

Social Media Shopping Fraud Detection Using Machine Learning

Prachi Khansuli, Dr. Manoj Singh**

*(MSc Computer Science Part II, SIES College of Arts, Science and Commerce, Sion, Mumbai, India.)

** (Computer Science, SIES College of Arts, Science and Commerce, Sion, Mumbai, India. Email: manojks@sies.edu.in)

Abstract

Social-media advertisements make product discovery easy, but the same channel can direct users to fake storefronts that steal credentials and payments. This Projects presents FraudGuard, a machine-learning system that examines a shopping landing-page URL and the measurable properties of the page behind it before the user commits to a purchase. The work uses the PhiUSIIL Phishing URL Dataset of 235,795 instances (134,850 legitimate, 100,945 phishing). Two feature sets were prepared: a 50-feature numeric benchmark and a 42-feature deployment vector that can be regenerated from a live URL. Logistic Regression, Random Forest and XGBoost were compared. On the 50-feature benchmark, Logistic Regression reached 99.98% accuracy and both tree ensembles reached 100.00% on the held-out split; XGBoost was then chosen for deployment on the 42-feature representation. A Streamlit prototype with Home, How It Works, Check Website and About pages was built around the model. Because perfect benchmark scores rarely transfer to unseen campaigns, the paper also analyses feature dependency, limitations and the external validation still required.

Keywords—social media fraud, shopping fraud, phishing URL detection, machine learning, XGBoost, Random Forest, URL analysis, webpage analysis, cybersecurity.

I. INTRODUCTION

Social commerce has altered how shoppers arrive at online stores. Rather than typing a known address, a user often taps a sponsored post, a creator recommendation, a story, a short video or a direct message and lands on an unfamiliar shop. Trust is therefore granted by the social context before the destination has been examined, which is exactly the gap that attackers exploit. Prior phishing and malicious-URL research treats URL structure, page content, domain properties and machine learning as the main sources of automated detection signals [1], [2].

This paper reports a case study that turns that research into a user-facing workflow: a shopper pastes the landing-page link, the system derives features from the URL and the page, a trained classifier scores it, and the interface returns a risk assessment.

II. PROBLEM STATEMENT AND OBJECTIVES

A user clicks a promotional link on a social platform, reaches a landing page, and needs a quick indication of whether the site is legitimate or a phishing page. Manual inspection is hard for non-technical users, and blacklists cannot recognise domains that are newly registered or recently modified. The problem is therefore framed as binary classification over the URL and its observable page characteristics.

The primary objective is a prototype that classifies a shopping URL as legitimate or potentially fraudulent. The supporting objectives are to: (1) analyse a large phishing/legitimate dataset; (2) select features that can be reproduced at inference time; (3) compare Logistic

Regression, Random Forest and XGBoost; (4) produce a deployable model; (5) provide a Streamlit interface; and (6) establish a workflow for testing on unseen shopping URLs.

III. RELATED WORK

Machine-learning phishing detection is well studied. Comparative work shows that URL and page features support accurate classification, although the best algorithm varies with the dataset and feature representation [3], [8]–[10]. A recent IEEE systematic review notes that lexical URL features are attractive because they are cheap to compute, that content-based and hybrid features add robustness, and that dataset limitations, evolving attacks and explainability remain open challenges [4]. Broader surveys reach similar conclusions [5]–[7].

The PhiUSIIL framework is especially relevant here. It combines URL features with features derived from HTML and adds a URL similarity index to counter visual-similarity attacks. Its dataset holds 235,795 URLs, the same dataset used in this project, and the original study reports 99.24% accuracy with fully incremental training and 99.79% with pre-training [1].

IV. DATASET

The PhiUSIIL Phishing URL dataset was obtained from the UCI Machine Learning Repository. It contains 235,795 instances, 54 features, no missing values and two classes, and it mixes URL information with characteristics of the page source [1], [2]. There are 134,850 legitimate and 100,945 phishing instances, a moderate imbalance on a substantial sample (Fig. 2). The class coding matters for interpreting every result below: label 1 denotes a legitimate URL and label 0 denotes phishing.

