Int. j. adv. multidisc. res. stud. 2026; 6(6):660-673

Received: 27-09-2026
Accepted: 07-11-2026

International Journal of Advanced Multidisciplinary Research and Studies

ISSN: 2583-049X

Smart Fraud Detection System for Shoprite Online Payments: A Random Forest Approach with Leakage-Controlled Evaluation and a Protocol for Collecting Verified Labels

[1] Gloria Ilunga Kasongo, [2] Sikwa

[1, 2] Department of Computer Science, DMI-St. Eugene University, Lusaka, Zambia

DOI: https://doi.org/10.62225/2583049X.2026.6.6.7173

Corresponding Author: **Gloria Ilunga Kasongo**

Abstract

Online grocery payments by card and mobile money are exposed to fraud, yet a retailer rarely holds the fraud-labelled data that supervised detection needs. This study presents a smart fraud detection system for the online payments of Shoprite, built as an academic case study with synthetic checkouts and public datasets; no Shoprite data were used. A transparent rule engine scores every order and sends suspicious ones to an administrator for review. A Random Forest model is consulted only when an order has the fields it was trained on; otherwise the system reports an explicit unsupported status and creates no score. Analysts record verified outcomes for later training. We tested the Random Forest and two baseline classifiers on two public datasets, 283,726 card transactions (473 frauds) and 235,370 web addresses (100,520 phishing), splitting before any fitting, grouping addresses by website and choosing thresholds without the test data. On held-out card data the Random Forest was the best model, with an area under the precision-recall curve of 0.800 (95% confidence interval 0.713 to 0.876), precision 0.926 and recall 0.789; oversampling before the split inflated the same model to 1.00. On web addresses a logistic model scored 0.999, but every legitimate address in the data was a secure site root starting with www and having no path; rewriting phishing addresses in the legitimate format cut its detection from 99.5% to 29.6%. A robustness gate now blocks such models, and a host-name model that replaced it detected 52% of phishing addresses at a 0.8% false-alarm rate. No labelled Shoprite fraud data exist yet, and the paper sets out how to collect them.

Keywords: Fraud Detection, Shoprite Online Payments, Random Forest, Mobile Money, Data Leakage, Probability Calibration

1. Introduction

1.1 Background

Mobile money is now a major way for people in sub-Saharan Africa to move value. The GSMA counted more than two billion registered mobile money accounts worldwide in 2024, more than one billion of them in sub-Saharan Africa, and almost 1.7 trillion US dollars passing through those accounts during the year [1, pp. 3, 14]. Online shops can reach these customers through payment aggregators that offer card and mobile money checkout, so a grocery retailer such as Shoprite can offer card and mobile money checkout online through a hosted payment page. The same report says that fraud remained a pervasive issue for the industry in 2024 and that most of it involves agents or customers [1, p. 63].

A grocery order differs from the transactions studied in most fraud research. The basket holds everyday items, the amounts are small, and the merchant sees only what the customer types at checkout and what the payment provider returns. The risk assessment in this study uses no device fingerprint, no network address or location, and no card numbers, which stay with the payment provider. Whatever assessment the store runs must work with the fields it has, and a member of staff must be able to explain a score to a customer who was flagged by mistake. Shoprite is used here as the case context. This paper is an academic study that uses synthetic checkouts and public datasets; it uses no Shoprite data and is not affiliated with or endorsed by Shoprite.

1.2 Problem Statement

Two problems follow. First, supervised models need labelled fraud, and a new store has none. Public datasets are the obvious substitute, but they describe other things: anonymised card transactions, simulated mobile money transfers and phishing web addresses. None carries the fields of a grocery order. Second, published performance on such datasets is hard to trust. A survey of machine-learning research found leakage of test information into training in 294 papers across 17 fields [2], and an analysis of card-fraud studies reproduces a recall of 99.9% with a deliberately flawed pipeline [3]. A shop that adopts a model because a paper reports near-perfect scores may be adopting an artefact of the evaluation.

1.3 Research Gap

- Reviews of e-commerce fraud detection describe many classifiers and report a shortage of real datasets [4]. Mobile money research leans on simulators because real transaction logs are private [5, 6].
- Studies of operational card-fraud systems show that labels arrive late and only for cases someone investigated [7]. When outcomes are observed only for the cases that received an action, models and decision makers are hard to compare fairly [8].
- Benchmark data can reward cues that do not carry over to real conditions [9], and published scores are often inflated by leakage [2, 3].
- The studies reviewed here do not combine, for an online retailer such as Shoprite, a decision aid built from its own fields, a refusal to score inputs a model never saw, and a plan for gathering the labels that the first two leave missing.

1.4 Research Objectives

General Objective

To design, implement and evaluate a smart, auditable fraud detection system for Shoprite online payments, in which a Random Forest model and a transparent rule engine help administrators decide which orders need human review, using only data the store records.

Specific Objectives

1. Define the fields the store records at checkout as a versioned schema that excludes outcome information and personal identifiers.
2. Implement a transparent rule engine, a Random Forest scoring service and a guard that refuses to score inputs a model was not trained on.
3. Prepare two public datasets and train baseline and tree-based models under a leakage-controlled protocol, with the Random Forest as the candidate production model.
4. Measure the uncertainty of the results and the effect of common evaluation shortcuts.
5. Specify how analyst-verified outcomes become training labels and when a trained model may be promoted.

1.5 Research Questions

1. Which fields can an online grocery store such as Shoprite use for risk assessment, and how can the assessment stay explainable to staff and customers?
2. How well do logistic regression, random forests and gradient boosting detect card fraud and phishing addresses when the evaluation avoids leakage, and how uncertain are the estimates?
3. How much do common evaluation shortcuts inflate reported scores?
4. Can public datasets support a supervised model for grocery orders and, if not, what protocol will produce usable labels?
5. Are the predicted probabilities calibrated, and which inputs drive the predictions?

1.6 Research Contributions

- A smart fraud detection framework that joins a rule engine, a Random Forest scoring service behind a schema guard and per-order model records, with append-only analyst outcomes (Section 3.1).
- A leakage-controlled benchmark on two public datasets, with bootstrap intervals, paired tests and five repeated splits (Section 4).
- Measured inflation from evaluation shortcuts: oversampling before the split, keeping duplicate rows, and formatting artefacts in the web-address data (Sections 4.9 and 5).
- A robustness gate that refuses address models whose verdicts change with cosmetic edits, motivated by the failure of the first model we trained, and a weaker host-name model that passes it (Sections 3.1 and 4.9).
- A label-collection protocol with an audit sample of unflagged orders, readiness thresholds, shadow-mode deployment for experimental models and a promotion gate that requires a model to beat the rules (Section 3.1).
- An open implementation with automated tests and continuous integration (Section 4.10). The contribution is the framework and its evaluation; none of the classifiers is new.

2. Literature Review

2.1 Existing Methods

Fraud detection has been reviewed many times. Recent reviews cover anomaly detection for financial fraud [10] and machine learning for e-commerce fraud, where a systematic review of 101 publications reports a rising use of neural networks and a shortage of real datasets [4]. Practitioners have described what makes card fraud hard: class imbalance, change in behaviour over time and the delay before labels arrive. Dal Pozzolo *et al.* [7] formalised these problems for a stream of more than 75 million transactions over three years and proposed a learning strategy for them. Hybrid designs add unsupervised outlier scores to a supervised classifier and report better detection [11]. In online marketplaces, supervised classifiers with custom preprocessing and imbalance handling can detect organised retail fraud from buyer and seller behaviour [12].

Mobile money has its own literature, but real transaction logs are private, so researchers use simulators. PaySim generates mobile money transactions from an original dataset [5], and MoMTSim is a multi-agent simulation calibrated for mobile money transactions whose synthetic data are compared with real data by statistical tests [6]. Simulated labels describe the simulator's assumptions about fraud, so they cannot confirm that a model finds fraud in a real store.

Phishing detection from web addresses has a similar history. Sahingoz *et al.* [13] compared seven classifiers on language-based address features and reported 97.98% accuracy for a random forest. A review of 80 studies groups the techniques

into list-based, visual-similarity, heuristic, machine-learning and deep-learning approaches [14]. Prasad and Chandra [15] released the dataset used here: 134,850 legitimate and 100,945 phishing addresses, with features drawn from the address and from the page.

2.2 AI and machine-learning approaches

Tree ensembles remain strong on medium-sized tabular data, even against deep networks [16], and random forests [17] are a common fraud baseline. Class imbalance is usually handled by resampling or by class weights. A study of balancing methods found that oversampling helps weak classifiers but not strong ones whose hyperparameters were tuned [18]. Rebalancing also distorts predicted probabilities, which matters when a threshold is meant to express a cost [19]. Isotonic regression and Platt scaling repair the distortion; Niculescu-Mizil and Caruana [20] found that boosted trees, random forests and support vector machines gave the best probabilities after calibration.

