Group 46 Progress 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 sensitive credentials. This is a growing problem, as phishing websites can change their visual appearance rapidly and are now easier than ever to create. Traditional detection systems, which rely on blacklists or manual, rule-based checks, often fail to detect new or slightly modified phishing websites. As a result, machine learning methods have emerged as an 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 for phishing based on features such as URL characteristics, SSL certificate information, domain metadata, and other HTML/JavaScript elements. A secondary goal is to ensure high interpretability of the model's predictions. Using visualization tools such as SHAP or LIME, we aim to provide transparency in the model's decision-making process, showing users why a particular site is classified as phishing, thereby increasing awareness and educating users.

2 Related Work

There have been several studies that applied machine learning methods for phishing detection, demonstrating significant advantages over traditional blacklist-based approaches. Prior works have utilized models such as neural networks, decision trees, and random forests to achieve competitive accuracy on phishing datasets.

In particular, Mohammad and McCluskey (Mohammad and McCluskey, 2012) developed a dataset and methodology using rule mining and classification to extract patterns from phishing websites. Their system leveraged data from PhishTank and the Yahoo Directory to show superior accuracy compared to traditional algorithms such as Decision Trees and Support Vector Machines (SVMs). Furthermore, their study emphasized transparency and interpretability through tools like SHAP and LIME, an aspect that strongly influenced our own focus on interpretability in this project.

Another related study by Abdelhamid et al. (Abdelhamid et al., 2014) proposed an associative classification data-mining approach for phishing detection. Their method generated classification rules based on mined feature associations and demonstrated high accuracy. However, while highly functional, many of these earlier works prioritized accuracy over explainability.

Additional studies have used similar datasets from the UCI Machine Learning Repository, such as Abdelhamid (Abdelhamid, 2014), which explored SVMs and Random Forest classifiers. These approaches achieved strong predictive performance but often lacked interpretability, making it difficult to explain model predictions to end users.

Recent advancements in phishing detection have incorporated neural networks and hybrid models. For example, Li et al. (Li et al., 2024) proposed KnowPhish, a system that combines large language models (LLMs) with multimodal knowledge graphs to enhance reference-based phishing detection. Similarly, Sahingoz et al. (Sahingoz et al., 2024) introduced DEPHIDES, a deep learning-based phishing detection system, demonstrating state-of-the-art accuracy but still facing challenges in transparency and overfitting.

Our project builds upon these prior studies by employing a feed-forward neural network and logistic regression as classifiers. What differentiates our work is the emphasis on **interpretability**. By combining the predictive power of neural networks with visual feature importance analysis, we aim to achieve a balance between model accuracy and explainability, bridging the gap between rule-based interpretability and data-driven performance.

3 Dataset

The project uses the **Phishing Websites Dataset** from the UCI Machine Learning Repository (Mohammad and McCluskey, 2018) 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.

3.1 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.

3.2 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.

3.3 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.

4 Features

The authors of the dataset performed feature extraction using rule-based scripts applied to website URLs and other sources such as WHOIS and Alexa. These features were grouped 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, such as -1 (*legitimate*), 0 (*suspicious*), or 1 (*phishing*), based on heuristic rules derived from observed URL statistics. Examples include:

- `having_ip_address`: whether the URL contains an IP address. $\{-1, 1\}$
- `url_length`: the length of the URL string. $\{-1, 0, 1\}$
- `shortening_service`: whether a URL shortening service was used. $\{-1, 1\}$

Since the dataset relies entirely on these explicit, rule-based numerical encodings, no learned embeddings were used. Therefore, no additional feature engineering or embeddings were added.

The 30 extracted features are stored in a 30-dimensional vector, represented as a `torch.Tensor`, which serves as the sole input to the neural network model.

5 Implementation

5.1 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.

5.2 Model Definition

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

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

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

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

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

The softmax operation is internally used by PyTorch's loss function, which will be discussed in Section 5.3.

5.3 Loss Evaluation

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

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

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

5.4 Optimization Technique

We used **Mini-batch Stochastic Gradient Descent (SGD)** for optimization, which balances computational efficiency with stable weight updates. The batch size was set to 32. The optimization procedure at each iteration is as follows:

1. Select a random mini-batch of 32 samples from the dataset.
2. Perform forward propagation to compute model predictions.
3. Compute the loss using `nn.CrossEntropyLoss()`.
4. Calculate gradients via backpropagation (`loss.backward()`).
5. Update model weights using `optimizer.step()`.

