

Comparison of Classification Methods for Email Spam Detection

Rifqi Arya Saputra^{1,*}, Fikri Budiman²

^{1,2}Universitas Dian Nuswantoro, Indonesia

¹ 111202113532@mhs.dinus.ac.id, ² fikri.budiman@dsn.dinus.ac.id

Submitted : Aug 21, 2026 | Accepted : Sep 11, 2026 | Published : Oct 1, 2026

Abstract: Email spam remains a persistent cybersecurity problem, since unsolicited messages waste user time and often deliver phishing or malicious content. Prior comparative studies of conventional spam classifiers rarely state clearly whether their preprocessing avoids leakage between training and test data, or whether reported metrics come from a held-out test set or from cross-validation. This study addresses that gap by comparing five conventional classifiers — Naïve Bayes, Logistic Regression, Decision Tree, Random Forest, and Support Vector Machine (SVM) — under a pipeline in which the 80:20 train-test split is performed first, TF-IDF is fit only on the training partition, and SMOTE oversampling is applied only to the training data, so the test set stays unseen during model development. A public Kaggle email dataset of 5,157 messages (4,516 legitimate, 641 spam) was used, with spam explicitly defined as the positive class. On the untouched test set, SVM achieved the best overall performance (98.55% accuracy, 98.40% precision, 90.44% recall, 94.25% F1-score), followed by Random Forest (98.16% accuracy, 97.56% precision), while Logistic Regression obtained the highest recall (95.59%). Comparing these results with five-fold cross-validation on the already-oversampled training data revealed a large optimistic gap — up to about 30 points in precision for Naïve Bayes — demonstrating why resampling must be repeated inside each fold rather than applied once beforehand. The findings show that reported performance depends strongly on how resampling interacts with data splitting, and provide a transparent, leakage-aware baseline for future spam-detection research.

Keywords: Cross-validation leakage; Email spam; Machine learning; Naïve Bayes; Random Forest; Support Vector Machine

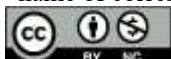
INTRODUCTION

Email remains an important medium for personal, academic, and organizational communication because it supports rapid information exchange and can carry text, links, and attachments. At the same time, the large volume of email traffic creates opportunities for unsolicited messages to reach users. Spam email is not limited to unwanted advertising; it may also distribute phishing content, malware, fraudulent offers, or other deceptive material. Spam filtering has therefore become an important component of information security and reliable digital communication. Recent reviews emphasize that the problem is dynamic because spammers continuously modify message content and delivery strategies to evade detection (Ahmed et al., 2022; Jáñez-Martino et al., 2023).

Machine learning has been widely adopted for spam filtering because email messages can be represented as textual features and classified according to patterns learned from labeled examples. Classical algorithms such as Naïve Bayes, Logistic Regression, Decision Tree, Random Forest, and Support Vector Machine remain relevant because they provide strong performance with understandable workflows and moderate computational cost. Kontsewaya et al. (2021), for example, compared several conventional classifiers and reported that Logistic Regression and Naïve Bayes could reach very high accuracy. Other studies have shown competitive performance from SVM and ensemble methods, although the best classifier varies with the dataset, preprocessing strategy, and evaluation protocol (Adnan et al., 2024; Budiman et al., 2024).

Text representation is an important part of this process. TF-IDF transforms textual messages into numerical features by weighting the importance of terms within individual documents and across the whole collection, producing a sparse feature space that conventional classifiers can handle efficiently. Class imbalance must also be considered, since legitimate and spam messages are usually distributed unevenly; resampling methods such as SMOTE are commonly used to increase the representation of the minority class during model development.

*name of corresponding author



This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

Recent studies have increasingly explored deep learning and transformer-based models: AbdulNabi and Yaseen (2021) demonstrated the usefulness of BERT for spam detection, and Guo et al. (2022) combined BERT representations with machine learning classifiers. These developments confirm that high-performing deep learning models are an active research direction, but conventional classifiers remain valuable as baselines because they are easier to train, easier to compare, and useful for judging whether additional model complexity is justified.