For evaluation, precision-recall curves are more informative than ROC curves when positives are rare [21], and the Matthews correlation coefficient summarises all four cells of the confusion matrix [22]. Leakage, meaning the use at training time of information that would not exist at prediction time, was described for data mining more than a decade ago [23]. In card-fraud research it takes the form of resampling or scaling before the split, vague reporting, weak temporal validation and thresholds tuned for recall [3]. Shortcut learning describes models that succeed on a benchmark through cues that fail under real conditions [9].

2.3 Comparative Analysis

Table 1: Comparison of related studies. Only figures stated in the cited sources are given

Author	Method	Dataset	Performance	Limitations
Dal Pozzolo *et al.* [7]	Learning strategy for imbalance, drift and label delay	Stream of more than 75 million card transactions over three years	Realistic evaluation on a long stream; no single score quoted here	Card transactions only; does not address retail checkout fields
Carcillo *et al.* [11]	Supervised classifier plus unsupervised outlier scores at several granularities	Real annotated card data	Detection accuracy improves over the supervised stage alone	Supervised stage still needs labels
Hayat and Magnier [3]	Critical examination of card-fraud evaluation	Card data of European cardholders	A flawed pipeline reaches 99.9% recall	Evaluation study; no deployable system
Lopez-Rojas *et al.* [5]	PaySim agent-based simulator of mobile money	Synthetic, based on an original private dataset	Not applicable (simulator)	Fraud labels come from the simulator's assumptions
Azamuke *et al.* [6]	MoMTSim multi-agent simulator calibrated to mobile money data	Synthetic, calibrated against real mobile money data	Synthetic and real data compared with Kolmogorov-Smirnov test, sum of squared errors and Bland-Altman plots	Synthetic labels
Sahingoz *et al.* [13]	Seven classifiers on language-based URL features; random forest best	Phishing and legitimate URLs described in the paper	97.98% accuracy	Address text only
Prasad and Chandra [15]	Framework with a URL similarity index and incremental learning	PhiUSIIL: 134,850 legitimate and 100,945 phishing URLs	Not quoted here	Page-content features need the page to be fetched
Emmanuel and Oghie [24] (preprint)	Multi-layer perceptron on 16 URL structure features	PhiUSIIL (235,795 URLs)	99.24% accuracy, 98.74% precision, 99.95% recall, F1 99.34%, ROC-AUC 99.65%	Preprint; split not stated in the abstract
Mutemi and Bacao [4]	Systematic review of machine learning for e-commerce fraud	101 publications	Not applicable (review)	Reports few real datasets
This study	**Rule engine, schema guard, leakage-controlled models, robustness gate, label protocol**	**ULB card data; PhiUSIIL; scripted store checkouts**	**Card PR-AUC 0.800; URL host-name model PR-AUC 0.863**	**No grocery fraud labels yet; weak URL model**

Table 2: Research gaps and how this study responds

Gap	Evidence	Response in this study
No labelled fraud data for a new store	Shortage of real datasets [4]; reliance on simulation [5, 6]	Rule engine from the store's own fields; label-collection protocol
Public benchmarks lack a retailer's fields	Card data holds anonymised components; PaySim holds account balances	Versioned schema and a guard that returns unsupported_input_schema
Inflated or leaky reported scores	[2, 3]	Leakage-controlled protocol; measured inflation from evaluation shortcuts
Benchmark shortcuts	[9]	Feature ablation, perturbation tests and a robustness gate on the phishing data
Biased labels from review	[7, 8]	Verified flag, audit sample of unflagged orders, rules baseline in the promotion gate

2.4 Research Gap

The studies above leave five gaps that matter to a small retailer. They are listed in Table 2 together with the part of this study that responds to each.

3. Materials and Methods

3.1 Proposed Framework

Fig 1 shows the system. A customer builds a cart and checks out through a React storefront. The FastAPI service creates the order and a payment attempt in one transaction, keyed by an idempotency header so that a repeated request returns the same order. The customer chooses card, MTN Mobile Money or Airtel Money, or cash on delivery, and the service asks the provider's hosted checkout page for the matching option. The service accepts an online payment only after asking the provider directly to verify the reference, currency and amount, and a background job re-checks pending payments every minute. We tested the provider integration against a fake provider server and made no real payment during this study.

When the order is saved, the rule engine store_risk_v1 computes a score from twelve fields (Table 4) and stores the inputs and the reasons with the assessment. A score of 30 or more opens a case for a person to review, a score of 60 or more is labelled high, and four rules (high value, bulk quantity, order velocity and repeated rejections) open a case on their own. Nothing blocks the order or the payment. The same event is offered to the model service, which first validates the input against the versioned schema grocery_txn_v1. A missing, unexpected or malformed field produces the status unsupported_input_schema or invalid_input and no score is created. The research card model expects 30 inputs (Time, Amount and 28 anonymised components) that the store schema does not contain, so for every order it answers with the field names it could not find. A model trained on store data would run in shadow mode while it is marked experimental: its probability is stored beside the rule score and it cannot raise alerts. Cases appear in an admin-only staff console, where an administrator inspects the stored reasons and decides what to do. The Random Forest is the model proposed for scoring store orders once enough verified labels exist; until then the rule engine alone decides which orders reach the administrator.

Staff can record an outcome for any order: confirmed fraud, false positive, legitimate or undetermined, with a flag that shows whether evidence outside the system verified it. Rows are only appended, and the newest row for an order wins. Only verified, definite outcomes become labels, and unreviewed orders are not assumed legitimate. A deterministic weekly sample puts about one in ten unflagged orders in front of reviewers, so that labels are not drawn only from orders the rules already suspected [8]. Training is attempted only with at least 150 verified fraud cases, 1,000 verified legitimate cases and 50 customers. The validation gate then requires 30 fraud rows in a customer-grouped test set, a score well above a do-nothing baseline, a successful reload in a fresh process and a better score than the rule engine on the same held-out customers. A model trained on fewer than 1,000 verified fraud cases stays experimental. Fig 2 shows the protocol used for every model in this paper.

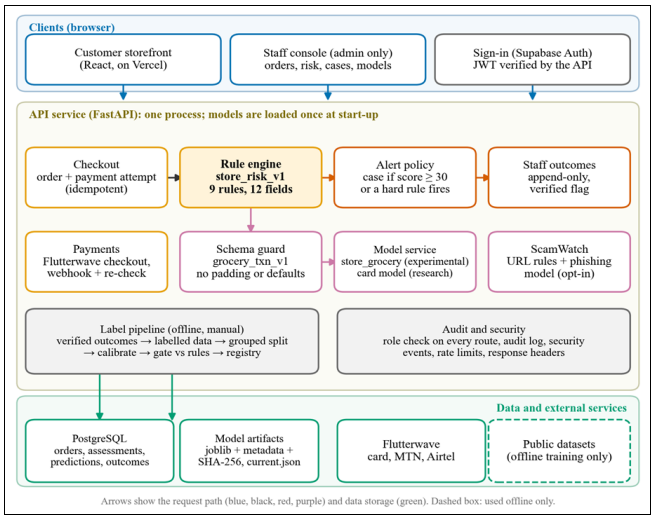


Fig 1: System architecture of the store and its risk-assessment services

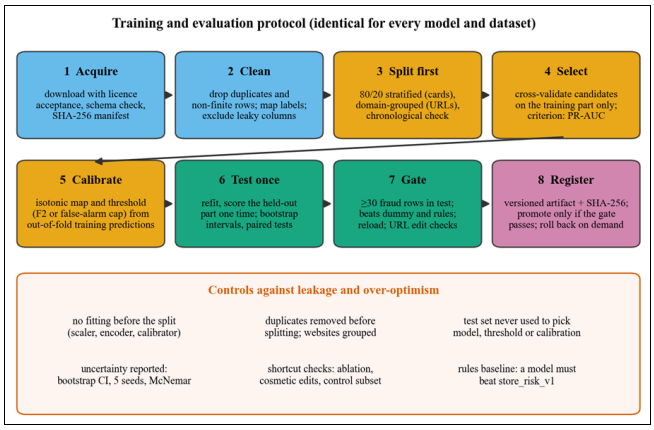


Fig 2: Training and evaluation protocol, with the controls against leakage

The address model has one more gate, added after the experiments in Section 4.9. On up to 4,000 held-out legitimate addresses it changes one cosmetic cue at a time (removing a leading www, switching to http, adding a trailing slash, adding a path), and it rewrites up to 4,000 held-out phishing addresses in the legitimate format. A model is refused if the flagged share of legitimate addresses rises by more than five percentage points after any change, or if it keeps less than 80% of its detections after the rewrite. Every address model is also saved as experimental, and the ScamWatch checker skips experimental models, so no verdict from one reaches a user.

