CS4403 Project
Signs of Trust:
Using Lightweight Models to Detect Phishing Websites

Chris Higgins - 3697087

University of New Brunswick

April 5, 2025

Abstract

The project explores the use of machine learning models to detect phishing websites using the PhiUSIIL Phishing URL dataset. Rather than maximizing performance with all available features, this project aims to balance interpretability and simplicity. This report details the steps of data cleaning, correlation and variance-based feature reduction, and feature selection using L1 regularization. The dataset was refined from 54 to 15 features across three models. A Random Forest model trained on the final combined feature set achieved an F1-score above 0.999, even when cross-validated. Finally this report reflects on limitations, performance concerns, and future improvement opportunities.

1 Introduction

Phishing is a social engineering tactic used by cybercriminals to steal sensitive information such as login credentials, financial data, or personal identity. These attacks manipulate individuals into willingly disclosing their personal information, and often without realizing they have been deceived. Phishing is both cheap to execute and highly effective. The rise of artificial intelligence (AI) has caused the scale and realism of these attacks to increase significantly, enabling the mass production of highly convincing content that can target users more efficiently [1]. These characteristics make phishing one of the most prominent and dangerous cybersecurity threats today.

A key component to many phishing attacks is the use of phishing websites, which are fraudulent pages that often mimic legitimate websites. Victims are lured to these websites through deceptive advertisements, emails, or other messages. Once there, users may be prompted to enter their credentials into fake login portals or unknowingly submit personal or financial details. For example, a phishing site may clone a bank's homepage, perfectly mirroring the favicon and design, to trick users into submitting their account credentials.

As phishing threats grow more frequent and more sophisticated, automated detection of phishing websites is becoming essential to internet safety. With millions of potential targets and new scams emerging daily, machine learning offers a scalable detection strategy, but it must carefully balance performance and interoperability to be useful and trustworthy in real-world applications.

2 Dataset and Project Direction

The PhiUSIIL Phishing URL (Website) dataset from UCI Machine Learning Repository was used for this project. It contains 235 795 labelled website instances, each of which includes 54 features either extracted from the source code of the webpage and URL, or features derived from the extracted information. Each instance is labelled as either phishing (0) or legitimate (1), making it ideal for binary classification.

The companion paper associated with this paper achieved exceptional results, reaching an accuracy of 99.993% using an XGBoost classifier and similar results with a variety of other models [2]. They sought to maximize performance by leveraging the full feature set, demonstrating that high detection rates are possible with this level of information. However, I wanted to explore a different question: *Can we achieve comparable performance with fewer features?* My project focused on reducing model complexity through feature selection whilst retaining as much performance as possible. Rather than using the full feature set, I wanted to identify the most informative features to build a lightweight but still powerful model.

There were several motivations behind my approach. Simpler models are generally faster, easier to deploy, and more explainable. Additionally, feature selection can reveal the characteristics that are truly important for classification. Ultimately, my goal was to better understand what the telltale signs of phishing websites are, from both a logical and machine learning perspective.

3 Dataset Exploration and Preprocessing

Before training any models, I spent time exploring the structure and distribution of the dataset. The PhiUSIIL dataset included 54 features, which was intimidating to face given my lack of experience in

the realm of phishing websites. These features ranged from simple binary indicators, to numerical scores. Most features are explained clearly in the companion article [2], and given the fact I didn't use each feature I won't provide an explanation to all 54, but I will explain the 15 features that I did use. The explanations can be found in the data dictionary provided at the end of the article.

3.1 Initial Observations

The dataset was already carefully curated, so I found no missing values or formatting issues during my initial observation. I did however find some features that were highly correlated, containing similar or overlapping information. Furthermore, some features showed almost no variance, suggesting they provided little signal across the dataset. To ensure that the models did not rely on noisy or redundant patterns, I performed several preprocessing steps.

