



Lightweight Hyperparameter Optimization Model for Enhancing Phishing Detection in IoT

Ansar YERTAYEV ¹, Hunaida AVVAD ^{1*}

Management Information Systems Department, İzmir Bakırçay University, İzmir, Türkiye

Received: 28/10/2024, **Revised:** 31/12/2024, **Accepted:** 18/03/2024, **Published:** 28/03/2025

Abstract

This study presents an enhanced machine learning approach that emphasizes the optimization of hyperparameters to improve phishing detection, particularly in resource-constrained environments like Internet of Things (IoT) devices. Phishing is considered one of the most dangerous cyberattacks, where attackers can reveal sensitive information about a user's identity, password, privacy, and properties. Machine learning techniques and tools play an important role in detecting phishing and have shown to be effective and advantageous methods for detection and classification, especially for the unified resource locator (URL). The proposed model presupposes a systematic approach for feature selection as well as finding the optimized hyperparameter values for the sake of increasing the detection quality while maintaining the low computational complexity of the process. This study examines how feature set selection from a training dataset and how hyperparameter tuning can significantly improve the performance of phishing attacks. Logistic regression, random forest, gradient boosting, support vector machine, and k-nearest neighbors are used in this study. According to the experiment, we found the best hyperparameter values for each classifier and comparative results of the implemented classification algorithms showed that the Random Forest classifier achieved the best performance with an accuracy of 98% and the best hyperparameter values were 200 for the `n_estimator` and 20 for the `max_depth`.

Keywords: classification, cyber security, hyperparameter tuning, industrial engineering, phishing attack.

IoT'de Kimlik Avı Tespitini Geliştirmeye Yönelik Hafif Hiperparametre Optimizasyon Modeli

Öz

Bu çalışma, oltalama (phishing) tespitini geliştirmek amacıyla hiperparametre optimizasyonuna odaklanan gelişmiş bir makine öğrenimi yaklaşımı sunmaktadır. Özellikle Nesnelerin İnterneti (IoT) cihazları gibi kaynak kısıtlı ortamlar için tasarlanan bu model, makine öğrenimi algoritmalarının hiperparametre değerlerini optimize ederek oltalama saldırılarının daha etkili bir şekilde tespit edilmesini hedeflemektedir. Oltalama, kullanıcıların kimlik bilgileri, şifreleri ve mahremiyetleri gibi hassas verilerini ele geçirmeyi amaçlayan en tehlikeli siber saldırılardan biri olarak kabul edilmektedir. Makine öğrenimi teknikleri, özellikle URL tabanlı oltalama tespitinde başarılı sonuçlar vermekte olup, bu çalışma özellik seçimi ve hiperparametre optimizasyonuna odaklanarak tespit kalitesini artırmayı amaçlamaktadır. Bu bağlamda, lojistik regresyon, rastgele orman, gradyan artırma, destek vektör makineleri ve en yakın k-komşu algoritmaları kullanılmıştır. Deneysel sonuçlar, rastgele orman algoritmasının %98 doğruluk oranı ile en iyi performansı sergilediğini ve en uygun hiperparametre değerlerinin `n_estimator` için 200 ve `max_depth` için 20 olduğunu göstermektedir. Bu çalışma, hiperparametre optimizasyonunun oltalama tespitinde kritik bir rol oynadığını ve doğru özellik seçiminin makine öğrenimi modellerinin performansını önemli ölçüde artırabileceğini ortaya koymaktadır. Elde edilen bulgular, kaynak kısıtlı ortamlarda güvenlik önlemlerini geliştirmek için önemli katkılar sunmaktadır.

Anahtar Kelimeler: sınıflandırma, siber güvenlik, hiperparametre ayarlama, endüstri mühendisliği, phishing saldırısı.

1. Introduction

With the exponential growth of the Internet of Things (IoT), the digital landscape has undergone a significant transformation. The modern industrial landscape depends extensively on IoT devices which operate in healthcare services and within smart cities and agriculture and manufacturing operations. The exponential growth in IoT device installation created newly security weaknesses for network systems. Phishing presents itself as a common cyber threat that-social engineering-tactics use to trick users into handing over their authentication details and bank information and personal data. IoT devices become easy targets for phishing attacks because they have minimal processing capabilities and weak security measures as well as limited human interaction during usage[1]. These attacks become more consequential because of expanding IoT device adoption in critical systems thus cybersecurity emerges as a critical issue.

The sophistication level of phishing attacks that aim at IoT systems continues to rise. An attacker takes advantage of advanced cloaking methods together with domain mimicry and obfuscated URLs methods. The attackers exploit specific vulnerabilities which affect IoT devices when conducting these attacks. Malicious URLs become virtually indiscernible to users because some IoT interfaces conceal their address bars. URLs. The massive heterogeneity found in IoT networks together with their large scale presents potential targets for attackers. The numerous weak security points provide attackers with the opportunity to exploit them.