3.2 Dataset Description

Table 3: Datasets used. No store data were used for training

Dataset	Source and licence	Rows (raw, cleaned)	Positive class	Inputs used	Split (seed 42)
Card transactions	ULB Machine Learning Group and Worldline, OpenML mirror 1597; licence field "Public" [19]	284,807; 283,726	473 fraud (0.167%)	Time (as 24-hour cycle), Amount, V1 to V28	Stratified 80/20: train 226,980 (378 fraud), test 56,746 (95 fraud)
Web addresses	PhiUSIIL, UCI data set 967; CC BY 4.0 [15]	235,795; 235,370	100,520 phishing (42.7%)	31 features computed from the address text (10 host-name features for the replacement model)	Grouped by domain 80/20: train 188,086, test 47,284 (20,118 phishing)
Store orders	Scripted checkouts in a throw-away database (synthetic customers)	10 scenarios	Not applicable	12 schema fields (Table 4)	Not applicable

The card data hold the transactions of European cardholders over two days in September 2013. The file has 284,807 rows with Time, Amount, 28 anonymised principal components (V1 to V28) and the label Class. After removing 1,081 exact duplicate rows, 283,726 rows remain, of which 473 are fraud. A classifier that always answers "legitimate" would be right 99.83% of the time, which is why accuracy is not used to rank models.

From the phishing data we took only the address text, the label and the registrable domain, which is the grouping key. After removing 425 duplicate addresses, 235,370 rows remain. In the file, label 1 means legitimate; we inverted it so that 1 means phishing. Both datasets were downloaded in October 2026 with their licence terms accepted, and each file was checked against a SHA-256 checksum recorded at download.

3.3 Data pre-processing

- Missing values: none occurred in either dataset. The pipeline would drop rows with missing or non-finite values and count them.
- Noise removal: exact duplicate rows (card data) and duplicate addresses (web data) were removed before the split, so that copies cannot sit on both sides.
- Normalisation: every input was standardised with the mean and variance of the training partition only, inside the pipeline.
- Encoding: the two public datasets have numeric inputs. The store schema has two categorical fields, encoded one-hot with fixed vocabularies.
- Data augmentation: none.
- Class balancing: class weights (balanced) instead of resampling, followed by probability calibration. We did not use SMOTE because balancing mainly helps weak classifiers [18] and because oversampling before the split is one of the leakage routes examined in Section 4.9.

3.4 Feature Engineering

For the card data, Amount was log-transformed. Time counts seconds from the first transaction of the file, so it identifies a position in the collection window instead of a property of a transaction; we converted it to a position in a 24-hour cycle (sine and cosine) and dropped the raw value. For the web addresses we built two feature sets. The first has 31 lexical features computed from the whole address string: lengths of the address, host, path and query; counts of dots, hyphens, underscores, at-signs, equals signs, question marks, ampersands, percent signs, digits and other special characters; digit and letter ratios; the number of subdomains; whether the host is an IP address or contains punycode; the entropy and longest label of the host; the length of the top-level domain and whether it is common; flags for the secure protocol, a non-standard port, a leading "www" and a double slash in the path; and a count of suspicious keywords. The second, built after the experiments in Section 4.9, has 10 features of the canonical host name only: its length, number of subdomains, digit ratio, hyphen count, entropy, longest label, top-level domain length and commonness, and flags for an IP address and for punycode. The canonical host ignores the scheme, a leading www, the port, the path, the query and any trailing slash. We excluded the dataset's page-content columns because they require fetching the page, which the checker never does; columns computed from dataset-wide statistics because they cannot be recomputed for one new address; and identifiers.

The store schema (Table 4) holds only fields known when the order arrives. It excludes the payment outcome of the order, order and delivery status, the analyst verdict, names, contact details, address text, card data and customer identifiers. All twelve fields are used; no feature selection was applied.

Table 4: Store feature schema grocery_txn_v1. Currency must be ZMW; other currencies are refused

Field	Type	Allowed values	Source at checkout	Privacy note
total_cents	int	1 to 100,000,000	orders.total_cents	order value only
max_line_quantity	int	0 to 10,000	max(order_items.quantity)	basket shape only, no product names
orders_last_hour	int	0 to 1,000	count of this customer's orders in the hour up to the event	behavioural count for the customer's own account
prior_orders	int	0 to 1,000,000	count of this customer's earlier orders	behavioural count for the customer's own account
prior_paid_orders	int	0 to 1,000,000	count of this customer's earlier paid orders	behavioural count for the customer's own account
average_paid_total_cents	int	0 to 100,000,000	mean total of this customer's earlier paid orders	aggregate of the customer's own history
failed_payments_24h	int	0 to 1,000	failed payment attempts by this customer in 24 h before the event	behavioural count; failure reasons are not used
rejected_checkouts_10m	int	0 to 1,000	checkouts rejected (stock/validation) in the 10 min before the event	behavioural count
delivery_address_seen_before	bool	true or false	whether the delivery address matches one this customer used before	the address itself is never a feature, only this yes/no flag
fulfilment	category	delivery, collection	orders.fulfilment	none
local_hour	int	0 to 23	order creation time converted to Zambian local time	hour only, not the full timestamp
payment_method	category	card, mtn_momo, airtel_money, cod, simulated	orders.payment_method (the chosen method, never card data)	method label only; card numbers never reach this store

Table 5: Rules of store_risk_v1. A hard rule opens a case even when the total score is below 30

Rule	Condition	Points	Hard
high_value	Order total at or above the high-value limit	30	yes
bulk_quantity	A line quantity at or above the bulk limit (15)	15	yes
order_velocity	Four or more orders by the customer within an hour	20	yes
repeated_rejections	Five or more rejected checkouts in ten minutes	20	yes
first_order_large	No earlier paid order and a total of at least half the high-value limit	15	no
spend_spike	At least two earlier paid orders, a total of at least four times their average, and at least ZMW 100	20	no
failed_payments	Three or more failed payments in 24 hours (two failures score 10)	25	no
new_address	Delivery to an address the customer has not used, for a customer with earlier orders	10	no
night_order	Order placed between 00:00 and 05:00 local time	5	no

3.5 Models

Four classifiers were compared on each public dataset: a dummy classifier that predicts the training prevalence; logistic regression with L2 penalty and balanced class weights; a random forest of 150 trees with balanced subsample weights [17]; and histogram-based gradient boosting with balanced class weights. All were implemented in scikit-learn [25]. We did not train deep networks: with 378 training frauds and 30 inputs, tree ensembles are a strong default for tabular data [16]. The scikit-learn gradient-boosting classifier belongs to the same family as XGBoost, LightGBM and CatBoost and needs no extra dependency.

The rule engine is a set of nine named rules with fixed points (See Table 5). The score is the sum of the points that fire, capped at 100, and every assessment stores the fields used and a plain-language reason for each rule. The limits are deployment parameters; the defaults are a high-value limit of ZMW 5,000, a bulk quantity of 15, four orders per hour and five rejected checkouts in ten minutes.

3.6 Model Optimisation

Hyperparameters were chosen by four-fold stratified cross-validation on the training partition, with average precision (PR-AUC) as the criterion; Table 6 lists the grids and the selected values. The chosen model's out-of-fold predictions were used twice: to fit an isotonic regression that turns scores into probabilities [20], and to choose the decision threshold. For the card model and the original web-address models the threshold maximises the F2 score, which weights recall twice as heavily as precision because a missed fraud costs more than an unnecessary review. For the host-name models, which may only raise suspicion, the threshold gives the highest recall at a false-positive rate of at most 1%. The test partition played no part in either step. The inner cross-validation for the web addresses was stratified, not grouped by domain, so its scores are optimistic; the outer test partition is grouped and its results are the ones reported.

Table 6: Hyperparameter grid and cross-validated PR-AUC for the card data (training partition, four folds)

Model	Settings	CV PR-AUC	Selected
Logistic regression	C = 0.1	0.732	
Logistic regression	C = 1.0	0.736	yes
Random forest	min_samples_leaf = 2, max_depth = none	0.834	yes
Random forest	min_samples_leaf = 5, max_depth = 14	0.831	
Gradient boosting (HistGB)	max_leaf_nodes = 31, learning_rate = 0.1, max_iter = 200	0.741	
Gradient boosting (HistGB)	max_leaf_nodes = 15, learning_rate = 0.05, max_iter = 300	0.745	yes

Note. Random forests used 150 trees. The web-address models used the same grids for logistic regression and gradient boosting and one random-forest setting (minimum leaf size 2).

3.7 Explainability

The rule engine explains itself: each assessment lists the rules that fired and their points. For the trained models we used permutation importance, which measures how much PR-AUC falls when one input is shuffled. It was computed on held-out rows with five repeats: all 95 test frauds plus 8,000 legitimate rows for the card data, and 8,000 random rows for the web addresses. For the web-address logistic model we also read the standardised coefficients. We did not use SHAP because permutation importance answers the question asked here, which inputs the held-out score depends on, at a fraction of the cost.

