

A Lightweight and Explainable Machine Learning Framework for SMS Spam Detection with Low False-Positive Rates

Abstract—SMS filtering must identify abusive messages without hiding legitimate authentication, banking, delivery, or service notifications. This study builds a transparent classical benchmark on the 5,572-message spam.csv derivative of the SMS Spam Collection. The recorded protocol uses an unstratified 80/20 split (random state 42), lowercase and letter-only normalization, English stop-word removal, WordNet lemmatization, and a training-fitted TF-IDF vocabulary capped at 3,000 features. Multinomial Naive Bayes, logistic regression, decision tree, k-nearest neighbors, linear support vector machine (SVM), and random forest are compared using accuracy, precision, recall, F1-score, false-positive rate, and explicit error counts, mostly at their library-default settings rather than a tuned configuration. On 1,115 test messages, Multinomial Naive Bayes records no false positives but 81.33% spam recall. Linear SVM gives the strongest measured balance: 98.12% accuracy, 97.78% precision, 88.00% recall, 92.63% F1-score, and three false positives among 965 legitimate messages (0.31%). LIME and linear coefficients provide complementary local and global lexical evidence. A data audit also finds 116 exact texts shared across the recorded training and test partitions, so the results are presented as a benchmark operating point rather than an unbiased deployment estimate. The work contributes a readable, false-positive-centered comparison and an explicit account of dataset, explanation, and validation limits; it does not claim dedicated smishing detection or measured on-device efficiency.

Index Terms—SMS spam detection, TF-IDF, false-positive rate, linear SVM, LIME, explainable artificial intelligence, corpus leakage.

1 INTRODUCTION

Short Message Service (SMS) still carries one-time passwords, account alerts, delivery notices, public-service information, and ordinary person-to-person text. That same direct reach is exactly what makes it a cheap channel for bulk advertising, scams, and social engineering. Content-based SMS filtering became its own research problem once labeled short-message corpora and early benchmark classifiers appeared [1] to [3]. Later comparative work showed that the apparent best model is really a function of corpus composition, representation, evaluation design, and the relative cost of an error [4], [5] not the classifier alone.

Accuracy by itself is not the objective. A false negative leaves an unwanted or harmful message visible; a false positive can suppress a legitimate banking alert or authentication code, and that failure is more expensive in almost any real deployment. In an imbalanced corpus dominated by legitimate traffic, a model can post a high accuracy figure while still missing a large share of spam. A conference-ready evaluation should therefore report precision, recall, F1-score, false-positive rate (FPR), and the underlying false-positive (FP) and false-negative (FN) counts rather than one aggregate percentage.

Spam and smishing are not the same thing. Spam is simply an unsolicited-message label; smishing normally denotes a phishing attempt delivered by SMS to steal information, install malware, or induce a harmful action. Security-oriented systems may draw on mobile context, URL behavior, or threat-specific labels [6], [7]. The benchmark used here contains only ham and spam, so it can support binary spam filtering but cannot establish a smishing detection rate or separate advertising from credential theft. That boundary is kept explicit throughout the study.

The literature now ranges from hybrid and rapidly selected classical models [8], [9], through multi-type feature fusion [10], to transformer and BERT-based systems [11] to [14]. Recent work adds explanation, privacy-preserving learning, adversarial testing, large language models, fuzzy reasoning, and explicit misclassification analysis [15] to [20]. These advances matter, but they do not remove the need for transparent, lower-complexity baselines: reported scores are not directly rankable when datasets, augmentation, split procedures, labels, or thresholds differ, and attaching an explanation does not by itself prove that the explanation is stable, faithful, or useful.

This paper addresses a narrower empirical gap: a controlled comparison of six common classifiers under the same bounded sparse representation, with false-positive behavior treated as a first-class selection criterion and with local and global lexical evidence

reported at the same operating point. LIME [21] supplies a local surrogate explanation, while direct linear coefficients provide a global view comparable in spirit but not identical in assumptions to additive attribution approaches such as SHAP [22]. The design favors auditability and reproducibility of the representation over novelty of the classifiers, and it takes for granted that spam vocabulary drifts over time and that a static random split cannot test that drift [23].

Three research questions guide the analysis. RQ1: what precision–recall and FP, FN trade-offs arise across the six classifiers under one 3,000-feature TF-IDF representation? RQ2: which model gives the strongest measured balance between spam recall and false-alarm suppression on the recorded test split? RQ3: do the local and global explanations produce plausible lexical evidence, and which conclusions can they not support?

The contribution is fourfold. First, the exact 5,572-row corpus derivative and fixed experimental partition are preserved and documented. Second, a controlled six-model comparison is reported with both rates and counts. Third, local LIME evidence and global SVM weights are joined in a single audit trail. Fourth, exact duplicate overlap and other threats to validity are identified and reported rather than smoothed over.

2 LITERATURE REVIEW

Early SMS filtering carried ideas from email spam research over to much shorter texts. Gómez Hidalgo et al. [2] showed that message content alone could support supervised filtering even in a sparse, heavily abbreviated vocabulary. Cormack et al. [3] examined spam filtering for short messages further, helping establish the field's comparative-experimentation norms. Almeida et al. [1] consolidated several sources into the SMS Spam Collection and compared established learners on it, producing the benchmark that underlies most later work. Delany et al. [4] reviewed methods and data in more depth, reinforcing that a result is inseparable from its corpus and evaluation protocol.

Classical text classifiers stay plausible for SMS because recurring campaign vocabulary creates strong lexical regularities: probabilistic models exploit token likelihoods, regularized linear models combine many weak indicators, margin classifiers tend to do well in high-dimensional sparse spaces, and tree- or instance-based methods add contrasting nonlinear and local biases. Their training and inference paths are also easier to inspect than a deep contextual encoder's. Blanzieri and Bryl's survey [5] is a useful check on that optimism, though: spam is nonstationary, class costs differ, and benchmark choice can dominate the apparent performance gap. All three cautions carry over directly to SMS.

The benchmark's popularity creates a second problem: corpus derivatives are not always identical. Files can differ by encoding, auxiliary columns, duplicate handling, or the two additional ham messages now present in the UCI archive. Random record-level splitting can also place repeated campaign texts on both sides of the boundary, so a classifier meets exact or near-exact language during both training and testing making the hold-out less independent than its size suggests. Reusing the collection for comparability is legitimate, but a careful report should name the file variant, disclose the split procedure, and stop short of treating one random partition as a deployment estimate.

Feature engineering extends the basic bag-of-words pipeline. Hybrid systems combine classifiers or join lexical evidence with message attributes, as in Baaqeel and Zagrouba [8]; Rapacz et al. [9] instead study rapid classifier selection, a reminder that practical filtering has computational as well as predictive constraints. Al-Kabbi et al. [10] integrate multiple feature types through early fusion to capture coverage a single word representation might miss. Character patterns, message length, punctuation, digits, currency symbols, sender information, and URL structure can all carry signal in principle.

Richer features come with trade-offs. Hand-designed attributes can capture obfuscation and contact patterns, but every extraction stage adds assumptions, maintenance work, and often a dependence on metadata that is unavailable or privacy-sensitive. Early fusion can widen coverage while making it harder to tell which feature family produced a gain; aggressive cleaning keeps a pipeline compact but can discard exactly the digits, punctuation, or domain syntax a fraudulent message relies on. This study keeps the recorded letter-only TF-IDF pipeline so its results stay unchanged, and treats the removed signals as a limitation rather than claiming the preprocessing is universally optimal.

Efficiency claims need precise language too. A sparse 3,000-feature linear model is architecturally smaller than a transformer, and its prediction path is simple but that alone does not prove low latency, memory, or energy use on a phone. Such claims need serialized-model size, peak memory, timing under controlled hardware, and an energy measurement protocol. "Lightweight" in this paper describes a bounded representation and a classical inference structure; it is not a completed mobile benchmark.