3.2 Feature Cleaning

- Dropping Irrelevant Columns
  - Columns like URL, Domain, and Title were dropped as they added little to no information. Other features already captured the important information from these.
- Encoding
  - I only encoded a single feature, with that being TLD (Top Level Domain). To extract some information out of this column I used frequency encoding, as it would represent how common the TLD being used was, which could be an indicator of phishing.

3.3 Feature Reduction

- Correlation-Based Feature Pruning
  - Since some of my features were highly correlated, I took this as an opportunity to reduce the feature set. I performed correlation-based feature pruning by calculating a Pearson correlation matrix for the features, which were all numeric at this point, and identifying pairs of features that were highly correlated. I used a threshold of 0.85, and for any pair of features with a correlation coefficient above the threshold, one of the two features would be removed. Since one feature may carry more information than the other, their removal was a manual process. The decision of which feature to drop primarily depended on how focused a feature was. For example, the features TLDLegitimateProb and TLD had a near 1.00 coefficient, so TLD was dropped because despite being frequency encoded it lacked the clear information provided by TLDLegitimateProb.

    Additionally, I found that the feature 'URLSimilarityIndex' had a high correlation ($\sim$0.86) with the label, indicating it would be an extremely powerful feature. This was unsurprising, given how the score is determined and that it played a large role in the dataset's companion paper. This feature also played a big role in my project, and will be discussed in a later section.

- Low-Variance Feature Removal
  - To further simplify the dataset and remove non-informative features, I performed low-variance feature removal. Features with extremely low variance provide almost no signal for classification, as they remain constant across most instances. These features are unlikely to aid the model in distinguishing between legitimate and phishing websites, and hence were removed.A variance threshold of 0.01 was used, meaning that features whose values varied less than this across all samples were removed. By filtering these features out early, noise and dimensionality in the dataset were reduced.

4 Efforts

Although this was not the goal initially, my project finished with three models (two base, and one final). The initial goal was to make an extremely compact model using URLSimilarityIndex, since it was an extremely powerful feature. I would use the second model to find other powerful features and characteristics, without URLSimilarityIndex as I thought it may take away from some of the other features. The following sections will detail the general process for building each model, and then the specifics of each.

4.1 General Process

This process was followed for both base models:

- **Model 1:** Included the URLSimilarityIndex and 4 additional features
- **Model 2:** Included 14 features without URLSimilarityIndex

4.1.1 Feature Selection

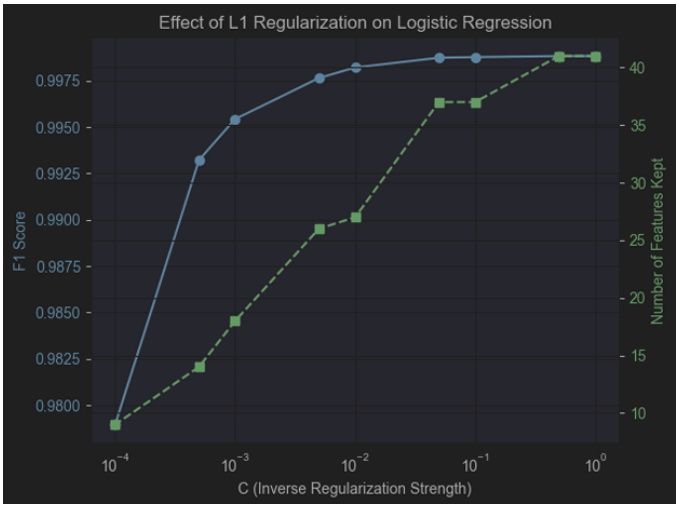

Figure 1: Effect of L1 Regularization on Logistic Regression (Model 2)

To align with my primary goal of finding the most important features, I applied L1 regularization using a Logistic Regression model to identify and retain the most powerful features. This method penalized less informative features by reducing their coefficients toward zero. By adjusting the 'c' parameter (the inverse of regularization strength), I explored how feature count and model performance were affected. In order to track model performance as feature count dropped, I kept track of F1 score, the 'c' parameter, and the count of features, which were then visualized to better understand this process. As c decreased, feature reduction became more aggressive. This process helped to identify a subset of features that maintained strong predictive performance while reducing dimensionality of the data.