3.8 Experimental Setup

Experiments ran in Python 3.14.0 with scikit-learn 1.8.0, pandas 3.0.3, NumPy 2.4.4 and SciPy 1.17.1 on Windows 11 with 12 logical processors and 8 GB of memory. All random seeds are fixed (42 for the main split; 11, 22, 33, 44 and 55 for the repeated splits). The software, the experiment scripts and the saved results are in a public repository (release tag paper-v1). The application is also tested on Linux with Python 3.12 and PostgreSQL 16 in continuous integration (Section 4.10).

3.9 Evaluation Metrics

For each model we report accuracy, precision, recall, specificity, F1, the Matthews correlation coefficient [22], the area under the ROC curve, and the area under the precision-recall curve (average precision), which is the primary metric [21]. Precision, recall and F1 use the threshold chosen on training data. Calibration is measured by the Brier score and by the expected calibration error in ten equal-count bins. Uncertainty is shown by 95% percentile intervals from a stratified bootstrap of 1,000 resamples that keeps the class counts fixed [26, 27]. Models are compared with a paired bootstrap of the same resamples and with the exact McNemar test on the rows where two classifiers disagree [28]. Variation caused by the split is measured with five further random splits.

4. Experimental Results

4.1 Dataset Analysis

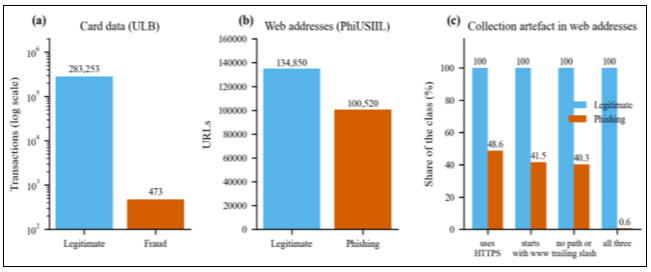


Fig 3: Class balance of the two public datasets (a, b) and the collection artefact in the web-address data (c)

Fig 3 shows the two class balances. Fraud is 0.167% of the card transactions, while phishing is 42.7% of the web addresses. Panel (c) shows a property of the web data that the later results depend on. All 134,850 legitimate addresses have the form https://www.domain: 100% use the secure protocol, 100% start with www, and 100% have no path, query or trailing slash. Among phishing addresses the shares are 48.6%, 41.5% and 40.3%, and only 0.6% (627 of 100,520) match the legitimate format in all three respects. The mean address length is 27 characters for legitimate and 46 for phishing.

4.2 Model Performance

Table 7: Card fraud: held-out performance (56,746 transactions, 95 fraud)

Model	Accuracy	Precision	Recall	F1-score	MCC	ROC-AUC	PR-AUC [95% CI]
Dummy (training prior)	0.9983	0.000	0.000	0.000	0.000	0.500	0.0017
Logistic regression	0.9993	0.830	0.768	0.798	0.798	0.963	0.690 [0.597, 0.790]
Random forest	**0.9995**	**0.926**	**0.789**	**0.852**	**0.855**	**0.929**	**0.800 [0.713, 0.876]**
Gradient boosting (HistGB)	0.9990	0.689	0.768	0.726	0.727	0.970	0.678 [0.589, 0.773]

Note. Precision, recall, F1 and MCC at each model's own threshold, chosen on out-of-fold training predictions. Bold marks the highest PR-AUC.

Table 8: Web addresses, original 31-feature models: held-out performance (47,284 addresses, 20,118 phishing, domains kept apart)

Model	Accuracy	Precision	Recall	F1-score	MCC	ROC-AUC	PR-AUC [95% CI]
Dummy (training prior)	0.5745	0.000	0.000	0.000	0.000	0.500	0.4255
Logistic regression	0.9973	0.999	0.995	0.997	0.994	0.998	0.999 [0.998, 0.999]
Random forest	0.9975	0.999	0.995	0.997	0.995	0.998	0.998 [0.998, 0.999]
Gradient boosting (HistGB)	**0.9977**	**1.000**	**0.995**	**0.997**	**0.995**	**0.998**	**0.999 [0.998, 0.999]**

Note. Thresholds chosen on out-of-fold training predictions. Bold marks the highest PR-AUC. Section 4.9 shows that these scores mostly reflect how the addresses were formatted.

On the card data the random forest has the highest PR-AUC (0.800). The paired bootstrap puts its advantage over logistic regression at 0.108 (95% interval 0.047 to 0.173) and over gradient boosting at 0.120 (0.060 to 0.178); both intervals exclude zero. By PR-AUC, logistic regression and gradient boosting cannot be separated (difference 0.012, interval −0.030 to 0.055). The random forest also has the best precision (0.926, interval 0.871 to 0.974). Its recall is 0.789 (interval 0.705 to 0.874), against 0.768 for both other models. The wide intervals come from the 95 fraud rows in the test partition: a single missed fraud moves recall by about one percentage point.

The exact McNemar test compares the decisions that each model makes at its own threshold. The random forest is right where logistic regression is wrong on 11 test rows, and the reverse never happens ($p < 0.001$). Against gradient boosting the counts are 32 rows in the forest's favour and 3 in the other's ($p < 0.001$). Unlike PR-AUC, this test also separates logistic regression from gradient boosting: logistic regression is right where gradient boosting is wrong on 26 rows, and the reverse holds on 8 ($p = 0.003$). Gradient boosting raises more false alarms (33, against 15 for logistic

regression and 6 for the random forest). All three p values stay below 0.017, the Bonferroni threshold for three comparisons.

The ROC-AUC ranks the models in the opposite order: gradient boosting 0.970, logistic regression 0.963, random forest 0.929. Calibration is not the cause: the raw forest score has the same ROC-AUC (0.929) and a PR-AUC of 0.810; calibration lowers PR-AUC slightly because it reduces the score to 28 distinct values. The cause is where the missed frauds land. The forest ranks 77 of the 95 test frauds above 99.9% of legitimate transactions, against 75 for logistic regression and 74 for gradient boosting, but 13 of its frauds fall below the median legitimate transaction (3 and 1 for the other two). ROC-AUC penalises those low-ranked frauds heavily, whereas PR-AUC rewards the top of the list, which is where an analyst works [21]. We therefore rank models by PR-AUC.

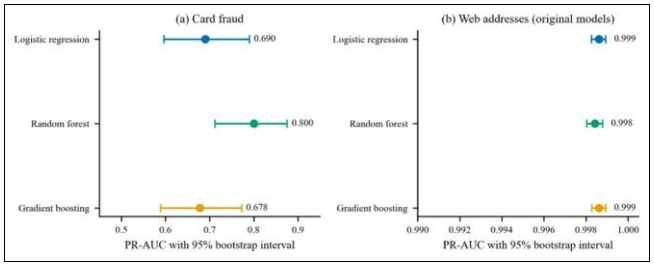


Fig 4: PR-AUC with 95% bootstrap intervals for the three classifiers. The no-skill baselines (the class prevalences, 0.0017 for cards and 0.43 for web addresses) lie outside the plotted ranges

On the web addresses all three original models reach a PR-AUC of at least 0.998, and the largest difference between them is 0.0002. The simplest model, logistic regression, ties for the best score (0.999, interval 0.998 to 0.999). Section 4.9 shows why these scores should not be read as accuracy on real links.

Table 9: Calibration on the held-out partition, before and after isotonic calibration

Data	Model	Brier (raw)	Brier (calibrated)	ECE (raw)	ECE (calibrated)
Card	Logistic regression	0.02280	0.00056	0.07189	0.00013
Card	Random forest	0.00046	0.00045	0.00037	0.00024
Card	Gradient boosting (HistGB)	0.00738	0.00064	0.04281	0.00023
Web addresses	Logistic regression	0.00241	0.00236	0.00080	0.00044
Web addresses	Random forest	0.00242	0.00220	0.00545	0.00031
Web addresses	Gradient boosting (HistGB)	0.00219	0.00220	0.00043	0.00029

Note. ECE is the expected calibration error in ten equal-count bins; lower is better.

Class weights shift a classifier's scores away from true probabilities. On the card data, isotonic calibration cuts the Brier score of logistic regression from 0.0228 to 0.00056, and its expected calibration error from 0.0719 to 0.00013. The random forest was already close to calibrated (expected calibration error 0.00037 before and 0.00024 after). Fig 5 shows the reliability of the two selected models. For the original web-address model the middle of the diagram is poor, but 99.6% of the held-out addresses score below 0.05 or above 0.95 and only 0.11% fall between 0.2 and 0.8, so the expected calibration error stays small.

