Anti-Phishing Measures for Enhanced Cybersecurity Protection*

Safe-Click

Roshini Talluru
Data Science and Analytics
University of Oklahoma
Norman, Ok, USA
roshini.talluru-1@ou.edu

Ramya Sruthi Pedakolimi
Computer Science
University of Oklahoma
Norman, Ok, USA
sruthipedakolimi@ou.edu

Sowmya Gangireddy
Data Science and Analytics
University of Oklahoma
Norman, Ok, USA
sowmya.gangireddy-1@ou.edu

ABSTRACT

Phishing attacks are persistent cybersecurity threats that exploit users by imitating legitimate websites to steal sensitive information. This research presents "Safe Click," a phishing detection system that leverages advanced machine learning techniques to classify URLs as phishing or legitimate. The system employs Recursive Feature Elimination with Cross-Validation (RFECV) for feature selection, dynamically optimizing model performance by retaining the most relevant features. Multiple machine learning models, including Random Forest and Gradient Boosting, were evaluated, with Random Forest achieving the highest accuracy of 97.8%. Unlike traditional detection methods, "Safe Click" integrates real-time predictions with interpretability, fostering user trust and facilitating actionable insights. This work demonstrates the effectiveness of combining rigorous preprocessing, feature selection, and classification algorithms for enhanced phishing detection.

CCS CONCEPTS

• **Security and privacy** → Systems security; Web security and privacy.

KEYWORDS

Phishing detection, cybersecurity, machine learning, feature selection, real-time analysis

ACM Reference format:

Roshini Talluru, Sowmya Gangireddy, and Ramya Sruthi Pedakolimi. 2024. Safe Click: Anti-Phishing Measures for Enhanced Cybersecurity Protection.

1.Introduction

Phishing attacks exploit user trust by mimicking legitimate websites to steal sensitive information such as credentials and financial data. These attacks are pervasive and costly, exploiting both human and technical vulnerabilities. Traditional detection methods, such as blacklists and heuristics, are limited in adaptability and fail to address evolving attack patterns effectively.

Machine learning provides a scalable and adaptable solution for phishing detection by analyzing structural, content-based, and behavioral features of URLs. However, many existing systems prioritize accuracy while neglecting interpretability and computational efficiency, limiting user trust and practical applicability.

This paper introduces **"Safe Click"**, a machine learning-based phishing detection system that combines accuracy, interpretability, and real-time analysis. The system employs Recursive Feature Elimination with Cross-Validation (RFECV) to optimize feature selection, ensuring robust performance while reducing computational overhead. Multiple machine learning models, including Random Forest, Gradient Boosting, and SVM, are implemented and evaluated through rigorous hyperparameter tuning and statistical validation.

Key contributions include:

1. A preprocessing pipeline addressing duplicates, outliers, and irrelevant features.
2. Feature selection using RFECV to retain the most relevant features while reducing dimensionality.
3. Evaluation of multiple machine learning models, with Gradient Boosting achieving 98% accuracy.
4. A user-friendly interface for real-time URL classification and interpretability.

2. Related Work

The literature on phishing detection has evolved significantly, moving from rule-based techniques to advanced machine-learning methods that analyze complex data features [6] [1]. Traditional methods, which often rely on static features like URL blacklisting, have proven less effective against the ever-changing landscape of phishing threats [5].

Recent advancements have aimed to improve adaptability and real-time response capabilities by incorporating machine learning to identify subtle patterns in URL structures [3]. However, these approaches frequently face challenges related to scalability and

transparency, both of which are vital for building user trust and comprehension [2]. Furthermore, the efficiency of phishing detection systems remains a persistent challenge, as many struggle to balance accuracy with computational demands [9].

