**CYBER SECURITY ASSIGNMENT-2**

# REPORT

**Name**: Vidiyala Ashritha

**Roll No**:160123737089

**Date**: 05/10/2025

**Github Repository**: <https://github.com/ashritha2006/phishing-url-detector.git>

**RESEARCH PAPER:** [**https://ieeexplore.ieee.org/abstract/document/10058201/**](https://ieeexplore.ieee.org/abstract/document/10058201/)

By: Karim, Abdul, et al. "Phishing detection system through hybrid machine learning based on URL." *IEEE Access* 11 (2023): 36805-36822.

## INTRODUCTION

The chosen research paper focuses on the application of **machine learning for phishing URL detection**, demonstrating the potential of ML models to automatically identify malicious websites based on URL characteristics. However, the paper primarily evaluates individual models such as LightGBM, Random Forest, Decision Tree, Logistic Regression, and SVM, reporting accuracy scores without deep feature engineering or interpretability techniques.

This project builds upon and extends the original work by developing a **prototype phishing URL detection framework** that integrates:

* **Enhanced Feature Engineering** using Shannon entropy and phishing keyword detection.
* **Hybrid Ensemble Learning** combining Random Forest and XGBoost for improved generalization
* **Reproducible Implementation** using Python, GitHub, and publicly available Kaggle data.
* **Explainability via SHAP** to interpret feature importance and model behavior.
* **Comprehensive Model Evaluation** using accuracy, precision, recall, F1-score, and ROC AUC.

## RESEARCH GAP

From the paper:

* No advanced lexical or semantic features (e.g., entropy, keyword analysis) were utilized.
* No explainability or feature importance analysis was provided.
* Evaluation was limited to accuracy without detailed performance metrics.
* Individual models were tested separately without ensemble approaches.

This project provides:

* **Improved feature set** including entropy and phishing keyword flags
* **SHAP-based explainability** to identify influential URL features.
* **Full quantitative evaluation** (precision, recall, F1, ROC AUC) for realistic performance analysis.
* **Open-source, reproducible Python implementation** hosted on GitHub.
* **Ensemble model** (Random Forest + XGBoost) for better phishing detection.

## METHODOLOGY

### 3.1 Dataset Preprocessing & Feature ExtractionAgents perform benign or malicious actions.

* Raw data is sourced from the Kaggle Phishing URLs dataset (≈ 5.4 lakh URLs).
* Data is cleaned by removing duplicates and null entries.
* A subset of 50,000 samples is selected for efficient processing and experimentation.
* Each URL is processed to extract lexical features, including:

1. URL length, number of dots, presence of @, hyphens, subdomain count, etc.

* Additional discriminative features are introduced:

1. Shannon Entropy of the domain to detect random-looking phishing domains.
2. Phishing Keywords (login, bank, secure, etc.) flagged as binary features.

Implementation: src/preprocess.py

Output: data/processed\_features.csv (baseline) and data/improved\_features.csv (enhanced feature set)

### Baseline Model – XGBoost

* A baseline model is trained using **XGBoost** on lexical features only.
* Train–test split: 80–20
* Metrics evaluated: Accuracy, Precision, Recall, F1 Score, ROC AUC.
* Implementation: src/train\_model\_baseline.py

Results stored in:

1. results/baseline\_metrics.csv
2. results/baseline\_confusion\_matrix.png

### Improved Model – Hybrid Ensemble (Random Forest + XGBoost)

* A **hybrid ensemble model** is developed by combining predictions from **Random Forest** and **XGBoost**.
* Predictions are averaged to produce the final label.
* Uses the **enhanced feature set** (lexical + entropy + keyword features).
* This approach improves recall and F1 score by leveraging complementary strengths of both models.

Implementation: src/train\_model.py

Results stored in:

* results/metrics.csv
* results/confusion\_matrix.png

### Model Explainability – SHAP Analysis

* **SHAP (SHapley Additive exPlanations)** is used to identify feature contributions.
* A subset of 50 test samples is used to compute SHAP values efficiently.
* The summary plot reveals top influential features such as entropy, URL length, and phishing-related keywords.

Implementation: Inside src/train\_model.py

Output: results/shap\_feature\_importance.png

results/shap\_force.png

results/ shap\_summary.png

### Example Workflow

The complete workflow is automated through sequential execution of scripts:

1. Run preprocessing to generate features:

python src/preprocess.py --input data/urls\_dataset.csv --output data/processed\_features.csv --sample\_size 50000

1. Train and evaluate the **baseline model**:

python src/train\_model\_baseline.py --data data/processed\_features.csv

1. Run improved preprocessing with extra features:

python src/preprocess.py --input data/urls\_dataset.csv --output data/improved\_features.csv --sample\_size 50000

1. Train and evaluate the **improved ensemble model**:

python src/train\_model.py --data data/improved\_features.csv

1. Collect metrics, confusion matrices, and SHAP plots from the results/ folder.
   1. **Output Artifacts**

* **Metrics:** Baseline & improved performance in .csv format.
* **Confusion Matrices:** PNG images for both models.
* **SHAP Plots:** Feature importance visualization.
* **Processed Datasets:** Baseline and improved feature CSV files.
* **Code Repository:** All scripts and outputs are hosted on GitHub for reproducibility.

## RESULTS

A blue squares with numbers and a white text

AI-generated content may be incorrect. A blue squares with numbers and a white background

AI-generated content may be incorrect.

## 

A close up of a graph

AI-generated content may be incorrect.

A graph with blue and pink dots

AI-generated content may be incorrect.

## DISCUSSION

* The baseline XGBoost model achieved **85.62% accuracy**, matching the paper’s individual model results.
* Adding **entropy** and **phishing keyword** features, and using a **Random Forest + XGBoost ensemble**, improved **recall to 46.31%** and **F1 score by 3.3%**, enhancing phishing detection.
* **SHAP analysis** revealed that **entropy**, **URL length**, and keywords like login, bank, and secure were the most influential features.
* The project delivers a **working prototype**, providing a reproducible, end-to-end phishing detection system — addressing the paper’s lack of practical implementation.
* The pipeline, built entirely in **Python** and hosted on **GitHub**, ensures transparency and repeatability.

## FUTUREIMPROVEMENTS

* Incorporate **deep learning models** (e.g., CNN/LSTM) for semantic URL analysis.
* Integrate **host-based features** (e.g., WHOIS, DNS, SSL certificate data) to further boost accuracy.
* Optimize SHAP calculations using GPU acceleration or approximation methods for large datasets.
* Develop a **real-time browser extension** or web dashboard to visualize phishing detections and feature importance.
* Experiment with **more advanced ensembles** (e.g., stacking, blending) to push performance further

.

## CONCLUSION

This project provides a working proof-of-concept phishing URL detection system that goes beyond the original research paper. By combining enhanced lexical features, entropy and keyword analysis, ensemble learning, and SHAP explainability, it delivers measurable improvements in recall and F1-score while maintaining accuracy.  
The approach is fully reproducible, leveraging open datasets and transparent code, and serves as a practical demonstration of how research gaps can be translated into real, testable cybersecurity solutions.