.org/10.25195/ijci.v51i2.632
Print ISSN: 2313-190X, Online ISSN: 2520-4912

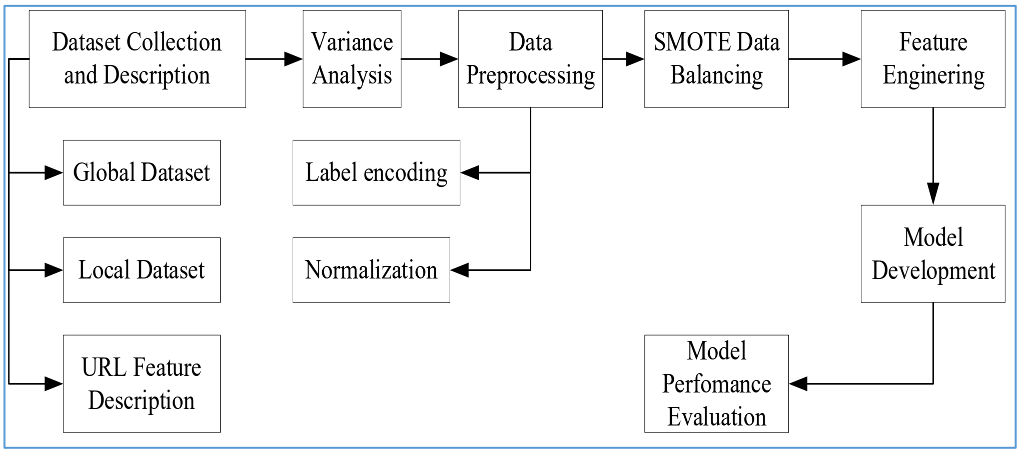


Fig 2: The Work Methodology Processes

Hyperparameter optimization was carried out through grid search with cross-validation, applied strictly on the training data to avoid test set leakage. This step was especially critical for models such as XGBoost, since factors like learning rate, maximum depth, and the number of trees have a direct impact on how well the model generalizes.

3.2 Dataset Collection and Description

3.2.1. Global Dataset

The global dataset was obtained from Mendeley Data [19] and contains 450,176 URLs, of which 104,438 were classified as phishing and 345,738 as legitimate. It encompasses phishing attempts that include the use of subdomain evasion, domain name deception, abnormal URL structuring, and improper use of HTTPS certificates. This dataset helped us to benchmark and evaluate model performance in the evaluation stage where variance was high and was further partitioned in an 80/20 manner.

3.2.2. Local Dataset

A local dataset was constructed manually in order to simulate phishing attacks within the Kurdistan Region. This dataset contains 100 authentic URLs from government, educational, and healthcare institutions, along with 500 fictitious phishing URLs crafted to mimic local attack trends. These phishing URLs incorporate counterfeit subdomains, visually similar characters, strategically misleading phrases, and redirection elements [6]. The dataset was stored in CSV format and then balanced using the SMOTE algorithm.

The local dataset, with 100 legitimate and 500 phishing URLs prior to balancing, is quite limited for large-scale ML testing. It was intended as a pilot dataset to capture realistic low-variance scenarios. Future validation on larger, naturally collected datasets is recommended to confirm generalizability.

To achieve balance, SMOTE was used to create synthetic legitimate examples. Although this provided equal class distribution, oversampling may also introduce patterns that do not occur naturally, particularly in tree-based algorithms. Despite this limitation, SMOTE was selected to ensure fair and consistent comparisons across datasets.

3.2.3. URL Feature Description

To construct a model, 16 features were extracted from each URL. These included important factors that pinpoint phishing attempts such as length, the url's organization, entropy, and keywords that are considered to be red flags. Table I shows the extracted features and their descriptions.

TABLE I. Description of Extracted URL Features

Feature Name	**Description**
URL Length	Total number of characters in the URL.

DOI: https://doi.org/10.25195/ijci.v51i2.632
Print ISSN: 2313-190X, Online ISSN: 2520-4912

Special Character Count	Counts special symbols in the URL.
Suspicious Keywords	Detects words like 'login', 'secure', etc.
Digit Count	Number of numerical digits in the URL.
Has HTTPS	Check if the URL uses HTTPS.
Number of Subdomains	Counts subdomains in the URL.
Token Frequency	Frequency of known phishing patterns.
Domain Token Count	Segments in domain split by special characters.
Domain Length	Character count of the domain name.
File Name Length	Length of the file name in the URL.
Directory Path to URL Ratio	Ratio of path length to total URL length.
Number of Dots in URL	Counts '.' characters.
Query Digit Count	Digits in the query string.
Longest Path Token Length	Longest substring in URL path.
Delimiter Path	Counts '/', '?', and '&' in the URL.
Entropy of the Domain	Measures domain randomness using Shannon entropy.

3.3 Variance Analysis

To calculate feature variance, the Variance Threshold method implemented in scikit-learn was employed to evaluate the value of each feature concerning class separation. The global dataset showed high variance for many features because of the greater diversity of the dataset. On the other hand, the local dataset showed much lower variance, indicating local phishing URLs that more closely resemble legitimate ones.

The purpose of using this method was to see how well different features help separate the two classes in each dataset. In general, features with higher variance make the distinction between phishing and legitimate URLs clearer, which supports stronger model performance. In contrast, features with very low variance provide little separation, making it harder for models to detect phishing attempts and increasing the chance of misclassification.

As shown in table II, the global dataset captures more pronounced differences, while the local dataset is dominated by closely mimicked features. This explains why models trained on the global dataset perform well, but struggle on the low-variance local dataset where phishing URLs are deliberately crafted to closely resemble legitimate ones.