Conventional phishing detection techniques, such Blacklists together with rule-based systems demonstrate limited adaptability toward dynamic phishing attacks because of their failure to adjust accordingly. particularly those targeting IoT environments. The inability of blacklisting to block zero-day phishing attacks becomes a problem since it detects no previously unknown domain names while rule-based systems provoke significant numbers of false positives when it applies to masked or dynamically created phishing URLs. The combination of high false-positive creation stems from security systems which detect phishing through obfuscated or dynamically produced phishing links. links [2], [3].

The technology of machine learning (ML) presents itself as an effective solution against the weaknesses of traditional phishing detection techniques. The analysis and classification of phishing URLs is possible through ML models dynamically, adapting to evolving threats. Modern ML research has produced hybrid detection methods based on recent developments. Algorithmic strategies that unite URL evaluation with textual content detection methods produce outstanding results accuracy[4]. Similarly, deep learning architectures, such as convolutional neural networks Deep learning networks known as CNNs demonstrate outstanding effectiveness in phishing detection applications. intricate patterns in large datasets [5]. The techniques require substantial processing power although they deliver their results. IoT devices lack sufficient processing capabilities since these methods need substantial computing power and make them unusable for constrained IoT systems [6].

The researchers in this study extend previous studies to handle specific challenges that IoT situations create. The proposed lightweight hyperparameter optimization model achieves accuracy-efficiency equilibrium which makes it suitable for processing power and energy capacity limited IoT devices. The system uses streamlined features and adjusts hyperparameters

for Support Vector Machines (SVM), Random Forests (RF) and Gradient Boosting (GB) machine learning algorithms. The model delivers practical real-time phishing detection capabilities to IoT networks because it enhances performance efficiency alongside detection precision. The research investigates universal data application across various datasets while examining operational efficiency as well as time-related complexities needed for IoT system implementations.

The research seeks to progress IoT phishing detection through evaluation of unmet research gaps. Numerous studies prioritize accuracy during their investigations yet they fail to consider the practical system limitations of IoT including power usage and response delay times[1]. The proposed model fills this existing gap through its efficient IoT-specific solution which helps enhance the ongoing security measures against phishing threats in IoT ecosystems.

The datasets in this research were not designed for IoT environments but the underlying structure of phishing URLs exists similarly in all platforms which include IoT devices. The extracted features which include URL length, suspicious characters, and domain anomalies have relevance for URLs targeting IoT-based services. These datasets can help optimize phishing detection models toward developing efficient lightweight algorithms which will operate properly in resource-limited IoT devices. The research establishes new methods to strengthen IoT security through immediate phishing threat detection.

The main contributions of this paper are the ability select features from a training dataset for the detection and classification of phishing URLs, exactly suitable for IoT systems and to serve as a starting point for researchers and practitioners in developing solutions to phishing attack classification problems for systems with limited properties like IoT.

Literature Review

Phishing attacks remain a significant cybersecurity threat, increasingly targeting IoT devices due to their inherent vulnerabilities, such as limited computational power, low-security features, and unsupervised user interactions. These devices are often deployed in environments where traditional cybersecurity mechanisms are impractical, necessitating the development of lightweight, efficient, and adaptable solutions. Existing research has made substantial progress in phishing detection using machine learning (ML) techniques, yet critical gaps remain in adapting these methods for resource-constrained IoT environments.

The initial systems for detecting phishing mainly depended on blacklist approaches that compared URLs against pre-established databases containing known phishing sites[2], [7], [8]. Operational efficiency is low for these systems because they cannot find zero-day phishing attempts or URLs that use disguising methods. The traditional rule-based system approaches [9], [10] produce numerous false positives because they depend on pre-defined heuristics that fail to detect new phishing methods. generalize across evolving phishing techniques.

The field of ML now offers different methods for phishing detection which exhibit better accuracy rates. The identification of legitimate and phishing URLs through Random Forest (RF) models as well as Support Vector Machines (SVM) and Gradient Boosting (GB) has proven effective according to research [3], [4], [11], [12]. RF models maintain robustness even in the presence of noise whereas other models are affected by this type of interference. The

accuracy level of SVM reaches its peak when identifying decision boundaries that are complex in nature. The computational demands coupled with the memory requirements of such models makes them inappropriate for IoT devices.

The development of lightweight models represents an effective solution to address specific IoT limitations according to research papers. [2] Employed the Rabbit Optimization Algorithm (ROA) functioned for feature selection to enhance phishing detection accuracy through reduced computational requirements. [1] paper describes a fog-based ML framework which distributes computational tasks between multiple devices and is achieving 99.37% accuracy in edge devices. The implementation of these methods proves the necessity of optimization strategies that maintain IoT systems between performance excellence and operational efficiency.

The detection capabilities of phishing have improved through deep learning models that include convolutional neural networks (CNNs) and probabilistic neural networks (PNNs) as described in [3] and [5]. The models demonstrate exceptional performance in identifying complex patterns from large datasets which results in 99% accuracy levels. Their computing requirements make deep learning models unsuitable for implementing real-time IoT systems. Research is needed to adequately solve the scalability and latency problems of deep learning adaptation for IoT systems through approaches like edge computing integration and distributed frameworks.