Once I had selected a reduced subset of features, I did a closer analysis on each feature, examining its coefficient to the LR model, as well as its distribution throughout the dataset per label.

4.1.2 Model Training and Testing

I experimented with three different models for the reduced features sets: **Random Forest, Logistic Regression, and K-Nearest Neighbours**. An 80/20 train-test split was used on each, and I computed the following for evaluation:

- Accuracy
- Precision
- Recall
- F1 Score
- Confusion Matrices

Across both models, Random Forest outperformed it's competition, though not be much. Each model obtained an accuracy of **99%+**. Training time was not taken into account, though it could've been an effective metric for helping to determine a model. Random Forest was chosen as the primary classifier for both models.

4.1.3 Hyperparameter Optimization and Threshold Tuning

For each Random Forest classifier I applied Grid Search Cross-Validation, which tested various combinations of some hyperparameters, such as:

- n_estimators (number of trees)
- max_depth
- min_samples_split
- min_samples_leaf

This was done with a goal of identifying the best hyperparameter settings based on F1 score. 5-fold cross-validation was used to evaluate each configuration and select the optimal model.

Once the best model configuration was selected, I tuned the decision threshold. By default, classifiers assign class labels on a threshold of 0.50. However, phishing detection is a potentially high-risk classification task and because of this, I wanted to reduce false positives (phishing websites predicted as legitimate).

I evaluated thresholds from 0.30 to 0.70 and recorded the number of false positives and false negatives for each. The final threshold was selected based on the following criteria:

- Minimal total errors (FP + FN)
- FPs $<$ FNs (if not possible, use the lowest possible FP count)

This optimization process, using grid search followed by threshold adjustment, was done with a goal of creating a model that was accurate but also aligned with real-world safety requirements.

4.2 Final Evaluation of Base Models

Each of the base models were evaluated with 5-fold cross validation and produced the following results:

4.2.1 Model 1: URLSimilarity and 4 Additional Features

Table 1: Final Cross-Validated Evaluation (Threshold = 0.50)

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Phishing	1.000000	0.999670	0.999835	100,137
Legit	0.999755	1.000000	0.999878	134,850
Accuracy		0.999860		
Macro Avg	0.999878	0.999835	0.999856	234987
Weighted Avg	0.999860	0.999860	0.999860	234987

Feature importance was also analyzed:

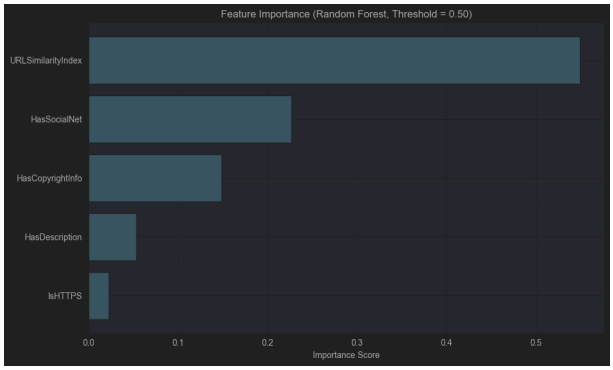

Figure 2: Feature Importance for Model 1

4.2.2 Model 2: No URLSimilarityIndex and 14 Features

Table 2: Final Cross-Validated Evaluation (Threshold = 0.64)

Class	Precision	Recall	F1-Score	Support
Phishing	0.997846	0.999191	0.998518	100137
Legit	0.999399	0.998398	0.998889	134850
Accuracy		0.998736		
Macro Avg	0.998622	0.998795	0.998708	234987
Weighted Avg	0.998737	0.998736	0.998736	234987