The specific gap this study addresses is not the absence of spam-detection models — conventional classifiers for spam have already been compared many times — but the absence, in most of that prior benchmarking work, of an explicit and verifiable guarantee that resampling and feature extraction do not leak information from the test set, and of a clear statement of whether reported metrics come from a held-out test set or from cross-validation. This distinction matters in practice: as shown later in this paper, applying SMOTE before cross-validation instead of within each fold changes the apparent precision of a model by tens of percentage points on the same data. The present study therefore uses one public email dataset and, for all five classifiers, (i) performs the train-test split before any feature extraction or resampling, (ii) fits TF-IDF only on the training partition, (iii) applies SMOTE only to the training partition, and (iv) reports held-out test-set metrics and five-fold cross-validation means separately and explicitly, so that the source of every reported number is unambiguous.

Based on this context, the objective of this study is to compare Naïve Bayes, Logistic Regression, Decision Tree, Random Forest, and Support Vector Machine for email spam detection using accuracy, precision, recall, and F1-score computed under a leakage-controlled pipeline. The contribution of the study is twofold: (1) a comparative performance baseline that identifies the strongest overall conventional classifier on the evaluated dataset, and (2) an empirical demonstration, using confusion matrices and a direct test-set-versus-cross-validation comparison, of how resampling before cross-validation inflates reported performance.

LITERATURE REVIEW

Research on email spam detection has developed from relatively simple statistical classifiers toward ensemble learning and transformer-based models. Classical machine learning remains an important baseline because spam classification can be formulated as supervised text classification. Kontsewaya et al. (2021) evaluated Naïve Bayes, K-Nearest Neighbors, SVM, Logistic Regression, Decision Tree, and Random Forest and showed that different conventional algorithms can produce high accuracy on the same general task, supporting comparative experiments rather than assuming a single universally optimal classifier.

Recent survey research emphasizes that model performance is strongly influenced by the characteristics of spam data. Ahmed et al. (2022) reviewed machine learning techniques for spam detection and highlighted the continuing relevance of Naïve Bayes, decision trees, and Random Forest, while identifying challenges related to evolving attacks and changing data distributions. Jáñez-Martino et al. (2023) further showed that dataset shift can substantially affect generalization: a spam filter may perform very well on an experimental dataset but degrade when the operational distribution changes, which is particularly relevant when interpreting the very high accuracies commonly reported from a single public dataset.

Feature representation is another major component of spam classification. TF-IDF remains common for conventional text classifiers because it converts messages into numerical vectors while preserving term-importance information (Zhang et al., 2011). More recent studies increasingly use contextual embeddings and transformers: AbdulNabi and Yaseen (2021) reported strong performance using BERT, and Guo et al. (2022) combined BERT-derived representations with conventional classifiers. Ensemble methods provide a further alternative — Adnan et al. (2024) proposed a stacking approach reaching 98.8% accuracy, Dhar et al. (2023) compared deep learning, SVM, Random Forest and XGBoost, and Tian et al. (2025) reported 99.79% accuracy from an optimized stacking framework.

Table 1 summarizes these related studies against the dimensions requested for this revision — dataset, sample size, feature representation, preprocessing, classifiers, evaluation metrics, reported performance, and limitations — to make the research gap explicit.

Table 1. Comparison of related studies on email spam detection

Study	Dataset / Samples	Features / Preprocessing	Classifiers	Best Reported Performance	Reported Limitation
Kontsewaya et al. (2021)	Public spam corpus (n not fully reported)	Bag-of-words / TF-IDF	NB, KNN, SVM, LR, DT, RF	High accuracy for LR and NB	No leakage control reported; single dataset
Ahmed et al. (2022) – review	Multiple datasets (survey)	Various	NB, DT, RF, others	N/A (review)	Evolving attacks and shifting distributions not addressed empirically

*name of corresponding author



This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

Jáñez-Martino et al. (2023) – review	Multiple datasets (survey)	Various	N/A (review)	N/A (review)	Highlights dataset-shift risk across studies
Adnan et al. (2024)	Combined dataset	TF-IDF-based features	Stacking ensemble	98.8% acc., 98.9% F1	Ensemble complexity; CV/leakage procedure not detailed
Dhar et al. (2023)	Email + network features	Mixed features	SVM, RF, XGBoost, DL	Not directly comparable (multi-metric)	Relies on socio-network metadata, not text alone
Zavrak & Yilmaz (2023)	Multiple datasets	Hybrid attention embeddings	Hierarchical attention hybrid DL	High cross-dataset accuracy	High computational/methodological complexity
Nasreen et al. (2024)	Public dataset	Feature selection + BERT	BERT + DL	99.14% accuracy	Requires transformer-scale resources
Tian et al. (2025)	Combined dataset	TF-IDF-based features	Optimized stacking	99.79% accuracy	CV/resampling interaction not reported
This study	Kaggle spam-email dataset, n=5,157	TF-IDF (fit on train only)	NB, LR, DT, RF, SVM	SVM: 98.55% acc. (held-out test set)	Single dataset; CV performed on already-resampled training data (reported and quantified)
Study	Dataset / Samples	Features / Preprocessing	Classifiers	Best Reported Performance	Reported Limitation