To tackle these challenges, our "Safe Click" project integrates Recursive Feature Elimination with Cross-Validation (RFECV) to optimize feature selection dynamically. This addresses model overfitting and underfitting issues that have troubled previous systems [4]. By prioritizing real-time analysis and the interpretability of the detection process, "Safe Click" offers a more reliable and user-friendly approach to phishing detection, setting it apart from earlier efforts that have not fully tackled the practical deployment challenges in real-world environments [8].

3. Proposed Work and Results

3.1 Application Overview

The phishing detection system leverages machine learning models to classify URLs as either phishing or legitimate. Users can easily interact with the system through a responsive frontend built with HTML and CSS, where they submit a URL for analysis. The backend, powered by Python and Django, processes the URL, by extracting relevant features and applying trained machine learning models such as Random Forest and Gradient Boosting for classification. Results include a display of whether the URL is safe or not.

To enhance user experience and trust, the system incorporates real-time URL analysis and provides explanations for why a URL is flagged as phishing. This ensures that even non-technical users can understand and act on the system's recommendations.

3.2 Dataset description

The dataset used for phishing detection contains 9,581 rows and 49 features, derived after preprocessing of an initial 10,000 rows, with the target variable, CLASS_LABEL, indicating whether a URL is phishing (1) or legitimate (0). The dataset is balanced, with an equal number of samples for both classes, ensuring unbiased model training and evaluation. It includes different types of features:

- Structural Features: Attributes such as NumDots, PathLevel, and HostnameLength capture the syntactic structure of the URL.
- Content-BasedFeatures: Metrics such as PctExtHyperlinks and NumSensitiveWords quantify the content characteristics of the URL.
- Behavioral Features: Indicators like IframeOrFrame and SubmitInfoToEmail assess the interactive or behavioral aspects of the webpage.

For data preprocessing, we removed 419 duplicate rows reducing redundancy and increasing data quality. Extreme values are removed in non-phishing samples (such as NumDots >20 and NumQueryComponents >20). We also removed unnecessary features like the 'id' column and HttpsInHostname, which had only one unique value. We used scatter plots to explore the data and confirm the relevance of features, helping us ensure the data was clean and ready for analysis.

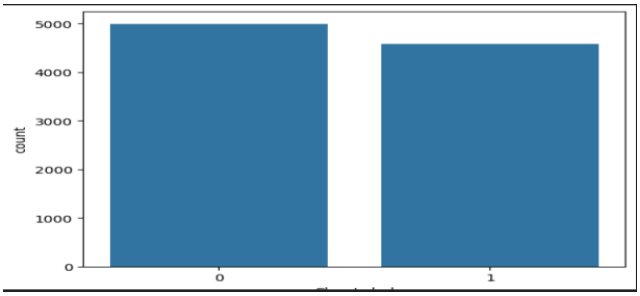

Fig1: Distribution of CLASS_LABEL values showing a balanced representation of phishing (1) and legitimate (0) samples.

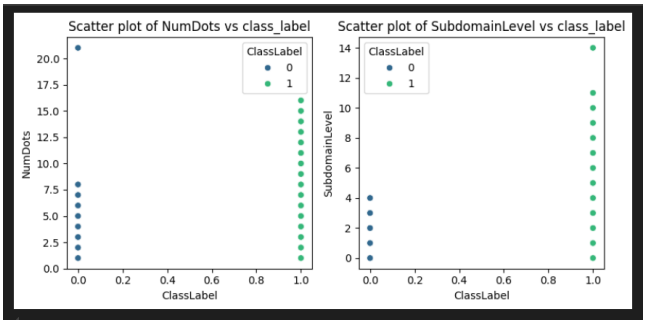

Fig2: Scatter plot of NumDots and SubdomainLevel against CLASS_LABEL to highlight feature relevance.