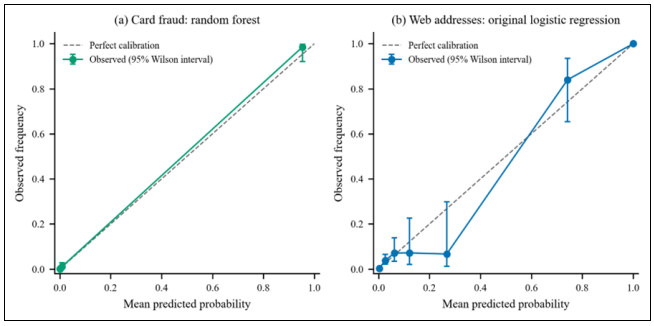


Fig 5: Reliability of the calibrated scores on the held-out partitions. Bars show 95% Wilson intervals; bins with fewer than 15 rows are omitted

4.3 Confusion Matrix

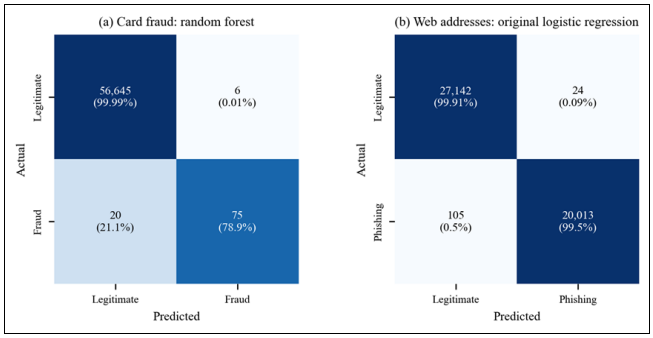


Fig 6: Confusion matrices at the chosen thresholds. Percentages are row shares

On the card data the random forest flags 81 of 56,746 transactions (1.4 per 1,000). Of these, 75 are fraud and 6 are not, so 92.6% of the alerts are real. The 20 missed frauds are 21.1% of all frauds, and 0.011% of legitimate transactions are flagged. On the web addresses the original logistic model makes 24 false alarms and 105 misses in 47,284 addresses.

4.4 ROC Curve

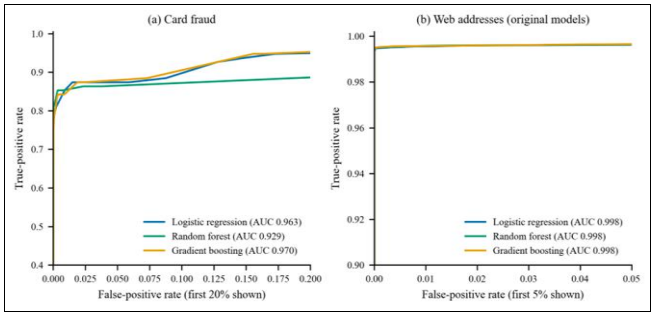


Fig 7: ROC curves on the held-out partitions (axes truncated to show the region where the curves differ)

4.5 Precision-recall curve

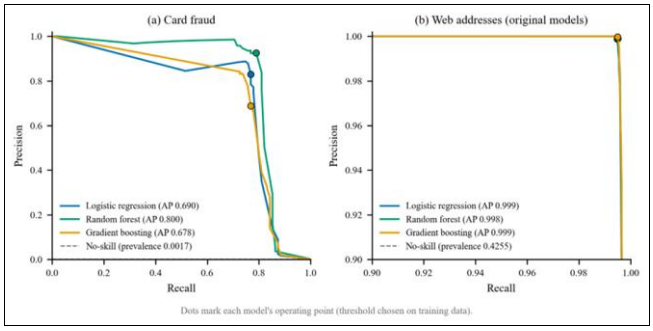


Fig 8: Precision-recall curves on the held-out partitions. The dashed line is the prevalence

The precision-recall curve for the card data shows the trade-off an analyst faces. The random forest keeps precision at or above 0.9 up to a recall of 0.79; neither of the other two models reaches a precision of 0.9 at any threshold. At a precision of 0.5 the three models reach recalls of 0.79 (logistic regression), 0.82 (random forest) and 0.78 (gradient boosting). The operating points marked on each curve come from thresholds that were fixed before the test partition was scored.

4.6 Training and validation curves

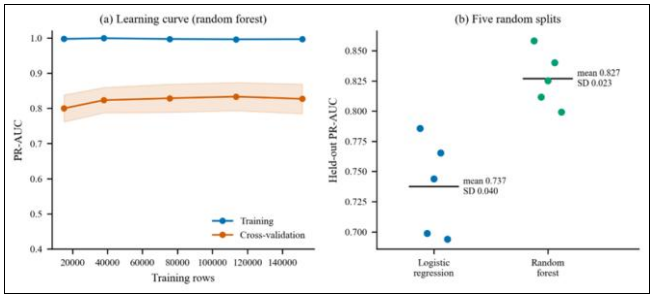


Fig 9: Learning curve of the random forest on the training partition (a) and held-out PR-AUC on five random splits (b)

The learning curve in Fig 9a shows a training PR-AUC between 0.996 and 0.999 at every size. The cross-validated PR-AUC (three folds) rises from 0.800 with 15,132 training rows to 0.829 with 75,660 rows (about 126 frauds), then stays between 0.827 and 0.833 up to 151,320 rows. The standard deviation across folds is about 0.04, so the later differences are within noise and more rows of the same kind would add little. The gap between the training and validation curves is expected for a random forest, whose trees fit the training rows almost perfectly; the validation curve is the one to read. Panel (b) shows how much a single split can mislead. Across five random splits the held-out PR-AUC of the random forest averages 0.827 with a standard deviation of 0.023 and a range of 0.799 to 0.858; for logistic regression the mean is 0.737 (standard deviation 0.040). The random forest is ahead on average in these splits, and the main split (0.800) falls below their mean.

Table 10: Mean (standard deviation) over five random 80/20 splits of the card data

Model	PR-AUC	ROC-AUC	Precision	Recall	F1-score
Logistic regression	0.737 (0.040)	0.972 (0.008)	0.821 (0.064)	0.792 (0.032)	0.805 (0.033)
Random forest	0.827 (0.023)	0.956 (0.009)	0.866 (0.045)	0.844 (0.017)	0.854 (0.019)

A chronological check trains on the oldest 80% of the transactions and tests on the newest 20% (74 frauds). PR-AUC is 0.816 for the random forest, 0.757 for logistic regression and 0.769 for gradient boosting, so the random forest also comes first when the split follows time. Precision and recall are omitted for this check because the threshold of the main split does not carry over to a different training set.

4.7 Feature Importance

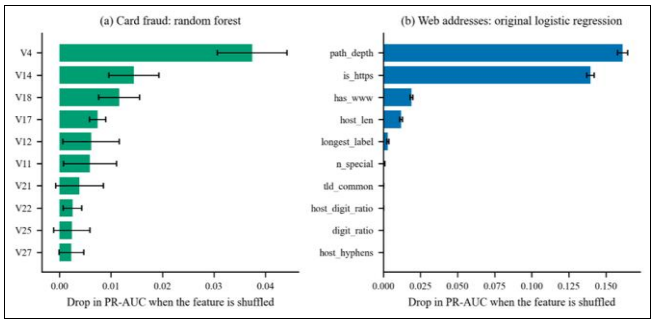


Fig 10: Permutation importance on held-out data (mean drop in PR-AUC; bars show standard deviation over five repeats)

For the card data, shuffling V4 (PR-AUC drop 0.037), V14 (0.014), V18 (0.012) or V17 (0.007) hurts the random forest most. The forest's own impurity-based importance puts V14, V10 and V12 first, so the two measures share only V14 among their leading features. The components are anonymised, so the ranking shows which components carry signal but not what they mean. For the original web-address model the largest drops come from path_depth (0.161), is_https (0.140), has_www (0.019) and host_len (0.012). The top three are formatting cues, the ones examined in Section 4.9. The standardised coefficients of the same model rank those three first as well (path_depth +8.64, is_https −6.69, has_www −5.16), with the signs that the data imply: a deeper path raises the phishing score, and a secure protocol or a www prefix lowers it.

4.8 Explainability Results

Table 11 and Fig 11 show ten scripted checkouts that between them make all nine rules fire. Each starts from a fresh customer, with earlier orders dated three days back unless the scenario is about recent behaviour. The limit for a high-value order was set to ZMW 300 so that basket sizes in the demonstration catalogue can reach it.