Smishing research widens the threat model past unsolicited content. S-Detector frames the problem as mobile attack detection [6]. Mishra and Soni [7] combine SMS content analysis with URL behavior, since phishing often depends on where a link leads rather than on the words around it. This matters because a harmless advertisement and a credential-stealing message can share the same urgent or promotional vocabulary while posing very different risks.

That richer threat model rules out a simple comparison with the SMS Spam Collection. A binary spam label does not confirm malicious intent, and plenty of legitimate messages contain links or urgent language. A security claim would need URL reputation, redirects, domain age, landing-page content, sender identity, and user interaction. So the present experiment can offer a first-stage content filter or a benchmark baseline, but it should not be called a validated smishing defense. A future dataset would need explicit categories benign bulk, advertising, scam, credential phishing, malware delivery and annotation rules that separate content category from security impact.

Transformer-based work goes after information unigram TF-IDF cannot encode. Liu et al. [11] propose a Spam Transformer, Oswald et al. [12] use BERT embeddings for intention-driven filtering, and Ghourabi and Alohalay [13] combine transformer-based

embeddings with ensemble learning all following the wider success of bidirectional transformer pretraining represented by BERT [14]. Contextual encoders can model word order, polysemy, and longer dependencies, which can help when a campaign avoids familiar keywords.

Higher representational capacity isn't a free upgrade, though. Fine-tuning costs more compute and adds sensitivity to initialization, optimization, and checkpoint choice; prompted or embedding-based approaches can add service latency, privacy exposure, or version dependence. Deep models can also inherit corpus leakage and split bias a more sophisticated learner does not fix a nonindependent test set. So a claim that one architecture beats another only holds up when the same examples, split constraints, preprocessing boundaries, and decision criteria are applied to both.

ExplainableDetector pairs fine-tuned transformer language models with explanation analysis and reports very high performance under its augmentation-based protocol [15]. That's a real demonstration that contextual performance and XAI can coexist, but it doesn't make a bounded classical baseline obsolete: augmentation, architecture, and validation choices change the experiment, and choosing between systems also needs error counts, resource measurements, calibration, and robustness evidence. The useful comparison is multidimensional, not a single ranking by accuracy.

Post-hoc explainability asks why a classifier produced a given label. LIME perturbs one instance and fits a simpler local surrogate around it [21]; SHAP assigns additive feature contributions through a game-theoretic framework [22]. Both can reveal whether a detector leans on plausible content, but they answer different formal questions and can shift with background data, perturbation choices, tokenization, or random sampling. A visually convincing explanation is not automatically faithful, stable, or causal local surrogate fidelity and repeated-seed stability are things that should be measured when explanations are used operationally.

Recent work treats explanation as more than a presentation layer. SpamDam assembles more than 76,000 recent spam messages from social platforms and studies centralized and federated learning alongside adversarial resistance [16], which exposes three weaknesses common to smaller benchmark studies: dated vocabulary, privacy constraints on data collection, and vulnerability to deliberate manipulation. SpaLLM-Guard evaluates open and commercial LLMs under zero-shot, few-shot, fine-tuning, adversarial, and concept-drift settings, and reports that fine-tuning beats convenient zero-shot use [17] relevant to adaptive filtering, though its computing and governance needs differ substantially from a sparse local model.

Fuzzy and hybrid studies look for middle ground between learned representations and human-interpretable uncertainty. Jasim et al. combine fuzzy logic with a bidirectional LSTM [18], and Birthriya et al. combine fuzzy reasoning with TF-IDF-oriented feature engineering for smishing detection [19]. These designs can make linguistic uncertainty explicit, but each added component still needs its own ablation to isolate what it actually contributes. X-MAP takes a different angle: it uses feature attributions and topic profiles to trace semantic patterns behind spam and phishing misclassifications and to support an error-repair layer [20], shifting explainability from describing a prediction toward diagnosing systematic failure.

Temporal robustness is a related concern. Delany et al. [23] study concept drift in spam filtering, where vocabulary and campaign strategy change over time. Randomly mixing an older corpus into training and testing cannot simulate that process. A

practical evaluation should split by time or campaign, measure how performance decays, and specify an update policy; explanations can help monitor drift by showing which tokens gain or lose influence, but they don't substitute for temporal validation.

A comparison table only means something when the underlying experimental units are comparable. Studies may use the original collection, a CSV derivative, a deduplicated version, an augmented training set, a translated corpus, or a newly collected threat dataset and may report a random hold-out, a cross-validation average, a temporal test, or an adversarial test. Even when the metric shares a name, these designs estimate different quantities: accuracy on a random record split cannot be ranked fairly against accuracy on a future-campaign split, and a spam/ham result is not a smishing result. Foundational comparative work already made this point about methods and data needing to be read together [1], [4].

Error reporting is a second comparability requirement. On a ham-dominated test set, two models can post similar accuracy while imposing very different costs. Precision describes how reliable a positive prediction is, recall describes how much spam gets recovered, and FPR describes the share of legitimate traffic wrongly blocked none of which communicates sample size on its own. Reporting FP and FN counts alongside the denominators keeps the operating point auditable and stops a rounded percentage from hiding a small or unstable number of events. A zero count should be read as zero observed errors, not as proof the population rate is zero.

Independence has to be checked separately from feature-fitting leakage. Fitting a vectorizer only on training text is necessary but not sufficient repeated campaign messages can still cross the boundary. Grouping exact and near-duplicate templates before splitting tests generalization to unseen campaigns more directly, and temporal separation asks a further question: does the model survive changing vocabulary and attacker behavior, the concern at

the center of concept-drift research [23]? These controls tend to lower the reported score, but they raise the credibility of the estimate.

Reproducibility, finally, is more than a fixed random seed. A complete record should identify the exact data file or checksum, publish split indices, preserve preprocessing resources, name library versions, state every nondefault parameter, and keep the raw predictions the metrics were computed from. Probability calibration and stochastic ensembles can shift across software releases even when the visible code looks unchanged, so a result can be computationally repeatable in one archived environment yet lack external validity or be externally relevant yet hard to reproduce. Both dimensions deserve their own report rather than being folded into the single word "reproducible."

Taken as a whole, the literature shows SMS filtering being approached through classical text models, engineered features, contextual encoders, threat-specific signals, hybrid uncertainty mechanisms, and post-hoc explanations. Four gaps stand out. First, many reported scores are hard to interpret without the exact corpus derivative and split. Second, accuracy still gets more attention than the operational FP–FN balance. Third, explanation studies tend to focus on either individual predictions or model-level patterns rather than presenting both with clear scope limits. Fourth, architectural efficiency sometimes substitutes for a measured deployment cost.

This paper does not claim to close every one of those gaps or introduce a new classifier. Its contribution is a transparent baseline that holds representation and test set constant across six common learners, selects an operating point from explicit false-alarm and miss counts, and pairs local and global lexical evidence while also auditing the file and partition for duplicate overlap. Table 1 positions this contribution against representative studies and explains why their reported performance figures are not simply copied into one ranking table.

