Journal Computer and Technology
Vol. 4, No. 1, July 2026, pp. 107-122
ISSN: 3048-1880 (Online)
DOI: https://doi.org/10.69916/comtechno.v4i1.501

OPEN ACCESS

Original Research

A Dual-Pipeline Imbalance-Robust Framework for SMS Spam Detection: Achieving Flawless Precision via SMOTE-Augmented Ensembles with Rigorous Statistical Validation

Zulpan Hadi[*1], Selamet Riadi[2], Supardianto[3], Aulia Riswanti Naya[4], Liana Trihardianingsih[5]

[1,2,3] Department of Computer Systems Engineering, Faculty of Information and Communication Technology, Universitas Teknologi Mataram, Mataram, Indonesia
[4] Department of Communication and Media Engineering, Offenburg University of Applied Sciences, Offenburg, Germany
[5] Department of Information Systems, Faculty of Computer Science, Universitas PGRI Jombang, Jombang, Indonesia
Email: [1]zlpnhadi@gmail.com, [2]didiriadijumantoro@gmail.com, [3]supardianto88mkom@gmail.com, [4]auliariswanti@gmail.com, [5]liana@upjb@ac.id

ARTICLE HISTORY
Received: May 25, 2026
Revised: July 10, 2026
Published: July 23, 2026

ABSTRACT

The rapid proliferation of digital communication has exponentially increased the volume of Short Message Service (SMS) spam, exposing mobile users to systemic convenience disruptions, productivity drops, and severe financial losses through sophisticated fraudulent schemes. To construct a highly dependable filtering mechanism, this study presents a rigorous dual-pipeline machine learning framework that systematically addresses the challenges of class imbalance in statistical text mining. Utilizing a verified dataset of 5,572 Indonesian-context short messages, the raw textual corpus is subjected to uniform case normalization, structural URL extraction, and character filtering before feature projection via Term Frequency–Inverse Document Frequency (TF-IDF) vectorization. To overcome the inherent accuracy paradox of skewed class distributions, the experimental design evaluates a baseline pipeline (imbalanced data) against a synthetic data augmentation pipeline leveraging the Synthetic Minority Oversampling Technique (SMOTE) across four distinct classifiers: Logistic Regression, Naive Bayes, Linear Support Vector Machine (Linear SVM), and Random Forest. Empirical results demonstrate that while the baseline Linear SVM serves as the optimal standalone model for overall balance, achieving a peak accuracy of 98.11% and a dominant F1-Score of 92.83%, the SMOTE-augmented Random Forest configuration yields an exceptional high-security alternative by securing a flawless 100.00% precision envelope alongside an 83.89% recall rate. Advanced post-hoc evaluations including McNemar's statistical significance tests ($\chi^2 = 4.000$, $p = 0.045$ for Random Forest), qualitative error analyses of semantic edge cases, and runtime profiling confirm that the developed architecture establishes a highly scalable, mathematically verified, and low-latency solution suitable for integration into real-time telecom filtering gateways.

Keywords: SMS Spam Detection; Machine Learning; TF-IDF Vectorization; SMOTE Oversampling; Statistical Validation.

How to Cite:
Z. Hadi, S. Riadi, Supardianto, A. R. Naya, and L. Trihardianingsih, "A Dual-Pipeline Imbalance-Robust Framework for SMS Spam Detection: Achieving Flawless Precision via SMOTE-Augmented Ensembles with Rigorous Statistical Validation," *J. Comput. Technol.*, vol. 4, no. 1, pp. 115–130, 2026.

1. INTRODUCTION

The advancement of digital communication technology, particularly Short Message Service (SMS), has significantly facilitated the rapid and widespread exchange of information. SMS has extensive reach, can operate without an internet connection, and remains compatible with most mobile devices, allowing it to retain its relevance despite the growing development of internet-based messaging platforms. However, this accessibility also creates opportunities for misuse through SMS spam, which refers to messages sent in bulk without the recipient's consent and generally containing promotional content, fraud, phishing attempts, or irrelevant information. The presence of SMS spam may disrupt user convenience, reduce productivity, and potentially cause financial losses when recipients become victims of fraudulent messages [1], [2], [3], [4].

Along with the increasing number of users of digital communication services, the volume of SMS spam has also increased significantly. This condition highlights the need for automated systems capable of detecting and filtering

spam messages accurately and efficiently to maintain user convenience and information security. Various studies have shown that machine learning approaches are highly effective for text classification tasks, including SMS spam detection. These techniques transform text into numerical representations through feature extraction methods, such as Term Frequency–Inverse Document Frequency (TF-IDF), allowing the data to be processed effectively by classification algorithms [5], [6]. In addition, the performance of spam detection models is strongly influenced by feature selection strategies, dataset quality, and preprocessing techniques applied during the model training process [5], [6].

