AI Security Final Project: Detecting Phishing Websites with Machine Learning

Farida Ahmed*, Hadeer Amr†, Muhammed Salah‡, Marwan Gaber§,
Omar Ibrahim¶, Youssef Tamer|| , Ahmed Saber**
*†‡§¶||** Faculty of Computer and Data Science, Alexandria University, Alexandria, Egypt

***Abstract*—Phishing attacks are a persistent cybersecurity challenge, exploiting users into disclosing sensitive information via deceptive websites. This project proposes a machine learning-based phishing detection system, utilizing supervised learning models (Random Forest, SVM, Logistic Regression) trained on URL and content-based features. The approach is designed to identify new phishing techniques and zero-day attacks in real-time. Our target is to achieve over 95% detection accuracy with low false positives, contributing a robust and scalable solution to phishing defense mechanisms.**

***Index Terms*—Phishing Detection, Machine Learning, Cybersecurity, Random Forest, SVM, URL Spoofing**

I. Introduction

Phishing continues to be a major threat in the cybersecurity landscape, with attackers evolving their methods to bypass traditional detection systems such as blacklists. This project aims to build an intelligent phishing detection system using machine learning models trained on various URL-based and content-based features.

II. Literature Review

Recent studies have shown the effectiveness of ML models in phishing detection:

- A 2018 study compared seven ML algorithms, showing Random Forest achieving 97.3% accuracy by leveraging URL and HTML features.
- A 2024 model used XGBoost with domain age and custom features, reaching 98.1% accuracy while minimizing false positives.
- A 2023 comparison between SVM and RF showed Random Forest outperformed SVM (96.2% vs. 94.7%), especially under imbalanced datasets.

These studies highlight the potential of combining feature types and using robust classifiers like Random Forest.

III. Dataset

We will utilize the following datasets:

1) Phishing Websites Dataset (UCI Repository)
2) Mendeley Phishing Dataset
3) PhishStorm Dataset
4) OpenPhish Live Feed

These sources provide labeled data of phishing and legitimate URLs, enriched with advanced features (e.g., domain age, JavaScript content, HTTPS usage).

IV. Proposed Methodology

A. Data Preprocessing

Data will be cleaned by removing duplicates and fixing missing values. Feature extraction includes URL length, special characters, HTTPS usage, and domain age.

B. Model Building

We will implement and compare three models:

- Random Forest (robust against noisy and imbalanced data)
- Support Vector Machine (efficient for binary classification)
- Logistic Regression (used as baseline)

C. Compilation and Evaluation

We will train models using Scikit-learn. Evaluation metrics include Accuracy, Precision, Recall, and F1-score. Testing will include real-world phishing URLs from sources like PhishTank.

D. Attack Types Covered

- URL Spoofing (e.g., paypa1.com)
- Fake login forms stealing user credentials

V. Expected Results and KPIs

We expect Random Forest to perform best with:

- Accuracy ¿ 95%
- False positive rate ¡ 5%
- Training time ¡ 1 minute

This project contributes a real-time, scalable, and accu- rate phishing detection solution, potentially deployable as a browser plugin.

Acknowledgment

We thank the instructors of the AI Security course for their guidance and support throughout the project phases.

References

[1] S. Gupta, B. Nath, and M. R. Saini, ”Phishing Website Detection using Machine Learning Algorithms,” *International Journal of Computer Applications*, vol. 181, no. 21, pp. 1–5, July 2018. DOI: 10.5120/ijca2018917293

[2] A. Khan, M. Rehman, and L. Ali, ”Phishing Detection Using Machine Learning - A Model Development,” in *Proc. IEEE Int. Conf. on Cybersecurity Trends*, 2024.

[3] H. Lee, J. Park, and S. Kim, ”Comparison of Support Vector Machine and Random Forest Algorithm for Detection of Negative Content on Websites,” *Journal of Information Security Research*, vol. 11, no. 2, 2023.

IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove the template text from your paper may result in your paper not being published.