Table 1. Critical comparison of representative SMS spam and smishing research

Study	Data / task	Method	Main contribution	Unresolved issue for this study
Gómez Hidalgo et al. [2] (2006); Cormack et al. [3] (2007)	Early SMS spam/ham corpora	Content features and classical filters	Established content-based filtering and comparative testing	Small, dated corpora; no modern XAI or deployment audit
Almeida et al. [1] (2011)	Public English SMS collection	Bag-of-words and classical learners	Released the widely reused 5.5k-message benchmark	Derivative files, duplicates, and random splits reduce comparability
Delany et al. [4] (2012)	SMS datasets and methods	Comparative classical evaluation	Linked method choice, data composition, and error behavior	Pre-dates contextual models and paired local/global XAI
Joo et al. [6] (2017); Mishra and Soni [7] (2020)	Mobile smishing and URL-oriented threats	Security model; text plus URL behavior	Extends filtering from nuisance spam to attack-oriented messages	Threat labels and URL signals differ from text-only spam/ham
Baaqeel and Zagrouba [8] (2020); Rapacz et al. [9] (2021)	SMS spam filtering	Hybrid learning and rapid classifier selection	Retains practical value of efficient classical models	Protocols and operational error costs remain inconsistent
Al-Kabbi et al. [10] (2023)	SMS spam	Multi-type features with early fusion	Uses evidence beyond one lexical representation	Added extraction stages increase maintenance and attribution difficulty
Liu et al. [11] (2021); Oswald et al. [12] (2022)	SMS spam and intention analysis	Transformer and BERT embeddings	Models context and word order beyond sparse unigrams	Compute, privacy, and low-FP behavior need separate measurement
Ghourabi and Alohalay [13] (2023)	SMS spam	Transformer embeddings and ensembles	Combines contextual features with multiple learners	Scores cannot be ranked against different corpora and splits
Uddin et al. [15] (2024)	Benchmark SMS with augmentation	Fine-tuned transformer and XAI	Pairs contextual classification with feature explanations	Augmentation and model capacity prevent a controlled baseline comparison
Li et al. [16] (2024); Salman et al. [17] (2025)	Recent and evolving SMS	Federated detectors and LLMs	Addresses privacy, adversaries, adaptation, and drift	Scale, governance, and evaluation design differ from this benchmark
Jasim et al. [18] (2026); Birthriya et al. [19] (2026)	SMS spam and smishing	Fuzzy-deep and fuzzy/TF-IDF hybrids	Represents uncertainty or interpretable rules	More components obscure which change produced a gain
Zhang et al. [20] (2026)	Spam/phishing error analysis	SHAP, topic profiles, and divergence	Uses explanations to diagnose misclassification patterns	Focuses on post-prediction repair rather than bounded baselines

Note: The table compares task design and evidence, not headline accuracy. Reported scores across studies are not normalized because their corpora, labels, preprocessing, splits, augmentation, and validation procedures differ.

Read together, the reviewed studies point to a layered research agenda: a bounded sparse classifier is a useful auditable reference point; smishing-specific signals are necessary once a security claim is being made; contextual and large language models have to be judged against their resource and governance costs; and explanations should be treated as diagnostic objects rather than decorative plots. The experiment below deliberately addresses only the first of these layers, while leaving clear interfaces to the rest.

3 METHODOLOGY

3.1 End-to-end project workflow

The project turns a raw SMS message into a ham or spam prediction and then produces human-readable evidence for the selected classifier. Figure 1 lays out the full experiment in eight stages: dataset loading, data audit and partitioning, text preprocessing, TF-IDF feature construction, model training, fixed-test prediction, performance evaluation, and explanation. The first five stages build the trained candidate models; the remaining three compare their frozen predictions and examine the model chosen from the recorded results. Test labels are used only to compute evaluation measures they never update the vocabulary, the classifier parameters, or the explanations.

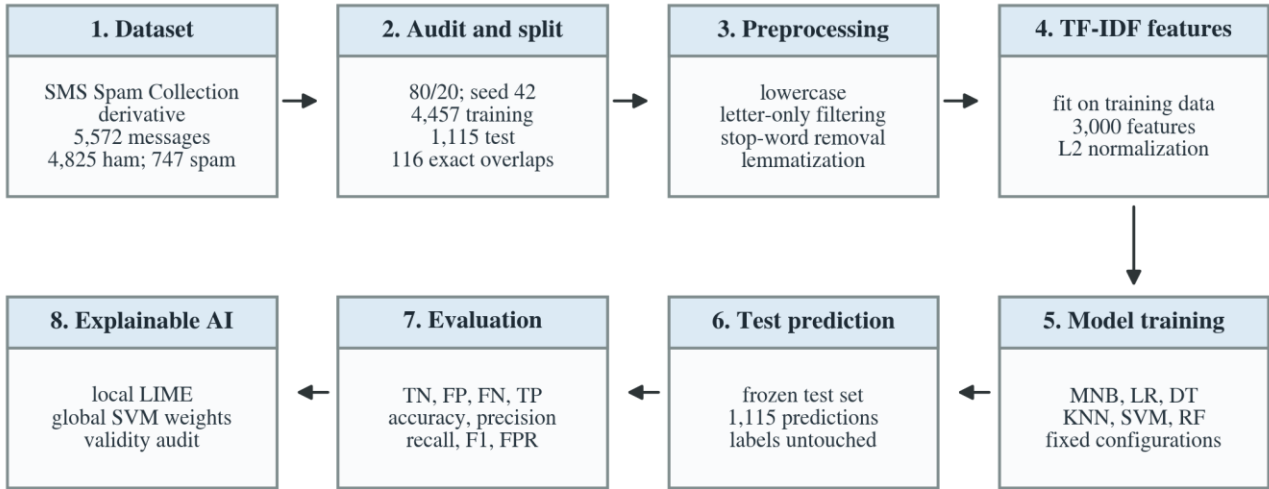
During data preparation, the CSV is reduced to the class label and message text, labels are mapped to ham = 0 and spam = 1, and exact duplicate counts are recorded before the preserved 80/20 random split is applied. The same cleaning function then strips non-alphabetic characters, lowercases each message, removes English stop words, and lemmatizes what's left. A TF-IDF vectorizer is fitted only on the cleaned training messages; its stored vocabulary and inverse-document-frequency values then transform both the training and held-out messages into vectors of 3,000 or fewer components.

Six classifiers receive the same training matrix and labels, and each fitted model predicts the same 1,115 held-out messages, so the

representation and the evaluation population stay constant across all six. Accuracy, precision, recall, F1-score, FPR, FNR, and the FP/FN counts are then computed from those predictions, with particular attention to false positives, since a legitimate authentication or banking message can be costly to hide. Linear SVM is carried into the explanation stage because it posts the highest recorded accuracy, recall, and F1-score while producing only three false positives.

Explainable artificial intelligence (XAI) is applied after prediction, purely to make the model's evidence legible to a reviewer it does not retrain the classifier or change a predicted label. The local view asks why the model treated one particular SMS as spam or ham; the global view asks which vocabulary terms generally push the fitted linear boundary toward spam. LIME supplies the local view and the learned linear SVM coefficients supply the global one; Section 3.7 explains how each is constructed, how to read the sign of each value, and why neither should be read as proof of malicious intent.

In a future deployment, a new SMS would enter at the preprocessing stage, pass through the stored TF-IDF transform, and receive a label from the selected classifier; a local explanation could be generated on review, and the global coefficient list could support periodic model audits. The present work evaluates the classification-and-explanation pipeline offline only it does not claim that a smartphone interface, a URL-analysis layer, or an on-device performance benchmark has already been built.



Training-derived parameters never use test text; duplicate overlap is reported as a validity threat.

Fig. 1. Complete eight-stage methodology generated from the recorded data and model settings. The training path creates the preprocessing rules, TF-IDF representation, and six fitted classifiers. The evaluation path freezes the 1,115-message test set, compares prediction errors, selects the recorded linear SVM operating point, and produces local and global explanation evidence. Exact duplicate overlap is disclosed as a validity limitation.

3.2 Dataset variant and label scope

Experiments use the widely circulated spam.csv derivative of the SMS Spam Collection [1], distributed in a related form by the UCI Machine Learning Repository [24]. The file contains 5,572 English messages: 4,825 ham (86.59%) and 747 spam (13.41%). The current UCI text archive contains two additional ham records, which is why this particular derivative is named rather than just the collection. Auxiliary empty columns are discarded, labels are mapped to $y = 0$ for ham and $y = 1$ for spam, and no attempt is made to infer a separate smishing label.