The previously mentioned methods prove effective but fail to consider vital aspects that would be relevant to IoT technological situations. Deploying these solutions on IoT devices depends on energy consumption levels as well as time complexity and memory usage factors. The conventional approach of ML together with deep learning models focuses on achieving high accuracy although they ignore the practical restrictions which arise with IoT's restricted resource availability. Such untested models on diverse datasets create limitations for their real-world performance because their effectiveness becomes restricted[6].

Phishing IoT devices create new phishing paths since they need basic user involvement but have limited processing capability. The automatic connections and embedded applications in IoT endpoints increase device vulnerability because they enable the receipt of malicious URLs which are distributed through software updates and compromised cloud services and SMS-based phishing. The IoT devices need security from phishing attacks because presently available phishing detection systems primarily protect web-based phishing yet maintain potential for modifying these solutions for similar purposes. The research datasets serve as foundational knowledge for phishing detection which can be transformed for use within IoT security platforms to find hazardous URLs within active operations.

The current study develops a streamlined hyperparameter optimization model which fits IoT contexts. This model opts for streamlined approaches involving feature selection and hyperparameter refinement since it seeks to boost performance alongside reducing computational complexity when compared to deep learning frameworks. The model combines SVM, RF, and GB classifiers to deliver high accuracy results without reducing its operational efficiency. The study fills knowledge gaps by documenting energy usage and computational performance which proves the model suitable for IoT system real-time implementation. Multiple datasets serve as assessment platforms for the model to demonstrate its wide-ranging capabilities when detecting different forms of phishing.

Basic Concepts and Techniques for Phishing Attacks

A common technique used for phishing attacks is to create a large number of URLs with all sorts of different variants. This provides a distraction for regular Internet users, which means that the likelihood of a successful phishing attack increases by several times the order of magnitude. In the URL string, introduce multiple slashes that point to multiple directories in the URL and look correct to inexperienced users of IoT devices. Similarly, introducing a few dots and some characters/digits in the domain name to create multiple sub-domains gives the impression of a correct URL.

Such generated malicious URLs very often replace alphanumeric characters with other characters, i.e. Unicode characters and/or hexadecimal character representation. English text has a relatively low entropy, i.e. it is predictable, and entropy changes more when different characters are introduced. Hence, the use of entropy can lead to the detection and correct classification of malicious URLs.

Considering such aspects of phishing attacks, this paper focuses on proposing a lightweight URL representation and a better-performance algorithm for detecting and classifying phishing URLs in IoT systems. Cybercriminals during a phishing attack very often deliver a malicious URL using a common application (email, Telegram, Twitter, Facebook, etc.). If an inexperienced Internet user accesses the phishing URL, the malicious activity will work in favor of the cybercriminal.

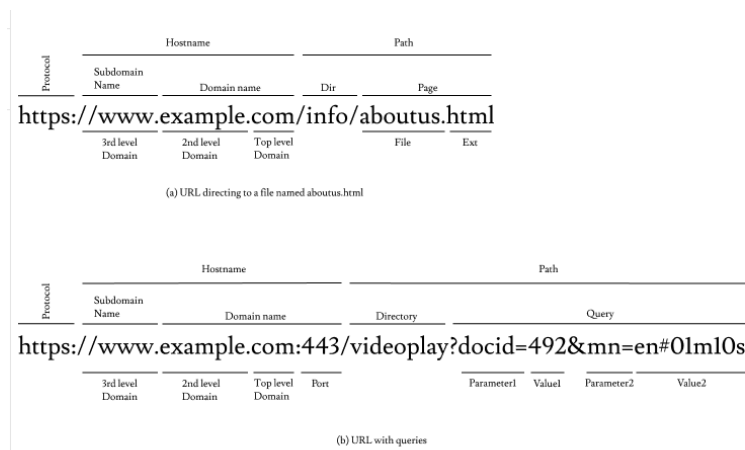


Figure 1. General view of the unified resource index form and its parts

2. Methodology

The objective of this study is to build a machine-learning model to classify URLs as either phishing or legitimate. This classification helps in identifying malicious websites and preventing phishing attacks on IoT devices. The dataset used for this study contains various features extracted from URLs, which are utilized to train the model.

Feature selection for phishing attack detection:

In [8], [13], [14], [15], [16], [17], the authors investigated some approaches for detecting phishing URLs. These approaches utilize several features extracted from URLs in practice. In this study, the proposed set of attributes was built by considering the nature of phishing attacks and projecting them onto URLs, e.g., phishers try to confuse ordinary users of IoT devices by making URLs unreadable and unintelligible. As a result, the generated phishing URLs become larger in number and use different characters/numbers than the correct addresses.

Based on this fact, in this study, we used such types of features measuring the length of some parts of the URL, the number of characters/digits, and features related to HTTP/S. For a better understanding of the structure, it is recommended to look at Figure 1.