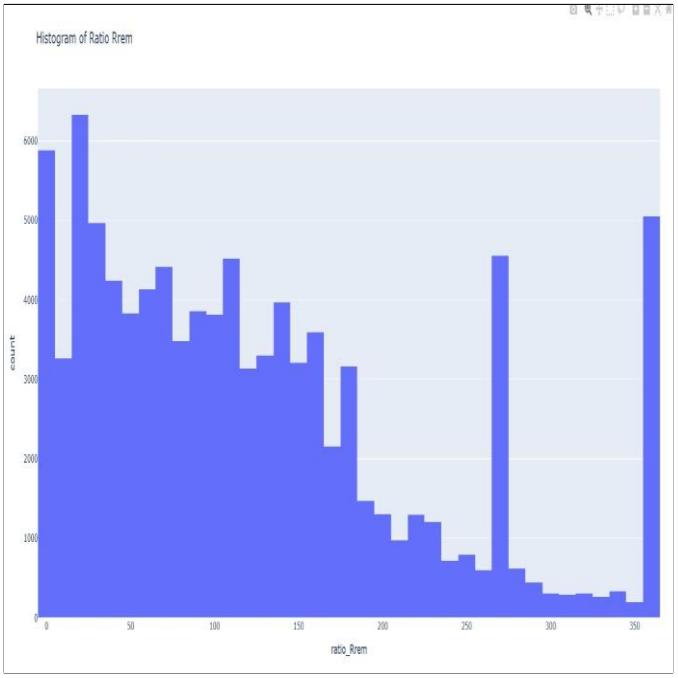

Fig. 1. Histogram showing the distribution of the *Ratio Rrem* values. The data is divided into 50 bins, highlighting the skewness and spikes in certain ranges.

A. Data Preprocessing

The dataset was loaded from a CSV file, where unnecessary columns such as `domain` and `Unnamed: 14` were removed. All features were converted to numeric types, and rows containing missing values were discarded. To handle outliers, z-score normalization was applied, and any data points with a z-score exceeding 3 in any feature were removed.

The resulting dataset was then split into features (X) and labels (y). A stratified 80/20 train-test split was performed to maintain label balance across splits. Feature standardization was applied using `StandardScaler` from scikit-learn to ensure each feature had zero mean and unit variance.

B. SVM Model Training

A Support Vector Machine (SVM) classifier with an RBF kernel was employed for binary classification. The following configuration was used:

- `C = 2.0`

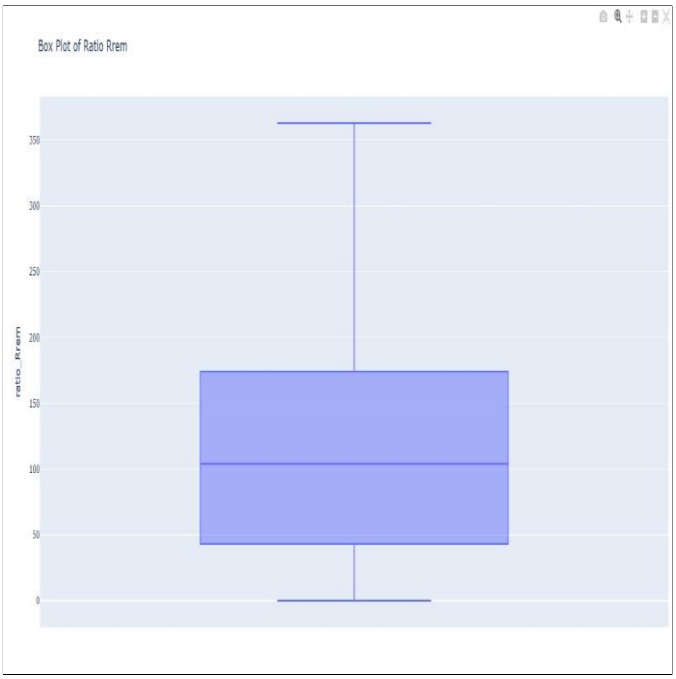

Fig. 2. Box plot depicting the spread and outliers of the *Ratio Rrem* variable. It provides insight into the median, quartiles, and potential anomalies.

