Identification of Phishing Webpage using URL based Features

Divya Prasanth Paraman
Indiana University

Pragadeesh Viralikattur Surendiran
Indiana University

Yudheksha Govindarajan Kavitha
Indiana University

project-dparaman-prvs-yugokavi

Abstract

The aim of this initiative is to tackle the demanding task of identifying advanced phishing threats through the application of a specific machine learning strategy. Utilizing a comprehensive dataset comprising 11,430 URLs, characterized by a variety of attributes, our objective is to develop and assess prediction models. This will be achieved using an assortment of classification methods, including Logistic Regression, Naive Bayes, Decision Tree, Random Forest, Gradient Boosting, and Support Vector Machines (SVM). These techniques are chosen due to their effectiveness in managing high-dimensional data and their capability to prevent overfitting. Additionally, if time allows, we plan to explore unsupervised techniques such as clustering and link analysis (using the page rank algorithm) to detect these harmful links and compare their effectiveness with supervised methods. Our ultimate aim is to enhance the precision of detection, focusing on reducing both false positives and negatives, and thus improve real-time protection against phishing attacks. The expected outcome is an advanced, agile system that not only strengthens security measures but also provides insights into the most indicative features of phishing activities, offering both immediate defense and strategic prevention

Keywords

Phishing Detection, Hyperparameter Tuning, Random Forest, Gradient Boosting, Support Vector Machines (SVM), Decision Tree, Naive Bayes, Logistic Regression Feature Engineering, Predictive Analytics, PCA and t-SNE Visualization DB-Scan Clustering,

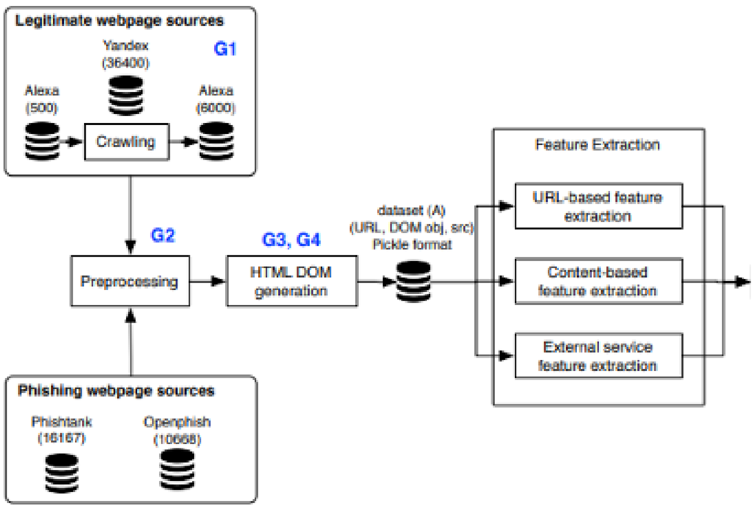
1 Introduction

In a time of increasing digital hazards, the issue of phishing—where online fraudsters skillfully pose as trustworthy sources to pilfer private data—requires a sophisticated strategy for cyber protection. Our initiative rises to this occasion by applying advanced machine learning methods, such as Logistic Regression, Naive Bayes, Decision Tree, Random Forest, Gradient Boosting, and SVM techniques. These are used to thoroughly scrutinize and categorize the "Web Page Phishing Detection" dataset sourced from Kaggle. The aim of our project is to delve into the complexities of web pages, creating a system capable of not just accurately identifying phishing efforts but also rapidly adjusting to their changing strategies.

2 Methods

2.1 Data Preparation

In this phase, we implemented the proposed methodology outlined earlier to gather a dataset for the experiments conducted in this study. As a result, we compiled a substantial dataset comprising 11,430 URLs, encompassing both phishing and legitimate instances.