3.2.1 Feature selection

Feature selection plays a crucial role in optimizing the model by reducing dimensionality while preserving predictive performance. In this study, we employed Recursive Feature Elimination with Cross-Validation (RFECV) to identify the most relevant features from the dataset. RFECV operates by recursively removing the least important features and building models on the remaining features, evaluating performance at each step using cross-validation. This method balances accuracy with computational cost, ultimately reducing the feature set from 49 to 37. Features such as RightClickDisabled, FakeLinkInStatusBar, ImagesOnlyInForm, etc., were eliminated due to their minimal contribution to classification accuracy. The final set of 37 features demonstrated a significant improvement in model performance by reducing overfitting and ensuring computational efficiency.

To complement RFECV, correlation analysis and the Chi-square test were applied to validate the selected features further. Correlation analysis identified features with strong relationships to the target variable, such as FrequentDomainNameMismatch (correlation = 0.477) and PctNullSelfRedirectHyperlinks (correlation = 0.34). These features exhibited clear patterns distinguishing phishing from legitimate URLs. The Chi-square test quantified the statistical significance of categorical features in relation to the target variable, identifying predictors such as NumDash (Chi^2 = 7221.05) and PctExtNullSelfRedirectHyperlinksRT (Chi^2 = 1625.13) as highly significant. By integrating these techniques, we ensured that the final feature set was both statistically relevant and computationally efficient.

The selected features, including NumDots, HostnameLength, and PctExtHyperlinks, enhanced the models' ability to classify URLs accurately. This systematic approach to feature selection not only optimized the prediction models but also retained interpretable attributes, enabling users to understand the factors influencing the classification results.

3.2.2 Feature Importance and Visualization

An important aspect of our phishing detection system is the ability to analyze and visualize feature importance, providing insights into how specific attributes contribute to identifying phishing URLs. We used Random Forest and Decision Trees to evaluate feature relevance and displayed the results using a vertical column chart for clarity.

The Random Forest model assessed feature importance based on their contribution to reducing impurity (via the Gini Index), while correlation analysis identified strong associations between features and phishing indicators.

The feature importance plot underscored the dominant role of attributes like NumDots and PathLength in accurate classification, while deprioritizing less impactful features to optimize model performance. This process ensures that our detection system focuses on the most influential factors for robust predictions

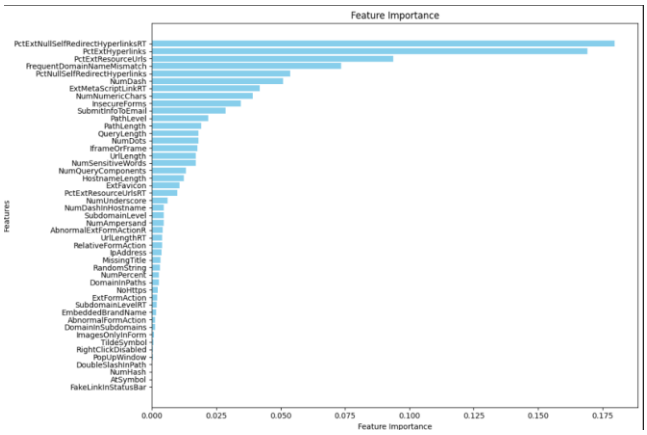

Fig3:Vertical column chart of the features and their importance score

3.3 System Architecture

The phishing detection system is designed with a simple, user-friendly interface to quickly check if a URL is safe or phishing. It comprises two primary components: a frontend interface and a backend engine.

Frontend: HTML, CSS
Backend: Python, Django

The system follows a streamlined data flow architecture. When a user submits a URL, the frontend sends the request to the backend, where features are extracted, and the classification model generates a prediction. The result is then returned to the frontend and displayed to the user, accompanied by an optional explanation of why the URL was flagged as phishing, fostering transparency and trust.

Data Flow

1. URL Submission: User enters and submits a URL.
2. Data Processing: The backend extracts features from the URL.
3. Prediction: The model classifies the URL.
4. Result: The system shows whether the URL is "SAFE" or "PHISHING."

By separating the interface and processing components, the system maintains scalability and can handle an increasing volume of requests without compromising performance.