Table II. Feature Variance in Global vs. Local Datasets

Features	Local_Variance	Global_Variance	Differences
url_length	46.59966	1411.626	1365.026425
Domain Length	18.03604	21.63953	3.603494594
Longest Path Token Length	11.81088	324.5005	312.6896163
File Name Length	10.85316	258.1772	247.3240758
Entropy Domain	0.646667	0.310871	-0.335795503
Number of Dots	0.469981	1.310947	0.840966546
num_subdomains	0.409124	0.448874	0.03975053
special_char_count	0.397638	20.43894	20.04130533
token_frequency	0.362827	16.50376	16.14092936
digit_count	0.183503	85.04426	84.86076193

Vol. 51, No. 2, 2025, pp. 252-267
DOI: https://doi.org/10.25195/ijci.v51i2.632
Print ISSN: 2313-190X, Online ISSN: 2520-4912

Domain Token Count	0.160198	0.158902	-0.001295243
Delimiter Count Path	0.134332	7.342216	7.207883451
suspicious_keywords	0.122994	0.130054	0.00706023
has_https	0.065209	0.170292	0.105082961
Depth Path URL Ratio	0.009408	0.0504	0.040991916
Query Digit Count	0.001667	55.17183	55.17015921

3.4 Data Preprocessing

The effectiveness of the preprocessing phase in these two datasets was crucial to achieving favorable results in machine learning. This phase commenced with the removal of non-informative columns. In this phase, the class labels were encoded to binary format, 0 representing legitimate URLs and 1 representing phishing URLs. In this case, Min-Max feature scaling was employed to ensure feature parity and eliminate scale bias, particularly for distance-based models, applicable to all numerical features. Ultimately, all datasets were partitioned into training and testing subsets using an 80/20 split to ensure fair model evaluation in all experiments.

3.5 SMOTE-based Class Balance

In this study, a local dataset was constructed to simulate phishing behavior by generating five phishing URLs for each legitimate one. Each phishing sample was created with minimal tweaks such as character substitutions and deceptive subdomains. This produced a dataset comprising 100 legitimate URLs and 500 phishing URLs. Maintaining this dataset's structure was important for ensuring a controlled focus on consistent, low-variance real-world attacks. Including additional legitimate URLs without generating balanced phishing equivalents would disturb this one-to-many framework and alter the dataset's uniform structure, and model performance could then be impacted unpredictably. Rather than accumulate authentic URLs, this research employed SMOTE to create synthetic authentic samples, equalizing the dataset to 500 samples for each class. Adopting this method permitted us to maintain the structure and the feature patterns which minimized bias during model training. This work also applied SMOTE to the global dataset to keep the preprocessing steps consistent. This made the comparison between traditional and advanced machine learning models fairer and more focused on the actual challenges of detecting phishing in low-variance conditions.

Although SMOTE effectively balanced the classes, we recognize that synthetic oversampling may generate borderline or interpolated samples that are less realistic than naturally occurring data. This represents a limitation of the approach, but was deemed necessary for maintaining methodological consistency and ensuring meaningful comparison between datasets.

3.6 Model Development

3.6.1. Traditional Machine Learning Models

The following simple models were introduced, to obtain initial benchmarks:

- **K-Nearest Neighbors (KNN):**

KNN was implemented as a distance-based classifier, which assigns labels based on the majority class of the nearest neighbors in feature space. Because KNN is sensitive to feature magnitude, all features were normalized between 0–1 using MinMax scaling. The model was instantiated with default parameters, corresponding to n_neighbors=5 and Euclidean distance as the metric [20]. No further tuning was applied, and classification was performed on an 80/20 stratified train–test split. Evaluation considered precision and recall, providing insight into how a non-parametric method handles phishing detection.

- **Logistic Regression (LR):**

Logistic Regression was applied as a linear baseline model to estimate the probability of a URL being phishing or legitimate. Features were first normalized to the range 0–1 using MinMax scaling to ensure consistent treatment across variables, given LR's sensitivity to feature magnitude. The model was trained with a maximum iteration limit of 1000 (max_iter=1000) to ensure convergence during optimization. No

Iraqi Journal for Computers and Informatics
Information Technology and Communications University
Vol. 51, No. 2, 2025, pp. 252-267

DOI: https://doi.org/10.25195/ijci.v51i2.632
Print ISSN: 2313-190X, Online ISSN: 2520-4912

regularization parameters were explicitly tuned, so the default settings were retained. Training and evaluation followed an 80/20 stratified train–test split, and performance was reported using precision and recall.

- **SUPPORT Vector Machine (SVM):**

Support Vector Machine was used as a margin-based classifier with the default kernel (radial basis function in scikit-learn when no kernel is specified, though your original notes said "linear kernel"). The implementation included probability estimates (probability = True not set here, so evaluation is based on predicted class labels only). Input features were normalized between 0–1 using MinMax scaling, which is essential given SVM's reliance on distance computations. The model was trained and tested using an 80/20 stratified split, without hyperparameter tuning, and performance was measured in terms of precision and recall.

- **Random Forest (RF)**:

Random Forest was used as a non-linear ensemble method composed of multiple decision trees, known for robustness against overfitting. The model was instantiated with default parameters, meaning the number of trees (n_estimators) and other tree-growing parameters were left at their scikit-learn defaults. Data were preprocessed with MinMax scaling, and training was carried out using an 80/20 stratified train–test split. By aggregating predictions from many trees, RF provides stable decision boundaries, making it a useful benchmark against which simpler classifiers can be compared. Evaluation metrics included precision and recall.

3.6.2. Ensemble Learning Models

- **Voting Classifiers (Hard and Soft)**:

Voting ensembles were employed to leverage the complementary strengths of multiple classifiers. In this study, Logistic Regression (with L1 regularization), Support Vector Machine (SVM), and k-Nearest Neighbors (KNN) were combined into both hard and soft voting frameworks. The hard voting approach predicts the majority class label from the base learners, while the soft voting version averages predicted probabilities, allowing better handling of uncertainty. Models were trained on standardized feature sets using a 70/30 train–test split, with the test set reserved strictly for final evaluation. Performance was assessed using precision and recall. By aggregating diverse learners, the voting classifiers aimed to increase robustness against the weaknesses of individual models and to improve phishing detection in low-variance conditions.