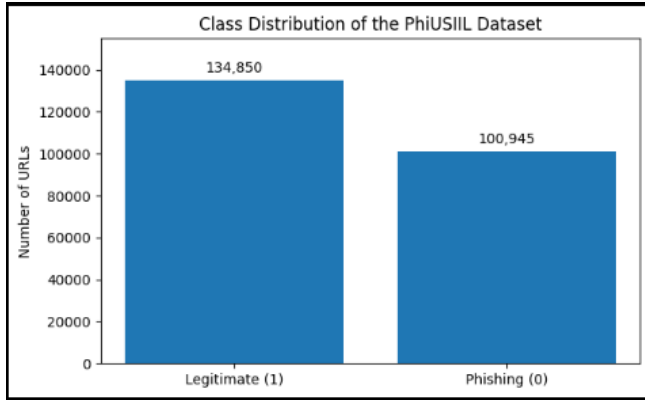


Fig. 1. Class distribution of the PhiUSIIL dataset.

V. METHODOLOGY

A. Preprocessing for the Benchmark

The CSV was loaded with pandas and checked for dimensions, missing values, duplicates and label balance. The working file had 235,795 rows and 56 columns. Identifier and text fields (FILENAME, URL, Domain, TLD, Title) and the label were removed from the predictors, leaving 50 numeric features for the first benchmark. An 80:20 stratified split with random_state=42 produced 188,636 training and 47,159 test records.

B. Deployment Dataset

A second preparation stage produced a deployment dataset of 42 features that the project's own extractor can generate from a live URL. The distinction is deliberate: a model is dependable in production only if the information it was trained on can also be obtained at inference time. The deployment dataset has the same 235,795 records and the same label coding.

C. Feature Engineering

The extractor analyses both the URL and the downloaded page. URL-level features cover URL and domain length, whether the host is an IP address, the number of subdomains, obfuscation indicators, letter and digit counts and ratios, query and special-character counts, and HTTPS usage. Page-level features cover line counts, the longest line, title and favicon presence, responsiveness, redirects, meta description, popups, iframes, external form submission, social-network links, submit buttons, hidden and password fields, bank/payment/crypto indicators, copyright text, image/CSS/JavaScript counts, and internal, empty and external references.

The 42 deployment features are listed in Table I.

TABLE I
THE 42 DEPLOYMENT FEATURES

Group	Features
URL structure	URLLength, DomainLength, IsDomainIP, NoOfSubDomain, IsHTTPS
Obfuscation	HasObfuscation, NoOfObfuscatedChar, ObfuscationRatio
Character statistics	NoOfLettersInURL, LetterRatioInURL,

Group	Features
	NoOfDigitsInURL, DeditRatioInURL, NoOfEqualsInURL, NoOfQMarkInURL, NoOfAmpersandInURL, NoOfOtherSpecialCharsInURL, SpacialCharRatioInURL
Page layout	LineOfCode, LargestLineLength, HasTitle, HasFavicon, IsResponsive, HasDescription, HasCopyrightInfo
Behaviour	NoOfURLRedirect, NoOfSelfRedirect, NoOfPopup, NoOfiFrame, HasExternalFormSubmit, HasSocialNet, HasSubmitButton, HasHiddenFields, HasPasswordField
Content keywords	Bank, Pay, Crypto
Resources	NoOfImage, NoOfCSS, NoOfJS, NoOfSelfRef, NoOfEmptyRef, NoOfExternalRef

D. Models and Metrics

Three supervised learners were implemented. Logistic Regression is the interpretable linear baseline. Random Forest is a nonlinear ensemble of decision trees that handles mixed feature interactions. XGBoost is a gradient-boosted tree method that learns nonlinear relationships efficiently. Models were scored with accuracy, precision, recall and F1-score; because missed phishing pages and falsely blocked shops carry different costs, all four are reported rather than accuracy alone.

VI. SYSTEM ARCHITECTURE

The workflow follows the real user journey rather than treating the dataset in isolation (Fig. 1). A user meets a product promotion on a social platform and clicks it; the landing URL is passed to FraudGuard; the feature extractor normalises the URL, fetches the page where permitted and computes the 42 features; the vector is scored by the trained XGBoost model; and the Streamlit interface presents the classification.