Feature importance was also analyzed:

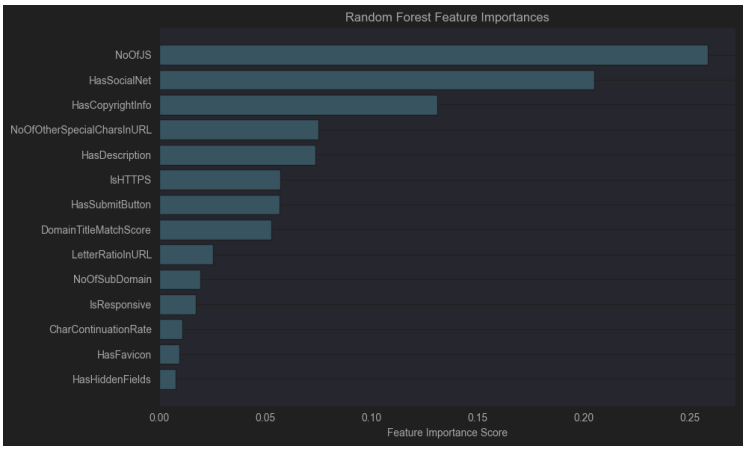

Figure 3: Feature Importance for Model 2

4.2.3 Main Oversight

While model performance was surely optimistic, I made one large oversight. Throughout my process of aggressive feature reduction, I was inadvertently reducing the variety of the dataset. I never thought of checking duplicates until I was almost finished working with these models. This issue was much more prominent in **Model 1**, as unique instances were reduced to **only 8 unique legitimate instances** and **445243 unique phishing instances**. In **Model 2**, there were **73776 unique legitimate instances** and **45182 unique phishing instances**. I didn't feel comfortable leaving my project like this, and this served as the main motivation for the third model, however, there is some information to extract from this issue.

The extremely aggressive feature reduction in Model 1 left it with the 5 most important features found by L1 regularization. The extremely low count of unique legitimate instances shows that in these features legitimate websites tend to be extremely similar, whereas phishing websites have far more variety. Model 2, had the opposite (and much less extreme) effect, where there was more variety in legitimate instances than phishing instances. This led to the final model, which would combine the feature set of Model 2 with URLSimilarityIndex, with hopes of having a more diverse dataset and more powerful model.

4.3 Final Model

Since Random Forest performed the best in the previous models, I skipped the model testing phase here and started with a Random Forest classifier. I then followed the same optimization process, by running Grid Search Cross-Validation followed by threshold tuning. I again had the goal of reducing FPs, which mainly came down to threshold tuning. **I was able to achieve less FPs than FNs with a threshold of 0.80**, though I found this threshold to be too high, and **settled at a threshold of 0.70**.

4.3.1 Model Performance

The model performed exceptionally:

Table 3: Final Cross-Validated Evaluation (Threshold = 0.70)

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Phishing	0.999970	0.999960	0.999965	100100
Legit	0.999970	0.999978	0.999974	134747
Accuracy		0.999970		
Macro Avg	0.999970	0.999969	0.999970	234887
Weighted Avg	0.999970	0.999970	0.999970	234887