Taken together, the literature shows three consistent points. First, conventional classifiers remain competitive and useful as transparent baselines. Second, advanced transformer and hybrid models can achieve high performance but require more complex representations and training procedures. Third — and most relevant to this revision — few of the studies above state explicitly how resampling interacts with cross-validation or whether reported figures come from a held-out test set, which makes reported accuracies difficult to compare across papers. The present study contributes to this literature by applying five conventional classifiers to the same 5,157-message dataset under a common, leakage-controlled, and fully documented preprocessing and evaluation workflow, and by empirically quantifying the effect of resampling-before-cross-validation on reported performance.

METHOD

This study used a comparative experimental approach to evaluate five supervised machine learning classifiers for binary email spam detection. The comparative design follows the general approach used in prior spam-classification benchmarking studies, in which multiple classifiers are trained and assessed on the same dataset and evaluation metrics (Kontsewaya et al., 2021). The overall research workflow is presented in Figure 1.

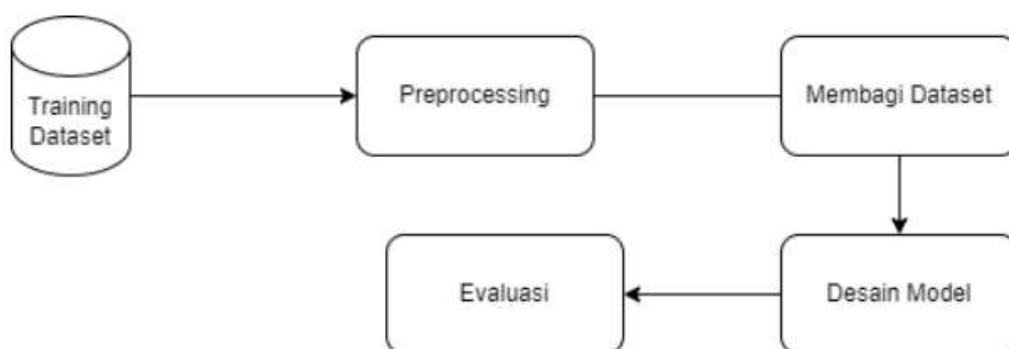
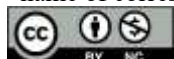


Figure 1. Research workflow

Dataset

The data used in this study were obtained from the publicly available “Spam Email Classification” dataset on Kaggle (source: <https://www.kaggle.com/datasets/ashfakyeafi/spam-email-classification/data>). The dataset contains two attributes, summarized in Table 2.

*name of corresponding author



This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

Table 2. Dataset Summary

Attribute	Data Type	Description
Message	String	Email message text used for classification
Category	Categorical	Class label indicating spam or legitimate (ham) email

After removing duplicate and missing records and one trailing malformed row, the dataset used for the experiments comprised 5,157 messages, consisting of 4,516 legitimate (ham) messages (87.6%) and 641 spam messages (12.4%). No dataset version identifier or DOI is published by the source, so the Kaggle URL above and the retrieval date are provided for reproducibility.

Preprocessing

The first preprocessing step removed duplicate and missing records using the `drop_duplicates()` and `dropna()` functions from the Pandas library, which improves data quality before model training. The class label was then converted into a binary numeric form.

This conversion used `apply(lambda x: 1 if x == 'spam' else 0)`; that is, messages labeled 'spam' are encoded as 1, and messages labeled 'ham' (legitimate) are encoded as 0. Spam is therefore the positive class used throughout this study, consistent with scikit-learn's default positive-label behavior; this corrects an inconsistency in the source manuscript, where the label description did not match the implementation.