Datasets and Data Pre-Processing:

The study used datasets from publicly available repositories that were created for web-based phishing detection but serve the purpose of this research for URL detection in phishing environments. The extraction of features and classification methods in this work applies to IoT environments that employ URL-based deceptive methods in phishing attacks even though the datasets lack IoT-specific origins. Research enhancement can be attained by implementing IoT-specific datasets that reflect IoT-specific vulnerabilities like IoT device attack paths and IoT network attack sequences.

Libraries such as Pandas, matplotlib, and Seaborn were imported for data manipulation and visualization. The datasets were downloaded from Kaggle dataset1; (<https://www.kaggle.com/code/akashkr/phishing-url-eda-and-modelling#Dataset-description>) and dataset2; (<https://www.kaggle.com/datasets/manishkc06/web-page-phishing-detection>). Dataset 1 contains 11,430 rows and 89 columns; it is a balanced data set with 50% of the dataset labeled as fishing and 50% as legitimate. Dataset 2 contains 11,481 rows, with 5,741 labeled as phishing and 5,740 as not phishing which means it is a balanced dataset. Both datasets include the same features. We calculated basic statistics for both datasets to understand the distribution of features. Then we checked missing values, and it was found that there were no missing values in both datasets.

Feature Selection:

A correlation matrix is constructed to identify the relationship between the features and the target variable status. The matrix here is a table, its cells contain correlation coefficient values when 1 represents a strong relationship between variables, 0 represents a neutral relationship, and -1 represents a negative relationship. Also, a heatmap is generated to visualize the correlations. Then the features with a correlation absolute value greater than 0.2 with the target variable are selected. This resulted in selecting 23 features that had a significant correlation with the status. To focus on features with a meaningful impact on the target variable, a threshold was applied, selecting only those features with an absolute correlation value greater than 0.2. This approach resulted in identifying 23 features that exhibited significant correlations with the status, highlighting their potential relevance for further analysis and model development.

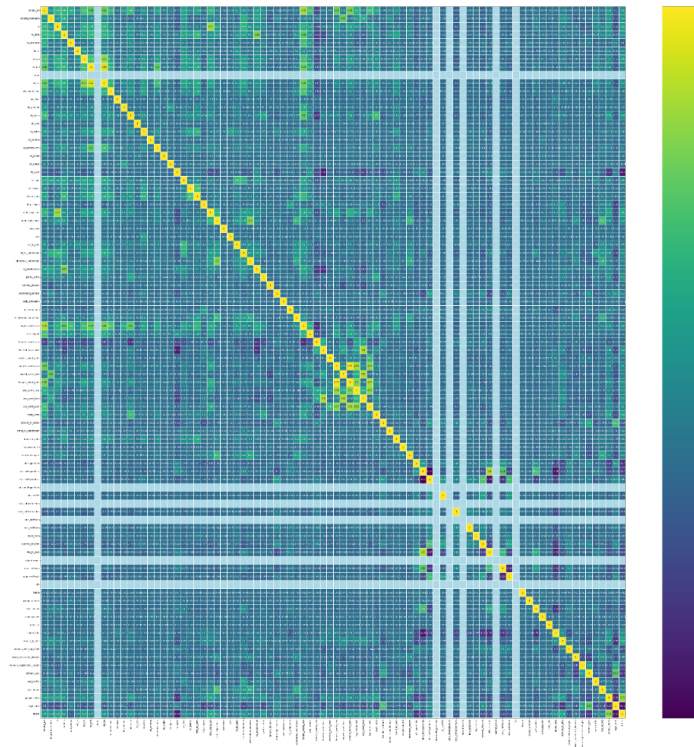


Figure 2. Correlation matrix heatmap for dataset1

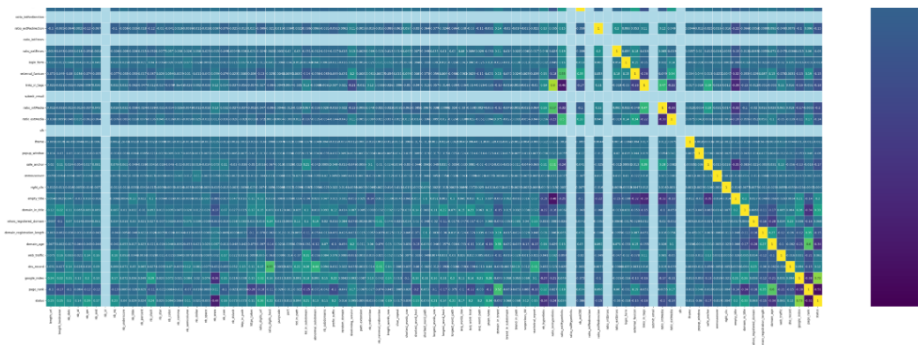


Figure 3. Correlation matrix heatmap for dataset2

As a result of the correlation matrix, the 23 features are as follows; F1: URL length, F2: Length hostname, F3: IP, F4: Number of dots, F5: Number of question marks, F6: Number of equals, F7: Number of slashes, F8: Number of “www”, F9: Ratio digits URL, F10: Ratio digits host, F11: TLD in a subdomain, F12: Prefix suffix, F13: Shortest word host, F14: Longest words raw, F15: Longest word path, F16: Phishing hints, F17: Number of hyperlinks, F18: Ratio int Hyperlinks, F19: Empty title, F20: Domain in the title, F21: Domain age, F22: Google index, F23: Page rank, F24: Status (0 for legitimate, 1 for phishing).