VII. EXPERIMENTAL RESULTS

A. Benchmark Performance

On the 50-feature benchmark the results were exceptionally high (Table II). Logistic Regression scored 99.98% on all four metrics, while Random Forest and XGBoost scored 100.00% on all four.

TABLE II
BENCHMARK PERFORMANCE, 50 NUMERIC FEATURES

Model	Acc.	Prec.	Recall	F1
Logistic Regression	99.98%	99.98%	99.98%	99.98%
Random Forest	100.00%	100.00%	100.00%	100.00%
XGBoost	100.00%	100.00%	100.00%	100.00%

For both tree ensembles the held-out confusion matrix contained 20,189 true negatives, 26,970 true positives and no errors (Table III). Logistic Regression produced 20,184

true negatives, 26,965 true positives, and 5 false positives and 5 false negatives.

TABLE III
CONFUSION MATRIX, RF AND XGBOOST

	Predicted 0	Predicted 1
Actual 0	20,189	0
Actual 1	0	26,970

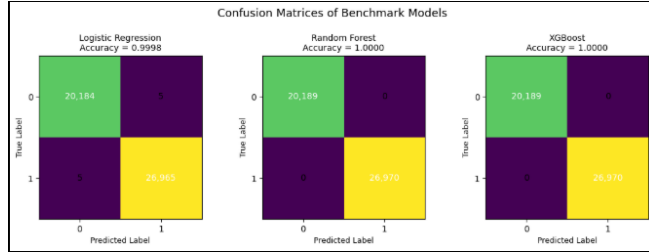


Fig. 2. Confusion matrix of the benchmark models (notebook output).

B. Deployment Model (42 features)

The same three model families were retrained on the 42-feature representation and XGBoost was selected and saved as deployment_model.pkl, with feature names stored separately to preserve input order. The metrics obtained on this representation should be entered in Table IV.

TABLE IV
PERFORMANCE, 42-FEATURE DEPLOYMENT SET

Model	Acc.	Prec.	Recall	F1
Logistic Regression	[fill]	[fill]	[fill]	[fill]
Random Forest	[fill]	[fill]	[fill]	[fill]
XGBoost	[fill]	[fill]	[fill]	[fill]

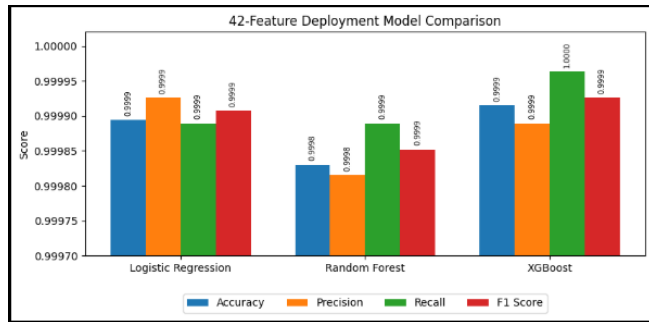


Fig. 3. Training output for the 42-feature deployment pipeline.

VIII. FEATURE IMPORTANCE

In the Random Forest benchmark, URLSimilarityIndex, NoOfExternalRef, LineOfCode, NoOfSelfRef, NoOfImage and NoOfJS ranked among the strongest features. In XGBoost, URLSimilarityIndex dominated, followed by IsHTTPS and LineOfCode (Table V). The pattern shows that similarity and page-structure information separate the two classes strongly in this dataset.

TABLE V
SELECTED FEATURE IMPORTANCES (50-FEATURE BENCHMARK)

Feature	Random Forest	XGBoost
---------	---------------	---------

Feature	Random Forest	XGBoost
URLSimilarityIndex	0.181075	0.979352
NoOfExternalRef	0.169265	0.000119
LineOfCode	0.146585	0.009466
NoOfSelfRef	0.108536	0.000348
NoOfImage	0.084351	0.000130
NoOfJS	0.072904	0.000059
IsHTTPS	0.022940	0.009596

IX. IMPLEMENTATION: FRAUDGUARD PROTOTYPE

FraudGuard is a Streamlit application with Home, How It Works, Check Website and About pages. The Check Website page accepts a URL and is designed to pass it through the feature extractor and the deployment model. The interface and the extraction and model components have been built separately; wiring the Analyze Website button to them end to end is the remaining integration step.