Table 11: Rule-engine behaviour on scripted checkouts

S. No	Scenario	Rules that fired	Score	Decision	Case
1	Routine repeat basket	none	0	allow	no
2	First-time medium basket	first order large	15	allow	no
3	First-time bulk basket	high value, bulk quantity, first order large	60	high	yes
4	Spend spike	spend spike	20	allow	no
5	Order burst	order velocity	20	allow	yes
6	Repeated failed payments	failed payments	25	allow	no
7	New delivery address	new address	10	allow	no
8	Night-time order	night order	5	allow	no
9	Repeated rejected checkouts	repeated rejections	20	allow	yes
10	Combined warning signs	high value, bulk quantity, spend spike, new address, night order	80	high	yes

Note. A score of 30 or more is review and 60 or more is high. An order burst and repeated rejections each score 20 but open a case through hard rules.

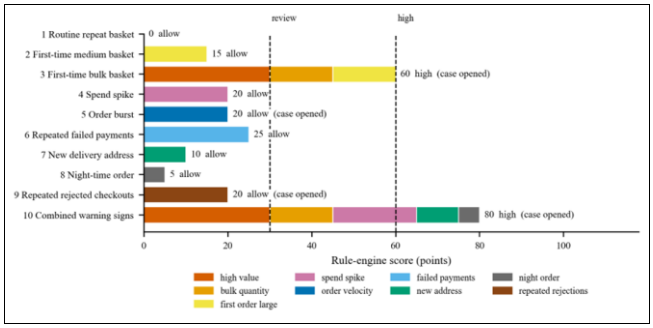


Fig 11: Rule points by scenario. Dashed lines mark the review and high bands

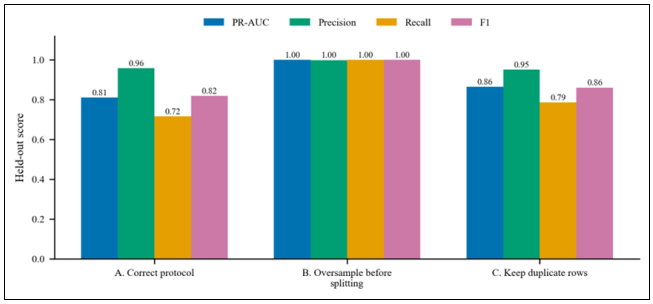


Fig 12: Held-out scores of the same random forest under three protocols

Weak signals do not escalate on their own: a spend spike, a new delivery address, a night-time order or three failed payments each score 25 points or fewer and open no case. A strong or combined pattern does escalate, as in the bulk first order (60) and the combined case (80). An order burst and repeated rejected checkouts each score only 20 but open a case, because their rules are hard rules. Three failed payments in a day score 25 and do not reach review. A store that wants them to count can raise that rule's points, and the stored reasons make the change easy to justify.

4.9 Comparison with existing methods

Direct comparison with published numbers is limited by differences in features, splits and thresholds. We therefore compare in three ways. Table 12 and Fig 12 reproduce, on the same card data, evaluation shortcuts of the kind that Hayat and Magnier [3] describe. Table 13, Table 14 and Fig 13 examine what the web-address scores depend on. Table 15 and Table 16 then compare models with each other and with published results.

Table 12: Effect of evaluation shortcuts on the same random forest (card data, threshold 0.5 on the raw score)

Protocol	PR-AUC	ROC-AUC	Precision	Recall	F1	Fraud rows in test
A. Correct protocol (deduplicated, split first)	0.810	0.929	0.958	0.716	0.819	95
B. Oversample fraud before splitting (×20)	1.000	1.000	0.998	1.000	0.999	1,892
C. Keep exact duplicate rows	0.865	0.957	0.951	0.786	0.860	98

Note. Rows A to C use the same model settings; only the protocol changes.

Oversampling the fraud class twenty-fold before the split gives a PR-AUC of 1.00 and an F1 of 1.00, because copies of each training fraud sit in the test partition. The correct protocol gives 0.810 and 0.819. Keeping the 1,081 duplicate rows is a smaller mistake but still raises PR-AUC from 0.810 to 0.865, since a duplicate of a training row often lands in the test partition. Choosing the threshold on the test partition lifts F1 from 0.819 to 0.852. The same model, data and code give a score anywhere from 0.81 to 1.00 depending on the evaluation, the pattern that Kapoor and Narayanan [2] and Hayat and Magnier [3] warn about.

Table 13: Web addresses: logistic regression with different feature sets (domain-grouped test partition, threshold 0.5)

Feature set	Features	PR-AUC	ROC-AUC	F1	Recall
All 31 URL-text features	31	0.999	0.998	0.997	0.994
Without protocol and prefix features (is_https, has_port, has_www)	28	0.974	0.978	0.892	0.846
Host-name features only	10	0.869	0.852	0.784	0.698
Path and query features only	8	0.770	0.800	0.750	0.600
URL length only	1	0.755	0.728	0.638	0.584
is_https only	1	0.713	0.750	0.667	0.501
has_www only	1	0.763	0.794	0.740	0.588
is_https + URL length	2	0.952	0.948	0.889	0.842
Three collection cues (is_https, has_www, path_depth)	3	0.996	0.996	0.996	0.993

The web-address score depends on how the addresses are written. The secure-protocol flag alone gives a PR-AUC of 0.713 (precision 1.000, recall 0.501: every legitimate address uses the protocol, so any address that does not is phishing), the www flag alone gives 0.763, and protocol plus address length gives 0.952. Three formatting cues (protocol, www and path depth) give 0.996, almost the

0.999 of all 31 features. The host-name features alone give 0.869.

Table 14: Share of held-out addresses flagged after one cosmetic change

Change applied to the held-out addresses	Original model (31 features)	Host-name model (10 features)
Legitimate addresses, as released	0.1%	0.8%
Legitimate: remove the www prefix	100.0%	0.8%
Legitimate: use http	100.0%	0.8%
Legitimate: add a trailing slash	100.0%	0.8%
Legitimate: add a path	100.0%	0.8%
Phishing addresses, as released	99.5%	52.3%
Phishing: add a www prefix	98.7%	52.3%
Phishing: use https	66.8%	52.3%
Phishing: drop a lone trailing slash	99.2%	52.3%
Phishing: rewrite as https://www.host	29.6%	52.3%

Note. 27,166 legitimate and 20,118 phishing addresses. The host-name model ignores the scheme, www, path and trailing slash by construction, so only changes that alter the host can move its output.

Table 14 and Fig 13(b) change one cue at a time. The original model flags 0.1% of the legitimate test addresses as released, but every one of them (100%) after any single change: removing www, switching to http, adding a trailing slash or adding a path. It detects 99.5% of phishing addresses as released and 29.6% after they are rewritten as https://www.host. Restricted to addresses that already match the legitimate format, its PR-AUC falls to 0.278 (precision 0.619, recall 0.271) on the 144 phishing addresses of the test partition.

Spot checks on a fixed list of 12 addresses agree. The original model flagged 4 of 5 ordinary addresses (google.com, a Gmail path, a GitHub repository page and example.co.zm) and passed only www.example.co.zm. It flagged all 3 phishing addresses taken from the data, but passed 2 of those 3 hosts once they were rewritten as https://www.host. For sparkasse-auth.com the score was 1.00 as released and 0.004 after the rewrite, against a threshold of 0.27. The original model therefore flags ordinary links and passes phishing addresses that are dressed in the dataset's legitimate format.

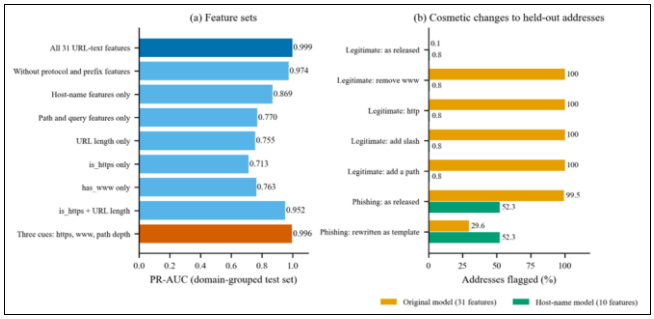


Fig 13: Feature-set ablation (a) and the effect of cosmetic changes on the original and host-name models (b), web-address data

Table 15: Original and host-name address models on the same held-out addresses

Model	PR-AUC [95% CI]	Precision	Recall	False-alarm rate	Passes robustness gate
Original logistic regression (31 features)	0.999 [0.998, 0.999]	0.999	0.995	0.09%	no (0 of 5 checks)
Host-name logistic regression (10 features)	0.823 [0.820, 0.827]	0.976	0.433	0.80%	yes (5 of 5 checks)
Host-name gradient boosting (10 features)	0.863 [0.859, 0.866]	0.979	0.523	0.82%	yes (5 of 5 checks)

Note. The host-name models use a threshold that allows at most 1% false alarms on out-of-fold training predictions; the original models use the F2 threshold.