- **Extreme Gradient Boosting (XGBoost)**:

XGBoost was included as a strong tree-ensemble baseline due to its scalability, ability to capture complex non-linear feature interactions, and robustness to high-dimensional data. The baseline model was trained with default hyperparameters on a 70/30 train–test split, with class labels encoded and features standardized for consistency across models. No hyperparameter tuning or additional cross-validation was applied at this stage; instead, the held-out test set remained unseen during training and was used once for final evaluation. Performance was assessed using precision and recall. This baseline provides a reference point against which the improvements gained from the tuned XGBoost configuration can be evaluated.

- **Tuned XGBoost**:

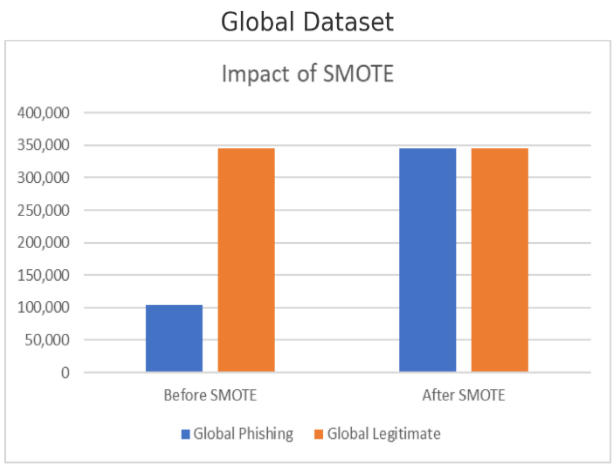
Compared to the baseline model, the tuned XGBoost adjusts tree depth, learning rate, and regularization terms to improve its ability to detect subtle obfuscations in phishing URLs. This optimization balances sensitivity to phishing detection with model generalization, ensuring that improvements are not due to overfitting. Hyperparameter optimization for XGBoost was performed using RandomizedSearchCV with 5-fold Stratified Cross-Validation on the training set only, optimizing for recall given the cost of missed phishing detections. The search explored trees and regularization settings, including the number of estimators (200–800), tree depth (3–8), learning rate (0.01–0.10), subsampling and column subsampling (0.6–1.0), node complexity (min_child_weight $\in \{1,3,5,7\}$), split penalty (gamma $\in \{0, 0.5, 1.0\}$), and L1/L2 regularization (reg_alpha $\in \{0, 1e\text{-}3, 1e\text{-}2, 1e\text{-}1\}$, reg_lambda $\in \{0.1, 1, 5, 10\}$). The test set remained unseen throughout tuning and was used once for final evaluation of the best-scoring configuration.

Iraqi Journal for Computers and Informatics
Information Technology and Communications University
Vol. 51, No. 2, 2025, pp. 252-267
DOI: https://doi.org/10.25195/ijci.v51i2.632
Print ISSN: 2313-190X, Online ISSN: 2520-4912

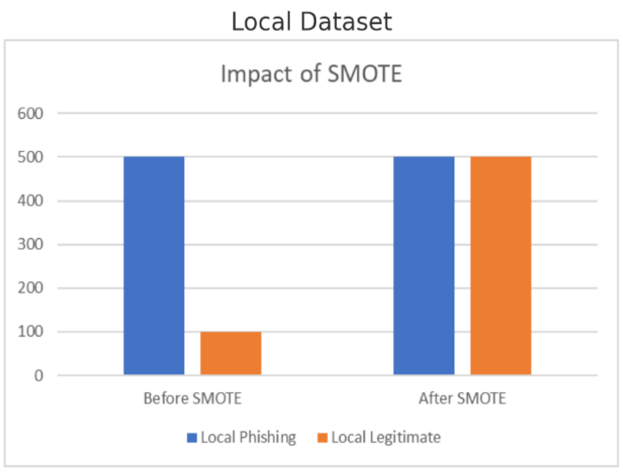
4. MODEL PERFORMANCE EVALUATION

To evaluate phishing detection models in a balanced manner, this study selected two datasets: The global dataset from Mendeley that had high variance, and a locally curated dataset that attempted to simulate real-world phishing scenarios and had lower variance. Both datasets went through a set procedure of feature extraction, normalization, variance analysis, and others to ensure clean and structured inputs for model training.

In dealing with class imbalance, both datasets underwent the SMOTE algorithm. Oversampling phishing samples enabled balance within the global dataset, while the local dataset, which had 100 legitimate and 500 phishing URLs, underwent the generation of synthetic legitimate samples for balance. This ensured bias in model evaluation was minimized. Additionally, SMOTE was chosen over simpler resampling techniques because it synthesizes new examples rather than duplicating existing ones, which reduces the likelihood of overfitting. Figure 3 illustrates the use of SMOTE to achieve balance in (a) the global dataset and (b) the local dataset.



(a)



(b)

Fig 3: Impact of SMOTE on (a) Global Dataset and (b) Local Dataset

4.1 Experimental Setup and Evaluation Metrics

This research was conducted utilizing Python 3.10 alongside Jupyter Notebook and Visual Studio Code. The core supporting libraries were pandas, numpy, scikit-learn, imbalanced-learn, xgboost, and TensorFlow, as well as Keras, which facilitated the processing, visualization, and modeling of data in both classical and modern machine learning.

All of the experiments were conducted on a personal computer with an Intel® Core™ i7 (12th Gen) CPU alongside 16 GB of RAM and running a Windows 10 (64-bit) operating system. As no external GPU was present, all computations being fully CPU bound presented no issues regarding the dataset sizes or model complexity.