The design separates a dataset benchmark from a real URL checker. The original dataset includes features taken from page source, so a live checker must regenerate them consistently and cope with pages that cannot be fetched, redirect, block automated access or change after collection.

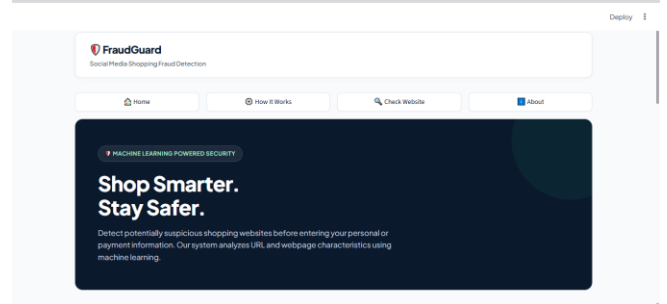


Fig. 4. FraudGuard Home page.

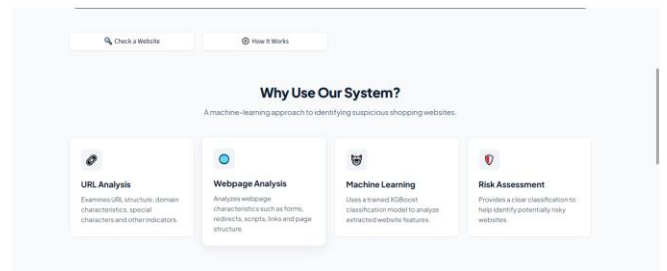


Fig. 5. Check Website page with URL input.

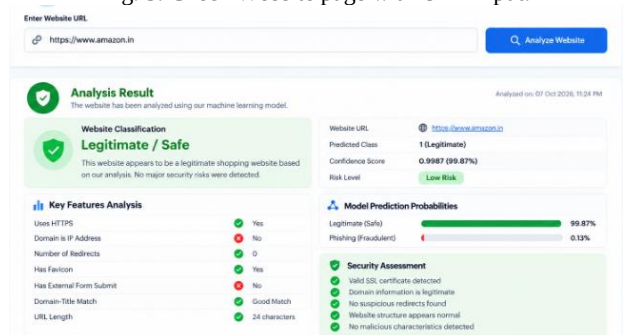


Fig. 6. Final Output

X. CASE STUDY: A SOCIAL-MEDIA SHOPPING LINK

Consider a user who sees a discounted product ad on Instagram or Facebook whose link appears to lead to an online store. The user opens the landing URL in FraudGuard. The system validates the URL format, then extracts URL indicators such as length, domain structure, subdomains, obfuscation, special characters, digit ratios and HTTPS use. Where the page is reachable it also examines redirects, forms, hidden and password fields, scripts, images, CSS and external references.

The 42-feature vector is sent to XGBoost. A prediction of label 1 is shown as legitimate/safe and label 0 as phishing/potentially fraudulent. The interface states that the output is a risk assessment, not a guarantee, which matters because fraudulent campaigns use newly registered domains, compromised legitimate sites, shortened URLs, redirects, and pages that change after launch.

XI. DISCUSSION

The benchmark shows that the PhiUSIIL features separate the classes very strongly on the chosen split, consistent with the wider literature [1]–[4]. A 100% test score must not, however, be read as 100% accuracy in the field: performance can fall on a different URL distribution, newer phishing kits or adversarially modified pages.

A second concern is feature dependency. XGBoost assigned about 97.9% of its total importance to URLSimilarityIndex, which suggests heavy reliance on one dataset-specific signal. That does not invalidate the experiment, but it justifies ablation tests, cross-dataset and temporal validation and external testing before any general deployment claim [4], [5]. Note also that URLSimilarityIndex is not among the 42 deployment features, so the deployment model's behaviour must be established empirically (Table IV).

