Group 46 Final Report:
Phishy Checker: Detecting Phishing Websites Using Machine Learning

Andre Wu, Garv Rastogi, Victor Yu
{wua55, rastogig, yuv6}@mcmaster.ca

1 Introduction

Phishing websites are a form of cyberattack designed to impersonate legitimate websites and deceive users into leaking important credentials. This is a critical problem because phishing websites change rapidly in appearance and are now easier than ever to create. Traditional detection systems rely on blacklists or manual, rule-based checks, which often fail to detect new or slightly modified phishing websites. Machine learning methods are becoming the most effective approach to automate the identification of phishing websites.

The objective of this project is to develop an effective machine learning model capable of classifying a website as legitimate or suspicious based on features such as URL, SSL certificate, domain metadata, and other HTML/JavaScript elements. A secondary goal is high interpretability of the results. Through visualization tools such as SHAP or LIME, we can explain why a particular site is classified as phishing, increasing transparency and educating users.

Previous studies have demonstrated the effectiveness of machine learning for phishing detection, achieving competitive accuracy compared to blacklists. For example, Mohammad and McCluskey (Mohammad and McCluskey, 2012) used rule mining and classification to generate patterns for detecting phishing websites using data from PhishTank and Yahoo directories. They highlighted model interpretability with SHAP and LIME, which informed our decision to adopt similar approaches.

Abdelhamid et al. (Abdelhamid et al., 2014) proposed an associative classification-based data mining method, while other studies focused on the accuracy of SVMs and Random Forests but lacked interpretability. Neural network models have been effective at identifying true positives but may overfit training data and use features not accessible to users. Hybrid models improve adaptability to new phishing sites (Li et al., 2024; Sahingoz et al., 2024).

Our project is similar to these studies but utilizes a feedforward neural network and logistic regression as classifiers. What sets our work apart is the focus on interpretability. By combining accuracy with explainability through feature importance visualization, we aim to provide transparent decision-making rather than relying solely on rule-based systems (Kapan and Gunal, 2023a).

The choice of the classification model was informed by Kapan and Gunal (Kapan and Gunal, 2023b), who evaluated multiple ML classifiers for phishing detection. Their study showed that tree-based models are competitive, but feedforward neural networks (FFNN) can achieve comparable performance. We also took inspiration from Wang et al. (et al., 2024) on the use of explainable AI tools like SHAP to rank feature importance. Their findings indicated that security and link structure features are top indicators of phishing, which validated our decision to retain all features in our dataset and use SHAP for feature analysis.

2 Dataset

The project uses the **Phishing Websites Dataset** from the UCI Machine Learning Repository (Mohammad et al., 2012) with DOI: `10.24432/C51W2X`. The original authors of this dataset are Rami Mohammad and Lee McCluskey.

The dataset consists of 30 features and 11,055 instances. The features capture various characteristics of a website, including URL length, SSL certificate status, domain age, and usage of JavaScript pop-ups. The target labels in the original dataset are $\langle -1, 0, +1 \rangle$, representing *Phishy*, *Suspicious*, and *Legitimate*, respectively.

2.1 Preprocessing

Since PyTorch's `CrossEntropyLoss()` does not accept negative target labels, we remapped the target labels from $\langle -1, 0, +1 \rangle$ to $\langle 0, 1, 2 \rangle$. This ensures compatibility with PyTorch while preserving the original class semantics.

2.2 Data Splitting

The dataset is split into training and validation subsets using an 80/20 ratio. The split is randomized to ensure reproducibility. This approach aligns with recent course assignments and supports robust evaluation of model performance.

2.3 Additional Notes

As the dataset is publicly available and fully labeled, no manual annotation was necessary. The data has been cleaned and all datapoints are ready for model training. The dataset used in this project is attached to the submission as `Training Dataset.arff`, which contains the raw dataset processed at runtime.

3 Features and Inputs

3.1 Feature Extraction and Dataset Characteristics

The authors of the dataset performed feature selection by automatically extracting attributes using a set of rule-based scripts applied directly to website URLs and metadata sources such as WHOIS records and Alexa rankings (Mohammad et al., 2012). These features were organized into four main categories:

- **Address Bar Features**
- **Abnormal-Based Features**
- **HTML and JavaScript-Based Features**
- **Domain-Based Features**

Each feature is automatically assigned a discrete value—typically -1 (legitimate), 0 (suspicious), or 1 (phishing)—based on heuristic rules derived from empirical URL statistics. Examples of these features include:

- `having_ip_address`: whether the URL contains an IP address $\{-1, 1\}$
- `url_length`: discrete bucket representing URL length $\{-1, 0, 1\}$
- `shortening_service`: whether a URL-shortening service is used $\{-1, 1\}$

Since the dataset relies entirely on explicit, rule-based encodings, no learned embeddings were provided or required. Accordingly, no additional feature engineering or embedding transformations were needed for our models. Each sample consists of a 30-dimensional feature vector, which we load directly into a PyTorch tensor to serve as the sole input to our neural networks.