The primary goal of the heatmap is to identify patterns of correlation among features, which is crucial for multiple analytical purposes. High variable correlation manifests as an important

indicator when performing feature selection because it allows researchers to eliminate noncritical attributes. The heatmap acts as an important tool to detect multicollinearity that produces negative effects on regression-based statistical models and other approaches. Such heatmap analysis allows researchers to predetermine data preprocessing strategies that result in enhanced model interpretability together with increased efficiency.

Figure 2 and Figure 3 display the correlation matrix heatmaps of dataset1 and dataset2 respectively. The strong relationships between features become visible through yellow diagonal areas in the heatmap display. When multiple features show very high correlation approaching 1 then their data contains redundancy thus allowing one to drop those features to prevent multicollinearity issues. Features showing minimal correlation indicated by dark blue or purple tones in the heatmap should be considered for removal since they have minimal effect on the target variable. This simplifies the model structure. Features with moderate correlation levels (green to light yellow segments at 0.4 to 0.7) deliver the most valuable information because they avoid duplicate variance aspects. The grid pattern in the heatmap shows related variable clusters which enables analysts to select the most representative group member from each cluster to simplify their dataset. You can achieve greater model performance through variable selection using these principles to keep major impact-driven variables thus improving both model efficiency and interpretability and predictive power and lowering computational complexity.

Model Training and Evaluation:

The selected features are extracted from the original datasets to form the predictor variables, and the target variable (status) is mapped to numeric values, where 'legitimate' is mapped to 0 and 'phishing' is mapped to 1. Each dataset was split into training (75%) and test (25%) sets using *train_test_split* from *sklearn.model_selection*. Increasing the number of estimators improves the model performance but also increases the computational cost as well. The model is trained on the training set using the fit method of the Random Forest classifier. Random Forest Classifier (RFC) is an ensemble learning method suitable for classification and regression. The concept here is to create multiple decision trees at training time and for classification, the output of the Random Forest will be the class that is selected by most of the trees. RFC is used in many applications such as fraud detection in banks, email spam detection, and in the health sector in identifying a patient's disease.

The features from the original datasets were extracted to form the predictor variables, and the target variable (status) was mapped to numeric values: 'legitimate' as 0 and 'phishing' as 1. The dataset was split into training (75%) and test (25%) sets using *train_test_split* from *sklearn.model_selection*, with a random state of 42 to ensure reproducibility. Data scaling was performed using *StandardScaler* and training data was scaled with *fit_transform*, and the test data was scaled using *transform*. For model training and hyperparameter Tuning, the following classifiers were evaluated; Logistic Regression (LR), Random Forest (RF), Gradient Boosting (GB), Support Vector Machine (SVM), and K-Nearest Neighbors (KNN). Hyperparameter tuning was performed using *GridSearchCV* with 5-fold cross-validation. The parameter grids for each model are shown in Table 2.

Table 1. Confusion matrix for all experiments

Classifier	Class		Dataset1		Dataset2	
			Predicted		Predicted	
			False	True	False	True
Logistic Regression (LR)	False	Actual	1331	91	1338	99
	True		89	1347	86	1348
Random Forest (RF)	False	Actual	1367	55	1408	29
	True		59	1377	31	1403
Gradient Boosting (GB)	False	Actual	1361	61	1387	50
	True		57	1379	142	1292
Support Vector Machine (SVM)	False	Actual	1376	46	1400	37
	True		62	1374	54	1380
K-Nearest Neighbors (KNN)	False	Actual	1350	72	1382	55
	True		75	1361	68	1366

Table 2. Summary Table

	Classifier	Hyperparameter Values	Best Hyperparameter Values	Best Score	Accuracy	Weighted average accuracy	Label	Precision	Recall	F1 score
Dataset1	Logistic Regression (LR)	C: [0.1, 1, 10]	C: 1	0.9319	0.9370	0.94	0 1	0.94 0.94	0.94 0.94	0.94 0.94
	Random Forest (RF)	n_estimators: [100, 200] max_depth: [None, 10, 20]	n_estimators: 200, max_depth: 20	0.9615	0.960	0.96	0 1	0.96 0.96	0.96 0.96	0.96 0.96
	Gradient Boosting (GB)	n_estimators: [100, 200] learning_rate: [0.01, 0.1, 1]	n_estimators: 200, learning_rate: 0.1	0.9591	0.9587	0.96	0 1	0.96 0.96	0.96 0.96	0.96 0.96
	Support Vector Machine (SVM)	C: [0.1, 1, 10] kernel: ['linear', 'rbf']	C: 10, kernel: 'rbf'	0.9565	0.9622	0.96	0 1	0.96 0.97	0.97 0.96	0.96 0.96
	K-Nearest Neighbors (KNN)	n_neighbors: [3, 5, 7, 9] p: [1, 2]	n_neighbors: 3, p: 1	0.9510	0.9486	0.95	0 1	0.95 0.95	0.95 0.95	0.95 0.95
	Dataset2	Logistic Regression (LR)	C: [0.1, 1, 10]	C: 1	0.9364	0.94	0.94	0 1	0.94 0.93	0.93 0.94
Random Forest (RF)		n_estimators: [100, 200] max_depth: [None, 10, 20]	n_estimators: 200, max_depth: 20	0.9743	0.98	0.98	0 1	0.98 0.98	0.98 0.98	0.98 0.98