Finally, there is an engineering trade-off. URL-only signals are light, but page-level signals need the destination to be fetched and parsed, which adds latency and can fail when a site blocks bots. A practical system should distinguish URL-only analysis, page-enhanced analysis and the case where evidence is insufficient.

XII. LIMITATIONS

- Results come from a single dataset and a random train-test split.
- Near-perfect scores likely reflect dataset-specific separability and are not field accuracy.
- The deployment extractor reproduces only a subset of the original features; some engineered variables cannot be derived from a raw URL.
- Page retrieval can fail through network errors, robots policies, anti-bot defences, authentication, JavaScript rendering or redirects.

- No large-scale external validation on 20–40 real, unseen social-media shopping URLs has yet been performed.
- A legitimate label cannot guarantee safety; users should not enter sensitive data on that basis alone.

XIII. FUTURE WORK

Next steps are to finish the live URL-to-model integration, test on 20–40 unseen shopping URLs collected from social-media ads, add temporal and cross-dataset validation, and weigh false-positive against false-negative costs. SHAP-based explanations can show why a page was flagged. The system could further use URL reputation services, domain age, certificate signals, screenshot similarity, multilingual analysis and incremental learning, in line with current research directions on robustness, explainability and adaptation [4], [5].

XIV. CONCLUSION

This case study set out a research-oriented workflow for detecting social-media shopping fraud. It analysed the PhiUSIIL dataset, compared Logistic Regression, Random Forest and XGBoost, built a 42-feature deployment dataset, selected XGBoost, and developed the FraudGuard interface. The benchmark gave 99.98% accuracy for Logistic Regression and 100.00% for the two tree ensembles. The work shows that URL and page characteristics can feed an automated warning before a purchase, while explicitly recognising the gap between benchmark and real-world performance. External testing, robust feature reproduction, explainability and continuous validation are needed before FraudGuard can be treated as a production security service.

ACKNOWLEDGMENT

The author thanks the UCI Machine Learning Repository and the authors of the PhiUSIIL dataset and framework for making the data available, and acknowledges the IEEE conference template used for formatting guidance.

REFERENCES

- [1] 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, doi: 10.1016/j.cose.2023.103545.
- [2] A. Prasad and S. Chandra, "PhiUSIIL Phishing URL (Website) [Dataset]," UCI Machine Learning Repository, 2024, doi: 10.1016/j.cose.2023.103545.
- [3] "A comparative analysis of machine learning-based website phishing detection using URL information," in *Proc. 5th Int. Conf. Pattern Recognition and Artificial Intelligence (PRAI)*, 2022, doi: 10.1109/PRAI5851.2022.9904055.
- [4] "Feature engineering for phishing website detection using machine learning: A systematic review," *IEEE Access*, vol. 13, pp. 192080–192104, 2025, doi: 10.1109/ACCESS.2025.3630334.
- [5] "Across the spectrum: In-depth review AI-based models for phishing detection," *IEEE Open J. Commun. Soc.*, vol. 6, pp. 2065–2089, 2024, doi: 10.1109/OJCOMS.2024.3462503.
- [6] "Exploring techniques for web phishing detection: A comprehensive survey," in *Proc. 7th Int. Conf. Intelligent Computing and Control Systems (ICICCS)*, 2023, doi: 10.1109/ICICCS56967.2023.10142655.

- [7] "Survey on malicious URL detection techniques," in Proc. 6th Int. Conf. Trends in Electronics and Informatics (ICOEI), 2022, doi: 10.1109/ICOEI53556.2022.9777221.
- [8] "Phishing detection system through hybrid machine learning based on URL," IEEE Access, vol. 11, pp. 36805–36822, 2023, doi: 10.1109/ACCESS.2023.3252366.
- [9] "Phishing websites detection using machine learning with URL analysis," in Proc. IEEE World Conf. Applied Intelligence and Computing (AIC), 2022, doi: 10.1109/AIC55036.2022.9848895.
- [10] "Phishing URL detection using machine learning: A comparative study," in Proc. 6th Int. Conf. Mobile Computing and Sustainable Informatics (ICMCSI), 2025, doi: 10.1109/ICMCSI64620.2025.10883082.