A file-level audit finds 5,169 unique message texts and 403 duplicate label-text rows, with no identical text carrying

conflicting labels. These figures describe exact duplicates only they do not count near-duplicates with changed numbers, spelling, or punctuation. The audit matters because repeated campaign templates can make a random record-level evaluation look better than it is, even when feature fitting is correctly restricted to the training set.

3.3 Fixed split and leakage boundaries

The recorded experiment uses `train_test_split` with `test_size = 0.20` and `random_state = 42`, without stratification, producing 4,457 training messages (3,860 ham, 597 spam) and 1,115 test messages (965 ham, 150 spam). Table 2 reports this composition, and Table 3 spells out every use of the fixed test set. The TF-IDF vocabulary

is learned only from training messages, and the fitted transform is applied unchanged to the test set, which rules out direct vocabulary fitting on held-out text.

The split is nevertheless not independent at the message level: 116 exact text strings occur in both partitions. This partition is preserved anyway, because the recorded result table and confusion

counts were generated from it quietly changing the split would invalidate those reported values. Instead, the overlap is disclosed and treated as a threat to external validity. A stronger confirmatory experiment would group exact and near-duplicate campaigns before splitting and keep a separate validation set for threshold selection.

Table 2. Dataset and fixed hold-out composition

Partition	Total	Ham	Spam	Spam share
Full 5,572-row derivative	5,572	4,825	747	13.41%
Training partition	4,457	3,860	597	13.39%
Test partition	1,115	965	150	13.45%

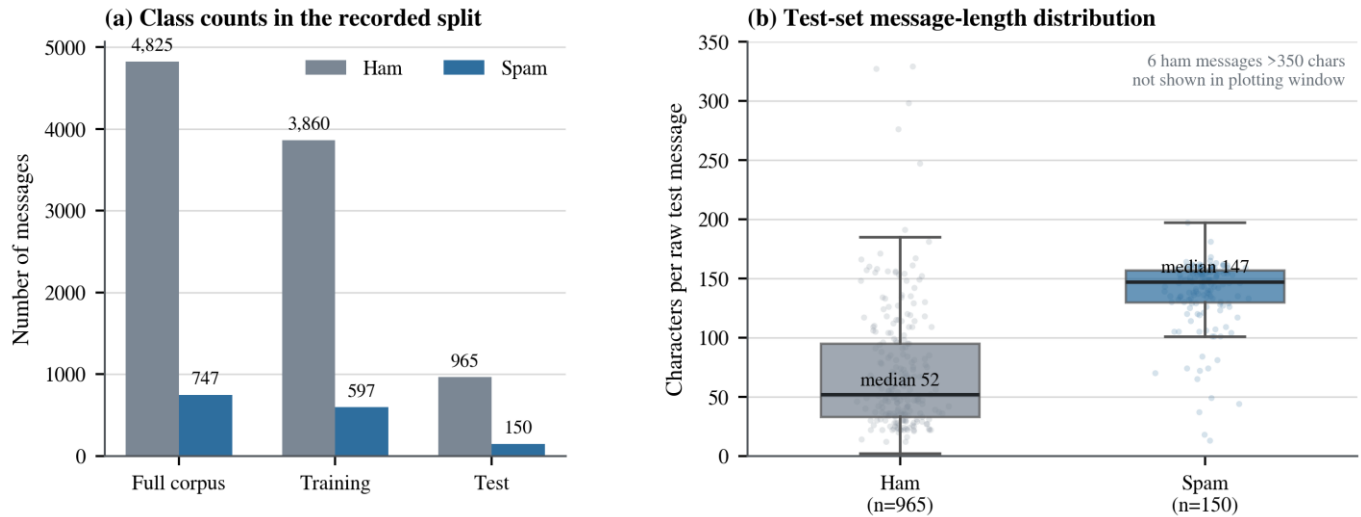


Fig. 2. Data profile generated directly from the 5,572-row corpus and fixed split. Panel (a) gives exact class counts. Panel (b) shows raw character-length observations in the 965 ham and 150 spam test messages; the boxes summarize the interquartile ranges and the overlaid points are a reproducible sample of actual observations. Six ham messages longer than 350 characters are included in the statistics but omitted from the plotting window for readability.

Table 3. Stepwise use of the fixed 1,115-message test set

Step	Test-set operation	Input	Output or safeguard
1	Lock the hold-out	965 ham and 150 spam from the recorded seed-42 split	No test row enters vocabulary fitting or parameter estimation
2	Normalize text	The deterministic lowercase, letter-only, stop-word, and lemma pipeline	The same transformation is applied to training and test messages
3	Create test vectors	Training-fitted vocabulary and IDF statistics	Produces a $1,115 \times 3,000$ sparse matrix without refitting
4	Generate predictions	Each of the six fitted classifiers	One locked spam/ham prediction is recorded for every test message
5	Count outcomes	Reference labels and recorded predictions	TN, FP, FN, and TP are computed before rounded percentages
6	Calculate metrics	The confusion counts in Step 5	Accuracy, precision, recall, F1-score, FPR, and FNR are derived
7	Select an operating point	Recall, F1-score, FP, and FN together	SVM is selected for explanation from the recorded comparison; no model is refitted
8	Explainable AI analysis	LIME, SVM coefficients, and the duplicate audit	Explanations do not change predictions; overlap is reported as a validity threat

3.4 Text preprocessing

Each raw message first has every character outside the English alphabet replaced by a space, then is lowercased and split into individual tokens. Common English stop words are removed and the remaining tokens are lemmatized with WordNet [25] in the same pass. Empty auxiliary columns are never used. This sequence

matches the recorded notebook and is applied identically to both partitions before feature extraction.

The procedure trims case and inflectional variation, but it is not a neutral choice. Stripping every non-letter character deletes telephone numbers, currency amounts, punctuation intensity, and URL syntax all signals that could matter for fraud or campaign detection and English stop words can still carry pragmatic weight

in a message this short. These choices are kept for result consistency and documented in Table 4, so a later ablation can determine which transformations actually help versus quietly hurt.

3.5 Bounded TF-IDF representation

The normalized messages are encoded with scikit-learn's TF-IDF vectorizer [26], called with its library defaults except for a vocabulary capped at the 3,000 highest-ranked training features; inverse document frequency is smoothed and each document vector is L2-normalized by default. One consequence worth stating plainly: the vectorizer's default tokenizer only keeps tokens of two or more word characters, so common single-letter texting shorthand ("u," "i") never becomes a feature. Only training text determines the vocabulary and IDF statistics. For raw term

frequency $tf(t,d)$, n training documents, and document frequency $df(t)$, the smoothed inverse document frequency is

$$idf(t) = \ln \left(\frac{1+n}{1+df(t)} \right) + 1 \quad (1)$$

and the weighted document vector is normalized to unit Euclidean length. For vocabulary V , its component for term t is

$$x_{t,d} = \frac{tf(t,d)idf(t)}{\sqrt{\sum_{j \in V} [tf(j,d)idf(j)]^2}} \quad (2)$$

The 3,000-feature cap bounds representation width and keeps the arithmetic sparse, but on its own it says nothing about serialized size, memory use, latency, or energy claims that would need their own measurement. TF-IDF's information-retrieval basis and the role of term and document frequency are described in [27].