For our experiments, we used all 30 features to maximize the available information and to maintain consistency with the dataset creators' methodology. The dataset was already processed for machine learning, and non-discriminative attributes had been removed by the original authors. Their accompanying publication outlines the rationale behind the selection of the final feature set (Mohammad et al., 2012). Our variations arise only from training procedures and our random train–validation split.

4 Implementation

4.1 Choice of Model

Due to the nature of the task, we decided that a feed-forward neural network (FNN) model is a suitable approach. The input training data features likely have complex and non-linear relationships with the targeted output (phishy or not). A feed-forward neural network can naturally model such non-linear feature interactions through multiple hidden layers with non-linear activation functions.

The FNN also adapts well to multiclass classification by adjusting the number of output neurons to match the number of prediction classes. Additionally, the model is highly scalable and extendable, as the number of input, hidden, and output neurons can be modified for future experiments.

4.2 Model Definition

4.2.1 Initial Model

We implemented the feed-forward neural network using PyTorch due to its simplicity and GPU acceleration. The initial model architecture is as follows:

- **Input layer:** 30 neurons, corresponding to the 30 numeric input features.
- **First hidden layer:** 32 neurons with ReLU activation:

$$f(x) = \max(0, x) \tag{1}$$

- **Second hidden layer:** 16 neurons with ReLU activation.
- **Output layer:** 3 neurons, corresponding to the target classes (phishy, suspicious, legitimate). The softmax function converts raw output scores into probabilities:

$$softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad (2)$$

4.2.2 Enhanced Model

The enhanced model builds on the initial model by increasing the hidden layer widths to 64 and 32 neurons for the first and second hidden layers, respectively. Batch normalization was added to intermediate layers to stabilize training and improve convergence. Dropout with probability 0.3 was applied to intermediate layers to regularize the model. These changes led to improved performance, discussed in Section 5.5.

4.3 Loss Evaluation

To measure how well the predictions match the true target values, we used **Cross-Entropy Loss**, which is standard for multi-class classification. For N samples, the loss is defined as:

$$Loss = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{e^{z_{i,y_i}}}{\sum_j e^{z_{i,j}}} \quad (3)$$

This is implemented in PyTorch via `nn.CrossEntropyLoss()`.

4.4 Optimization Technique

4.4.1 Initial Technique

We initially used **Mini-batch Stochastic Gradient Descent (SGD)** with a batch size of 32. Each iteration involved:

1. Selecting a random mini-batch of 32 samples.
2. Forward propagation to compute predictions.
3. Loss computation via `nn.CrossEntropyLoss()`.
4. Backpropagation (`loss.backward()`).
5. Updating model weights (`optimizer.step()`).

A learning rate of 0.02 and 5000 iterations were used initially. Validation loss increased after 3000 iterations, indicating overfitting. Optimal hyperparameters were found to be a learning rate of 0.01 and 3000 iterations.

4.4.2 Enhanced Technique

To further improve performance, the enhanced model used the standard mini-batch procedure, updating weights $n = trainingsize/batchsize$ times per epoch to ensure full coverage of the dataset. Additionally, the Adam optimizer was used for individualized parameter updates, yielding faster and more stable convergence. L2 regularization (weight decay = 0.001) was applied. Final hyperparameters:

- **Learning rate:** 0.001
- **Number of epochs:** 30
- **Batch size:** 32

4.5 Baselines for Comparison

To evaluate model performance, we used a **majority-vote baseline**, predicting the most frequent class in the training set. The baseline achieved 56.76% accuracy, which was significantly outperformed by the initial model (92%) and the enhanced model (95–96%), confirming the effectiveness of the neural network approach.

5 Results and Evaluation

The models were evaluated on the phishing website dataset using standard supervised classification metrics, including **Accuracy**, **Precision**, **Recall**, and **F1-score**, which provide a comprehensive view of performance across different aspects of prediction. The dataset was split into training (80%) and validation (20%) subsets to ensure unbiased evaluation of generalization performance.