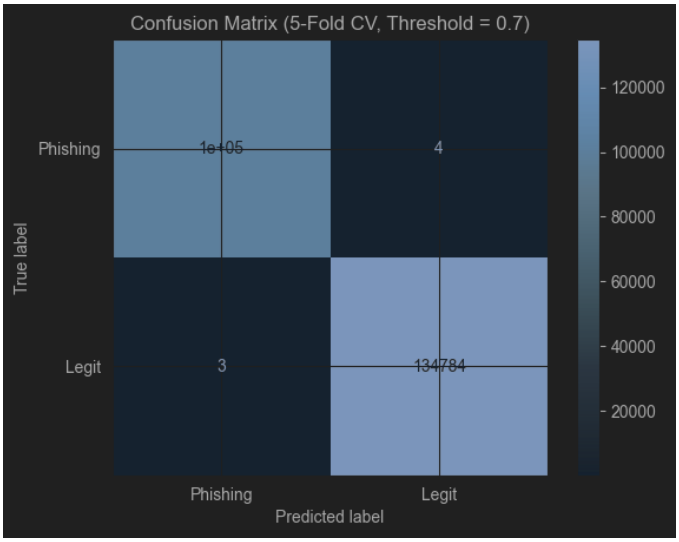

Figure 4: Confusion Matrix (Final Model)

4.3.2 Potential Overfitting

The models performance worried me, as I was suspicious of overfitting. I used a learning curve to assess how the model's performance scaled with the increased training data, and to check for signs of potential

Figure 5: Learning Curve (Final Model)

overfitting. The learning curve plotted F1 scores for training and validations sets as a function of training set size.

The training score demonstrates how well the model performs on the data it was trained on, and the validation score demonstrates how well the model performs on unseen data. The learning curve shows that the model achieved strong performance even with a smaller subset of the data. Both training and validation scores increased slightly, but remained close to each other. This indicated that the model was generalizing well, and was not overfitting or underfitting. While the extremely high performance with 10% of the data concerned me, features like URLSimilarityIndex are highly informative and allow for strong performance with less data. Additionally, 10% of the data is still is tens of thousands of instances.

For some peace of mind, I held out 100 samples from before training and testing the model, and tested these at the very end. The model achieved 100% accuracy on these samples.

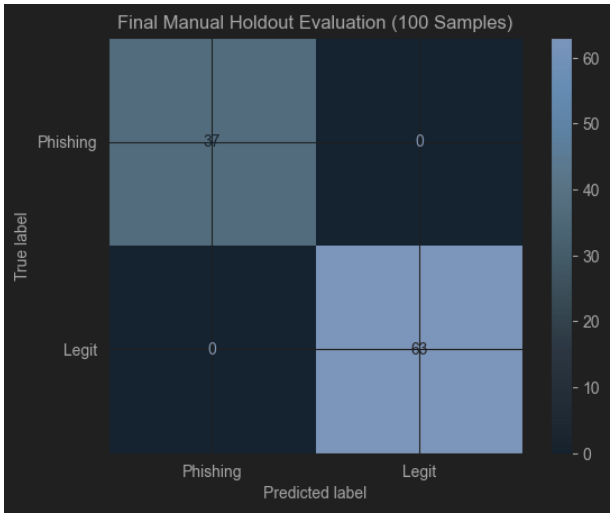

Figure 6: Holdout Confusion Matrix (Final Model)

4.3.3 Feature Importance

To understand how this model was making its predictions, I analyzed feature importance from the final Random Forest classifier. As shown in the figure, the top three features were:

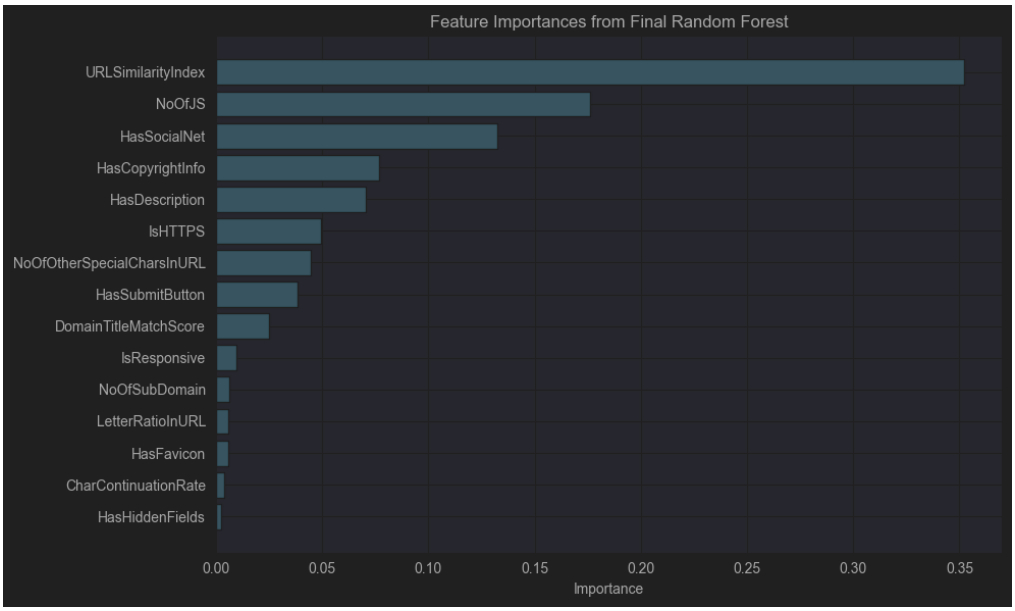

Figure 7: Feature Importance (Final model)

- **URLSimilarityIndex**
  - This is by far the most important feature. This feature is calculated by comparing the source URL to 10 million legitimate websites from Open RageRank Initiative [2]. In effect, instances with a 100% score are likely legitimate as they completely match one of the pages, meaning they are the legitimate website. Scores below this are suspicious and could be phishing. For example, an instance with 90% accuracy likely means this website is trying to mimic another. This was the primary reason I decided to not include this feature in the second model, as the classification can almost be made entirely off of this. I also had concerns that if a website was not in the list of 10 million legitimate websites, it was at risk as being labelled as phishing. Nonetheless, this feature is extremely useful.
- **NoOfJS**
  - This feature corresponds to the number of JavaScript elements on a page. After visualizing the distribution of JS elements per label, phishing sites in this dataset often have very little or no JavaScript. Legitimate sites have a much more spread out and varied distribution. The visualization can be found in the data dictionary.
- **HasSocialNet**

- This feature indicates whether the website has links to a social media presence. After visualizing the distribution of social media links per label, almost all phishing sites had no social media links. The majority of legitimate sites included links to social media. The visualization can be found in the data dictionary.

Other impactful features included **HasCopyrightInfo**, **HasDescription**, and **HasHTTPS**. Features such as **HasHiddenFields**, **CharContinuationRate**, and **HasFavicon** were much less informative.

This feature importance analysis shows that a small number fo well-chosen features can carry the majority of the predictive power. That is not to say that the other features add nothing, rather that they contribute much less to the predictions. Smaller sets of features add interpretability to the model, which can be just as important as the prediction itself.

5 Conclusion and Future Improvements

This project explored the use of machine learning models to detect phishing (and legitimate) websites based on structured features. I experiented with feature selection and reduction, model comparisons, hyperparameter tuning, and decision threshold tuning. Through this I was able to develop what appears to be a highly accurate and efficient Random Forest classifier, achieving near-perfect performance using the set of 15 features. The performance concerns me, as I haven't worked with models this accurate before, and I still wonder if something went wrong despite the model still performing well through cross-validation and held-out data. Because of this, I won't conclude that this model is in a state that is ready to shipped, and even if it was I lack the functions to extract the data from webpages. Despite this, I think this serves as a great proof of concept, and shows that effective and lean models can be built using a small subset of powerful features to accurately detect the presence of phishing or legitimacy.

I encountered several challenges through the process, from confusion about duplicate instances after feature selection to concerns about possible overfitting due to extremely high accuracy. After careful consideration, and to avoid the destruction of Model 1 entirely, I decided to keep the duplicates, as in the real world, it is almost certain that legitimate websites share similar characteristics amongst each other, and the same for phishing websites. This is an area that I should've analyzed far earlier on, as I'm not sure I made the best choice. I also learned the importance of careful threshold tuning when trying to balance false positives and false negatives. In my case this was a battle between reducing overall error, and trying to keep false positives lower than false negatives.