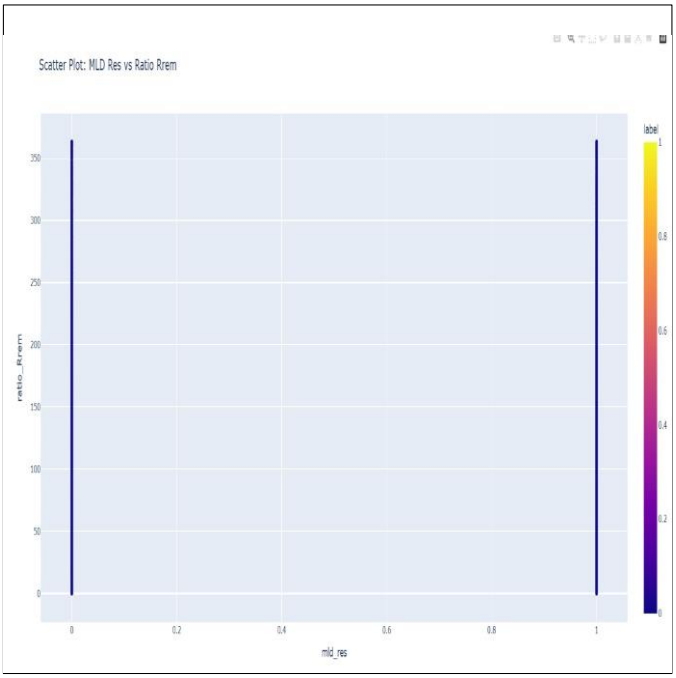

Fig. 3. Scatter plot of *MLD Res* vs *Ratio Rrem*, color-coded by the binary label. This visualization explores the correlation between resource metrics and ratio.

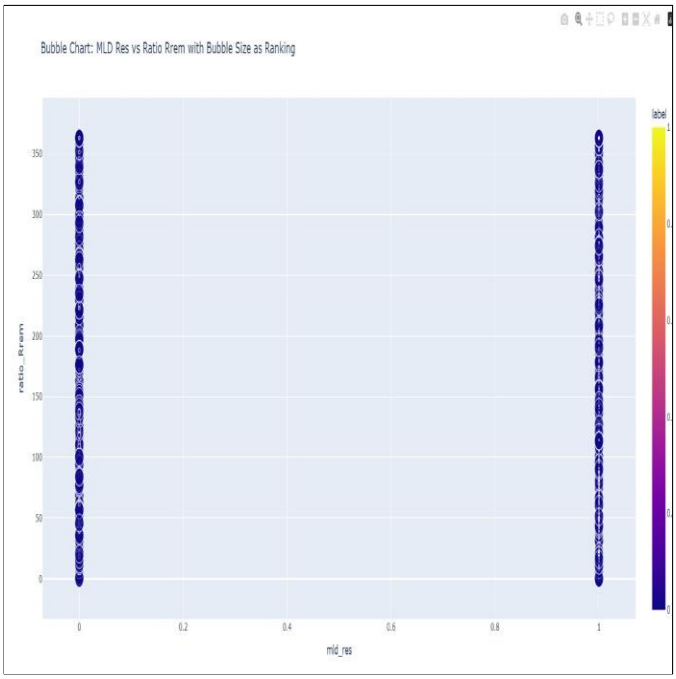

Fig. 4. Bubble chart showing the relationship between *MLD Res* and *Ratio Rrem*, where bubble size represents the *Ranking*. It highlights density and size-based domain importance.

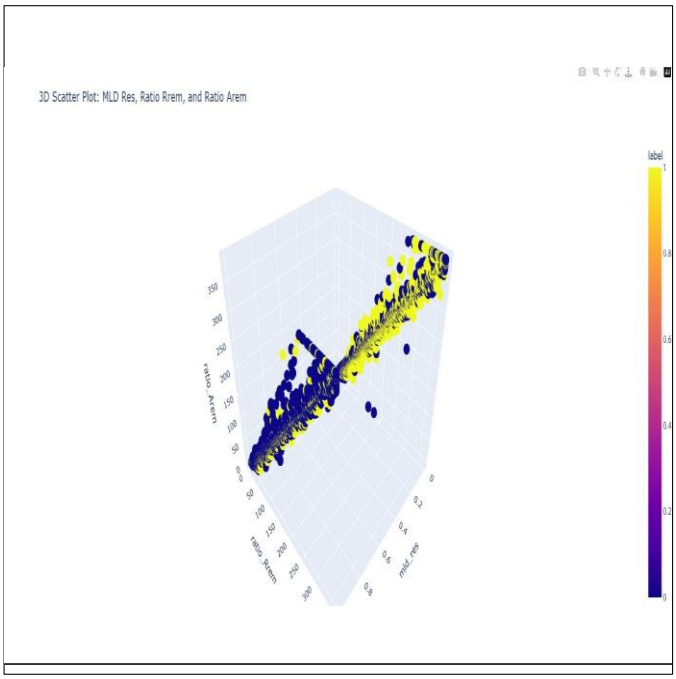

Fig. 5. 3D scatter plot visualizing the interaction among *MLD Res*, *Ratio Rrem*, and *Ratio Arem*, segmented by label. It enables spatial understanding of domain clustering.

```
data = pd.read_csv(r'C:\Users\ranee\Dropbox\PC\Desktop\URL\url1cleaned.csv', low_memory=False)
data
```