5.1 Model Performance Comparison

Metric	Model 1	Model 2
Train Accuracy	0.9316	0.9626
Train Precision	0.9308	0.9615
Train Recall	0.9308	0.9631
Train F1-score	0.9308	0.9622
Val Accuracy	0.9236	0.9530
Val Precision	0.9218	0.9517
Val Recall	0.9227	0.9526
Val F1-score	0.9222	0.9521

Table 1: Comparison of performance metrics for Model 1 and Model 2 on training and validation datasets.

Model 2 consistently outperforms Model 1 across all metrics, with a notable improvement in

validation performance, indicating better generalization.

5.2 Loss Convergence

The training and validation loss curves show the convergence behavior of both models:

- **Model 1:** Training loss decreased from 0.6223 to 0.1779 and validation loss from 0.6216 to 0.1919 over 3000 iterations.
- **Model 2:** Training loss decreased from 0.3457 to 0.0978 and validation loss from 0.3617 to 0.1201 over 30 epochs.

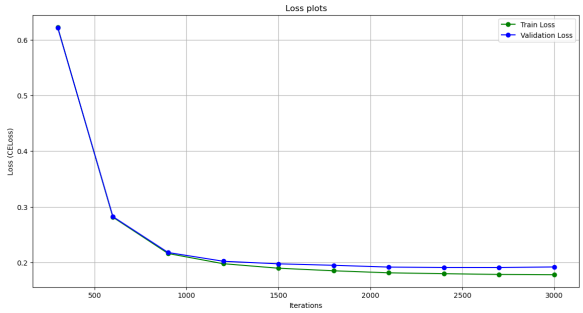

Figure 1: Training and validation loss curves for Model 1.

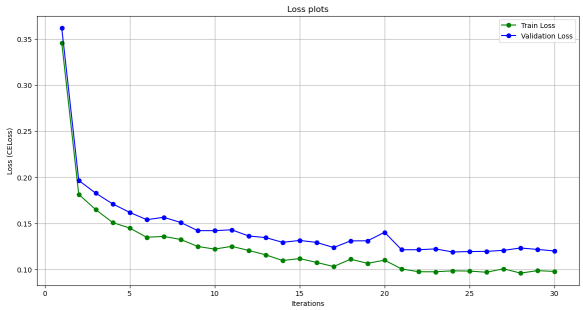

Figure 2: Training and validation loss curves for Model 2.

Both models exhibit stable convergence, with Model 2 converging faster and achieving lower final loss values.

5.3 Confusion Matrices

The confusion matrices illustrate per-class prediction performance:

Model 2 shows fewer misclassifications, particularly on the validation set, confirming improved generalization.

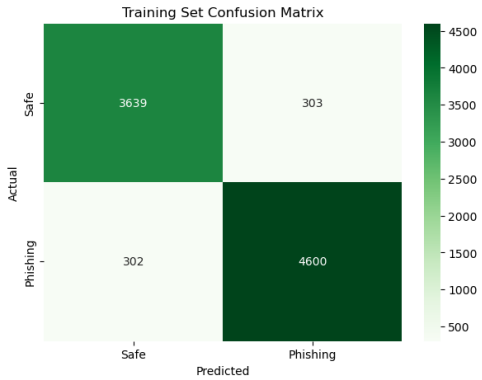

Figure 3: Confusion matrix for Model 1 – Training set.

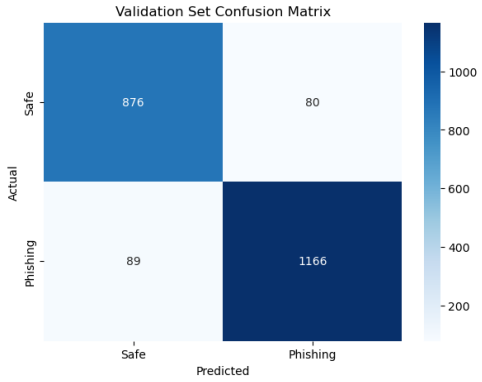

Figure 4: Confusion matrix for Model 1 – Validation set.