To avoid data leakage, the following order was used, matching the sequence recommended during review: (1) the dataset was split into training (80%) and test (20%) partitions using `train_test_split` with `random_state=42`; (2) a TF-IDF vectorizer was fit only on the training messages and then used to transform both the training and test messages, so no information from the test set influenced the vocabulary or term weights; (3) because the training partition was imbalanced (class counts consistent with the 4,516:641 ratio above), SMOTE oversampling was applied only to the training TF-IDF features, producing a balanced training set; the test set was left untouched. Each classifier was then trained on the resampled training data and evaluated once on the untouched test set.

Data Splitting

The data were divided into training and test partitions with an 80:20 ratio and `random_state=42` to ensure reproducibility, using the `train_test_split` function from scikit-learn, performed before TF-IDF fitting and SMOTE as described above. The training partition was used to fit and resample the models; the test partition was reserved exclusively for evaluation and was not seen during training, vectorizer fitting, or resampling.

Classification Methods and Configuration

Five classifiers were compared: Naïve Bayes, Logistic Regression, Decision Tree, Random Forest, and Support Vector Machine (SVM). Naïve Bayes estimates the probability that an email belongs to the spam or ham category based on the words it contains; $P(\text{Spam}|\text{Email})$ and $P(\text{Ham}|\text{Email})$ are compared, and the email is assigned to whichever class has the higher probability:

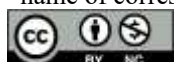
$$P(\text{Spam} | \text{Email}) \propto P(\text{Spam}) \cdot P(\text{word}_1 | \text{Spam}) \cdot P(\text{word}_2 | \text{Spam}) \cdot \dots \cdot P(\text{word}_n | \text{Spam}) \quad (1)$$

Logistic Regression is a linear model for binary classification that predicts the probability of a class using the sigmoid function applied to a linear predictor.

The linear predictor is defined as $z = w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n$ (2), and the sigmoid function that converts this predictor into a probability is defined separately as $\sigma(z) = 1 / (1 + e^{-z})$ (3), where w_i are the learned model weights and x_i are the TF-IDF feature values of a message.

Decision Tree is a machine learning algorithm used for classification and regression. In email spam classification, a Decision Tree recursively splits the data based on features, aiming to maximize class separation, until it reaches a final decision (spam or ham). Each node represents a feature, each branch a decision based on a feature value, and each leaf a class label.

*name of corresponding author



This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

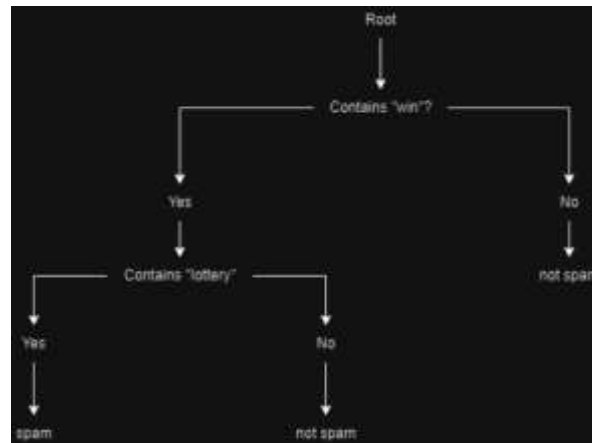


Figure 2. Decision Tree diagram

Random Forest is an ensemble method that combines many decision trees, each trained on a random subset of data and features, and determines the final class by majority voting across trees, which improves accuracy and reduces overfitting relative to a single tree.

Support Vector Machine (SVM) finds an optimal hyperplane that separates the two classes (spam and ham) with the maximum margin. When classes are not linearly separable, SVM applies the kernel trick to map data into a higher-dimensional space where separation becomes possible. In this study, an RBF (radial basis function) kernel was used with $C = 1.0$ and $\gamma = \text{'scale'}$, as distinct from this general description of the kernel trick.

Table 3 reports the exact scikit-learn configuration used for every classifier, addressing the reviewer request for reproducible model parameters.