Table 4. Processing controls, rationale, and known risks

Stage	Recorded choice	Scope	Rationale and risk
Label mapping	ham = 0; spam = 1	Entire corpus	Matches recorded experiment; does not create a smishing class
Normalization	Lowercase; replace nonletters	Before vectorization	Reduces sparsity but removes digits, punctuation, currency, and URL syntax
Token filtering	English stop words; WordNet lemma	Each message	Reduces common variants; may erase short-message style cues
Feature fitting	TF-IDF, maximum 3,000 features	Training text only	Bounds width and prevents direct vocabulary fitting on test text
Evaluation	Fixed unstratified 80/20 split, seed 42	One hold-out	Preserves recorded scores but does not quantify split variance
Explainable AI	LIME plus linear coefficients	Selected SVM	Provides local and global associations; neither proves causality

3.6 Classifier formulations

Six classifiers sit on the same TF-IDF input but bring different assumptions. Table 5 records the configurations actually used, and as the table notes most of these are scikit-learn's library defaults rather than the product of a separate hyperparameter search; only the linear kernel, probability output, and the fixed `random_state` values were set deliberately. The equations below state each classifier's decision rule; unlisted behavior follows scikit-learn [26].

3.6.1 Multinomial Naive Bayes

MNB combines the class prior with smoothed conditional feature probabilities. The multinomial model is technically motivated by counts, but scikit-learn accepts nonnegative fractional TF-IDF input without complaint. The predicted class is

$$\hat{y} = \arg \max_{c \in \{0,1\}} \left[\log P(c) + \sum_{j=1}^m x_j \log \theta_{cj} \right] \quad (3)$$

3.6.2 Logistic regression

Logistic regression maps a regularized linear score to a spam probability. At the default 0.5 decision threshold, its rule is

$$z = \mathbf{w}^T \mathbf{x} + b, \quad \sigma(z) = \frac{1}{1 + e^{-z}} \quad (4)$$

$$\hat{y} = \mathbb{I}[\sigma(z) \geq 0.5]$$

3.6.3 Decision tree

The decision tree splits on Gini impurity, choosing the split that most reduces weighted child impurity. For node sample S and child nodes S_k ,

$$G(S) = 1 - \sum_{c \in \{0,1\}} p_c^2 \quad (5)$$

$$\Delta G = G(S) - \sum_k \frac{|S_k|}{|S|} G(S_k)$$

3.6.4 K-nearest neighbors

KNN compares a test vector against stored training vectors by Euclidean distance and returns the majority label among its $k = 5$ nearest neighbors (scikit-learn's default k , kept as-is here):

$$d(\mathbf{x}, \mathbf{x}_i) = \sqrt{\sum_{j=1}^m (x_j - x_{ij})^2} \quad (6)$$

$$\hat{y} = \text{mode}\{y_i : i \in \mathcal{N}_5(\mathbf{x})\}$$

3.6.5 Linear support vector machine

The soft-margin linear SVM trades margin width against hinge-loss slack. At $C = 1.0$, the primal problem is

$$\min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{i=1}^n \xi_i \quad (7)$$

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

Probability output is switched on only to support the recorded LIME workflow, which adds an internal calibration step to the fitted estimator; classification metrics still use the recorded class predictions, and no probability threshold is tuned on the test set.

3.6.6 Random forest

The random forest aggregates $B = 100$ decision trees (again scikit-learn's default forest size) trained through bootstrap sampling and random feature selection. Its output is the majority vote

$$\hat{y} = \text{mode}\{h_b(\mathbf{x})\}_{b=1}^B, \quad B = 100 \quad (8)$$

Package versions were not pinned in the archived experiment record, so an exact future reproduction should preserve the

software environment alongside the data file, split indices, preprocessing resources, and random seed.

Table 5. Classifier configurations preserved from the experimental materials

Classifier	Configuration	Role in comparison
Multinomial Naive Bayes (MNB)	$\alpha = 1.0$ (library default)	Probabilistic sparse-text baseline
Logistic regression (LR)	L2 penalty; max_iter = 1000	Regularized linear decision boundary
Decision tree (DT)	Gini; random_state = 42	Nonlinear, rule-partition baseline
k-nearest neighbors (KNN)	k = 5; Euclidean distance (library defaults)	Local instance-based comparison
Linear support vector machine (SVM)	Linear kernel; C = 1.0 (default); probability = True; random_state = 42	Maximum-margin sparse classifier; selected for XAI
Random forest (RF)	100 trees (library default); random_state = 42	Bagged nonlinear ensemble

3.7 Evaluation and explainable artificial intelligence protocol

Let TP and TN denote correctly classified spam and ham, and FP and FN denote ham flagged as spam and spam predicted as ham. Because spam is only 13.45% of the test set, accuracy is supplemented with precision, recall, F1-score, and FPR precision recall analysis is particularly informative under this kind of class imbalance [28]. The measures are

$$\mathbf{C} = \begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix} \quad (9)$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN} \quad (10)$$

$$F_1 = 2 \frac{Precision \cdot Recall}{Precision + Recall}$$

$$FPR = \frac{FP}{FP + TN}, \quad FNR = \frac{FN}{FN + TP}$$

Rates are reported alongside raw counts so a high precision figure can't quietly hide severe under-detection. Wilson intervals accompany the selected-model results to show the uncertainty around small error counts, though model selection itself still follows the recorded point estimates there are no confidence intervals over resampled train/test splits here.

Explainable artificial intelligence, in this context, means methods that expose the input evidence behind a prediction in a form a person can actually inspect a different question from performance evaluation. Accuracy and recall say how often predictions agree with labels; an explanation says which represented words influenced one decision or the fitted boundary as a whole. That evidence can reveal reliance on sensible campaign terms or on suspicious shortcuts, but it cannot prove the message is malicious, that the prediction is correct, or that the same evidence will still matter on future data.

The local explanation comes from Local Interpretable Model-agnostic Explanations (LIME) [21]. One test message x_0 and the selected SVM prediction are fixed; LIME generates nearby versions of that message by keeping or dropping tokens; the trained classifier scores each perturbed version for spam; perturbations closer to the original get larger locality weights; and a small weighted linear surrogate g is fit to imitate the classifier only in that

neighborhood. With candidate interpretable models G , locality weights π , a fidelity loss L , and a complexity penalty Ω , the objective is

$$g_{x_0}^* = \arg \min_{g \in \mathcal{G}} \mathcal{L}(f, g, \pi_{x_0}) + \Omega(g) \quad (11)$$

A positive LIME weight means the presence of that token pushes the local surrogate toward spam; a negative weight pushes it toward ham, and its magnitude reflects strength within that one neighborhood only it says nothing about the dataset as a whole, since LIME approximates the classifier near a single message rather than opening up the full SVM. Local fidelity is how well g reproduces f on the weighted perturbations; stability is how much the explanation changes when the perturbation sample or random seed changes. The archived experiment keeps one recorded explanation but no repeated-seed fidelity or stability measurement.

The global explanation instead reads the fitted linear SVM directly. For TF-IDF vector $\mathbf{x}$, coefficient vector $\mathbf{w}$, and intercept b , the decision score and each represented token's exact contribution are

$$s(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = b + \sum_{j=1}^m w_j x_j, \quad \phi_j(\mathbf{x}) = w_j x_j \quad (12)$$

A large positive coefficient for term j means that when its TF-IDF feature is present, it generally pushes the SVM score toward spam; a large negative coefficient generally supports ham. Ranking the coefficient vector gives a model-level view, and multiplying a term's coefficient by its TF-IDF value gives that term's contribution to one specific decision. This differs from LIME in using the linear model's own internal parameters rather than a model-agnostic local approximation agreement between the two views can strengthen an audit observation, but neither one establishes a causal link between a word and malicious intent.

4 RESULTS

4.1 Overall classifier comparison

Table 6 reproduces the saved hold-out results and derives FPR from each confusion count. Five of the six classifiers clear 96% accuracy, but their error profiles diverge sharply. MNB and KNN both post zero false positives on the 965 ham messages, yet MNB catches 122 of 150 spam messages while KNN catches only 65 a reminder that 100% precision says nothing about detection strength on its own. KNN's 43.33% recall and 60.47% F1-score make it the weakest operating point here, despite having no observed false alarms.