Gradient Boosting (GB)	n_estimators: [100, 200] learning_rate: [0.01, 0.1, 1]	n_estimators: 200, learning_rate: 1	0.9735	0.93	0.93	0 1	0.91 0.96	0.97 0.90	0.94 0.93
Support Vector Machine (SVM)	C: [0.1, 1, 10] kernel: ['linear', 'rbf']	C: 10, kernel: 'rbf'	0.9659	0.97	0.97	0 1	0.96 0.97	0.97 0.96	0.97 0.97
K-Nearest Neighbors (KNN)	n_neighbors: [3, 5, 7, 9] p: [1, 2]	n_neighbors: 5, p: 1	0.9512	0.96	0.96	0 1	0.95 0.96	0.96 0.95	0.96 0.96

Some explanations about the hyperparameter values from Table 2 for each classifier:

- In LR, C is the inverse of the regularization parameter, and it determines how regularization (penalty on large model coefficients) the model applies; a lower C results in more regularization, and a higher C results in less regularization.
- In RF, $n_estimators$ control the number of trees in the forest; more trees usually mean better accuracy but higher computation. The max_depth limits the depth of each tree; higher values lead to more complex trees and smaller values create simpler trees.
- In GB, $n_estimators$ indicate the number of boosting stages (or trees) in the ensemble; more trees can improve accuracy but may increase the risk of overfitting. The $learning_rate$ controls the contribution of each tree to the final model; smaller learning rates usually result in better accuracy but require more trees to compensate.
- In SVM, C controls the regularization and tolerance to misclassifications; a larger C makes the model more focused on classifying training points correctly, while a smaller C makes the model more focused on finding a simpler decision boundary. The $Kernel$ function shapes the decision boundary according to data distribution complexity because it determines a linear or non-linear boundary. RBF kernel demonstrates strong capability when dealing with data distributions that resist linear separability. The SVM can learn sophisticated decision limits because of this feature thus making it appropriate for situations with strongly non-linear relationships between features and targets.
- KNN uses $n_neighbors$ parameter to determine the neighbor count for predictions along with p parameter to select distance metrics. Smaller $n_neighbors$ values yield sensitive models while larger values produce generalized models. The distance measurement in the model uses p for notation where p equals one for Manhattan distance and p equals two for Euclidean distance. The selected distance metric impacts model performance levels because of the characteristics found within the data set.

3. Results and Discussion

To measure the model's performance, we used a confusion matrix. Table 1 shows the confusion matrix for the different classifiers that have been used in the study, and it has four values for

each classifier; True Positive (TP) which indicates that the model correctly predicted a true outcome. This means the instance was actually true, and the model predicted it as true. True Negative (TN) which indicates that the model correctly predicted a false outcome. This means the instance was actually false, and the model predicted it as false. False Positive (FP) which means that the model incorrectly predicted a true outcome. This means the instance was actually false, but the model predicted it as true. False Negative (FN) which means that the model incorrectly predicted a false outcome. This means the instance was actually true, but the model predicted it as false. From TP, TN, FP, and FN we can calculate different model performance metrics; accuracy, recall, precision, and F1 score. The definitions and equations for the performance metrics are as follows:

Accuracy: Accuracy is the ratio of correctly predicted positive and negative observations to the total number of observations.

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

Precision: Precision is the ratio of correctly predicted positive observations to the total predicted positives.

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

Recall (Sensitivity): Recall is the ratio of correctly predicted positive observations to all observations in the actual class.

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

F1 Score: The F1 score is the weighted average of Precision and Recall.

$$F1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

The experiments showed in Table 2 that in LR, the best hyperparameter value of C is 1 for both datasets and the accuracy for dataset1 and dataset2 are 93.7% and 94% respectively. In RF, the experiments showed that the best hyperparameter value of the $n_estimators$ is 200, the max_depth is 20 for both datasets, and the accuracies for dataset1 and dataset2 are 96% and 98% respectively. In GB, experiments showed that the best hyperparameter value of the $n_estimators$ is 200 for both datasets, the $learning_rate$ is 0.1 for dataset1 and 1 for dataset2, and the accuracy for dataset1 and dataset2 are 95.9% and 93% respectively. In SVM, experiments showed that the best hyperparameter value of the C is 10, the Kernel is rbf for both datasets, and the accuracy for dataset1 and dataset2 are 96.2% and 97% respectively. In KNN, experiments showed that the best hyperparameter value of the $n_neighbors$ is 3 for dataset1 and 5 for dataset2 and the p is 1 for both datasets, and the accuracy for dataset1 and dataset2 are 94.9% and 96% respectively.