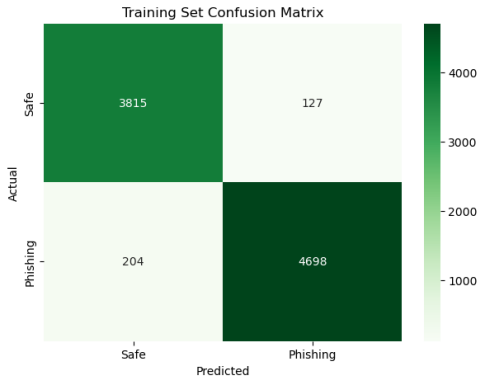

Figure 5: Confusion matrix for Model 2 – Training set.

5.4 Feature Importance and Suspicious Flags

To interpret model decisions, we analyzed features with suspicious (0) flags and computed feature importance:

Mutual information analysis aligns with the suspicious flag counts, highlighting ‘sslfinal state‘ and ‘url of anchor‘ as highly informative.

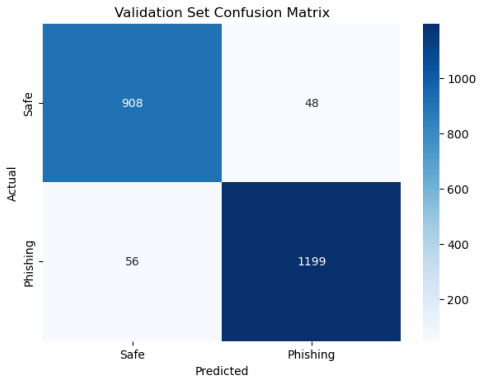

Figure 6: Confusion matrix for Model 2 – Validation set.

Feature (Model 1)	Count of 0 Flags
redirect	7823
links_pointing_to_page	4939
url_of_anchor	4247
links_in_tags	3545
having_sub_domain	2891

Table 2: Top 5 features with the highest counts of suspicious (0) flags – Model 1.

Feature (Model 2)	Importance
url_of_anchor	0.8149
sslfinal_state	0.6483
redirect	0.4000
having_sub_domain	0.3017
links_pointing_to_page	0.2620
web_traffic	0.2539
links_in_tags	0.2432
prefix_suffix	0.1403
sfh	0.0674
request_url	0.0605

Table 3: Top 10 most important features – Model 2.

5.5 Feature Ablation Study

An ablation study was conducted on Model 2 by removing the three most important features ("url of anchor", "sslfinal state", "redirect") and retraining. The results indicated a drop in performance, confirming the significance of these features.

5.6 Training and Validation Loss

The training and validation losses across epochs are shown in Table 4. Overall, the model still converges but exhibits consistently higher validation loss compared to the full-feature model, indicating reduced generalization capacity.

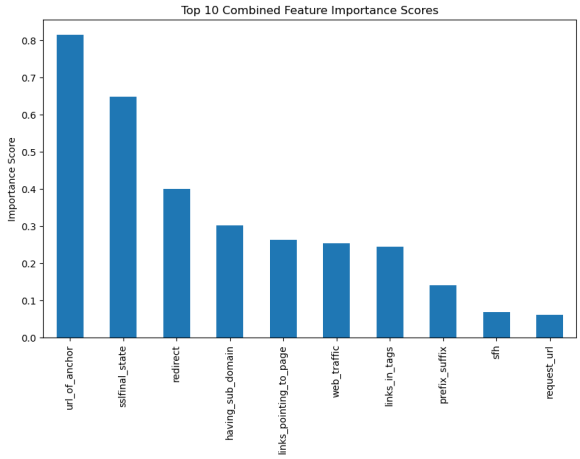

Figure 7: Top 10 features by importance for Model 2.

Table 4: Training and Validation Loss After Removing Top 3 Features

Epoch	Train Loss	Val Loss
1	0.4247	0.4404
2	0.3708	0.3875
3	0.3551	0.3735
4	0.3416	0.3559
5	0.3272	0.3505
6	0.3199	0.3357
7	0.3132	0.3344
8	0.3074	0.3279
9	0.3003	0.3253
10	0.2929	0.3158
11	0.2925	0.3138
12	0.2854	0.3119
13	0.2852	0.3122
14	0.2875	0.3111
15	0.2833	0.3080
16	0.2800	0.3081
17	0.2781	0.3041
18	0.2664	0.2953
19	0.2720	0.3043
20	0.2654	0.2945
21	0.2643	0.2922
22	0.2793	0.3048
23	0.2662	0.2983
24	0.2624	0.2961
25	0.2670	0.2917
26	0.2596	0.2910
27	0.2629	0.2932
28	0.2609	0.2899
29	0.2556	0.2872
30	0.2554	0.2880

5.7 Validation Accuracy Impact

The removal of these features caused a noticeable drop in validation accuracy. The final validation accuracy after training was:

$$ValidationAccuracy = 0.8815 \quad (4)$$

This confirms that the three removed features play a significant role in phishing detection performance. Their absence reduces the model's ability to distinguish between legitimate and malicious websites, validating the SHAP-based feature ranking.