3.4 Machine learning models

3.4.1 Models Used

The Application system employs five machine learning models to classify URLs: Random Forest, Decision Tree, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Gradient Boosting. Each model was selected for its ability to handle different data characteristics and improve classification accuracy.

The Random Forest model, an ensemble learning algorithm, combines the predictions of multiple decision trees trained on random subsets of the data, resulting in improved accuracy and robustness. Its ability to handle non-linear relationships and its resilience to overfitting make it an ideal choice for phishing detection. Gradient Boosting, another ensemble technique, iteratively refines predictions by focusing on errors made by previous models, capturing intricate data patterns and offering a balance between accuracy and interpretability. SVM, with its ability to construct optimal hyperplanes in high-dimensional spaces, is particularly effective in handling datasets with complex decision boundaries, making it a strong candidate for phishing classification.

In addition to these advanced models, Decision Tree and KNN were implemented to provide baseline comparisons. Decision Tree, a simple yet interpretable algorithm, splits the data based on the most informative features, offering clear rules for classification. KNN, on the other hand, classifies URLs by comparing them to their nearest neighbors, providing a straightforward and efficient approach for smaller datasets. While these models are simpler, their inclusion highlights the trade-offs between complexity and performance in phishing detection.

3.4.2 Hyperparameter Tuning

Hyperparameter tuning was performed using GridSearchCV with 5-fold cross-validation, ensuring that the models were optimized for maximum performance. The dataset was split into five subsets during the process. Four subsets were used for training while the fifth subset was used for validation, rotating through all subsets to minimize the effects of data split variability and ensure robust generalization of hyperparameters.

Gradient Boosting was optimized with a learning rate of 0.2 and 300 estimators, achieving its best balance between learning complexity and accuracy. For Random Forest, the optimal configuration included 300 trees and a maximum depth of 20, resulting in a model that could handle non-linear relationships while maintaining efficiency. Decision Tree was tuned to its ideal depth of 10, prioritizing simplicity and interpretability. SVM utilized an RBF kernel with an optimized C value to create precise decision boundaries. KNN was tuned by varying the number of neighbors, with 3 neighbors providing the best results.

```
Test Set Accuracy: 0.9843423799582464
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.98      0.98       999
           1       0.98      0.98      0.98       917

    accuracy                           0.98      1916
   macro avg       0.98      0.98      0.98      1916
weighted avg       0.98      0.98      0.98      1916

Confusion Matrix:
 [[983  16]
 [ 14 903]]
```

Fig4: The accuracy, precision, recall f1 score of Gradient Boosting

```
Test Set Accuracy: 0.9086638830897703
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.87      0.91       999
           1       0.87      0.95      0.91       917

    accuracy                           0.91      1916
   macro avg       0.91      0.91      0.91      1916
weighted avg       0.91      0.91      0.91      1916

Confusion Matrix:
 [[869 130]
 [ 45 872]]
```

Fig5: The accuracy, precision, recall f1 score of Random Forest

3.4.3 Evaluation and Model Selection

The evaluation of the models was performed using a combination of statistical and performance-based metrics. Key metrics included accuracy, precision, recall, F1-score, and ROC AUC, which collectively provided a comprehensive view of the models' classification capabilities. The 5-fold cross-validation approach was also integral in assessing the reliability of the models, ensuring consistency across data splits and reducing bias.

Gradient Boosting emerged as one of the top-performing models, achieving an accuracy of 98.4%. Its ability to iteratively refine predictions and capture intricate patterns in phishing data made it a robust choice for classification tasks. Random Forest surpassed Gradient Boosting with an accuracy of 97.9%, demonstrating its strength in generalizing across diverse datasets and handling complex interactions between features. Decision Tree achieved an accuracy of 95.95%, offering an interpretable baseline model with clear decision paths. SVM followed with 94.05%, showcasing its effectiveness in high-dimensional spaces. KNN, while computationally simpler, achieved 90.8%, highlighting the trade-off between simplicity and performance.