We initially used a learning rate of 0.02 with 5000 training iterations (epochs). Validation loss began to increase after 3000 iterations, indicating overfitting. After experimentation, we found optimal hyperparameters: a learning rate of 0.01 and 3000 training iterations.

5.5 Baselines for Comparison

To evaluate model performance, we used a simple **majority-vote baseline**, which predicts the most frequent class in the training set. The baseline achieved an accuracy of 56.76%, which is significantly lower than our model's validation accuracy of over 90%. This confirms the practical usefulness of our neural network approach.

6 Results and Evaluation

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

6.1 Model Performance

Metric	Training	Validation
Accuracy	0.9316	0.9240
Precision	0.9308	0.9223
Recall	0.9307	0.9231
F1-score	0.9308	0.9227

Table 1: Model performance metrics on training and validation datasets.

To better understand classification performance per class, the confusion matrices for both training and validation datasets are shown below.

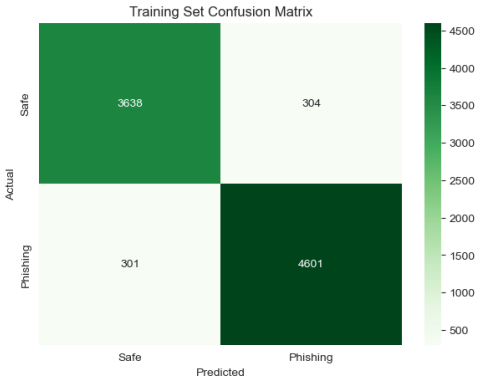

Figure 1: Confusion matrix for training set.

These results indicate strong predictive performance, with a minor drop from training to validation, suggesting the model generalizes well without significant overfitting. The loss curves over iterations further confirm this, showing a steady

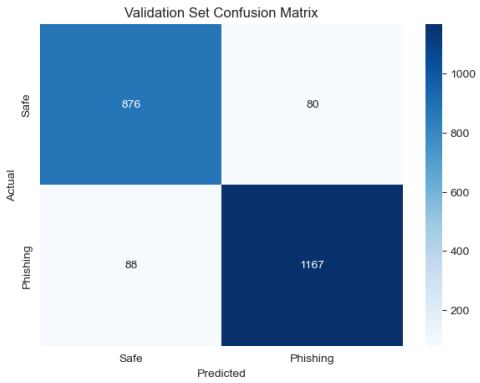

Figure 2: Confusion matrix for validation set.

decrease in both training and validation loss, indicating stable convergence.

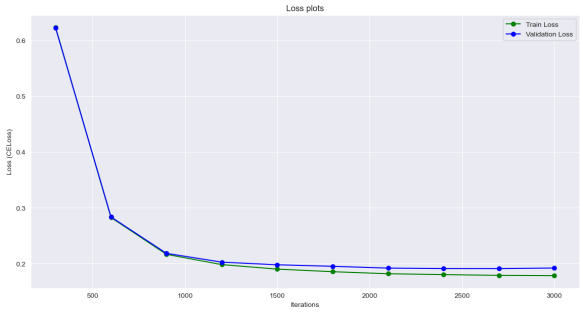

Figure 3: Training and validation loss curves over iterations.

6.2 Analysis of Suspicious (0) Flags

To gain further interpretability, we examined which features contain the 0 value, indicating potentially unsafe or anomalous characteristics in URLs. This analysis helps understand which features might be driving model decisions in identifying phishing websites. The top five features with the highest counts of suspicious flags are:

- **redirect** – 7823 suspicious instances
- **links_pointing_to_page** – 4939 suspicious instances
- **url_of_anchor** – 4247 suspicious instances
- **links_in_tags** – 3545 suspicious instances
- **having_sub_domain** – 2891 suspicious instances

This shows that features related to URL structure and external link references are often flagged as suspicious. For example, a high number of redirects or anchor links is a known indicator of phishing attempts.