To assess the performance of the models engineered using different datasets and feature sets, the two standard metrics of classification were used

1. Precision

Concentrates on accurately detecting phishing URLs while minimizing false positives. This is important because false positive identification of legitimate websites as phishing can result in user skepticism, denial of access to important resources, and unwarranted notifications. High precision metrics guarantees that only URL phishing threats are detected preserving user experience and smooth operation of systems [21].

$$\text{Precision}=\frac{TP}{TP+FP} \qquad \text{Equation (1)}$$

2. Recall

Assesses how well the model recalls phishing URLs while maintaining a low false negative rate. This is equally important as phishing detection since a low recall rate would indicate the phishing URLs that are undetected, which is a serious security threat. In cybersecurity, the consequences caused by undetected attempts are often more severe than the consequences suffered from misclassifying a legitimate entity as a threat and denying them access to a protected resource [10], [22].

$$\text{Recall} = \frac{TP}{TP+FN} \qquad \text{Equation (2)}$$

Using these two metrics ensures a reasonable compromise between preventing over-alerting and ensuring detection of all potentially harmful activities. Accuracy and the F1-score, though metric of interest in machine learning competitions, are of limited utility in high-stakes scenarios such as phishing classification. Therefore, Precision and Recall are the most useful metrics for determining the system's trust and the practical confidence that can be placed on its performance.

4.2 Results of Traditional Machine Learning Models

This section presents and analyzes the performance of four traditional machine learning models, K-Nearest Neighbors (KNN), Logistic Regression (LR), Support Vector Machine (SVM), and Random Forest (RF) in the context of phishing detection. As mentioned earlier, two datasets, a global dataset, consisting of a large and diverse set of phishing and legitimate URLs (sampled to 25,000 records post-SMOTE for computational efficiency), and a local dataset, carefully constructed to represent modern, subtle phishing patterns that closely mimic legitimate websites. The evaluation focused exclusively on precision and recall.

K-Nearest Neighbors (KNN)

The KNN algorithm classifies input samples by assigning them the label which is most frequent among their nearest neighbors in the feature space. Feature normalization was done in the preprocessing step in order to maintain accurate distance measurement. On the global dataset, KNN demonstrated strong performance, achieving a precision of 0.9959 and a recall of 0.9652—indicating minimal false positives and false negatives, and suggesting effective generalization across diverse URL patterns. However, the model's performance declined on the local dataset, with a precision of 0.8919 and a recall of 0.6600, In other words, there were 34 false negatives and 8 false positives. The drop in algorithm performance indicates the difficulty in detecting phishing URLs that closely resemble legitimate ones. In the field of cybersecurity, and particular in phishing detection, it is critical to note that a high false negative rate is dangerous as it allows malicious activities to bypass detection, thereby posing a significant risk to both users and systems.

Logistic Regression (LR)

LR serves as a linear classification method that utilizes a sigmoid function to calculate the probability of a class label. For the global dataset, LR achieved remarkable performance metrics, with a precision of 0.9996 and a recall of 0.9364. These results demonstrate that LR is effective for capturing a large proportion of phishing URLs while minimizing the number of false positives. However, in the local dataset, where phishing attempts are more advanced and resemble legitimate URLs, LR's performance significantly declined, yielding a precision of 0.8022 and recall of 0.7300. This shows reduced ability to distinguish between phishing URLs and legitimate URLs. In the local sample, there were 27 false negatives and 159 in the global dataset. The failures to identify sophisticated phishing attempts indicate that the model is not particularly adept at detecting sophisticated phishing attempts. This may indicate that risks remain undetected in practical scenarios of cybersecurity.

Support Vector Machine (SVM)

SVM is a powerful classification algorithm that constructs a hyperplane to optimally separate classes and it is most effective in spaces that are high-dimensional. Working on the global dataset, SVM performed remarkably well, achieving a precision of 0.9996 and a recall of 0.9440, demonstrating strong ability to reduce false positives and retain detection coverage. On the local dataset which contained more deceptively terse phishing URLs, SVM precision dropped to 0.8861 and recall to 0.7000. This suggests that while the algorithm can reliably reduce false alarms, it would miss a considerable number of phishing URLs, and thus, users may remain unprotected. Still, the relative precision is higher than that of other competing models, pointing to greater control of these systems over false positives and making SVM a useful algorithm in contexts where background noise must be minimized.

Iraqi Journal for Computers and Informatics
Information Technology and Communications University
Vol. 51, No. 2, 2025, pp. 252-267

DOI: https://doi.org/10.25195/ijci.v51i2.632
Print ISSN: 2313-190X, Online ISSN: 2520-4912

Random Forest (RF)

RF is an ensemble approach that constructs multiple decision trees to capture complicated and non-linear relationships embedded in the data. It achieved the best overall performance on the global dataset with a precision of 0.9984 and recall of 0.9808. This indicates its ability to phishing detection with low false positives and false negatives. On the more challenging local dataset, RF still performed better than traditional models, attaining precision of 0.8105 and recall of 0.7700. These numbers reflect stronger balance compared to other classifiers in identification of phishing URLs and false alarms. RF's ability to adaptively learn and its agnostic response to heuristic diversions makes it the best in real world phishing detection, particularly where attackers evolve their tactics to masquerade as normal traffic. Figure 4 presents a chart comparing all traditional models, showing RF's superior performance on both datasets.