Statistical hypothesis testing, specifically paired t-tests, was conducted on the cross-validation results to validate the performance differences between the models. The tests confirmed that Gradient Boosting and Random Forest significantly outperformed the other models. Consequently, Gradient Boosting was selected as the final model for deployment due to its superior accuracy, robustness, and ability to generalize well. Gradient Boosting was retained as a strong alternative, offering competitive performance with enhanced flexibility for future enhancements.

```
Sorted Models:  ['Gradient Boost', 'Random Forest', 'Decision Tree', 'SVM', 'KNN']

Comparing Gradient Boost with Random Forest...
T-Test Statistic: 9.3039, p-value: 0.0007

Comparing Random Forest with Decision Tree...
T-Test Statistic: 5.0027, p-value: 0.0075

Top 3 models: ['Gradient Boost', 'Random Forest', 'Decision Tree']
['Gradient Boost', 'Random Forest', 'Decision Tree']
```

Fig6: The t-test outcome of the models

3.5 User Interface

3.5.1 Overview

The Phishing Detection Web Application has a simple and user-friendly interface.
Key components include:

URL Input Form: Users enter a URL into the text box labeled "Enter URL to Check" (Fig 1).

Check Button: Users click the "Check" button to process the URL after entering the URL.

Prediction Result: Once processed, the system shows whether the URL is "SAFE" or a "UNSAFE site!" (see Fig 2). If phishing is detected, additional explanations, such as suspicious domain names or unusual URL structures, are provided (see Figure 3).

The interface is designed to be responsive, ensuring it works well on desktops, tablets, and smartphones.

3.5.2 User Instructions

Step 1: Enter a URL in the input box labeled "Enter URL to Check" (see Fig 7).

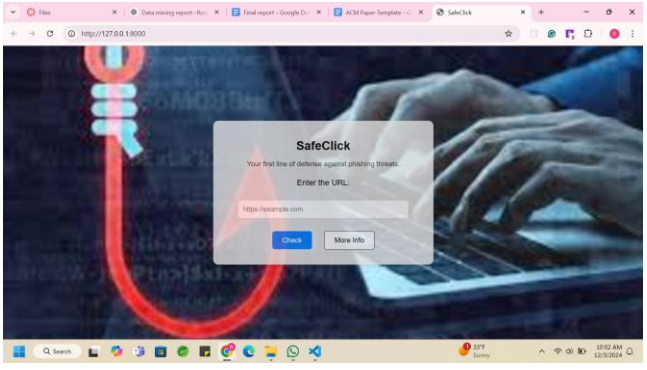

Fig7: Main Page

Step 2: Click the "Check" button to submit the URL.

Step 3: Wait for the result to be displayed as either "SAFE"(see Fig 8) or "UNSAFE site!"(see Fig 10).