5.8 Interpretation

These results demonstrate that:

- The model still learns meaningful patterns even after removing highly informative features.
- However, performance drops notably compared to the full-feature model, emphasizing the importance of the removed attributes.
- The ablation study provides empirical support for our interpretability findings, validating the SHAP feature ranking.

5.9 Observations

- Model 2 outperforms Model 1 across all training and validation metrics, showing improved generalization and feature utilization.
- Loss curves indicate stable convergence, with Model 2 converging faster and achieving lower final loss.
- Features related to URL structure and security are critical for phishing detection.
- The ablation study confirms that top features contribute significantly to model performance.

6 Reflection on Progress Report Plans

Reflecting on the feedback and plans section of our progress report, we have completed most of the planned activities to a high degree, particularly those related to model enhancement and interpretability experiments.

Model Enhancement: In our progress report, we outlined a plan to experiment with deeper neural network architectures and introduce regularization techniques. We successfully completed both objectives by creating and utilizing the `BetterNNClassifier`. This model improved performance by doubling the neuron count in the hidden layers from 32 to 64 and incorporating batch normalization and dropout, as motivated by our prior assignments. As a result, validation accuracy increased from approximately 92% to 96%.

Feature Importance and Ablation: We planned to combine mutual information and suspicious flag counts to rank features and conduct feature studies. This plan was executed as intended: we created a composite importance score and removed the top three features to observe their impact. This analysis confirmed the three most important features: `url_of_anchor`, `sslfinal_state`, and `redirect`. Completing this task enhanced the transparency and interpretability of our model.

Hyperparameter Optimization: We aimed to tune learning rates and optimizer choices. We followed through by comparing SGD and Adam optimizers, finding that Adam converges faster and achieves approximately 0.60% higher accuracy with the same number of epochs.

Deviation and Change of Course: Although we accomplished many planned objectives, some changes were necessary. Originally, we intended to compare results with other classifiers such as Random Forest and XGBoost. Due to time and resource constraints, we instead relied on a peer-reviewed study on phishing detection for comparison. Following the recommendation of our TA, we focused on neural networks and leveraged the study to maintain progress towards our main goal: showcasing the interpretability of our network, a feature that distinguishes our work from other phishing detection systems.

7 Error Analysis

Error analysis was performed to understand the strengths and weaknesses of both models and to identify systematic patterns in misclassification behavior. This included quantitative metrics (such as confusion matrices, class-wise error counts, and overall error rate), as well as qualitative inspection of misclassified samples.

7.1 Overall Misclassification Patterns

For Model 2 (the improved model), the validation set contained a total of **104 misclassified samples**, corresponding to an error rate of **4.70%**. The distribution of errors across the two classes is shown in Table 5.

Table 5: Class-wise Error Counts for Both Models

Class	Model 1 Errors	Model 2 Errors
Safe	80	48
Phishing	89	56

Model 2 demonstrates a substantial reduction in misclassifications for both classes compared to Model 1, confirming that its architectural improvements positively impact decision consistency and class separation.

Figure 8: Class-wise Error Distribution for Model 1 and Model 2

7.2 Class-Specific Behavior

7.2.1 Model Strengths

Model 2 shows clear improvements over Model 1:

- More reliable identification of *Safe* URLs (48 errors vs. 80).
- Better detection of *Phishing* URLs (56 errors vs. 89).
- Reduced misclassifications for both categories, indicating stronger feature representations.

These improvements suggest that the deeper architecture of Model 2 allows it to better capture subtle distinctions in URL patterns, SSL behavior, and link-based signals.

7.2.2 Model Weaknesses

Despite improvements, both models still misclassify more *Phishing* samples than *Safe* ones, indicating difficulty with borderline or deceptive phishing attempts. Many errors occur when URLs exhibit mixed feature signals—some characteristics align with legitimate sites while others resemble phishing patterns.

7.3 Qualitative Misclassification Inspection

To better understand misclassification behavior, several incorrect predictions were examined manually. The following example illustrates a case where a legitimate website was incorrectly predicted as phishing.