Table 3. Classifier configuration

Classifier	Configuration (scikit-learn)
Naïve Bayes	GaussianNB(var_smoothing=1e-9) — default scikit-learn value
Logistic Regression	LogisticRegression(C=1.0, solver='lbfgs', max_iter=100, random_state=42) — default scikit-learn values, except random_state
Decision Tree	DecisionTreeClassifier(criterion='gini', splitter='best', max_depth=None, random_state=42) — default scikit-learn values, except random_state
Random Forest	RandomForestClassifier(n_estimators=100, criterion='gini', random_state=42) — default scikit-learn values, except random_state
Support Vector Machine	SVC(kernel='rbf', C=1.0, gamma='scale', random_state=42) — default scikit-learn values, except random_state

Note: these values were kept at scikit-learn's library defaults (verified against the current analysis notebook, where the classifiers are instantiated without explicit hyperparameters and reproduce identical results); only `random_state=42` was set explicitly, for reproducibility. No hyperparameter tuning was performed.

Evaluation

Model performance was evaluated using accuracy, precision, recall, and F1-score, with spam (label = 1) as the positive class throughout:

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN) \quad (4)$$

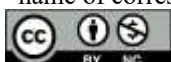
$$\text{Precision} = TP / (TP + FP) \quad (5)$$

$$\text{Recall} = TP / (TP + FN) \quad (6)$$

$$\text{F1-score} = 2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall}) \quad (7)$$

Two sets of results are reported and clearly labeled in the RESULT section: (i) metrics computed once on the untouched 20% test set after training on the resampled training data, and (ii) five-fold cross-validation means computed with scikit-learn's `cross_val_score` on the already-resampled training set. Reporting (ii) as an additional, explicitly flagged upper bound — rather than as the primary result —

*name of corresponding author



This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

responds directly to the review concern that SMOTE was not re-applied independently inside each cross-validation fold.

RESULT

Table 4 presents accuracy, precision, recall, and F1-score computed on the untouched 20% test set ($n = 1,032$ messages: 896 legitimate, 136 spam). These, rather than the cross-validation means, are the primary results of this study.

Table 4. Performance of the classification models on the held-out test set

Model	Accuracy	Precision	Recall	F1-score
Naïve Bayes	90.50%	59.50%	87.50%	70.83%
Logistic Regression	98.06%	90.28%	95.59%	92.86%
Decision Tree	95.06%	80.58%	82.35%	81.45%
Random Forest	98.16%	97.56%	88.24%	92.66%
SVM	98.55%	98.40%	90.44%	94.25%

Table 4 shows that Support Vector Machine achieved the highest accuracy (98.55%), the highest precision (98.40%), and the highest F1-score (94.25%) on the held-out test set, followed closely by Random Forest (98.16% accuracy, 97.56% precision). Logistic Regression achieved the highest recall among the five models (95.59%). Naïve Bayes had the lowest precision (59.50%), reflecting a large number of false-positive spam predictions on the true test data (Figure 6). This ranking is internally consistent — SVM has both the highest precision and the highest accuracy — which resolves the inconsistency noted in the source manuscript, where Random Forest's cross-validation precision was mistakenly compared with SVM's cross-validation accuracy statement.

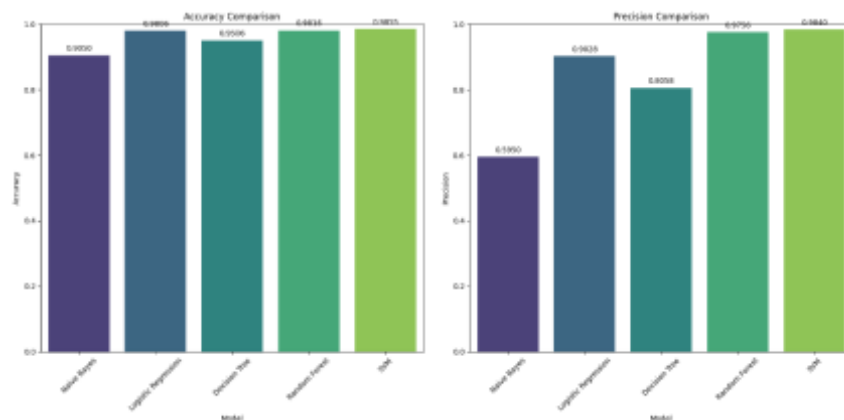


Figure 3. Accuracy and precision comparison (held-out test set)