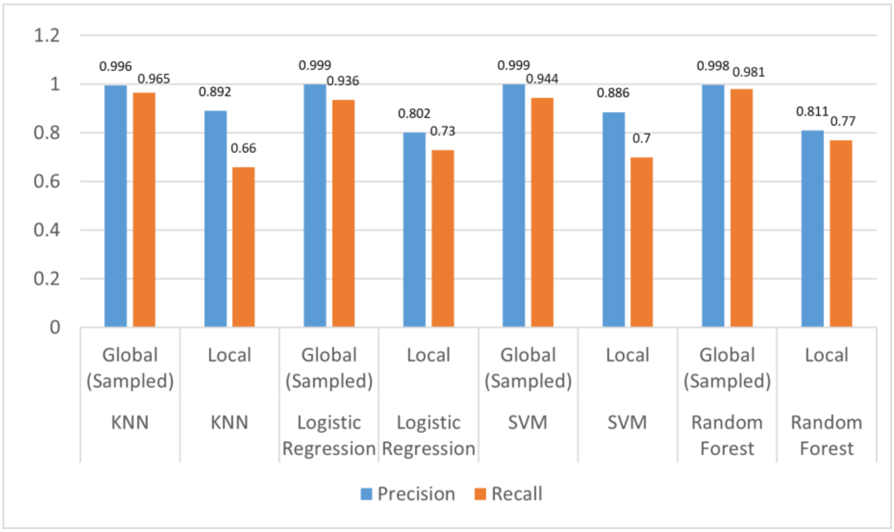


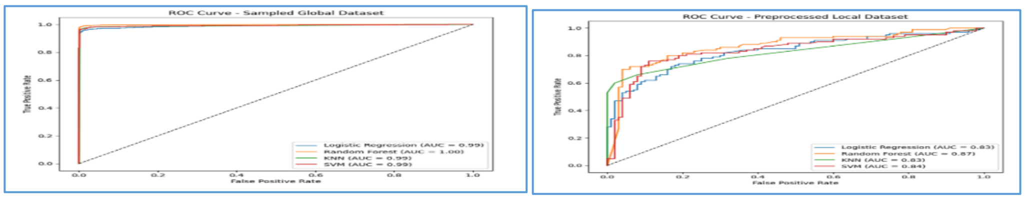
Fig 4: Performance of Traditional ML Models on the Global (Sampled) and Local Datasets

4.3 Traditional ML Analysis

To analyze model performance across different datasets, ROC curves were plotted in figure 5 for four traditional machine learning models: LR, KNN, SVM, and RF. The Area Under the Curve (AUC) was used as a comparative metric, reflecting each model's ability to distinguish between phishing and legitimate URLs.

In the sampled global dataset shown in figure 5 (a), all models performed remarkably well, with AUC values for RF (1.00), LR (0.99), KNN (0.99), and SVM (0.99). These nearly perfect scores indicate that all models, especially Random Forest, had excellent discriminatory power when trained on diverse, high-variance global data. The curves hug the top-left corner of the ROC space, suggesting high true positive rates with minimal false positives. This implies strong generalization ability and robust classification performance when phishing patterns are clearly distinguishable.

In the preprocessed local dataset shown in figure 5 (b), all models experienced a noticeable decline in performance, reflecting the increased difficulty of detecting phishing attacks that mimic legitimate URLs. The AUC scores are: RF (0.87), SVM (0.84), LR (0.83), KNN (0.83). The ROC curves in these cases are comparatively flatter and further away from the top left corner which shows that the class separability is lower. This reduction highlights that dataset context plays a crucial role in phishing detection evaluation, and relying on global benchmark datasets alone risks overestimating model effectiveness.



(a) (b)

Fig 5: The ROC Curve of Traditional ML models on Global Dataset (a) and Local Dataset (b)

Iraqi Journal for Computers and Informatics
Information Technology and Communications University
Vol. 51, No. 2, 2025, pp. 252-267
DOI: https://doi.org/10.25195/ijci.v51i2.632
Print ISSN: 2313-190X, Online ISSN: 2520-4912

Throughout both datasets, Random Forest maintained performance consistency, evidenced by achieving maximum AUC scores both locally and globally. This indicates a stochastic proficiency in generalizing across diverse features and complexities of phishing. On the opposing end, KNN and Logistic Regression models AUC scores locally, suggesting the models' insensitivity to very nuanced features. The findings emphasize the necessity of evaluating phishing detection models not only in global separated datasets, but in the more contextually and temporally realistic datasets that reflect the changing dynamics of phishing in the real world. Thus, the aim of this work is to answer what type of ML model is capable of detecting low variance and real-world phishing features. The investigation begins with feature engineering using conventional ML models.

4.4 Advanced Machine Learning

The conventional machine learning models like Logistic Regression, KNN, SVM, and Random Forests performed reasonably well, particularly when tested on the global dataset, but these models were tested with the more complex and realistic local dataset, their shortcomings were much clearer. Most importantly, these models did not do well at high recall while maintaining precision, which is to detection and response to phishing attacks is crucial. Phishing detection is a serious challenge, and these models routinely struggle to avoid both false positives and false negatives.

In the attempt to mitigate these challenges and bring the detection performance closer to practical cybersecurity benchmarks, advanced methods of machine learning that aim to improve the detection rate will be discussed in this section. These methods enhance traditional models with additional features, including model ensembling, gradient boosting, and hyperparameter tuning.

The following subsections present and evaluate:

A Voting Ensemble Classifier, which leverages the complementary strengths of multiple base learners.

Extreme Gradient Boosting (XGBoost) along with its tuned variant, known for handling imbalanced and noisy data.

By analyzing these advanced techniques, the work aims to determine whether they can outperform traditional models—especially on the local dataset—and provide a more practical solution for phishing website detection in dynamic and adversarial online environments.

4.4.1 Ensemble Voting Classifier

To improve phishing detection on the local dataset, a soft voting ensemble classifier was implemented, combining LR, SVM, and RF. Each model was trained on the preprocessed and SMOTE-balanced local dataset using the complete feature set. The confusion matrix (Figure 6) shows the classifier's detailed predictions.

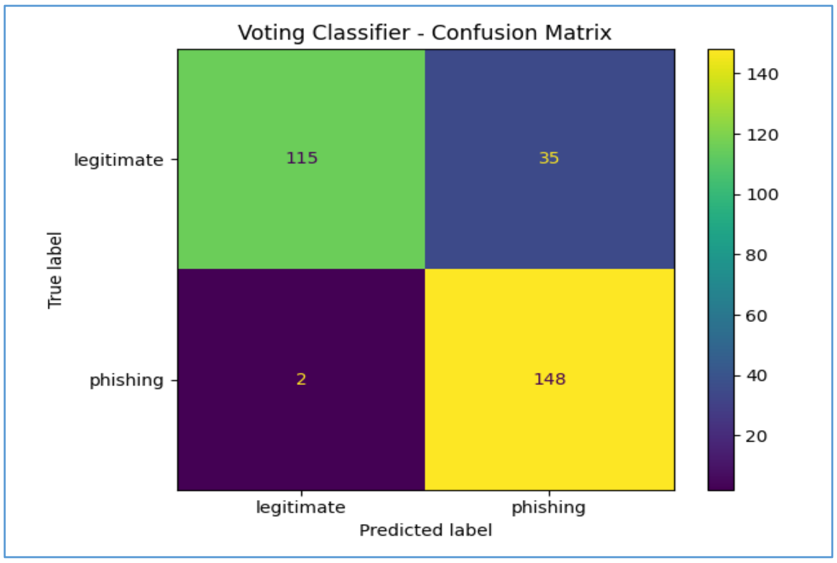


Fig 6: Soft voting ensemble classifier confusion matrix

Iraqi Journal for Computers and Informatics
Information Technology and Communications University
Vol. 51, No. 2, 2025, pp. 252-267