The host-name model that replaced it ignores those cues, and the robustness gate confirms it: all five checks pass, whereas the original model's flagged share of legitimate addresses rises by at least 99.9 percentage points under each change. That stability costs accuracy: the gradient-boosting host-name model reaches a PR-AUC of 0.863 (95% interval 0.859 to 0.866), with precision 0.979, recall 0.523 and a false-alarm rate of 0.82%. On the spot-check list it flagged 0 of the 5 ordinary addresses and 1 of the 3 phishing addresses from the data; sparkasse-auth.com scored 0.29, below the threshold of 0.83. The legitimate addresses in the benchmark belong to popular sites with short names, so a small business with a long, hyphenated name can be flagged. An invented address of that kind (my-small-grocery-lusaka-fresh-produce.com) scored 0.93. We therefore keep the model experimental, and the ScamWatch checker does not use it.

Table 16 places the original result beside published results on the same dataset and the same task. The comparison is indicative only: our split keeps domains apart and our features come from address text alone, whereas the preprint does not state its split in the abstract [24]. Given the findings above, scores near 99% on this dataset say more about how the legitimate addresses were written than about detecting phishing, and we do not claim that either result transfers to real links.

Table 16: Web-address detection: this study and published studies

Study	Data and features	Accuracy	Precision	Recall	F1	ROC-AUC
This study, original logistic regression	PhiUSIIL; 31 address-text features; domain-grouped split	99.73%	99.88%	99.48%	99.68%	99.85%
This study, host-name gradient boosting	PhiUSIIL; 10 host-name features; domain-grouped split; 1% false-alarm threshold	79.23%	97.92%	52.30%	68.18%	85.57%
Emmanuel and Oghie [24]	PhiUSIIL; 16 structure features; split not stated in the abstract	99.24%	98.74%	99.95%	99.34%	99.65%
Sahingoz *et al.* [13]	Authors' own URL data; language-based features	97.98%	not quoted	not quoted	not quoted	not quoted

Note. Published figures are copied from the cited abstracts.

4.10 System verification and performance

Table 17: Verification and performance of the implementation

Item	Result
Backend automated tests (pytest)	564 passed on Windows with Python 3.14.0; the suite also passes on Linux with Python 3.12 in continuous integration
Frontend unit tests (vitest) and browser tests (Playwright)	236 and 25, all passing
Continuous integration	backend: success, frontend: success, backend-postgres: success, e2e: success (run 5, commit d46f67c)
Route security test	Every API operation refuses anonymous callers unless it is on a reviewed list of 8 public operations; every staff operation refuses a customer token
API operations; database tables	75; 24
Checkout request (POST /orders), in-process, SQLite	median 125 ms, 95th percentile 234 ms (150 orders)
Rule engine computation alone	median 5.0 ms, 95th percentile 11.5 ms
Card model scoring (one transaction)	median 26.4 ms, 95th percentile 42.4 ms
Address model scoring (one address)	median 4.3 ms, 95th percentile 6.9 ms
Training time, card data (one run each)	logistic regression 28 s, gradient boosting 64 s, random forest 379 s, including cross-validation, on 12 logical processors

Note. Latencies are indicative: single thread, no network, local database.

The tests cover the behaviours this paper depends on: idempotent checkout, unsupported and invalid model input, the alert policy for experimental and validated models, append-only outcomes, label export that ignores unverified outcomes, refusal to train without enough labels, the address-model robustness gate, and access separation between customers and staff. The route-security test addresses broken function-level authorisation, one of the risks in the OWASP API Security Top 10 [29]; it is not a penetration test. Running continuous integration on a newer FastAPI release also exposed a flaw in that test: it listed routes through an internal structure that the release had changed, and found none. The test now reads the routes from the OpenAPI schema and asserts a minimum route count.

5. Discussion

- *RQ1.* A store can assess orders with twelve fields it already records, and the rule engine can explain every score in words.
- *RQ2.* A random forest is the best of three standard classifiers on the card data (PR-AUC 0.800), but the estimate is uncertain: the interval runs from 0.713 to 0.876, and five random splits span 0.799 to 0.858.
- *RQ3.* Careless evaluation can turn the same model into a near-perfect one, and the web-address data reward formatting cues that real links will not share.
- *RQ4.* The public datasets cannot support a grocery model because they lack the store's fields, so the store records an explicit refusal and collects verified outcomes instead.
- *RQ5.* Class-weighted models need calibration before their scores can be read as probabilities, and the card model relies on a handful of anonymised components.

We trained a web-address model, validated it on held-out domains and obtained a PR-AUC of 0.999. Only a perturbation test, which no standard metric requires, showed that it had learned how the legitimate list was written. A held-out score cannot reveal a flaw that the training and test partitions share. The remedy we adopted is a test that edits one cue at a time, applied as a gate before promotion, together with the rule that no address-model verdict reaches a user while it is experimental.

The framework avoids claims the data cannot support. A model that has not seen a field is not asked about it, an order that no one has reviewed is not labelled legitimate, and a model with too little evidence is kept out of the alert path. The price is some convenience, because the store starts with rules and not with a learned model.

The inflation from oversampling before the split matches the warning that leakage can produce near-perfect recall [3], and the web-address data show the kind of cue that shortcut learning predicts [9]. Our random forest beat logistic regression and gradient boosting, in line with the strength of tree ensembles on tabular data [16], but the same ordering is not guaranteed on another dataset, and the PR-AUC and ROC-AUC rankings disagree here. The practical implication for a retailer is to report precision-recall results with intervals and to ask any vendor which protocol produced a quoted score.

The experiments have limits of their own. The card data are from 2013 and hold 95 fraud rows in the test partition, so precision and recall are imprecise. The five repeated splits reuse the same data and are not independent. The host-name model ranks address texts and says little about a link to a compromised legitimate site. The rule engine's points were set by judgement, not fitted, and its performance on real fraud is unknown. Section 7 discusses the remaining limits.

6. Real-world applications

The framework suits any merchant, including an online supermarket such as Shoprite, that accepts mobile money or card online and has no fraud history to learn from. The rule engine gives a new store a documented, adjustable policy from its first order. The schema guard and the verified-outcome log let it collect labels at the same time, and the promotion gate says when a trained model has earned a place in the alert path.

Payment aggregators and mobile money providers could use the same pattern for merchants that are too small to receive a bespoke model: a rule baseline, an explicit unsupported status, and a shadow-mode comparison. ScamWatch, the address checker in the application, is a consumer-facing use. It combines offline address heuristics with a Safe Browsing lookup when a key is configured; we have not tested that lookup live. It only ever raises suspicion, and it tells the user that a result of no known threat does not mean a website is safe. The trained address model is not part of its verdicts. The audit trail of assessments and outcomes also gives a regulator or auditor something concrete to inspect.

The evaluation protocol in Section 3 does not depend on the store's fields, so other teams can apply it to their own fraud or phishing data, in banking, cybersecurity or public-sector payment services.

7. Challenges and Limitations

- *Dataset limitations.* The card data are anonymised, two days long and from 2013. PaySim and MoMTSim are simulations. In the phishing dataset every legitimate address is a bare, secure, www-prefixed site root, so it cannot support a deployable address model; that needs legitimate addresses sampled like real traffic, with paths, plain http and small sites. No public dataset carries a grocery order's fields, and the store has no verified fraud labels yet, so no claim is made about fraud detection on store orders.
- *Computational complexity.* All models trained in minutes on one ordinary computer (random forest 6 minutes including cross-validation), and scoring takes milliseconds, so complexity is not a barrier for a small store.
- *Privacy and security.* The schema holds no names, contact details, address text or card data; the application does not store card numbers, which stay with the payment provider. Zambia's Data Protection Act regulates the collection, use, storage and transfer of personal data [30], and a deployment would have to meet its duties as a data controller. We did not assess compliance in this study.
- *Ethical concerns.* A false alarm costs a customer time or a cancelled order. The system therefore never blocks an order or a payment; it asks a person to look. We did not test whether flag rates differ between groups of customers, and behavioural fields such as order timing and address reuse could correlate with such groups.
- *Scalability.* The service runs as one process with an in-memory rate limiter and background jobs inside the process. This suits a small store; a larger one would need shared rate limiting and a job queue.
- *Label quality.* Reviewers can verify only what evidence allows, so some fraud will be labelled legitimate or undetermined, and some verified labels will come from the orders the rules flagged. The audit sample limits but does not remove this selection effect [8].

8. Future Work

- Collect verified outcomes for several months, including the weekly audit sample, and train the first store model in shadow mode.
- Compare that model with the rule engine on the same held-out customers, as the promotion gate requires, and publish the comparison with intervals.
- Build an address benchmark whose legitimate half is sampled like real traffic, retrain under the robustness gate, and test on links reported by users.
- Monitor drift in order fields and in model scores, and retrain on a schedule with the same gate.
- Add mobile-money-specific rules and test them on simulated data from PaySim and MoMTSim, clearly labelled as simulation.
- Study whether merchants could pool signals without sharing orders, for example through federated learning, and measure analyst workload with the alert policy.