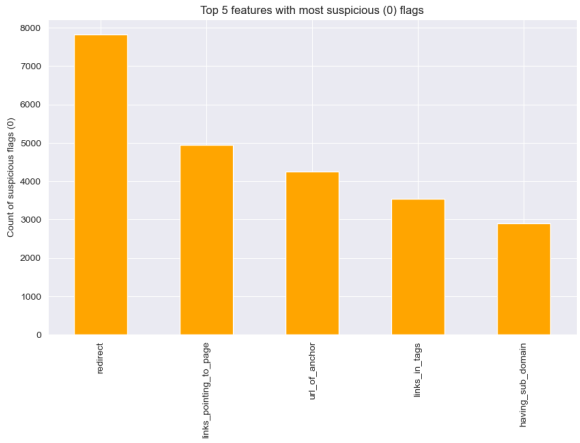

Figure 4: Top 5 features with highest counts of suspicious (0) flags.

6.3 Mutual Information Analysis

To complement the suspicious-flag analysis, we computed the mutual information between each feature and the target label. The results indicate that:

- **sslfinal_state** (MI = 0.3479)
- **url_of_anchor** (MI = 0.3294)

are the most informative features, confirming the relevance of the suspicious-flag findings. This alignment reinforces the idea that features reflecting security protocols and link structures are critical for phishing detection.

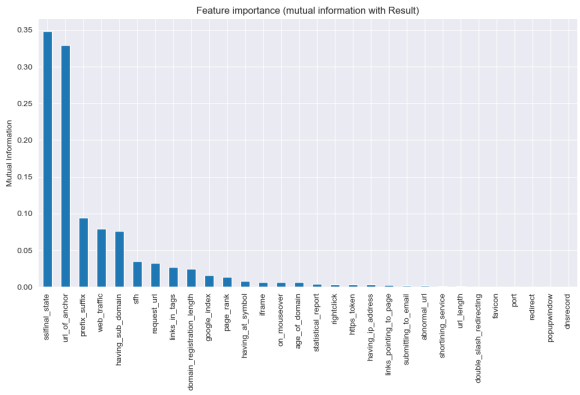

Figure 5: Mutual information of features with the target label.

6.4 Observations and Insights

- Features with high counts of suspicious flags often correspond to high mutual information scores, suggesting that these features contribute significantly to model prediction.

- The combination of metric evaluation, suspicious-feature analysis, and mutual information provides a holistic understanding of model behavior and feature importance.

7 Feedback and Plans

The TA recommended implementing at least one additional experiment and adding more graphical analyses to better understand model behavior. Following this feedback, the suspicious-feature experiment was introduced as an additional analysis to enhance interpretability. This approach allows us to identify features that are both frequently flagged as suspicious and highly informative for classification, bridging model predictions with domain knowledge.

7.1 Planned Next Steps

7.1.1 Advanced Feature Importance Experiments

- Combine mutual information and suspicious-flag counts to rank features.
- Conduct feature ablation studies, where top features are iteratively removed to observe performance impact. This will clarify which features are truly essential.

7.1.2 Model Enhancements

- Experiment with deeper neural network architectures, including additional hidden layers and neurons.
- Introduce regularization techniques (L2 weight decay, dropout) to reduce overfitting.
- Benchmark against other classifiers (Random Forest, XGBoost).

7.1.3 Enhanced Visualization

- Plot training vs. validation loss curves and metrics over epochs to visually monitor overfitting or underfitting.
- Include heatmaps or correlation plots for features flagged as suspicious to identify redundancy or interactions between features.

7.1.4 Hyperparameter Optimization

- Systematically tune learning rates, batch sizes, and optimizer choices to maximize convergence speed and final accuracy.
- Employ grid search or Bayesian optimization for efficient exploration of hyperparameter space.

7.2 Overall Assessment

The current model demonstrates high accuracy and robust generalization, and the suspicious-feature experiment provides actionable interpretability. Moving forward, refining the feature set, experimenting with model architecture, and adding comprehensive visualizations will further strengthen both performance and explainability of the phishing detection system.

Team Contributions

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

Table 2: Summary of team member contributions.

References

Neda Abdelhamid. 2014. Website phishing [dataset]. UCI Machine Learning Repository.

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

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 preprint arXiv:2403.02253*.

Rami Mohammad and Lee McCluskey. 2012. Phishing websites [dataset]. https://doi.org/10.24432/C51W2X. UCI Machine Learning Repository.

Rami Mohammad and Lee McCluskey. 2018. Phishing websites dataset. https://archive.ics.uci.edu/ml/datasets/Phishing+Websites.

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