If I were to continue this work in the future, or complete this project again, there are some aspects I would approach differently. As previously mentioned, I would like to keep track of duplicate instances more carefully. Another aspect I missed out on was model testing. While I initially tried to test more models, some took far too long to run, making it difficult to continuously analyze them if adjustments needed to be made. Finally, developing the tools to extract this information from websites to enable potential real-time and real-world detection would be an extremely interesting and rewarding project to work on in the future.

6 Data Dictionary

- URLSimilarityIndex (Continuous)
  - This feature represents a percentage similarity between the source URL and the target URL

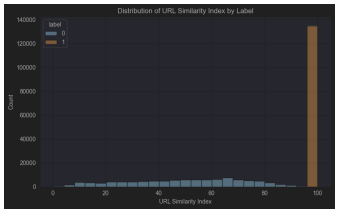

Figure 8: URLSimilarityIndex Distribution

- NoOfJS (Discrete)
  - This feature corresponds to the number of JavaScript elements on a page

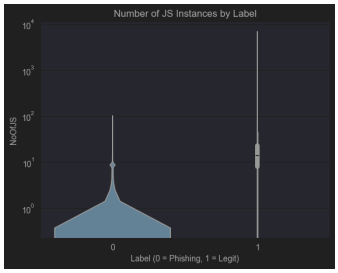

Figure 9: NoOfJS Distribution

- HasSocialNet (Binary)
  - This feature indicates whether the website has links to a social media presence

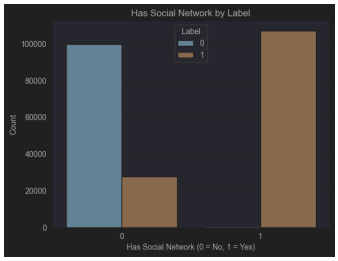

Figure 10: HasSocialNet Distribution

- HasCopyrightInfo (Binary)
  - Indicates whether copyright information is visible on the webpage

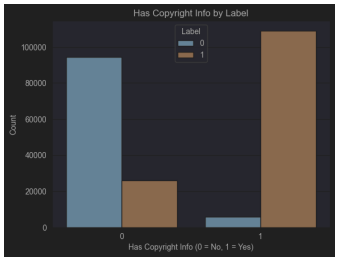

Figure 11: HasCopyrightInfo Distribution

- HasDescription (Binary)
  - Indicates whether the page includes an HTML description meta tag

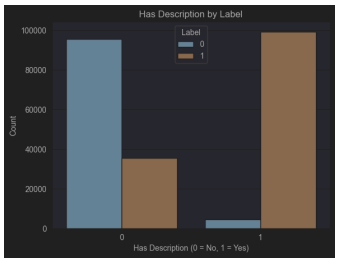

Figure 12: HasDescription Distribution

- IsHTTPS (Binary)
  - Indicates whether the URL uses HTTPS protocol

Figure 13: IsHTTPS Distribution

- NoOfOtherSpecialCharsInURL (Discrete)
  - Counts the number of special characters, excluding '&', '?', and '='

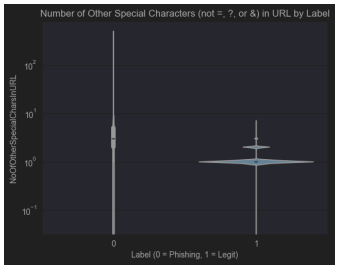

Figure 14: NoOfOtherSpecialCharsInURL Distribution

- HasSubmitButton (Binary)
  - Indicates the presence of a form submit button

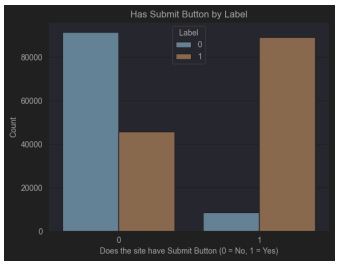

Figure 15: HasSubmitButton Distribution

- DomainTitleMatchScore (Continuous)
  - A similarity score between the domain name and page title