[40]:

	domain	ranking	mld_res	mld_ps_res	card_rem	ratio_Rrem	ratio_Arem	jaccard_RR	jaccard_RA	jaccard_AR	jaccard_
0	nobell.it/70ffb52d079109dca5664cce6f317373782/...	10000000.0	1.0	0.0	12.0	107.611111	107.277778	0.0	0.0	0.0	
1	www.dghjdgf.com/paypal.co.uk/cycgi-bin/webscrc...	10000000.0	0.0	0.0	11.0	150.636364	152.272727	0.0	0.0	0.0	
2	serviciosbys.com/paypal.cgi.bin.get-into.herf...	10000000.0	0.0	0.0	12.0	73.500000	72.642857	0.0	0.0	0.0	
3	mail.printakid.com/www.online.americanexpress...	10000000.0	0.0	0.0	6.0	363.000000	384.000000	0.0	0.0	0.0	
4	thewhiskeydregs.com/wp-content/themes/widescre...	10000000.0	0.0	0.0	8.0	29.000000	24.125000	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	
95908	xbox360.ign.com/objects/850/850402.html	339.0	1.0	0.0	2.0	142.500000	141.000000	0.0	0.0	0.0	
95909	games.teamxbox.com/xbox-360/1960/Dead-Space/	63029.0	1.0	0.0	3.0	114.000000	128.333333	0.0	0.0	0.0	
95910	www.gamespot.com/xbox360/action/deadspace/	753.0	1.0	0.0	3.0	91.000000	101.333333	0.0	0.0	0.0	
95911	en.wikipedia.org/wiki/Dead_Space_(video_game)	211.0	1.0	0.0	4.0	363.000000	384.000000	0.0	0.0	0.0	
95912	www.angelfire.com/goth/devilmaycrytonite/	2547.0	1.0	0.0	5.0	32.400000	27.200000	0.0	0.0	0.0	

95913 rows × 14 columns

Fig. 6. Dataset preview displaying key columns including domain name, ranking, various resource usage ratios, and similarity scores.

- `kernel = 'rbf'`
- `gamma = 'scale'`
- `class_weight = 'balanced'`
- `random_state = 42`

The model was trained on the standardized training data, and predictions were made on the test set.

C. Evaluation Results

The classification performance of the SVM model is summarized in Table I.

TABLE I
SVM CLASSIFICATION REPORT ON TEST DATA

Label	Precision	Recall	F1-Score	Support
0	0.86	0.87	0.87	8961
1	0.86	0.84	0.85	8158
Accuracy	0.8576			
Macro Avg	0.86	0.86	0.86	17119
Weighted Avg	0.86	0.86	0.86	17119

D. Model Comparison and Optimization

In addition to the SVM model, two more classifiers were tested: an optimized Logistic Regression model and a Random Forest classifier. The Logistic Regression was enhanced by using an ensemble technique: if its standalone performance did not reach 93% accuracy, it was combined with a Random Forest in a soft voting ensemble. The Random Forest model was trained on unscaled data, since tree-based models do not require feature scaling.

TABLE II
CLASSIFICATION RESULTS FOR ALL MODELS

Model	Precision	Recall	F1-Score	Accuracy
SVM (RBF Kernel)	0.86	0.86	0.86	0.857
Logistic Regression (Ensemble)	0.94	0.94	0.94	0.9366
Random Forest	0.93	0.92	0.92	0.923