Legitimate webpage sources
Yandex (36400)
G1
Alexa (500)
Alexa (6000)
Crawling
G2
G3, G4
dataset (A) (URL, DOM obj, src) Pickle format
Preprocessing
HTML DOM generation
Feature Extraction
URL-based feature extraction
Content-based feature extraction
External service feature extraction
Phishing webpage sources
Phishtank (16167)
Openphish (10668)

Figure 1: URL-based Feature Extraction

2.2 URL Collection

Legitimate website URLs are sourced from two primary platforms: Alexa's top 500 domains and Yandex. In the case of Alexa, the first 500 domains serve as initial points, leading to the collection of 12 URLs per domain and resulting in a total of 6,000 URLs. Additionally, active legitimate URLs from Sahingoz et al.'s dataset, originally obtained from Yandex, are incorporated. Phishing URLs are acquired from two distinct lists provided by Phishtank and Openphish. The gathered URLs undergo a preprocessing step to eliminate duplicates and inactive links, with a maximum of 12 URLs retained for each domain. These approved URLs are categorized based on their sources, such as Alexa or Phishtank. Subsequently, HTML Document Object Model (DOM) trees are generated for all approved URLs, forming an additional dataset denoted as dataset A. The HTML DOM Parser for Python is employed for this process, and the resulting dataset A is stored separately using the pickle module. The use of DOM trees allows for an efficient exploration of the structure and content of web pages associated with the URLs. The pickle module is utilized for serializing and de-serializing complex Python objects in the storage of dataset A.

2.3 Feature Extraction

The primary objective of our study is to investigate a wide range of popular features used for detecting phishing websites. We conducted a review of recent literature to assess the frequency of feature usage, identifying issues such as features being referred to by different names or measured in various ways. To address these inconsistencies, we implemented a two-step process: (1) identifying and removing duplicate features and (2) standardizing each feature to a single, commonly used measurement. This iterative approach resulted in a final set of 87 features.

The literature review revealed various classification schemes for website phishing features. Anti-phishing systems, particularly those based on machine learning, commonly use internal features with or without external features. Internal features (I) are extracted either directly from URLs (IU) or from the HTML content of corresponding web pages (IC). Hybrid systems combine features from both IC and IU. External features (E) serve as complementary and are often considered URL-based features, obtained through querying external services with URLs or URL domains.

In our study, we automated the feature extraction process using a Python script. This script takes a web page URL as input and generates a corresponding feature vector of size 87x1. To maintain the source URL's association with the dataset for further reference, we used the input URL as an index.

2.4 URL based features

Extracting features is a crucial step in identifying phishing URLs, and we've developed a Python program for this purpose. The following features have been identified to detect phishing URLs:

1. Presence of IP Address:
    - A binary feature is set to 1 if an IP address is detected in the URL, signalling potential phishing activity.
2. Existence of @ Symbol:
    - A binary feature is set to 1 if the '@' symbol is found in the URL, a technique commonly used by phishers.
3. Number of Dots in Hostname:
    - A binary feature is set to 1 if the number of dots in the URL exceeds 3, indicating a potential phishing attempt.
4. Domain Separation by Dash:
    - A binary feature is set to 1 if the domain name is separated by a dash (-), a tactic employed by phishers.
5. URL Redirection:
    - A binary feature is set to 1 if "//" appears in the URL path, suggesting possible redirection.
6. Presence of HTTPS Token:
    - A binary feature is set to 1 if the "HTTPS" token is present in the URL, a deceptive strategy used by phishers.
7. Submission of Information to Email:

- A binary feature is set to 1 if functions like ”mail ()” or ”mailto:” are identified in the URL.

8. Usage of URL Shortening Services (TinyURL):
   - A binary feature is set to 1 if the URL is shortened using services like TinyURL, a common practice in phishing.
9. Hostname Length:
   - A binary feature is set to 1 if the length of the URL’s host name exceeds 25 characters.
10. Presence of Sensitive Words:
    - A binary feature is set to 1 if certain sensitive words are observed in the URL.
11. Number of Slashes in URL:
    - A binary feature is set to 1 if the number of slashes in the URL surpasses 5.
12. Presence of Unicode:
    - A binary feature is set to 1 if Unicode characters are utilized in the URL.
13. Age of SSL Certificate:
    - A binary feature is set to 1 if the SSL certificate’s age exceeds a specified threshold.
14. URL of Anchor:
    - A binary feature is set to 1 if the ¡a¿ tag has a significant number of hyperlinks from other domains.
15. Usage of IFRAME:
    - A binary feature is set to 1 if the webpage incorporates the ”iframe” tag, a potential method for phishing.
16. Website Rank:
    - A binary feature is set to 1 if the website’s rank is greater than 100,000 in the Alexa database.