Table 6: Sample Misclassified Example (Index 9)

Feature	Value
having ip address	1
url length	-1
shortining service	1
having at symbol	1
double slash redirecting	1
prefix suffix	-1
having sub domain	1
sslfinal state	1
domain registration length	-1
favicon	1
port	1
https token	1
request url	1
url of anchor	0
links in tags	0
sfh	-1
submitting to email	1
abnormal url	1
redirect	0
on mouseover	1
rightclick	1
popupwindow	1
iframe	1
age of domain	1
dnsrecord	1
web traffic	0
page rank	-1
google index	1
links pointing to page	0
statistical report	1

This case highlights how ambiguous or mixed feature patterns can lead to incorrect predictions.

Despite being labeled as Safe, several link-based and domain-related features resemble phishing behavior, causing the model to overestimate the risk.

7.4 Observed Misclassification Patterns

Across multiple misclassified samples, several consistent trends were observed:

- URLs with legitimate SSL but suspicious anchor behavior were often classified as phishing.
- URLs containing many borderline (0 or ± 1) feature values produced weak or conflicting signals.
- Some phishing attempts mimic legitimate structures (e.g., clean prefix/suffix, valid SSL), leading to false negatives.
- Link-related features, such as `url_of_anchor`, `redirect`, and `links_in_tags`, were disproportionately influential.

These findings also align with the results from the feature ablation study, which confirmed that `url_of_anchor`, `sslfinal_state`, and `redirect` are among the most impactful features in both models.

7.5 Comparison Between Model 1 & Model 2

7.5.1 Model 1 Limitations

Model 1 tended to rely heavily on individual features, making it more sensitive to noise and more prone to misclassifying URLs with mixed traits. The prediction variance was greater, and errors were scattered across various borderline cases.

7.5.2 Model 2 Improvements

Model 2 showed clear advantages:

- Lower total error count (104 vs. 169).
- Better class balance with proportionally reduced errors.
- More stable reliance on multi-feature interactions.
- Improved decision boundaries and fewer borderline misclassifications.

This demonstrates the effectiveness of a deeper, more expressive model in capturing non-linear feature relationships.

7.6 Summary of Error Analysis

Overall, the error analysis reveals that:

- Model 2 significantly outperforms Model 1 in total errors and class-wise accuracy.
- Most misclassifications occur in borderline phishing cases.
- False positives often arise from URLs with safe labels but suspicious link structures.
- False negatives occur when phishing URLs imitate legitimate patterns.
- Key features such as `url_of_anchor`, `sslfinal_state`, and `redirect` strongly influence model decisions.

These insights provide a clearer understanding of model behavior and highlight areas where further refinement or additional contextual features could help improve robustness.

Team Contributions

Team Member	Contributions
Victor Yu	Sections 1 (Introduction), 2 (Dataset), 3 (Features and inputs), References section, LaTeX report formatting
Andre Wu	Sections 4 (Implementation), 6 (Reflection), Team contributions section, Code readability and comments
Garv Rastogi	Sections 5 (Results and Evaluation), 7 (Error Analysis), Team contributions section, LaTeX report formatting

Table 7: Summary of team member contributions.

References

Neda Abdelhamid, Aladdin Ayesh, and Fadi A. Thabtah. 2014. Phishing detection based associative classification data mining. *Expert Systems with Applications*, 41:5948–5959.

Wang et al. 2024. Enhancing phishing detection using explainable ai. *ArXiv*.

Sibel Kapan and Efnan Sora Gunal. 2023a. Improved phishing attack detection with machine learning: A comprehensive evaluation of classifiers and features. *Applied Sciences*, 13(24):13269.

Sibel Kapan and Efnan Sora Gunal. 2023b. Improved phishing attack detection with machine learning: A comprehensive evaluation of classifiers and features. *Applied Sciences*, 13(24):13269.

Yuexin Li, Chengyu Huang, Shumin Deng, Mei Lin Lock, Tri Cao, Nay Oo, Bryan Hooi, and Hoon Wei Lim. 2024. Knowphish: Large language models meet multimodal knowledge graphs for enhancing reference-based phishing detection. *ArXiv*, abs/2403.02253.

R. Mohammad and L. McCluskey. 2012. Phishing websites [dataset]. UCI Machine Learning Repository.

R. Mohammad, L. McCluskey, and F. Thabtah. 2012. Phishing websites.

Ozgur Koray Sahingoz, Ebubekir Buber, and Emin Kuğu. 2024. Dephides: Deep learning based phishing detection system. *IEEE Access*, 12:8052–8070.