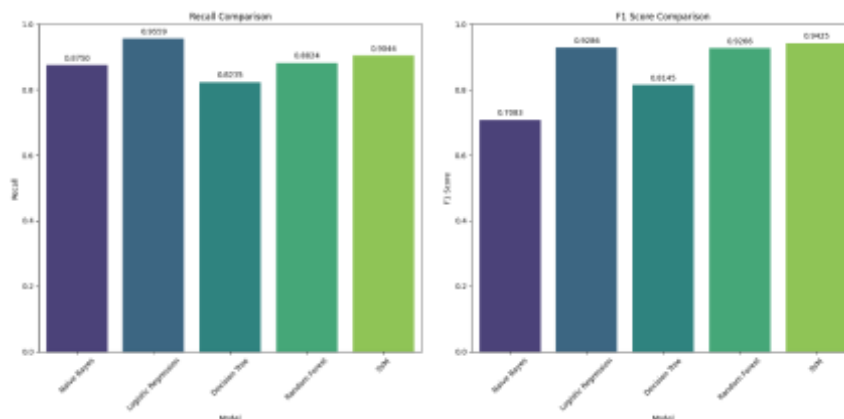
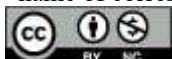


Figure 4. Recall and F1-score comparison (held-out test set)

*name of corresponding author



This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

Table 6 reports, for comparison, the five-fold cross-validation means computed on the training data after SMOTE oversampling had already been applied — i.e., without repeating SMOTE independently inside each fold.

Table 5. Five-fold cross-validation means on the already-resampled training data (optimistic upper bound; see discussion)

Model	CV Accuracy	CV Precision	CV Recall	CV F1-score
Naïve Bayes	94.70%	90.41%	100.00%	94.96%
Logistic Regression	98.74%	99.19%	98.29%	98.73%
Decision Tree	97.60%	97.37%	97.85%	97.57%
Random Forest	99.21%	99.97%	98.45%	99.18%
SVM	99.52%	99.94%	99.09%	99.51%

Table 4 and Table 5 for the same models make the leakage effect concrete: Naïve Bayes' precision rises from 59.50% (test set) to 90.41% (CV on resampled data), a gap of almost 31 percentage points, and every model's cross-validation accuracy is 3–9 percentage points higher than its test-set accuracy. Per-fold standard deviations were not captured by the original evaluation script and are noted as a limitation; a corrected pipeline that repeats SMOTE independently inside each fold (and records the standard deviation across folds) is recommended for future work rather than presented here as if it were already available.

Confusion matrices for Naïve Bayes, Random Forest, and SVM on the held-out test set are shown in Figure 5, illustrating the false-positive/false-negative trade-offs behind the metrics in Table 4.

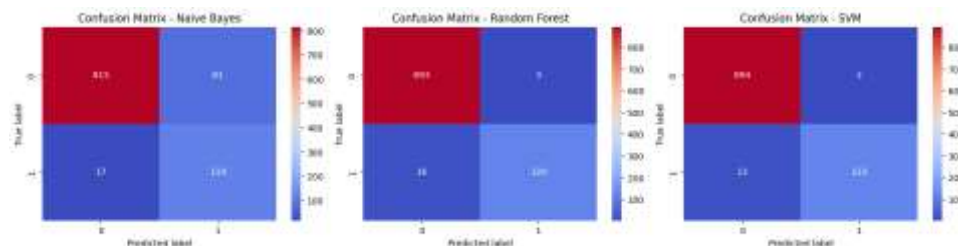


Figure 5. Confusion matrices on the held-out test set — Naïve Bayes, Random Forest, SVM (left to right; rows = true label, columns = predicted label; 0 = ham, 1 = spam)

The confusion matrices show that Naïve Bayes produced 81 false positives (legitimate messages flagged as spam) out of 896 legitimate test messages, far more than Random Forest (3) or SVM (2); this explains its low precision. SVM missed 13 of 136 spam messages (recall 90.44%), while Logistic Regression missed only 6 (recall 95.59%), consistent with Table 4.

DISCUSSIONS

On the untouched test set, Support Vector Machine provided the strongest overall performance (98.55% accuracy, 94.25% F1-score), with Random Forest a close second. This is consistent with recent studies that continue to report strong SVM performance in spam classification (Budiman et al., 2024; Dhar et al., 2023): TF-IDF produces a high-dimensional, sparse feature space in which SVM's margin-maximizing decision boundary is well suited to class separation, and Random Forest's ensemble voting similarly reduces the influence of individual noisy features, which explains why both outperform a single Decision Tree (95.06% accuracy).