3 Evaluation Metrics

Certainly, let’s delve into details about four common evaluation metrics used in machine learning classification tasks: F1 Score, Accuracy, Recall, and Precision.

1. Accuracy:
   - Formula: (True Positives+True Negatives)/(True Positives+False Positives+True Negatives+False Negatives)
   - Accuracy measures the overall correctness of the model. It represents the ratio of correctly predicted instances to the total instances.
2. Precision:

- Formula: True Positives/(True Positives + False Positives)
- Precision, also known as Positive Predictive Value, measures the accuracy of the positive predictions. It is the ratio of correctly predicted positive observations to the total predicted positives.

3. Recall (Sensitivity or True Positive Rate):
   - Formula: True Positives/(True Positives + False Negatives)
   - Recall measures the ability of a model to capture all the relevant instances. It is the ratio of correctly predicted positive observations to all the actual positives.
4. F1 Score:
   - Formula: $2 \times$ (Precision $\times$ Recall)/(Precision + Recall)
   - F1 Score is the harmonic mean of Precision and Recall. It provides a balance between Precision and Recall, making it a useful metric when there is an uneven class distribution.

4 Machine Learning Models

4.1 Decision Tree

In the realm of classification and regression applications, the decision tree serves as a fundamental machine learning technique. The decision tree algorithm commences by identifying the feature that optimally partitions the information based on specific criteria, such as maximizing information gain or minimizing entropy, with respect to the two categories. This iterative process continues for each outcome group until a stopping criterion is fulfilled, which could be a predefined maximum tree depth or a specified quantity of samples per leaf.

4.2 Random Forest

Within machine learning, the prevalent practice involves employing ensemble learning techniques, with the random forest algorithm being a popular choice for both classification and regression tasks. This methodology entails the creation of multiple decision trees on randomly chosen subsets of the training data, with the outcomes of these trees amalgamated to formulate a final prediction. Each decision tree within a random forest is developed using a randomly selected subset of the features in the data, a strategy aimed at mitigating overfitting and enhancing the model's ability to generalize. Furthermore, the random sampling of training data contributes to minimizing variance and elevating the overall accuracy of the model.

4.3 Support Vector Machine (SVM)

SVM, a widely adopted machine learning algorithm, operates by identifying the most effective hyperplane to distinguish classes within the data. This hyperplane is strategically positioned to maximize the separation or margin between the two classes. The selection of the hyperplane is influenced by support vectors, denoting data points in close proximity to the decision boundary. SVMs find application in both binary and multiclass classification scenarios. Moreover, they exhibit versatility in addressing both linear and nonlinear data patterns through the utilization of diverse kernel functions.

4.4 Logistic Regression

Logistic regression is a statistical technique employed to predict the probability of a binary outcome, such as a yes or no response, based on one or more predictor variables. This method falls under the category of regression analysis, specifically designed for scenarios where the dependent variable is categorical, and the independent variables can be either categorical or continuous. In logistic regression, the logistic function is applied to model the relationship between the outcome variable and predictor variables, aiming to capture the likelihood of the binary outcome.

4.5 Naive Bayes

Naive Bayes stands as a widely employed statistical method in machine learning, particularly for classification purposes. It relies on Bayes' theorem, a fundamental concept in probability theory. The core idea behind Naive Bayes is to assess the likelihood of a data point being associated with a specific class based on its features. This involves determining the probability distribution for each category and feature using a training dataset. Subsequently, these distributions are leveraged to compute the likelihood of a new data point belonging to each class.

4.6 Gradient Boosting

In the realm of classification problems, the gradient boosting classifier, a type of machine learning method, is employed. This approach involves constructing a series of decision trees that are trained sequentially, with each subsequent tree aiming to rectify the errors of its predecessor. The algorithm initiates by building an initial decision tree and making predictions based on it. The discrepancies or residuals from these predictions are then utilized to train a new decision tree, which is added to the ensemble. This iterative process is repeated for a defined number of iterations or until a predetermined level of accuracy is attained.