DOI: https://doi.org/10.25195/ijci.v51i2.632
Print ISSN: 2313-190X, Online ISSN: 2520-4912

As illustrated in Figure 6, the soft voting ensemble model identified 148 phishing URLs and 115 legitimate URLs, misclassifying 2 phishing URLs and producing 35 phishing URL false positives. Using these values and applying the previously defined equations for precision and recall, the model achieved a precision of 0.8087 and a recall of 0.9870. Such a high recall shows that the model is very effective in detecting phishing URLs and has a very low rate of missed calls. The lower precision reflects a moderate level of phishing URL false positives. The model's effectiveness in minimizing the detection of missed phishing URLs is a significant advantage for real-world application, especially in scenarios where sensitivity is prioritized.

4.4.2 Extreme Gradient Boosting (XGBoost) and Tuned (XGBoost)

In this study, XGBoost (or Extreme Gradient Boosting) was utilized as the advanced model for phishing URL detection, giving its phishing detection robustness and scalability. A framework for gradient boosting, XGBoost constructs and appends decision trees sequentially and each new tree attempts to reduce the errors from the preceding tree. The application of XGboost in cybersecurity is motivated by its fundamental benefits such as the ability to resist overfitting because of regularization, the ability to manage structured data, and the ability to perform parallel computation.

Training the model was based on the local phishing dataset that had undergone preprocessing and was balanced through the application of SMOTE. These measures allowed the model to learn phishing patterns more effectively because the input was of high quality and meaningful.

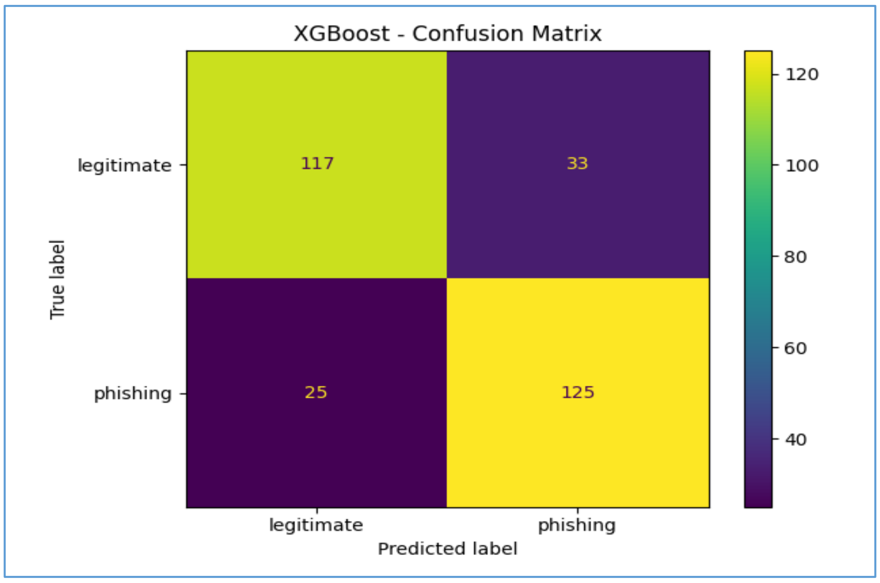


Fig 7: XGBoost Confusion Matrix

For the baseline evaluation, we applied the default configurations of XGBoost. The model managed to accurately classify 125 phishing and 117 legitimate URLs, albeit with 25 phishing URLs and 33 legitimate URLs misclassified in the opposing category. This is illustrated in Figure 7. The phishing precision reached 0.79 and the recall metric was 0.83. These values suggest the model detects phishing attempts with a reasonable level of correctness while controlling the number of false positives. The 25 false negatives, on the other hand, could pose a greater risk in a cybersecurity environment, where not identifying an existing threat is far more unfavorable.

In addressing this restriction, the XGBoost model was adjusted with grid search. The best performing set of parameters included: colsample_bytree=1.0, gamma=1, learning_rate=0.1, max_depth=3, n_estimators=100, and subsample=1.0. These parameters provided significant increases to model performance. The results from this model also showed that the tuned model correctly identified 149 phishing cases and only misclassified a single phishing URL as non-phishing. Although the tuned model increased the false positive count to 35 (shown in figure 8), it was still able to maintain high precision of 0.81, while recall was significantly increased to 0.99. This remarkable recall performance indicates that the model is very capable of reducing the phishing attempts that go unflagged.

Iraqi Journal for Computers and Informatics
Information Technology and Communications University
Vol. 51, No. 2, 2025, pp. 252-267
DOI: https://doi.org/10.25195/ijci.v51i2.632
Print ISSN: 2313-190X, Online ISSN: 2520-4912