As a Summary, Table 2 shows that RF has the highest overall accuracy (98%) and a strong F1-score (0.98), making it one of the best-performing classifiers in this study. SVM also performs well, with an accuracy of around 97% and F1-scores of 0.97. The best-achieved accuracy in this study used dataset2, while dataset1's best accuracy is 96.2% for SVM. LR accuracies are almost the same for the two datasets with 93.7% and 94%, RF performs better on dataset2 with a 2% difference, GB performs better on dataset1 with almost a 3% difference, and SVM performs almost the same on both datasets with 96.2 and 97%, and KNN performs better on dataset2 with 1.2% difference. LR achieves the lowest performance of 93.7% accuracy and a reasonable F1-score of 0.94 on dataset1 but is slightly less accurate than the tree-based methods and SVM. GB achieves the lowest performance of 93% accuracy and a reasonable F1-score of 0.94 on dataset 2. LR and GB still performed well but were outperformed by the other classifiers.

In conclusion, RF and SVM with *rbf* kernel provide the best balance between accuracy and generalization, while LR works well if data is linearly separable. KNN is slightly less effective in this context.

Table 2 shows also the best hyperparameter values to achieve the best score. For LR, the best-achieved result is when the hyperparameter value for C is 1, which provides the right balance between regularization and accuracy. Regularization parameter $C = 1$ appears to offer the best performance by avoiding overfitting and underfitting. In RF, the best values found were $n_estimators = 200$ and $max_depth = 20$, indicating a large ensemble of deep trees performs best. In GB, the best combination is $n_estimators = 200$ and $learning_rate = 0.1$ for dataset1 and 1 for dataset2, which balances the model's convergence and accuracy. In SVM, the best values are $C = 10$ and kernel = '*rbf*', indicating a more flexible decision boundary (non-linear kernel) and higher regularization. In KNN, the best values are $n_neighbors = 3$ for dataset1 and 5 for dataset2 and $p = 1$ (Manhattan distance).

All the classifiers in the study achieved better results with dataset2 except GB; the lower accuracy of dataset2 (93%) compared to dataset1 (95.9%), despite both using $n_estimators = 200$, is primarily due to differences in learning rate, data complexity, overfitting risk, and feature distribution. Dataset2's higher learning rate (1.0) causes aggressive weight updates, leading to suboptimal convergence, while dataset1's lower learning rate (0.1) ensures stable learning and better accuracy. Additionally, dataset2 may contain more noise, overlapping classes, or imbalanced data, making generalization harder. The high $n_estimators$ setting can also lead to overfitting, especially if dataset2 has fewer or noisier samples. Moreover, differences in feature distribution may weaken classification performance if dataset2 lacks strong distinguishing features. To improve accuracy, reducing the learning rate (e.g., 0.1 or 0.05), addressing data imbalance, enhancing feature selection, and testing alternative models like XGBoost or LightGBM could provide better results.

Different evaluation models for phishing URL detection techniques are presented in Table 2. The RF classifier achieved the highest accuracy level of 98% on test data which demonstrates its effective ability to detect phishing threats. The SVM model displayed robust evaluation results because it produced 97% accuracy. The research revealed that Random Forest together with SVM proved to be optimum options for malicious URL detection because of their exceptional performance abilities. Research efforts should concentrate on combining

sophisticated machine learning methods together with feature selection approaches to enhance the reliability and speed of phishing threat detection systems.

This research established a lightweight classification model which improved web address understanding and categorization, recognition and categorization of malicious web addresses. This model was focused on IoT. The selection of ideal features alongside the best hyperparameter values enabled device identification. Improving phishing prevention by utilizing training data sets allowed the model to enhance its performance. Experimental The outcome of our model demonstrates excellent phishing URL detection accuracy reaching 98 percent. The effectiveness of our method for identifying phishing risks successfully appears through these results. The proposed method shows great suitability for implementing on IoT devices through its emphasis on efficient algorithms. This study makes a major advancement to phishing research through its findings. The proposed solution stands out as an effective practical detection method that suits IoT devices operating within specific constraints, needs of IoT environments. Further research should expand from this work by developing new methods to address phishing threats. Future research will incorporate new advanced machine learning techniques combined with better approaches for feature selection to enhance the security system. Future detection systems for phishing attacks can become more strenuous and effective. Finally, the Random Forest classification model presented in this paper represents a significant leap forward in detecting phishing attacks against cyber threats that target IoT devices at large.