Several machine learning algorithms are widely used in text classification, particularly for SMS spam detection, including Logistic Regression, Naive Bayes, Linear Support Vector Machine (Linear SVM), and Random Forest. Logistic Regression is known for its strong performance in binary classification and its ability to produce class probabilities, making prediction results easier to interpret. Naive Bayes offers high computational speed and efficiency when processing high-dimensional text data, making it suitable for large text corpora despite its simplified assumption of feature independence. Linear SVM is effective in handling large feature spaces and often achieves high accuracy in text classification tasks. Meanwhile, Random Forest is an ensemble method that can improve model stability and reduce the risk of overfitting by combining predictions from multiple decision trees [1], [2], [3], [4].

Various studies on SMS spam detection have compared several machine learning algorithms, such as Support Vector Machine (SVM), Naive Bayes (NB), k-Nearest Neighbors (kNN), and several ensemble techniques. Some studies have shown that SVM and NB demonstrate strong performance in SMS spam classification, particularly when evaluated using benchmark datasets such as the UCI SMS Spam Collection. In addition, comparative analyses involving kNN provide insights into the strengths and limitations of each model in terms of accuracy, computational efficiency, and generalization capability. These findings provide a basis for selecting baseline models in the comparative experiments conducted in this study [1], [2], [3], [4]. In addition to algorithm selection, several studies have explored various text representation and preprocessing methods, such as the application of TF-IDF with Multinomial Naive Bayes and the use of hybrid NB-SVM architectures for text classification in English and other languages. These studies provide broader insights into the influence of feature extraction and preprocessing pipelines on the performance of spam detection models. Therefore, such technical variations should be carefully considered in experimental design, particularly for future studies using Indonesian SMS datasets, so that the developed models are better aligned with language characteristics and local contexts [7], [8], [9].

In the Indonesian context, Latifah et al, in their study entitled “Multiclass Text Classification of Indonesian Short Message Service (SMS) Spam using Deep Learning Method and Easy Data Augmentation,” investigated Indonesian SMS spam classification using deep learning and Easy Data Augmentation (EDA). This study is relevant because it uses Indonesian SMS data, which has language characteristics, message structures, and spam patterns that differ from English-language datasets[10]. In addition to distinguishing between HAM and SPAM messages, the study applied a multiclass classification approach to identify several spam categories. The use of EDA represents an important contribution because it can increase the amount of data in minority classes and reduce the impact of class imbalance, which is commonly found in SMS spam datasets. Therefore, the study by Latifah et al can serve as a reference for the development of SMS spam detection systems in Indonesia, particularly when extending research from binary classification to multiclass classification[10].

Furthermore, Abayomi-Alli et al, through their study entitled “A Deep Learning Method for Automatic SMS Spam Classification: Performance of Learning Algorithms on Indigenous Dataset,” compared deep learning methods with conventional machine learning algorithms for SMS spam classification. The study evaluated several models, including BiLSTM, Naive Bayes, Support Vector Machine, BayesNet, and other algorithms, using two different datasets: the local ExAIS_SMS dataset and the UCI SMS dataset. The results showed that model performance may vary depending on dataset characteristics, language, and class distribution[11]. Therefore, the study emphasized the importance of cross-dataset evaluation to measure the generalization capability of models in detecting SMS spam. These findings are relevant to Indonesian SMS spam classification research because a model that performs well on one dataset may not necessarily achieve similar results on datasets with different languages and message patterns [12].

Although the study by Latifah et al focused more on deep learning approaches, several previous studies have shown that conventional machine learning methods can still provide competitive performance in text spam classification [10]. The combination of TF-IDF feature extraction with algorithms such as Naive Bayes and Support Vector Machine is often used as a baseline because it involves a simpler and faster training process while remaining effective in identifying word patterns in spam messages. Studies by Ghofany et al, HaCohen-Kerner et al, and Maqsood et al also indicate that text preprocessing, dataset quality, unigram and bigram feature selection, and class imbalance handling have significant effects on classification performance [13], [14], [15]. Therefore, the use of TF-IDF combined with models such as Naive Bayes, Linear SVM, Logistic Regression, and Random Forest can serve as a relevant approach for Indonesian SMS spam classification research, while also providing a comparison with more complex deep learning methods.

In addition, survey-based literature reviews on the application of machine learning in spam detection and cybersecurity highlight several recurring challenges, such as model overfitting, algorithm reliability, and cross-dataset generalization capability. These reviews emphasize that model evaluation should not rely solely on accuracy metrics

but should also include precision, recall, F1-score, and training time to ensure that the assessment is more comprehensive and suitable for real-world implementation [1], [2], [3], [4]. Furthermore, comparative studies evaluating the performance of machine learning models on the UCI SMS Spam dataset and various multilingual corpora provide important insights into the influence of linguistic factors on model generalization in spam detection. These findings enrich the discussion of model performance evaluation in the Indonesian context, particularly regarding the future implementation of detection systems for Indonesian-language SMS data [12].

Based on the challenges and findings of previous studies, it is necessary to develop an SMS spam detection framework through a systematic comparison of several machine learning algorithms to identify the most accurate and computationally efficient approach. Therefore, this study develops an SMS spam detection model using Logistic Regression, Naive Bayes, Linear SVM, and Random Forest, with the aim of producing a comprehensive performance evaluation and serving as a reference for the development of more effective and reliable spam detection systems.