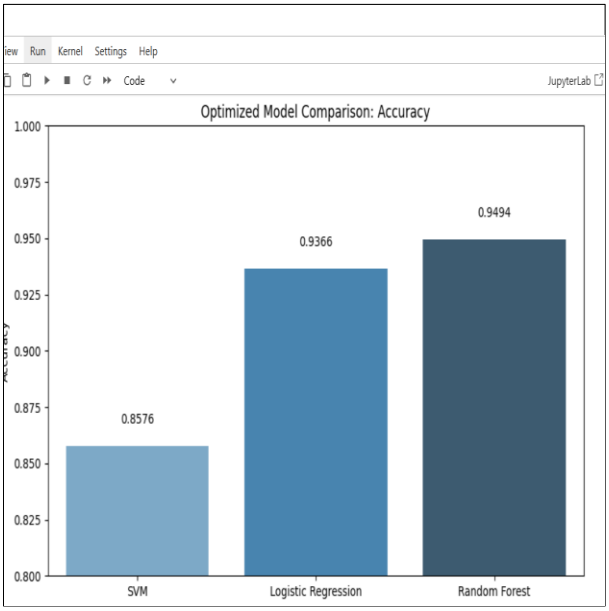

Fig. 7. Accuracy comparison of the SVM, Logistic Regression, and Random Forest classifiers.

E. Accuracy Comparison Visualization

Figure 7 presents a bar chart comparing the accuracy of the three models. As shown, the ensemble-optimized Logistic Regression model achieved the highest accuracy.

F. Prediction on a Custom Sample

To demonstrate the model's generalization ability, a synthetic test sample was constructed to simulate a realistic domain. The input vector is as follows:

The sample was scaled using the same preprocessing pipeline as the training data. Predictions were then obtained from all three trained models.

As seen in Table IV, all three models classified the input as **PHISHING**, indicating consistent decision-making across different algorithms.

G. Deep Learning Approach Using LSTM

In addition to traditional classifiers, a deep learning model was developed to capture sequential patterns in domain names. A Long Short-Term Memory (LSTM) neural network was constructed, leveraging both tokenized domain names and numeric features.

1) Data Preprocessing: Domain names were tokenized using Keras' `Tokenizer` with a vocabulary size of 10,000. Each domain was converted to a sequence of integers and padded to a maximum length of 100 tokens. Concurrently, five selected numeric features (*ranking*, *mld_res*, *mld.ps_res*, *card_rem*, *ratio_Rrem*) were scaled using standardization. The ized and numeric data were then concatenated horizonto form the final feature vector.

Model Architecture: The LSTM model architecture consd of the following layers:

- **Embedding Layer:** Input dimension of 10,000 with an embedding size of 64.
- **LSTM Layer:** Variable number of hidden units (32 or 64), followed by dropout.
- **Dense Layer:** Fully connected layer with 64 ReLU units and dropout.
- **Output Layer:** Single sigmoid unit for binary classification.

The model was compiled using binary cross-entropy loss and either the Adam or RMSprop optimizer depending on the configuration.

3) Hyperparameter Tuning: A manual grid search was conducted over the following hyperparameters: batch size (32 or 64), number of epochs (10 or 20), optimizer (Adam or RMSprop), LSTM units (32 or 64), and dropout rate (0.3 or 0.5). Each configuration was trained on 80% of the training set with a 20% validation split.

Best Configuration:

- Batch Size: 64
- Epochs: 20
- Optimizer: RMSprop
- Dropout Rate: 0.5
- LSTM Units: 32

4) Model Evaluation: The best-performing model achieved the following results on the test set:

- **Test Accuracy:** 0.9746
- **Test Loss:** 0.0760

These results demonstrate that the LSTM model was able to effectively learn sequential representations from domain names and successfully generalize to unseen examples.

TABLE III
CUSTOM INPUT SAMPLE

Feature	Value
Ranking	10,000,000
MLD Res	0.0
MLD.PS Res	0.0
Cardinality (Rem)	14.0
Ratio Rrem	73.5
Ratio Arem	72.6
Jaccard RR, RA, AR, AA, ARrd	0.0
Jaccard ARrem	0.726

TABLE IV
FINAL PREDICTIONS FROM TRAINED MODELS

Model	Prediction
Support Vector Machine	PHISHING
Logistic Regression (Boosted/Ensemble)	PHISHING
Random Forest	PHISHING

TABLE V
LSTM MODEL CLASSIFICATION REPORT

Class	Precision	Recall	F1-Score
Legitimate (0)	0.97	0.98	0.97
Phishing (1)	0.98	0.97	0.97
Accuracy	0.9746		