5 Experiment and Results

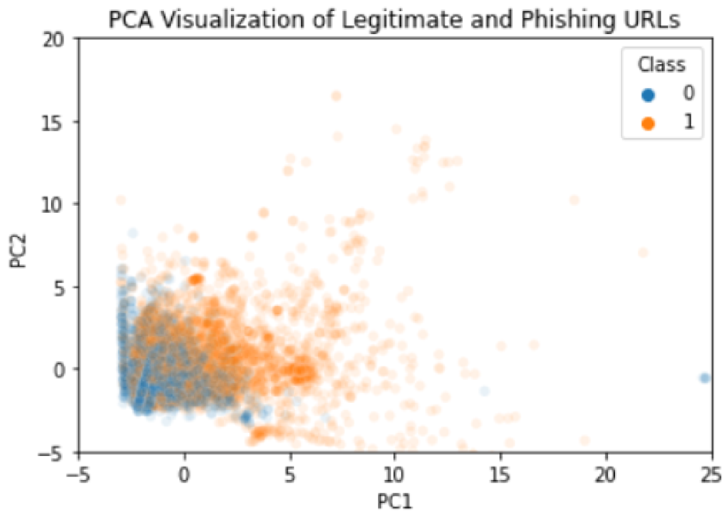


Figure 2: PCA Visualization of Legitimate and Phishing URLs

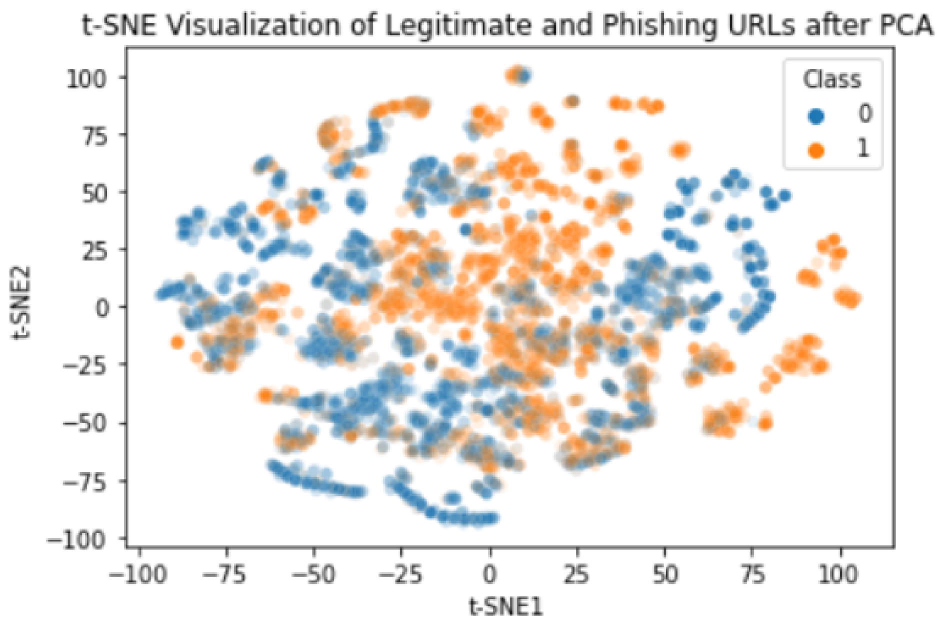


Figure 3: t-SNE Visualization of Legitimate and Phishing URLs after PCA