Table 6. Recorded performance on the 1,115-message hold-out test set

Model	Acc. (%)	Prec. (%)	Rec. (%)	F1 (%)	FP	FN	FPR (%)
MNB	97.49	100.00	81.33	89.71	0	28	0.00
LR	96.05	96.49	73.33	83.33	4	40	0.41
DT	96.86	89.66	86.67	88.14	15	20	1.55
KNN	92.38	100.00	43.33	60.47	0	85	0.00
Linear SVM	98.12	97.78	88.00	92.63	3	18	0.31
RF	97.76	98.45	84.67	91.04	2	23	0.21

The highlighted SVM row is selected for interpretation because it posts the highest recorded recall and F1-score while producing only three false positives. Zero observed FP for MNB and KNN is not, and should not be read as, a population guarantee.

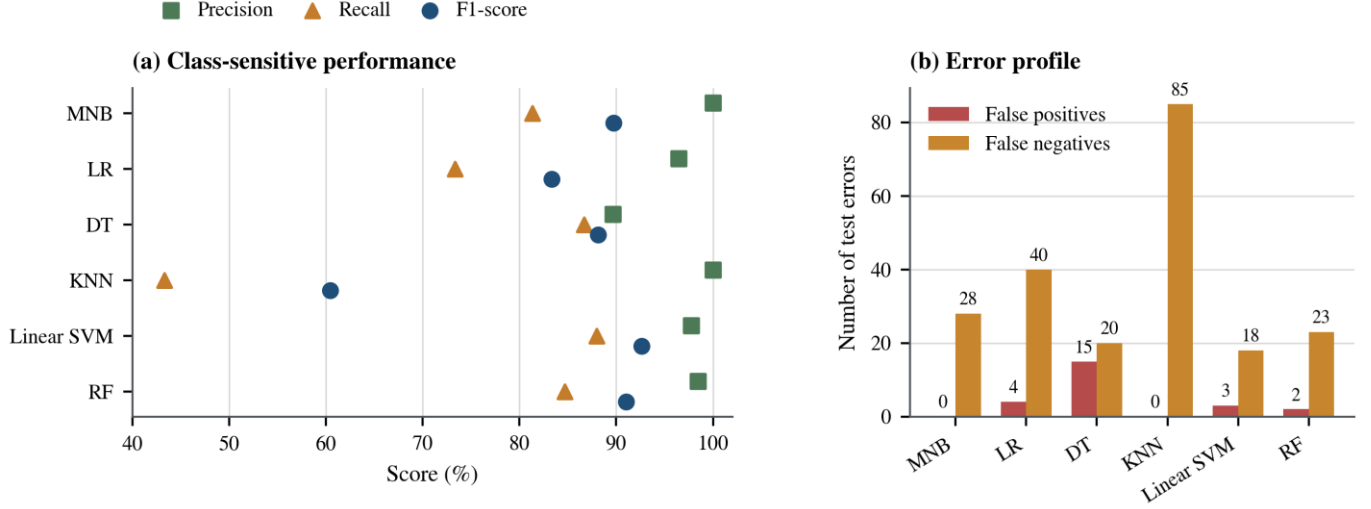


Fig. 3. Recorded class-sensitive scores and error counts under the same hold-out split. Panel (a) shows why accuracy alone is insufficient; panel (b) makes the false-positive and false-negative trade-off directly visible.

The decision tree catches 130 spam messages but at the cost of 15 false alarms, the largest FP count in the table. Logistic regression sits at the conservative end, with four FP and 40 FN and comparatively weak spam coverage. Random forest brings the error counts down to two FP and 23 FN for a 91.04% F1-score, and MNB remains the most conservative useful option, with zero FP against 28 FN. Together these answer RQ1: identical TF-IDF features do not produce identical operating behavior, and which model is "best" depends entirely on the relative cost assigned to a false alarm versus a miss.

4.2 Selected SVM operating point

Linear SVM records the best accuracy (98.12%), recall (88.00%), and F1-score (92.63%) of the six. It catches 132 of 150 spam messages and misses 18, while flagging three legitimate messages as spam. Figure 4 shows the corresponding confusion matrix; substituting its four counts into (10) gives

$$FPR_{SVM} = \frac{3}{3 + 962} \times 100 = 0.31\%$$

$$FNR_{SVM} = \frac{18}{18 + 132} \times 100 = 12.00\%$$
(13)

Relative to MNB, SVM trades three additional observed false alarms for 6.67 percentage points of spam recall.

Reference label	Actual ham	962 true ham	3 false alarm
	Actual spam	18 missed spam	132 detected spam
		Predicted ham	Predicted spam
		Predicted label	

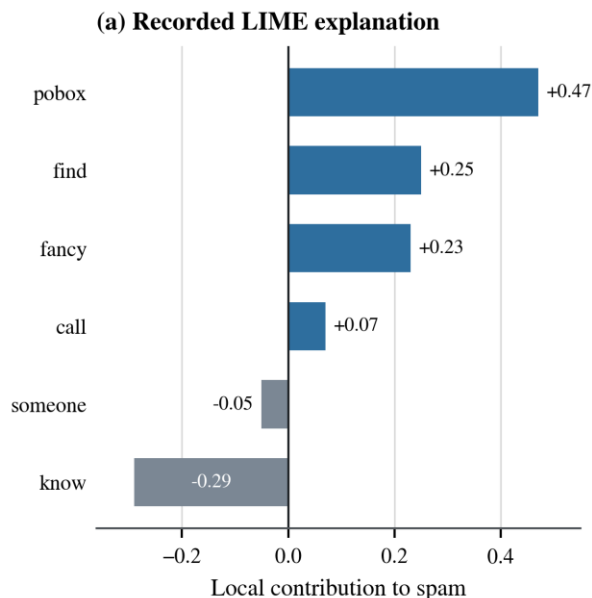
Fig. 4. Linear SVM confusion matrix for the 1,115-message test partition.

That small number of false alarms should still be read with some caution. The 95% Wilson interval for the SVM's FPR is roughly 0.11%–0.91%; for MNB, zero observed false positives out of 965 still carries an upper 95% bound near 0.40%. The SVM's recall interval sits at roughly 81.83%–92.27%. None of these intervals correct for the duplicate overlap or for split selection, but they're a useful reminder that 0.31% is a point estimate, not a guaranteed production rate.

RQ2, then, is answered conditionally. SVM gives the strongest measured balance on this split because it combines the top recall and F1-score with only three FP. MNB stays reasonable wherever local policy treats any false alarm as unacceptably costly. A production system should tune its threshold or cost function on a separate validation set, and a recoverable quarantine interface is a safer default than silent deletion.

4.3 Local explanation

The left panel of Fig. 5 reconstructs the LIME values preserved in the experimental notebook for one correctly classified spam message. In that local neighborhood, “pobox” contributes the most toward spam (+0.47), followed by “find” (+0.25), “fancy” (+0.23), and “call” (+0.07); “know” (−0.29) and “someone” (−0.05) pull the surrogate toward ham. Seeing both supporting and opposing evidence together is more informative than only showing the winning tokens.



This explanation is not the classifier, it approximates the classifier’s behavior near one message, under one chosen perturbation distribution. A different sample, random seed, tokenization, or kernel width could shift the weights. That makes it a useful audit example, but not enough on its own to establish global feature importance, stability, or trust.