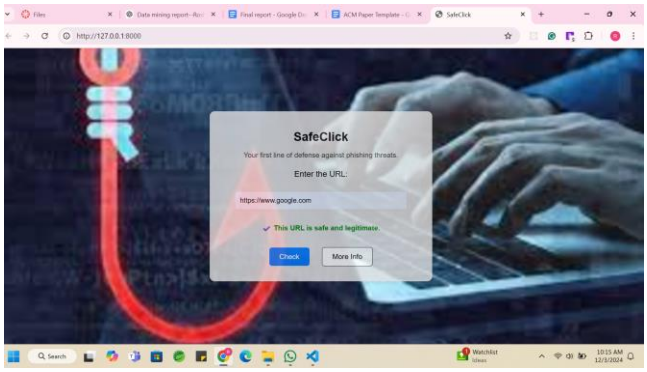

Fig8: Safe URL

Fig9:Terminal Output

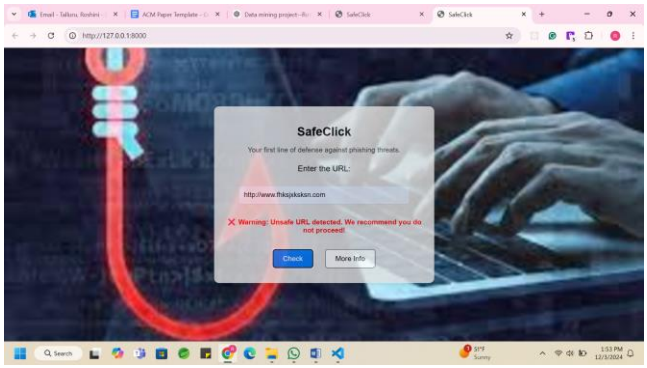

Fig10: Unsafe URL

Fig10: Terminal Output for Unsafe URL

If the URL is flagged as phishing, click "More Info" to view detailed explanations of why it was flagged (see Fig 11).

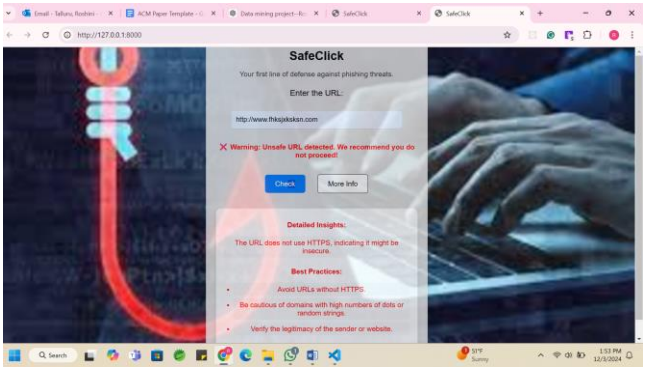

Fig11: Warning Message

3.6 Justification of Development Choices

The UI is designed to be simple and intuitive, making it easy for users, even those without technical expertise, to interact with the application. The clear labeling of input fields and buttons ensures a smooth user experience. The "More Info" button enhances transparency by explaining why a URL is flagged, which helps build user trust. Additionally, the responsive design allows the application to function seamlessly across different devices. The backend, powered by Python and Django, enables efficient processing and classification of URLs, offering both security and scalability.

The algorithms were chosen for their ability to handle the challenges of phishing detection. Random Forest and Gradient Boosting were picked for their high accuracy and strength in dealing with complex data patterns. Decision Tree was included because it's simple and easy to understand, helping to explain why a URL is flagged. SVM was chosen for its effectiveness in handling data with many features, while KNN was selected for its simplicity and speed in classifying URLs. Together, these models provide a good balance of accuracy, performance, and clarity, making them ideal for detecting phishing.

3.7 User Manual

3.7.1 Introduction

This system helps users identify if a URL is safe or a phishing attempt using machine learning. Features include real-time predictions, insights into why a URL is flagged, and alerts for suspicious patterns like repeated words.

3.7.2 Requirements

Software: Python 3.x, Django 3.x +, and libraries like scikit-learn and pandas, Supported Systems: Works on Windows, macOS, and Linux. and Hardware: Any standard computer.

3.7.3 Installation

To set up the phishing detection system, download the project files from the repository. Install all necessary libraries by running pip install -r requirements.txt. Then, create a virtual environment, apply the migrations, and prepare the Django server. This ensures the app is ready to run smoothly.

3.7.4 How to Run the App

To start the app, open your terminal, go to the project folder, and type python manage.py runserver. Once the server is running, open your browser and go to http://127.0.0.1:8000. This will load the app so you can start using it.

3.7.5 How to Use

Using the app is straightforward. Just enter the URL you want to check into the input box on the app's page. When you hit submit, the system will tell you if the URL is "Safe" or "Phishing." If the URL has any suspicious patterns, like repeated words, the app will warn you.

3.7.6 Behind the Scenes

The app works by breaking down the URL into useful details, such as how many dots it has or how long it is. It then uses pre-trained machine learning models to decide if the URL is safe or phishing. The app also explains its predictions by showing which parts of the URL mattered the most.