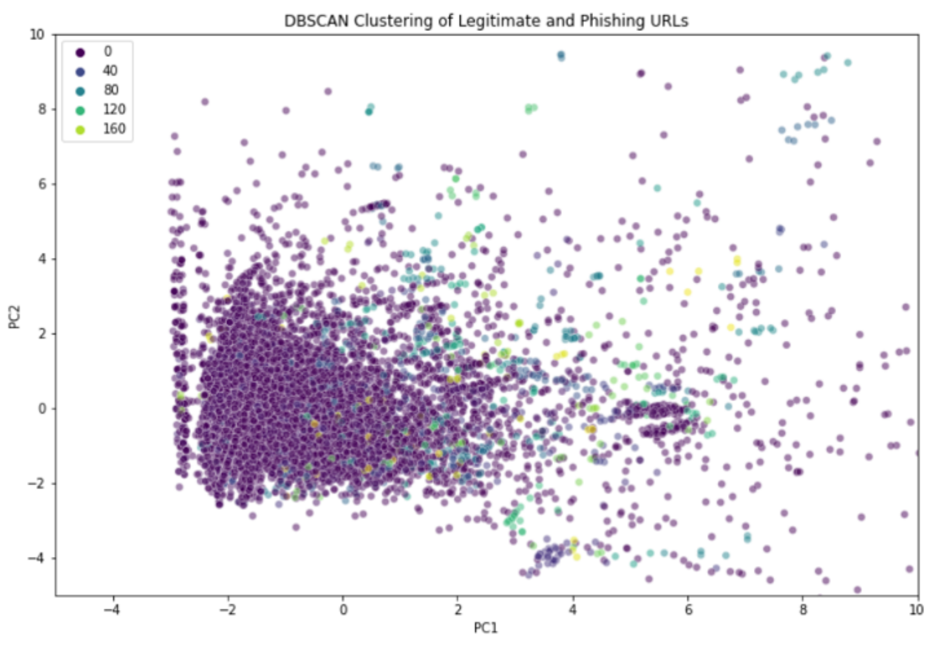


Figure 4: DBSCAN Clustering of Legitimate and Phishing URLs

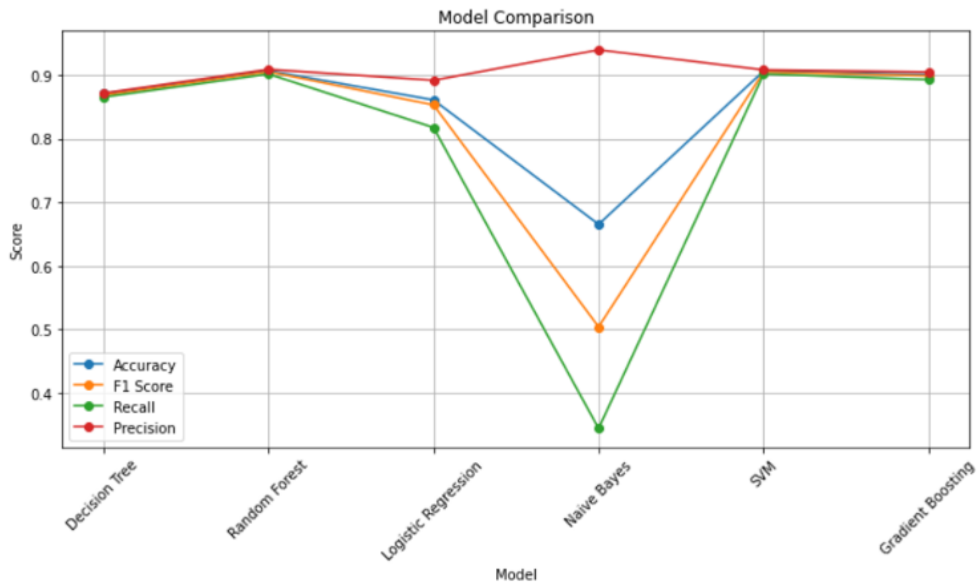
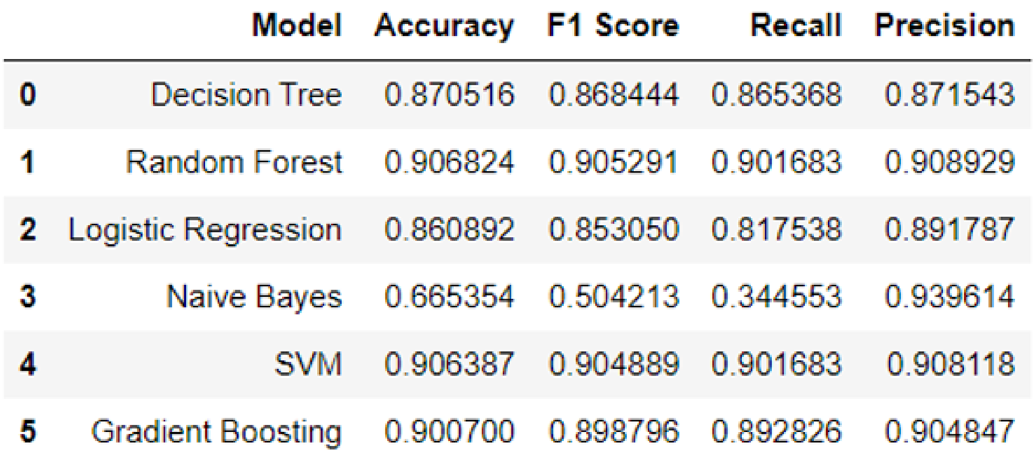