4.4 Global coefficient evidence

The right panel of Fig. 5 ranks the positive coefficients from the recorded SVM vocabulary. The largest are “txt,” “uk,” “claim,” “service,” “mobile,” “ringtone,” “www,” “urgent,” and “prize.” These line up with the promotional, prize-oriented language typical of this collection, and partly echo the local LIME example—though they also expose a dependence on dated, region-specific campaign vocabulary that may not hold up once that vocabulary shifts.

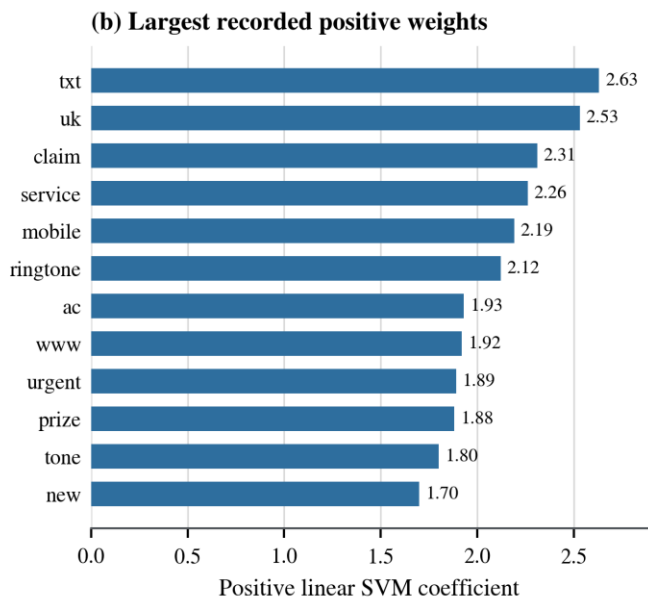


Fig. 5. Complementary explanation views from the experimental record. Panel (a) is one recorded LIME neighborhood; positive values support spam and negative values support ham. Panel (b) gives the twelve largest positive linear SVM coefficients in the fitted vocabulary. Neither plot is a causal explanation.

RQ3 has a two-part answer. The local and global views do surface intelligible lexical evidence and can flag spurious reliance, but they cannot verify malicious intent, explanation fidelity, stability, fairness, or performance on a new campaign. Their real value in this study is transparency, letting a reviewer see which words drive one decision and which words define the model’s overall positive direction.

5 LIMITATIONS AND FUTURE WORK

5.1 Limitations

The selected operating point is conditional on both the recorded test split and on whatever relative cost gets assigned to each error type. KNN and MNB share 100% observed precision, but KNN misses 85 of 150 spam messages. The decision tree recovers eight more spam messages than MNB while adding 15 false positives. Random forest trims one false positive off SVM’s count but misses five more spam messages. Linear SVM is preferred here for its measured balance, but a different application might reasonably prefer MNB if blocking any legitimate message carries a much steeper cost.

The message-length profile is a property of this particular historical corpus, not a universal rule. Spam messages in the fixed test set run a median 147 raw characters versus 52 for ham, with

means near 137.5 and 72.8 respectively. Length isn’t used as a model feature here, and a long message can perfectly well be a legitimate notification, but the gap does suggest campaign-level regularities that may help on this dataset and weaken as messaging styles change.

Comparing this work against published results is limited by incompatible experimental settings. The strong showing from MNB, SVM, and random forest lines up with foundational studies of content features and conventional learners [1]–[4], [8], [9], but that is not evidence that sparse models beat contextual systems [11]–[15] those studies differ in corpus construction, augmentation, feature representation, hyperparameter selection, and validation. Smishing studies [6], [7], [19] also target a richer security label than the plain spam/ham used here, so their numbers can’t be placed in a fair ranking without a shared dataset and protocol.

Internal validity is limited by a single unstratified random split, unpinned software versions, no separate validation set, and mostly-default hyperparameters rather than nested model selection. The corpus carries 403 duplicate label–text rows, and 116 exact texts sit in both the recorded training and test partitions an overlap that can reward recognition of repeated campaign language rather than genuine generalization. Removing or grouping those duplicates

would change every reported count, so this paper reports the overlap instead of quietly running a different experiment.

External validity is limited by one older English corpus, dated regional vocabulary, and coarse binary labels; the design says nothing about future campaigns, adversarial spelling, multilingual messages, sender behavior, URL destinations, or out-of-domain traffic. Construct validity is limited to accuracy, and F1-score don't capture the real cost of hiding an authentication code or missing a credential-stealing link. FPR and the raw error counts clarify the trade-off that was recorded here, but a genuine user-impact study and an application-specific cost model are still missing.

The preprocessing stage removes digits, punctuation, emoji, and URL structure, all of which may carry fraud signal it discards. The local LIME evidence comes from a single correctly classified message with no recorded fidelity or repeated-seed stability score, and the global coefficient view identifies learned associations without establishing causation or fairness. Finally, the bounded TF-IDF representation suggests a compact inference path, but serialized size, memory use, latency, energy consumption, calibration, and usability were never measured on an actual mobile device.

5.2 Future work

The natural next experiment groups exact and near-duplicate templates before creating separate training, validation, and test partitions. Repeated stratified group splits or campaign-level cross-validation could estimate variance across samples, and a chronological hold-out could measure performance on later vocabulary. The data checksum, split indices, preprocessing resources, package versions, and random seeds should all be archived together, so this design would show how much of the recorded performance survives once repeated campaign content is isolated.

Feature ablation should then restore digits, punctuation, character n-grams, message length, URL tokens, and sender or domain signals one family at a time. A threat-labeled dataset—separating benign personal messages, legitimate bulk notifications, advertising, scams, credential phishing, and malware delivery is a further necessary step, as is coverage of multilingual and code-mixed messages, relevant for settings such as Bangladesh where Bangla, English, and transliterated text often share an inbox. Under a shared protocol, this linear SVM baseline could then be compared fairly against contextual, ensemble, fuzzy, and large-language-model approaches.

The explanation pipeline needs direct evaluation of its own. Future work should report LIME surrogate fidelity, repeat each explanation across random seeds, and measure agreement among LIME, exact linear contributions, and SHAP-style attributions [21], [22]. Grouping misclassified messages by campaign and error type would help show whether the model leans on obsolete terms, misses new obfuscation, or mistakes legitimate promotions for spam—and a small user study could test whether showing supporting and opposing tokens together helps a reviewer without the explanation being mistaken for a security verdict.

Deployment-oriented work should serialize the vectorizer and classifier, profile memory and latency on representative low-power phones, and measure energy use under realistic message volumes. A recoverable quarantine folder is a safer default than automatic deletion, since it lets a user restore a false positive; suspicious URLs could be routed to a separate reputation or landing-page analyzer, and decision thresholds should be chosen on validation data according to the relative cost of a false alarm versus a missed threat. Monitoring global coefficients and recent error samples over

time could also flag vocabulary drift and signal when retraining is due.

6 CONCLUSION

This study built a complete, inspectable machine-learning workflow for binary SMS spam detection. The methodology starts from the 5,572-message spam.csv derivative of the SMS Spam Collection, records its duplicate and label properties, preserves the documented 80/20 split, and applies one preprocessing function identically to both partitions. A TF-IDF vectorizer fitted only on the 4,457 training messages produces a sparse representation capped at 3,000 features; Multinomial Naive Bayes, logistic regression, decision tree, KNN, linear SVM, and random forest are then trained on that common representation and evaluated on the same 1,115 held-out messages, which makes their FP and FN behavior directly comparable.

The results make the case for why accuracy alone falls short on an imbalanced task like this one. KNN produces no observed false positives but catches only 43.33% of spam. MNB also produces no false positives while lifting recall to 81.33%, making it a defensible conservative alternative. Linear SVM gives the strongest overall balance recorded here: it correctly identifies 132 of 150 spam messages, misses 18, and incorrectly flags three of 965 ham messages 98.12% accuracy, 97.78% precision, 88.00% recall, 92.63% F1-score, and a 0.31% FPR. That selection reflects both spam recovery and legitimate-message protection, not a single headline score.