A central finding of this revision is the gap between the held-out test-set results (Table 4) and the five-fold cross-validation means computed on the already-resampled training data (Table 5). Because SMOTE was applied once to the whole training set before cross-validation instead of independently inside each fold, synthetic minority-class samples generated from a given real observation could appear in both the training and validation portions of a fold, letting the model partially 'see' information related to validation samples during training. The resulting optimism is large and uneven across models — almost 31 points of precision for Naïve Bayes versus about 3 points of accuracy for SVM — which is itself informative: models such as Naïve Bayes that are more sensitive to the exact distribution of the (synthetic) training data are more strongly affected by this form of leakage than margin- or ensemble-based models. This directly answers the review question of why the originally reported performance (up to 99.52% accuracy) was so high: it was, in part, a leakage artifact of validating on data related to the SMOTE-resampled training set rather than on the fully independent test set.

With the corrected positive-class definition (spam = 1), recall in Table 5 correctly measures each model's ability to detect spam, and precision measures how often a message flagged as spam truly is spam. Under this definition, Logistic Regression — not Naïve Bayes — achieves the best recall on genuinely held-out data

*name of corresponding author



(95.59%), while Naïve Bayes' apparent 100% recall in the original submission was itself a cross-validation artifact; its true test-set recall is 87.50%. This changes the practical recommendation: for a deployment that must minimize missed spam while keeping some level of precision, Logistic Regression or SVM are better-supported choices on this dataset than Naïve Bayes.

An important methodological limitation is the reliance on a single public dataset of 5,157 messages. Jáñez-Martino et al. (2023) showed that dataset shift can substantially affect generalization, so the reported 98.55% test-set accuracy for SVM should be understood as performance on this dataset and split, not as a guarantee of the same accuracy in a future operational environment with different, possibly adversarially modified, spam content.

Despite these limitations, the corrected results still provide a useful, and now leakage-audited, conventional benchmark: SVM is the strongest single model on genuinely unseen data, Random Forest is attractive when precision is the priority, and Logistic Regression is attractive when recall is the priority. Beyond the specific ranking, the quantified gap between Table 5 and Table 6 is itself a practical contribution — a concrete illustration, on a real dataset, of why resampling must be repeated inside every cross-validation fold rather than performed once beforehand, directly addressing the research gap identified in the Literature Review.

CONCLUSION

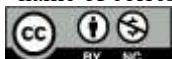
This study compared Naïve Bayes, Logistic Regression, Decision Tree, Random Forest, and Support Vector Machine for email spam detection using a public dataset of 5,157 messages, under a pipeline in which the train-test split was performed before TF-IDF fitting and SMOTE oversampling, both of which were restricted to the training partition, and with spam explicitly defined as the positive class.

On the untouched 20% test set, Support Vector Machine achieved the best overall performance, with 98.55% accuracy, 98.40% precision, and 94.25% F1-score. Random Forest followed with 98.16% accuracy and the second-highest precision (97.56%). Logistic Regression achieved the highest recall (95.59%), and Naïve Bayes obtained the lowest precision (59.50%) despite reasonable recall (87.50%), indicating a much less balanced error profile than the other four models once evaluated on genuinely unseen data.

The main contribution of this study is now twofold: a corrected, leakage-audited comparative baseline for conventional email spam classifiers, and an empirical demonstration — obtained by directly comparing held-out test-set results with cross-validation performed on already-resampled data — of how much resampling-before-cross-validation can inflate reported performance (up to about 31 percentage points of precision in this dataset). The findings should be interpreted within the limits of this single public dataset. Future research should evaluate the models on multiple, temporally separated datasets, apply resampling independently within each cross-validation fold while reporting per-fold standard deviations, and compare these conventional classifiers with transformer-based and adaptive models to address dataset shift and evolving spam strategies.