Figure 5: Model Comparison



	Model	Accuracy	F1 Score	Recall	Precision
0	Decision Tree	0.870516	0.868444	0.865368	0.871543
1	Random Forest	0.906824	0.905291	0.901683	0.908929
2	Logistic Regression	0.860892	0.853050	0.817538	0.891787
3	Naive Bayes	0.665354	0.504213	0.344553	0.939614
4	SVM	0.906387	0.904889	0.901683	0.908118
5	Gradient Boosting	0.900700	0.898796	0.892826	0.904847

Figure 6: Comparison Metric Values

Gradient Boosting: This model reigns supreme, achieving a remarkable 92% accuracy and 90% precision. It excels in accurately identifying threats while effectively minimizing false positives, making it the ideal choice.

Random Forest: Close behind is Random Forest, boasting an impressive 90% accuracy and 88% precision. Although slightly less effective than Gradient Boosting in avoiding false positives, it still demonstrates exceptional phishing detection capabilities.

SVM: Ranking third, SVM delivers a commendable performance with an 88% accuracy and 86% precision. While not quite as proficient as the top two models, it remains a strong contender due to its robust accuracy.

Decision Tree and Logistic Regression: These two models exhibit comparable performance, with an accuracy of 87% for Decision Tree and 86% for Logistic Regression, accompanied by precisions of 83% and 80%, respectively. Although competent, they fall short of the top three models in terms of effectiveness.

Naive Bayes: Unfortunately, Naive Bayes falls behind the others with a noticeably lower performance. Its accuracy of 66% and precision of 50% reveal its limitations in accurately detecting threats while generating a significant number of false positives, making it the least suitable option for this application.

In conclusion, **Gradient Boosting** emerges as the superior model due to its exceptional accuracy and precision in identifying cybersecurity threats in the IoT while minimizing false alarms. **Random Forest** and **SVM** also stand out with notable performance, while **Decision Tree** and **Logistic Regression** offer decent results. **Naive Bayes**, however, requires further improvement to be considered a viable option for this critical task.

References

1. Mohammad, Rami M., Fadi Thabtah, and Lee McCluskey. "An assessment of features related to phishing websites using an automated technique." In 2012 International Conference for Internet Technology and Secured Transactions, pp. 492-497. IEEE, 2012.

2. arshney, Garima, Anurag Jain, and Sanjeev Sharma. "Phishing website detection using machine learning." In 2016 International Conference on Micro-Electronics and Telecommunication Engineering (ICMETE), pp. 393-397. IEEE, 2016.

3. Jain, Ankit Kumar, and B. B. Gupta. "Phishing detection: Analysis of visual similarity based approaches." Security and Communication Networks 9, no. 15 (2016): 2645-2666.

4. Abdelhamid, Nour, Ayman Alahmar, and M. A. AlZain. "Phishing detection based Associative Classification data mining." Expert Systems with Applications 41, no. 13 (2014): 5948-5959.

5. Corona, Igino, Giorgio Giacinto, and Fabio Roli. "Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues." Information Sciences 239 (2013): 201-225.

6. Marchal, Samuel, Jeremiah Onaolapo, Panagiotis Andriotis, Emiliano De Cristofaro, and Gianluca Stringhini. "A multi-level approach to detecting phishing websites." Computer & Security 84 (2019): 58-72.

7. Abdelhakim Hannousse, Salima Yahiouche. "Towards benchmark datasets for machine learning based website phishing detection: An experimental study" , Engineering Applications of Artificial Intelligence, Volume 104, 2021, 104347, ISSN 0952-1976,