9. Conclusion

This study set out to build and evaluate a smart, auditable fraud detection system for Shoprite online payments. The framework scores orders with a documented rule engine, refuses to score inputs a model never saw, and records verified outcomes for later training. Under a leakage-controlled protocol, the Random Forest selected for the system detected card fraud with a PR-AUC of 0.800 (95% interval 0.713 to 0.876), and evaluation shortcuts moved the same model's score between 0.81 and 1.00. A web-address model that scored 0.999 had learned the formatting of the legitimate list, which a perturbation test exposed; a robustness gate now blocks such models, and the host-name model that replaced it reaches a PR-AUC of 0.863 and is kept experimental. Ten scripted checkouts showed the rule engine behaving as specified, and automated tests and continuous integration support the implementation.

The next step is empirical: collect verified grocery outcomes and test whether a model can beat the rules on held-out customers. Until then, the store's answer to the question "is this order fraud?" is a score with reasons for a person to review, and an explicit statement when no model can say more.

10. Acknowledgement

The ULB Machine Learning Group and Worldline collected and released the card data, which we obtained through OpenML. Prasad and Chandra released the phishing data under a CC BY 4.0 licence, which we obtained from the UCI repository. The study used open-source software, including scikit-learn, FastAPI, React and PostgreSQL. An AI assistant (Anthropic's Claude) helped to write the software, the analysis scripts and the text. The experiments were run with the scripts in the repository, every number that reports our own results is generated from the saved result files, and the author has reviewed the work and is responsible for it.

11. Declarations

Conflict of interest: the author declares no conflict of interest. Funding: no external funding was received. Data and code availability: the datasets are public (sources and licences are given in Section 3.2). The software, tests, experiment scripts and saved results are available at https://github.com/lemmymununga0-eng/Shopingrite (release tag paper-v1). Ethics: no human participants or personal data were involved; the customers in the scripted checkouts are synthetic. Relationship to Shoprite: Shoprite is the case context only; no Shoprite data were used, and the study is not affiliated with or endorsed by Shoprite.

12. References

1. GSMA. The state of the industry report on mobile money 2025. GSM Association, 2025. Retrieved from: https://www.gsma.com/sotir/wp-content/uploads/2025/04/The-State-of-the-Industry-Report-2025_English.pdf
2. Kapoor S, Narayanan A. Leakage and the reproducibility crisis in machine-learning-based science. Patterns. 2023; 4(9):100804. Doi: https://doi.org/10.1016/j.patter.2023.100804
3. Hayat K, Magnier B. Data leakage and deceptive performance: A critical examination of credit card fraud detection methodologies. Mathematics. 2025; 13(16):2563. Doi: https://doi.org/10.3390/math13162563
4. Mutemi A, Bacao F. E-commerce fraud detection based on machine learning techniques: Systematic literature review. Big Data Mining and Analytics. 2024;

7(2):419-444. Doi: https://doi.org/10.26599/BDMA.2023.9020023
5. Lopez-Rojas EA, Elmir A, Axelsson S. PaySim: A financial mobile money simulator for fraud detection. In: Proceedings of the 28th European Modeling and Simulation Symposium (EMSS 2016), 2016. Retrieved from: https://msc-les.org/proceedings/emss/emss2016/emss2016_249.html
6. Azamuke D, Katarahweire M, Bainomugisha E. MoMTSim: A multi-agent-based simulation platform calibrated for mobile money transactions. IEEE Access. 2024; 12:120226-120238. Doi: https://doi.org/10.1109/ACCESS.2024.3439012
7. Dal Pozzolo A, Boracchi G, Caelen O, Alippi C, Bontempi G. Credit card fraud detection: A realistic modeling and a novel learning strategy. IEEE Transactions on Neural Networks and Learning Systems. 2018; 29(8):3784-3797. Doi: https://doi.org/10.1109/TNNLS.2017.2736643
8. Lakkaraju H, Kleinberg J, Leskovec J, Ludwig J, Mullainathan S. The selective labels problem: Evaluating algorithmic predictions in the presence of unobservables. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2017, 275-284. Doi: https://doi.org/10.1145/3097983.3098066
9. Geirhos R, Jacobsen JH, Michaelis C, Zemel R, Brendel W, Bethge M, *et al*. Shortcut learning in deep neural networks. Nature Machine Intelligence. 2020; 2(11):665-673. Doi: https://doi.org/10.1038/s42256-020-00257-z
10. Hilal W, Gadsden SA, Yawney J. Financial fraud: A review of anomaly detection techniques and recent advances. Expert Systems with Applications. 2022; 193:116429. Doi: https://doi.org/10.1016/j.eswa.2021.116429
11. Carcillo F, Le Borgne YA, Caelen O, Kessaci Y, Oblé F, Bontempi G. Combining unsupervised and supervised learning in credit card fraud detection. Information Sciences. 2021; 557:317-331. Doi: https://doi.org/10.1016/j.ins.2019.05.042
12. Mutemi A, Bacao F. A numeric-based machine learning design for detecting organized retail fraud in digital marketplaces. Scientific Reports. 2023; 13(1):12499. Doi: https://doi.org/10.1038/s41598-023-38304-5
13. Sahingoz OK, Buber E, Demir O, Diri B. Machine learning based phishing detection from URLs. Expert Systems with Applications. 2019; 117:345-357. Doi: https://doi.org/10.1016/j.eswa.2018.09.029
14. Safi A, Singh S. A systematic literature review on phishing website detection techniques. Journal of King Saud University - Computer and Information Sciences. 2023; 35(2):590-611. Doi: https://doi.org/10.1016/j.jksuci.2023.01.004
15. Prasad A, Chandra S. PhiUSIIL: A diverse security profile empowered phishing URL detection framework based on similarity index and incremental learning. Computers & Security. 2024; 136:103545. Doi: https://doi.org/10.1016/j.cose.2023.103545
16. Grinsztajn L, Oyallon E, Varoquaux G. Why do tree-based models still outperform deep learning on typical tabular data? Advances in Neural Information Processing Systems. 2022; 35 (Datasets and Benchmarks Track).
17. Breiman L. Random forests. Machine Learning. 2001; 45(1):5-32. Doi: https://doi.org/10.1023/A:1010933404324
18. Elor Y, Averbuch-Elor H. To SMOTE, or not to SMOTE? arXiv preprint, 2022. arXiv:2201.08528. Retrieved from: https://arxiv.org/abs/2201.08528
19. Dal Pozzolo A, Caelen O, Johnson RA, Bontempi G. Calibrating probability with undersampling for unbalanced classification. In: 2015 IEEE Symposium Series on Computational Intelligence. IEEE, 2015, 159-166. Doi: https://doi.org/10.1109/SSCI.2015.33
20. Niculescu-Mizil A, Caruana R. Predicting good probabilities with supervised learning. In: Proceedings of the 22nd International Conference on Machine Learning. ACM, 2005, 625-632. Doi: https://doi.org/10.1145/1102351.1102430
21. Saito T, Rehmsmeier M. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. PLoS One. 2015; 10(3):e0118432. Doi: https://doi.org/10.1371/journal.pone.0118432
22. Chicco D, Jurman G. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. BMC Genomics. 2020; 21:6. Doi: https://doi.org/10.1186/s12864-019-6413-7
23. Kaufman S, Rosset S, Perlich C, Stitelman O. Leakage in data mining: Formulation, detection, and avoidance. ACM Transactions on Knowledge Discovery from Data. 2012; 6(4):15. Doi: https://doi.org/10.1145/2382577.2382579
24. Emmanuel UU, Oghie GF. A lightweight hybrid MLP-based framework for real-time phishing URL detection using structural URL features. arXiv preprint, 2026. arXiv:2606.00889. Retrieved from: https://arxiv.org/abs/2606.00889
25. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, *et al*. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research. 2011; 12:2825-2830.
26. Efron B. Bootstrap methods: Another look at the jackknife. The Annals of Statistics. 1979; 7(1):1-26. Doi: https://doi.org/10.1214/aos/1176344552
27. Boyd K, Eng KH, Page CD. Area under the precision-recall curve: Point estimates and confidence intervals. In: Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2013). Springer, 2013, 451-466. Doi: https://doi.org/10.1007/978-3-642-40994-3_29
28. Dietterich TG. Approximate statistical tests for comparing supervised classification learning algorithms. Neural Computation. 1998; 10(7):1895-1923. Doi: https://doi.org/10.1162/089976698300017197
29. OWASP Foundation. OWASP API security top 10: 2023, 2023. Retrieved from: https://owasp.org/API-Security/editions/2023/en/0x00-header/
30. Republic of Zambia. The Data Protection Act, 2021 (Act No. 3 of 2021), 2021. Retrieved from: https://zambialii.org/akn/zm/act/2021/3/eng@2021-03-24