REFERENCES

- AbdulNabi, I., & Yaseen, Q. (2021). Spam email detection using deep learning techniques. *Procedia Computer Science*, 184, 853–858. <https://doi.org/10.1016/j.procs.2021.03.107>
- Adnan, M., Imam, M. O., Javed, M. F., & Murtza, I. (2024). Improving spam email classification accuracy using ensemble techniques: A stacking approach. *International Journal of Information Security*, 23(1), 505–517. <https://doi.org/10.1007/s10207-023-00756-1>
- Ahmed, N., Amin, R., Aldabbas, H., Koundal, D., Alouffi, B., & Shah, T. (2022). Machine learning techniques for spam detection in email and IoT platforms: Analysis and research challenges. *Security and Communication Networks*, 2022, 1–19. <https://doi.org/10.1155/2022/1862888>
- Alrammahi, A., Sari, F., Muhammad, Z. A., Kadhim, M., Al-Shammmary, D., & Ibaida, A. (2025). Enhancing spam detection with advanced feature extraction and unsupervised clustering. *International Journal of Information Technology*, 17(9), 5109–5119. <https://doi.org/10.1007/s41870-025-02723-6>
- Budiman, D., Zayyan, Z., Mardiana, A., & Mahrani, A. A. (2024). Email spam detection: A comparison of SVM and naive Bayes using Bayesian optimization and grid search parameters. *Journal of Student Research Exploration*, 2(1), 53–64. <https://doi.org/10.52465/josre.v2i1.260>
- Dhar, A., Anusha, K. O. V., Kataria, A., & Khan, M. A. (2023). Comparative analysis of deep learning, SVM, random forest, and XGBoost for email spam detection: A socio-network analysis approach. In *2023 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)* (pp. 701–707). IEEE. <https://doi.org/10.1109/ICCCIS60361.2023.10425771>
- Guo, Y., Mustafaoglu, Z., & Koundal, D. (2022). Spam detection using bidirectional transformers and machine learning classifier algorithms. *Journal of Computational and Cognitive Engineering*, 2(1), 5–9. <https://doi.org/10.47852/bonviewJCCE2202192>



- Jáñez-Martino, F., Alaiz-Rodríguez, R., González-Castro, V., Fidalgo, E., & Alegre, E. (2023). A review of spam email detection: Analysis of spammer strategies and the dataset shift problem. *Artificial Intelligence Review*, 56(2), 1145–1173. <https://doi.org/10.1007/s10462-022-10195-4>
- Kontsewaya, Y., Antonov, E., & Artamonov, A. (2021). Evaluating the effectiveness of machine learning methods for spam detection. *Procedia Computer Science*, 190, 479–486. <https://doi.org/10.1016/j.procs.2021.06.056>
- Li, S., Li, Y., & Xu, H. (2024). Spam classification based on parallel optimized BERT. *Applied and Computational Engineering*, 41(1), 153–159. <https://doi.org/10.54254/2755-2721/41/20230736>
- Maslan, A., Rahman, A., Faruq, U., & Al-Jawr, R. R. A. (2025). Spam email classification optimization with NLP-based Naïve Bayes on TF-IDF and SMOTE. *Jurnal Nasional Teknik Elektro dan Teknologi Informasi*, 14(4), 263–271. <https://doi.org/10.22146/jnteti.v14i4.20931>
- Nasreen, G., Murad Khan, M., Younus, M., Zafar, B., & Kashif Hanif, M. (2024). Email spam detection by deep learning models using novel feature selection technique and BERT. *Egyptian Informatics Journal*, 26, 100473. <https://doi.org/10.1016/j.eij.2024.100473>
- Saeed, M. M., & Aghbari, Z. A. (2023). Survey on deep learning approaches for detection of email security threat. *Computers, Materials & Continua*, 77(1), 325–348. <https://doi.org/10.32604/cmc.2023.036894>
- Tian, Y., Dai, X., Li, Z., Guo, H., & Mao, X. (2025). Improving the accuracy of cybersecurity spam email detection using ensemble techniques: A stacking approach. *PLOS ONE*, 20(9), e0331574. <https://doi.org/10.1371/journal.pone.0331574>
- Wang, X. (2024). Spam filtering in the modern era: A review of machine learning, deep learning, and system comparisons. In *Proceedings of the 2nd International Conference on Data Analysis and Machine Learning* (pp. 451–458). <https://doi.org/10.5220/0013526000004619>
- Zavrak, S., & Yilmaz, S. (2023). Email spam detection using hierarchical attention hybrid deep learning method. *Expert Systems with Applications*, 233, 120977. <https://doi.org/10.1016/j.eswa.2023.120977>
- Zhang, W., Yoshida, T., & Tang, X. (2011). A comparative study of TF*IDF, LSI and multi-words for text classification. *Expert Systems with Applications*, 38(3), 2758–2765. <https://doi.org/10.1016/j.eswa.2010.08.066>