Explainable artificial intelligence adds an audit layer on top of that operating point. LIME approximates the selected SVM near one message and shows which present tokens locally push its score toward spam or ham the recorded example assigns the strongest positive local influence to 'pobox,' with 'know' pulling the other way. The global view ranks the fitted linear coefficients and surfaces terms such as 'txt,' 'claim,' 'urgent,' and 'prize' as general spam-associated directions in the learned vocabulary. Together, these views let a reviewer inspect both one decision and the broader model pattern, but they remain evidence about classifier behavior, not proof of sender intent, explanation stability, or causal importance.

The paper's contribution is an auditable baseline with explicit operating counts, equations, workflow controls, and clearly stated explanation boundaries. Its scope stays limited by the spam/ham labels, one historical English corpus, the 116 exact texts shared across the fixed train-test boundary, letter-only preprocessing, and the absence of temporal, adversarial, mobile-performance, and user studies. The reported values should be read as results for this preserved benchmark, not as guaranteed production rates or validated smishing performance. Grouped and chronological evaluation, richer threat labels, feature ablation, explanation-fidelity testing, and measured on-device profiling remain the necessary next steps before this framework could support a real mobile filtering system.

REFERENCES

- [1] T. A. Almeida, J. M. G. Hidalgo, and A. Yamakami, "Contributions to the study of SMS spam filtering: New collection and results," in *Proc. 11th ACM Symp. Document Eng.*, 2011, pp. 259–262, doi: 10.1145/2034691.2034742.
- [2] J. M. Gómez Hidalgo, G. Cajigas Bringas, E. Puertas Sández, and F. Carrero García, "Content based SMS spam filtering," in *Proc. 2006 ACM Symp. Document Eng.*, 2006, pp. 107–114, doi: 10.1145/1166160.1166191.
- [3] G. V. Cormack, J. M. Gómez Hidalgo, and E. Puertas Sández, "Spam filtering for short messages," in *Proc. 16th ACM Conf. Inf. Knowl. Manage.*, 2007, pp. 313–320, doi: 10.1145/1321440.1321486.
- [4] S. J. Delany, M. Buckley, and D. Greene, "SMS spam filtering: Methods and data," *Expert Syst. Appl.*, vol. 39, no. 10, pp. 9899–9908, 2012, doi: 10.1016/j.eswa.2012.02.053.

- [5] E. Blanzieri and A. Bryl, "A survey of learning-based techniques of email spam filtering," *Artif. Intell. Rev.*, vol. 29, no. 1, pp. 63–92, 2008, doi: 10.1007/s10462-009-9109-6.
- [6] J. W. Joo, S. Y. Moon, S. Singh, and J. H. Park, "S-Detector: An enhanced security model for detecting Smishing attack for mobile computing," *Telecommun. Syst.*, vol. 66, no. 1, pp. 29–38, 2017, doi: 10.1007/s11235-016-0269-9.
- [7] S. Mishra and D. Soni, "Smishing Detector: A security model to detect smishing through SMS content analysis and URL behavior analysis," *Future Gener. Comput. Syst.*, vol. 108, pp. 803–815, 2020, doi: 10.1016/j.future.2020.03.021.
- [8] H. Baaqeel and R. Zagrouba, "Hybrid SMS spam filtering system using machine learning techniques," in *Proc. 2020 Int. Arab Conf. Inf. Technol.*, 2020, pp. 1–8, doi: 10.1109/ACIT50332.2020.9300071.
- [9] S. Rapacz, P. Cholda, and M. Natkaniec, "A method for fast selection of machine-learning classifiers for spam filtering," *Electronics*, vol. 10, no. 17, Art. no. 2083, 2021, doi: 10.3390/electronics10172083.
- [10] H. A. Al-Kabbi, M.-R. Feizi-Derakhshi, and S. Pashazadeh, "Multi-type feature extraction and early fusion framework for SMS spam detection," *IEEE Access*, vol. 11, pp. 123756–123765, 2023, doi: 10.1109/ACCESS.2023.3327897.
- [11] X. Liu, H. Lu, and A. Nayak, "A Spam Transformer model for SMS spam detection," *IEEE Access*, vol. 9, pp. 80253–80263, 2021, doi: 10.1109/ACCESS.2021.3081479.
- [12] C. Oswald, S. E. Simon, and A. Bhattacharya, "SpotSpam: Intention analysis-driven SMS spam detection using BERT embeddings," *ACM Trans. Web*, vol. 16, no. 3, pp. 1–27, 2022, doi: 10.1145/3538491.
- [13] A. Ghourabi and M. Alohal, "Enhancing spam message classification and detection using transformer-based embedding and ensemble learning," *Sensors*, vol. 23, no. 8, Art. no. 3861, 2023, doi: 10.3390/s23083861.
- [14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT*, 2019, pp. 4171–4186, doi: 10.18653/v1/N19-1423.
- [15] M. A. Uddin, M. N. Islam, L. Maglaras, H. Janicke, and I. H. Sarker, "ExplainableDetector: Exploring transformer-based language modeling approach for SMS spam detection with explainability analysis," *arXiv:2405.08026*, 2024, doi: 10.48550/arXiv.2405.08026.
- [16] Y. Li, R. Zhang, W. Rong, and X. Mi, "SpamDam: Towards privacy-preserving and adversary-resistant SMS spam detection," *arXiv:2404.09481*, 2024, doi: 10.48550/arXiv.2404.09481.
- [17] M. Salman, M. Ikram, N. Basta, and M. A. Kaafar, "SpaLLM-Guard: Pairing SMS spam detection using open-source and commercial LLMs," *arXiv:2501.04985*, 2025, doi: 10.48550/arXiv.2501.04985.
- [18] A. K. Jasim, F. A. Al-Ibraheemi, and H. A. Alkaabi, "Explainable AI for SMS spam filtering: A novel hybrid architecture combining fuzzy logic and bidirectional LSTM networks," *Franklin Open*, vol. 14, Art. no. 100466, 2026, doi: 10.1016/j.fraope.2025.100466.
- [19] S. K. Bithriya, P. Ahlawat, and A. K. Jain, "Machine learning-based smishing detection using fuzzy logic and TF-IDF feature engineering," *Franklin Open*, vol. 14, Art. no. 100506, 2026, doi: 10.1016/j.fraope.2026.100506.
- [20] Q. Zhang, D. Chen, L. M. Kaplan, A. Jøsang, D. H. Jeong, F. Chen, and J.-H. Cho, "X-MAP: cXplainable misclassification analysis and profiling for spam and phishing detection," *arXiv:2602.15298*, 2026, doi: 10.48550/arXiv.2602.15298.
- [21] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you?: Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1135–1144, doi: 10.1145/2939672.2939778.
- [22] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Adv. Neural Inf. Process. Syst.* 30, 2017, pp. 4765–4774.
- [23] S. J. Delany, P. Cunningham, and A. Tsybal, "A case-based technique for tracking concept drift in spam filtering," *Knowl.-Based Syst.*, vol. 20, no. 2, pp. 187–195, 2007, doi: 10.1016/j.knosys.2006.11.014.
- [24] T. A. Almeida and J. M. G. Hidalgo, "SMS Spam Collection," *UCI Machine Learning Repository*, 2012, doi: 10.24432/C5CC84.
- [25] G. A. Miller, "WordNet: A lexical database for English," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995, doi: 10.1145/219717.219748.
- [26] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
- [27] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, U.K.: Cambridge Univ. Press, 2008.
- [28] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets," *PLoS ONE*, vol. 10, no. 3, Art. no. e0118432, 2015, doi: 10.1371/journal.pone.0118432.