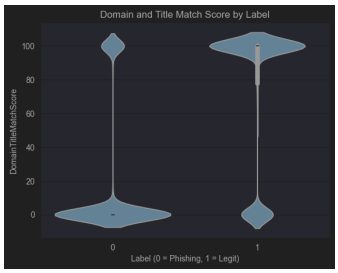

Figure 16: DomainTitleMatchScore

- IsResponsive (Binary)
  - Indicates whether the site uses responsive design (adjusts to screen sizes)

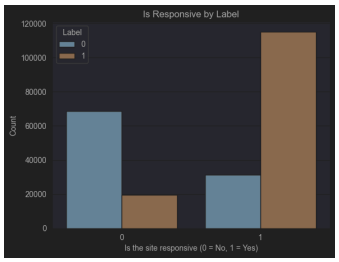

Figure 17: IsResponsive Distribution

- NoOfSubDomain (Discrete)
  - The number of subdomains in the URL

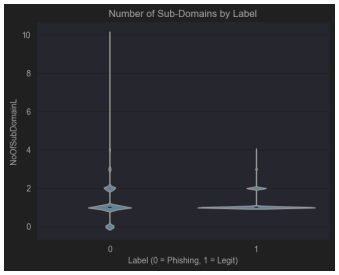

Figure 18: NoOfSubDomain Distribution

- LetterRatioInURL (Continuous)
  - The proportion of alphabetic characters in the URL

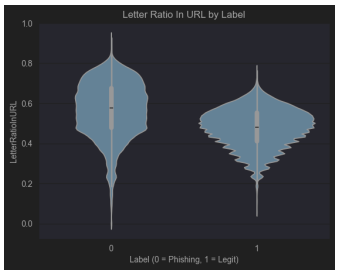

Figure 19: LetterRatioInURL Distribution

- HasFavicon (Binary)
  - Indicates whether the website includes a favicon (the icon in the browser tab)

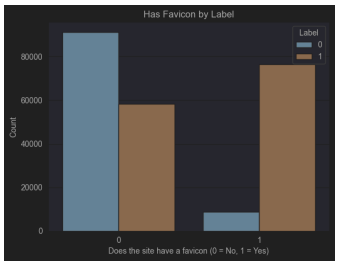

Figure 20: HasFavicon Distribution

- HasHiddenFields (Binary)
  - Indicates the presence of HTML fields hidden from the user

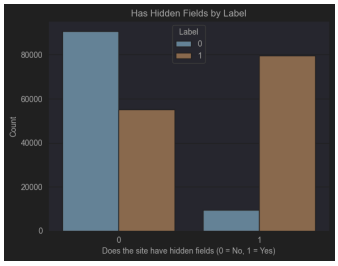

Figure 22: HasHiddenFields Distribution

- CharContinuationRate (Continuous)
  - This features indicates how a URL contains the alphabetical characters, digits, or special characters

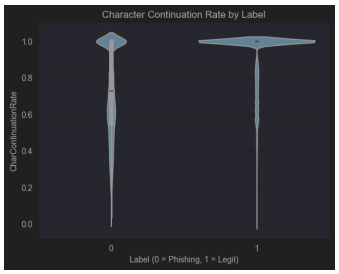

Figure 21: CharContinuationRate Distribution

References

[1] F. Heiding, B. Schneier, and A. Vishwanath, "Ai will increase the quantity-and quality-of phishing scams," *Harvard Business Review. https://hbr. org/2024/05/ai-will-increase-the-quantity-and-quality-of-phishing-scams*, 2024.

[2] A. Prasad and S. Chandra, "Phiusiil: A diverse security profile empowered phishing url detection framework based on similarity index and incremental learning," *Computers & Security*, vol. 136, p. 103545, 2024.