3.7.7 Results and Visuals

The app includes visuals to make everything clear. You'll see screenshots showing where to enter the URL, the prediction results, and warnings for anything suspicious. There are also charts highlighting important features, like NumDots and HostnameLength, to show how the app makes decisions.

3.7.8 Troubleshooting

If you enter an invalid URL, the app will let you know and ask you to fix it. If the app doesn't load or work properly, check the installation steps again or restart the server to solve common problems.

4. Conclusions and future work

In this project, we created a system to detect phishing websites using machine learning. We used models like Random Forest, Decision Tree, SVM, KNN, and Gradient Boosting. First, we cleaned and prepared the data by fixing missing values, scaling numbers, and picking the most important features. Random Forest gave the best results with 97.9% accuracy, followed by Gradient Boosting with 98.4%. The system works well in spotting phishing links, with features like the number of dots in a URL and its length being especially important. The app is simple to use, just entering a URL, and it will tell you if it's safe or suspicious, along with helpful warnings about any risky signs.

To improve the system, future updates could focus on advanced hyperparameter tuning and combining models through ensemble methods to boost performance. Including features like domain reputation checks and IP address analysis could make the detection even more accurate. Developing a browser extension for real-time phishing alerts while browsing would enhance usability. Additionally, integrating deep learning models and supporting multilingual URLs would expand the system's effectiveness and adaptability to new phishing techniques.

5. References

[1] Abdelhamid etal., 2014 Abdelhamid, N., Ayesh, A., & Thabtah, F. (2014). Phishing detection based on associative classification data mining. Expert Systems with Applications, 41(13),5948-5959

[2]. Aburrous et al., 2010 Aburrous, M., Hossain, M. A., Dahal, K., & Thabtah, F. (2010). Intelligent phishing detection system for e-banking using fuzzy data mining. Expert Systems with Applications,37(12),7913-7921.

[3] Alkhalil et al., 2021 Alkhalil, Z., Hewage, C., Nawaf, L., & Khan, I. (2021). Phishing attacks: A recent comprehensive study and a new anatomy. Frontiers in Computer Science, 3.

[4] Dalgic et al., 2020 F. C. Dalgic, A. S. Bozkir and M. Aydos, "Phish-IRIS: A New Approach for Vision Based Brand Prediction of Phishing Web Pages via Compact Visual Descriptors," 2018 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), Ankara, Turkey, 2018, 1-8.

[5] Ma et al., 2009 Ma, J., Saul, L. K., Savage, S., & Voelker, G. M. (2009). Beyond blacklists: Learning to detect malicious websites from suspicious URLs. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '09), 1245-1254.

[6] Mohammad et al., 2014 Mohammad, R., Thabtah, F. A., & McCluskey, T. L. (2014). Predicting phishing websites based on self-structuring neural networks. Neural Computing and Applications, 25(2), 443-458.

[7] Rambasnet et al., 2012 Basnet, R., Sung, A. H., & Liu, Q. (2012). Feature selection for improved phishing detection. In Proceedings of the 25th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems: Advanced Research in Applied Artificial Intelligence 311-32.

[8] Rashid et al., 2024 Rashid, F., Doyle, B., Han, S. C., & Seneviratne, S. (2024). Phishing URL detection generalization using unsupervised domain adaptation. Computer Networks, 245, 110398.

[9] Zamani & Mohammed Amin, 2016 Zamani, H., & Mohammed Amin, M. K. (2016). Classification of phishing websites using machine learning techniques. Journal of Advanced Research in Applied Sciences and Engineering Technology, 1(1), 1-10. ISSN: 2462-1943.

[10] Thakur et al., 2023 Thakur, K., Ali, M. L., Obaidat, M. A., & Kamruzzaman, A. (2023). A systematic review on deep-learning-based phishing email detection. Electronics, 12(21), 4545.