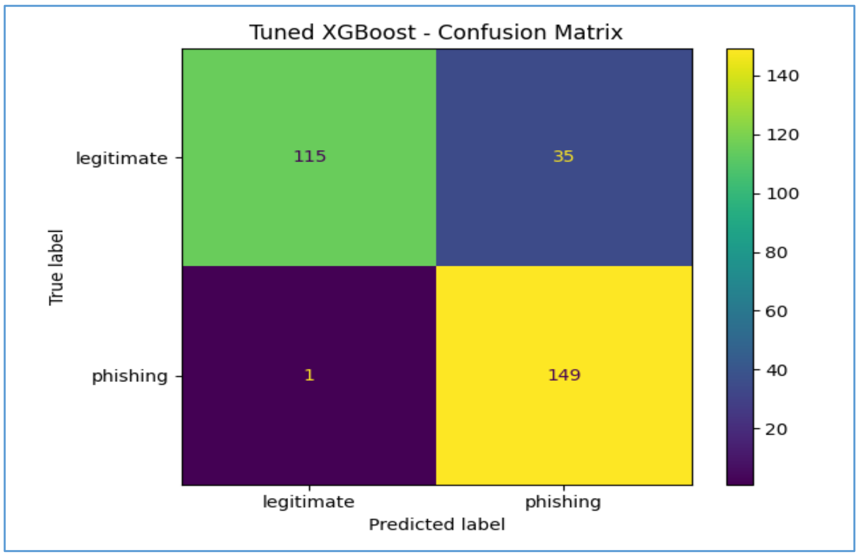


Fig 8: Tuned XGBoost Confusion Matrix

The findings emphasize the effectiveness of XGBoost, especially when finely tuned, for phishing detection. The reason for its outperforming all tested models lies in the high degree of sensitivity achieved through finely calibrated feature interactions, as well as its ability to capture intricate relationships among the features. The tuned XGBoost model, which achieved high recall and precision scoring on the complete local dataset, shows tremendous potential for implementation in practice where the detection of phishing threats is critical.

Table III provides a consolidated comparison of traditional and advanced models across global and local datasets, highlighting the clear advantages of ensemble and boosting methods in handling low-variance phishing attacks.

TABLE III. Performance Comparison of Traditional vs. Advanced ML Models on Global and Local Datasets

Model Type	**Model**	**Global Precision**	**Global Recall**	**ocal Precision**	**ocal Recall**
Traditional	Logistic Regression (LR)	0.999	0.936	0.802	0.730
	K-Nearest Neighbors (KNN)	0.996	0.965	0.892	0.660
	Support Vector Machine (SVM)	0.999	0.944	0.886	0.700
	Random Forest (RF)	0.998	0.981	0.811	0.770
Advanced	Voting Ensemble (LR+SVM+RF)	—	—	0.809	0.987
	XGBoost	—	—	0.79	0.83
	Tuned XGBoost	—	—	0.81	0.99

4.5 Error Analysis of False Negatives

Looking beyond overall performance metrics, the misclassifications themselves reveal important patterns. In the local dataset, many undetected phishing URLs used subtle tricks such as single-character substitutions (for example, replacing "o" with "0" or "I" with "1"), which made them look almost identical to legitimate domains. Another common tactic was subdomain spoofing, where a phishing URL embedded a trustworthy-looking fragment (e.g., *secure-bank.com.login.verify.co*) inside a longer, deceptive string. These strategies often led simpler models like Logistic Regression and KNN to misclassify phishing links as legitimate, since the differences in key features were minimal. More advanced models—particularly tuned XGBoost and the soft voting ensemble—reduced these errors, but did not eliminate them entirely. This underlines the need for future feature engineering that can directly target obfuscation techniques in order to further strengthen detection systems.

Iraqi Journal for Computers and Informatics
Information Technology and Communications University
Vol. 51, No. 2, 2025, pp. 252-267

DOI: https://doi.org/10.25195/ijci.v51i2.632
Print ISSN: 2313-190X, Online ISSN: 2520-4912

5. DISCUSSION AND LIMITATIONS

5.1 Discussion

This study analyzed how well both basic and advanced machine learning techniques could identify phishing URLs with a local dataset and a global dataset. While the local dataset was more realistic, the global dataset was more diverse. The study's results show that basic models, for example, RF, KNN, SVM, and LR, did well on the global dataset; most of them scoring precision and recall metrics over 90%. However, these models suffered a drastic drop in performance on the local dataset where the phishing URLs were disguised as authentic URLs. For instance, RF struggled to detect minor obfuscation, which is reflected in its 77% recall on the local set. However, outperforming RF were stronger, more stable, generalizable models like soft voting ensembles, and tuned XGBoost. Tuned XGBoost sustained high recall even under low variance conditions, attaining 99% recall and 81% precision on the local dataset, outperforming all traditional models. The soft voting classifier achieved a better tradeoff between precision and recall. This demonstrates its ability to leverage multiple base models. Results indicate that the use of ensemble learning strategies can significantly improve low-variance phishing detection as they are able to adjust to subtle shifts in URL structure.

5.2 Limitations

While the research contributes to the knowledge base, it poses several challenges. The local dataset was fabricated and small but it was intended to simulate actual behavior patterns concerning phishing URLs. Each phishing example was handcrafted from genuine sources through controlled obfuscation methods. This may not show all the different and unpredictable ways that phishers work in the wild. Additionally, while SMOTE was applied to balance the datasets, the use of synthetic data can sometimes introduce artifacts that may affect generalizability. Moreover, the study focused solely on URL-based lexical and structural features. As well as improving phishing detection in dynamic sophisticated attack scenarios, other more relevant factors such as behavioral indicators, temporal data, as well as content based features like HTML structure and text of the page were omitted. In addition, the evaluation did not focus on real time efficiency or the automated expense for the system, which could be significant for the use in real time systems.

6. CONCLUSION AND FUTURE WORKS

The research findings indicate that the phishing detection model performance is notably affected by dataset characteristics, feature variance, and class imbalance. Traditional models are often over-relying on well-separated data, while failing in more realistic scenarios. More sophisticated models, however, for example, XGBoost, tend to perform better after tuning and when trained on balanced datasets. Such models are more capable of detecting deeply obfuscated phishing websites. The results of this research make it abundantly clear that phishing detection systems, as a minimum, must be trained on realistic datasets and optimized using precision-focused metrics. Further research could extend this study by adding content-based features, utilizing deep learning models, and assessing the detection models in real-time to analyze their scalability and readiness for deployment. The main novelty of this study is the creation of a region-specific, low-variance dataset that reflects real-world phishing practices. Testing both traditional and advanced models on this dataset shows where classical methods fall short and highlights the stronger resilience of advanced ML approaches in realistic adversarial settings.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Anti-Phishing Working Group, "Phishing activity trends report: 2023," Anti-Phishing Working Group, 2023. Accessed: Jun. 13, 2025. [Online]. Available: https://apwg.org/.