This research proves the effectiveness of hyperparameter optimization in phishing detection, yet it faces a crucial drawback because it does not utilize IoT-specific datasets. Current datasets that study generic phishing URLs need to be expanded to address the specific challenges that IoT environments create such as attacks targeting devices and phishing through APIs and deceptive techniques at the network layer. Upcoming research must create exclusive phishing datasets targeted at IoT systems because the goal is to confirm the effectiveness of machine learning methods for phishing detection in resource-limited scenarios. Real-world applications of these models integrated into IoT security frameworks accompanied by deployment testing shall generate essential findings about their operational feasibility and efficiency.

Ethics in Publishing

There are no ethical issues regarding the publication of this study.

Author Contributions

The authors did not declare any contribution.

References

- [1] M. gad Awwad, M. M. Ashour, E. S. A. Marzouk, and E. Abdelhalim, “Anti-Phishing approach for IoT system in Fog networks based on machine learning algorithms,” *Mansoura Engineering Journal*, vol. 49, no. 3, 2024, doi: 10.58491/2735-4202.3196.
- [2] L. Shahba, A. Heidary-Sharifabad, and M. Mollahoseini Ardakani, *Detection of fake web pages and phishing attacks with rabbit optimization algorithm*, vol. 81, no. 1. Springer US, 2025. doi: 10.1007/s11227-024-06658-w.
- [3] A. H. Alsadig and M. O. Ahmad, “Phishing URL Detection Using Deep Learning with CNN Models,” *2nd International Conference on Intelligent Cyber Physical Systems and Internet of Things, ICoICI 2024 - Proceedings*, no. ICoICI, pp. 768–775, 2024, doi: 10.1109/ICoICI62503.2024.10696243.
- [4] V. Malamas, P. Kotzanikolaou, K. Nomikos, C. Zonios, V. Tenentes, and M. Psarakis, “HA-CAAP: Hardware-Assisted Continuous Authentication and Attestation Protocol for IoT Based on Blockchain,” *IEEE Internet Things J*, vol. PP, no. 8, p. 1, 2025, doi: 10.1109/JIOT.2025.3530775.
- [5] H. Ghalechyan, E. Israyelyan, A. Arakelyan, G. Hovhannisyan, and A. Davtyan, “Phishing URL detection with neural networks: an empirical study,” *Sci Rep*, vol. 14, no. 1, p. 25134, 2024, doi: 10.1038/s41598-024-74725-6.
- [6] J. Hong *et al.*, “Combating phishing and script-based attacks: a novel machine learning framework for improved client-side security,” *Journal of Supercomputing*, vol. 81, no. 1, 2025, doi: 10.1007/s11227-024-06551-6.
- [7] P. Prakash, M. Kumar, R. Rao Kompella, and M. Gupta, “PhishNet: Predictive blacklisting to detect phishing attacks,” *Proceedings - IEEE INFOCOM*, pp. 1–5, 2010, doi: 10.1109/INFCOM.2010.5462216.
- [8] R. S. Rao and A. R. Pais, “Detection of phishing websites using an efficient feature-based machine learning framework,” *Neural Comput Appl*, vol. 31, pp. 3851–3873, 2019.
- [9] M. Moghimi and A. Y. Varjani, “New rule-based phishing detection method,” *Expert Syst Appl*, vol. 53, pp. 231–242, 2016, doi: 10.1016/j.eswa.2016.01.028.
- [10] K. S. Adewole, A. G. Akintola, S. A. Salihu, N. Faruk, and R. G. Jimoh, *Hybrid Rule-Based Model for Phishing URLs Detection*, vol. 285. Springer International Publishing, 2019. doi: 10.1007/978-3-030-23943-5_9.

- [11] A. K. Jain and B. B. Gupta, “A machine learning based approach for phishing detection using hyperlinks information,” *J Ambient Intell Humaniz Comput*, vol. 10, no. 5, pp. 2015–2028, 2019, doi: 10.1007/s12652-018-0798-z.
- [12] E. Gandotra and D. Gupta, “An Efficient Approach for Phishing Detection using Machine Learning,” pp. 239–253, 2021, doi: 10.1007/978-981-15-8711-5_12.
- [13] Y. A. Alsariera and others, “AI meta-learners and extra-trees algorithm for the detection of phishing websites,” *IEEE Access*, vol. 8, pp. 142532–142542, 2020.
- [14] J. H. Nezhad, M. V Jahan, M.-H. Tayarani-N, and Z. Sadrnezhad, “Analyzing new features of infected web content in detection of malicious webpages,” *ISC International Journal of Information Security*, vol. 9, no. 2, pp. 63–83, 2017.
- [15] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, “Machine learning based phishing detection from URLs,” *Expert Systems and Applications*, vol. 117, pp. 345–357, 2019.
- [16] N. M. Shekokar, C. Shah, M. Mahajan, and S. Rachh, “An ideal approach for detection and prevention of phishing attacks,” *Procedia Comput Sci*, pp. 49–82, 2015.
- [17] A. Tewari and B. B. Gupta, “Security, privacy and trust of different layers in Internet-of-Things (IoTs) framework,” *Future Gener. Comput.*, 2020.