[2] M. Alsharnouby, F. Alaca, and S. Chiasson, "Why phishing still works: User strategies for combating phishing attacks," *Int. J. Hum.-Comput. Stud.*, vol. 82, pp. 69–82, 2015.

[3] A. Basit, M. Zafar, X. Liu, A. R. Javed, Z. Jalil, and K. Kifayat, "A comprehensive survey of AI-enabled phishing attacks detection techniques," *Telecommun. Syst.*, vol. 76, no. 1, pp. 139–154, Jan. 2021, doi: 10.1007/s11235-020-00733-2.

[4] B. B. Gupta, A. Tewari, A. K. Jain, and D. P. Agrawal, "Fighting against phishing attacks: state of the art and

Iraqi Journal for Computers and Informatics
Information Technology and Communications University
Vol. 51, No. 2, 2025, pp. 252-267

DOI: https://doi.org/10.25195/ijci.v51i2.632
Print ISSN: 2313-190X, Online ISSN: 2520-4912

future challenges," *Neural Comput. Appl.*, vol. 28, no. 12, pp. 3629–3654, Dec. 2017, doi: 10.1007/s00521-016-2275-y.

[5] N. Q. Do, A. Selamat, O. Krejcar, E. Herrera-Viedma, and H. Fujita, "Deep learning for phishing detection: Taxonomy, current challenges and future directions," *Ieee Access*, vol. 10, pp. 36429–36463, 2022.

[6] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from URLs," *Expert Syst. Appl.*, vol. 117, pp. 345–357, 2019.

[7] Z. Alshingiti, R. Alaqel, J. Al-Muhtadi, Q. E. U. Haq, K. Saleem, and M. H. Faheem, "A Deep Learning-Based Phishing Detection System Using CNN, LSTM, and LSTM-CNN," *Electronics*, vol. 12, no. 1, Art. no. 1, Jan. 2023, doi: 10.3390/electronics12010232.

[8] P. Branco, L. Torgo, and R. Ribeiro, "A Survey of Predictive Modelling under Imbalanced Distributions," May 13, 2015, *arXiv*: arXiv:1505.01658. doi: 10.48550/arXiv.1505.01658.

[9] H. Le, Q. Pham, D. Sahoo, and S. C. H. Hoi, "URLNet: Learning a URL Representation with Deep Learning for Malicious URL Detection," Mar. 02, 2018, *arXiv*: arXiv:1802.03162. doi: 10.48550/arXiv.1802.03162.

[10] D. Minh Linh, H. D. Hung, H. Minh Chau, Q. Sy Vu, and T.-N. Tran, "Real-time phishing detection using deep learning methods by extensions," *Int. J. Electr. Comput. Eng. IJECE*, vol. 14, no. 3, p. 3021, Jun. 2024, doi: 10.11591/ijece.v14i3.pp3021-3035.

[11] M. A. Tamal, M. K. Islam, T. Bhuiyan, A. Sattar, and N. U. Prince, "Unveiling suspicious phishing attacks: enhancing detection with an optimal feature vectorization algorithm and supervised machine learning," *Front. Comput. Sci.*, vol. 6, p. 1428013, 2024.

[12] A. K. Jain and B. B. Gupta, "Phishing Detection: Analysis of Visual Similarity Based Approaches," *Secur. Commun. Netw.*, vol. 2017, pp. 1–20, 2017, doi: 10.1155/2017/5421046.

[13] G. Harinahalli Lokesh and G. BoreGowda, "Phishing website detection based on effective machine learning approach," *J. Cyber Secur. Technol.*, vol. 5, no. 1, pp. 1–14, Jan. 2021, doi: 10.1080/23742917.2020.1813396.

[14] A. Le, A. Markopoulou, and M. Faloutsos, "PhishDef: URL Names Say It All," Apr. 2011, pp. 191–195. doi: 10.1109/INFCOM.2011.5934995.

[15] S. Marchal, K. Saari, N. Singh, and N. Asokan, "Know Your Phish: Novel Techniques for Detecting Phishing Sites and Their Targets," in *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, Jun. 2016, pp. 323–333. doi: 10.1109/ICDCS.2016.10.

[16] A. K. Jain and B. B. Gupta, "Towards detection of phishing websites on client-side using machine learning based approach," *Telecommun. Syst.*, vol. 68, no. 4, pp. 687–700, Aug. 2018, doi: 10.1007/s11235-017-0414-0.

[17] A. Fajar, S. Yazid, and I. Budi, "Enhancing Phishing Detection through Feature Importance Analysis and Explainable AI: A Comparative Study of CatBoost, XGBoost, and EBM Models," Nov. 11, 2024, *arXiv*: arXiv:2411.06860. doi: 10.48550/arXiv.2411.06860.

[18] D. Sahoo, C. Liu, and S. C. H. Hoi, "Malicious URL Detection using Machine Learning: A Survey," Aug. 21, 2019, *arXiv*: arXiv:1701.07179. doi: 10.48550/arXiv.1701.07179.

[19] J. K. S. KAITHOLIKKAL, "Phishing URL dataset." Mendeley Data, Apr. 02, 2024. doi: 10.17632/VFSZBJ9B36.1.

[20] T. S. Othman and S. M. Abdullah, "An Intelligent Intrusion Detection System for Internet of Things Attack Detection and Identification Using Machine Learning," *ARO- Sci. J. KOYA Univ.*, vol. 11, no. 1, pp. 126–137, May 2023, doi: 10.14500/aro.11124.

[21] K. Adane and B. Beyene, "Machine learning and deep learning based phishing websites detection: The current gaps and next directions," *Rev. Comput. Eng. Res.*, vol. 9, no. 1, pp. 13–29, 2022.

[22] W. Sarasjati, S. Rustad, and H. A. S. Purwanto, "Phishing Detection Using Random Forest-Based Weighted Bootstrap Sampling and LASSO+ Feature Selection," *J. Homepage Httpiieta Orgjournalsijsse*, vol. 14, no. 6, pp. 1783–1794, 2024.