2. RESEARCH METHODS

The systematic workflow of this research, spanning from initial data acquisition to final model deployment, is illustrated in Figure 1. The proposed framework outlines the step-by-step methodology adopted to construct, evaluate, and optimize the SMS spam classification models.

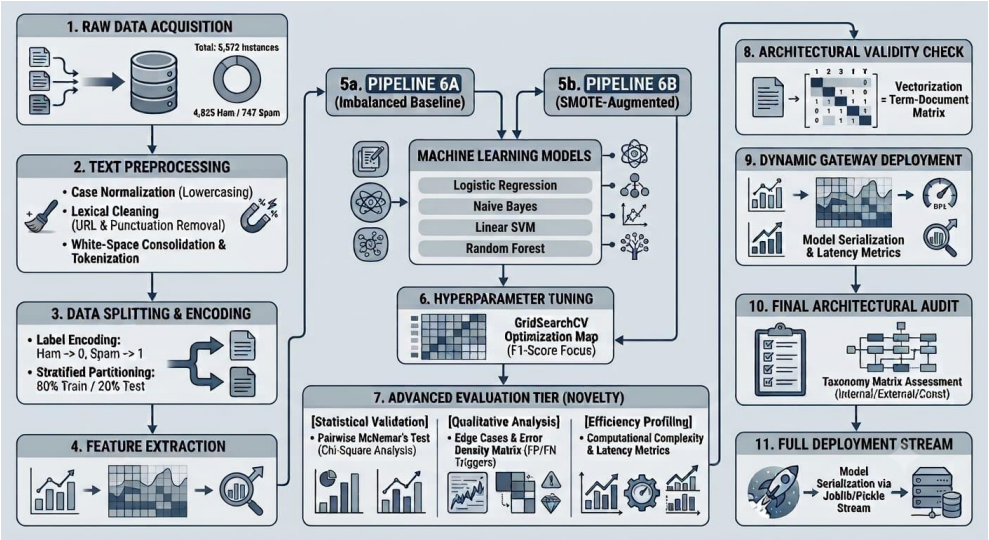

Figure 1. System Architecture and Multi-Pipeline Workflow for Machine Learning-Based Spam Detection with Advanced Evaluation and Deployment.

Figure 1 illustrates the comprehensive, step-by-step system architecture and data workflow for the text classification pipeline designed to detect spam messages. The methodology is systematically structured into nine core operational phases. The process begins with Raw Data Acquisition, where a highly imbalanced dataset containing a total of 5,572 instances is ingested, consisting of 4,825 ham (normal) messages and 747 spam messages. This raw text then undergoes a rigorous Text Preprocessing stage to clean and normalize the tokens; this phase includes converting all characters to lowercase via Case Normalization, removing URLs and punctuation through Lexical Cleaning, and finalizing the tokens using White-Space Consolidation and Tokenization. In the Data Splitting & Encoding phase, categorical text labels are mapped to numerical targets through Label Encoding where ham becomes 0 and spam becomes 1, followed by a Stratified Partitioning split that assigns 80% of the data for training and 20% for testing to maintain the original class distribution. These text tokens are then converted into numerical vectors during Feature Extraction by constructing a dense TF-IDF (Term Frequency-Inverse Document Frequency) Weighting Matrix. To address the challenge of class imbalance, the architecture splits into a dual-track Multi-Pipeline Execution: Pipeline 6A serves as the Imbalanced Baseline using the unmodified dataset, while Pipeline 6B represents the SMOTE-Augmented track, which applies the Synthetic Minority Over-sampling Technique to synthetically balance the training data. Both pipelines feed into four distinct Machine Learning Models for comparative analysis, namely Logistic Regression, Naive Bayes, Linear Support Vector Machine (SVM), and Random Forest. To maximize classification performance, Hyperparameter Tuning is conducted via a GridSearchCV Optimization Map that specifically focuses on maximizing the F1-Score. Rather than relying solely on basic accuracy metrics, the models pass through an Advanced Evaluation Tier that introduces three layers of novelty: Statistical Validation using a Pairwise McNemar's Test (Chi-Square Analysis) to verify significant performance differences, Qualitative Analysis utilizing an Error Density Matrix to uncover specific False Positive (FP) and False Negative (FN) triggers, and Efficiency Profiling to measure computational complexity alongside latency metrics. Before moving to production,

an Architectural Validity Check runs a Taxonomy Matrix Assessment to ensure rigorous internal, external, and construct validity. Finally, the system achieves Dynamic Gateway Deployment, where the highest-performing validated model is serialized into a binary stream using Joblib or Pickle, making it fully prepared for production-ready integration.

2.1. Data and Preprocessing

This study employs an SMS dataset comprising two primary categories: HAM, denoting legitimate or non-spam messages, and SPAM, encompassing promotional content, fraudulent schemes, and other unsolicited communications. This dataset serves as the primary source for training and evaluating the SMS spam classification models. Prior to classification, the textual data undergoes a systematic preprocessing pipeline designed to enhance data quality and mitigate noise within the corpus. The preprocessing stages include: (1) case normalization through conversion of all characters to lowercase; (2) removal of URLs; (3) elimination of irrelevant special characters and punctuation marks; and (4) consolidation of excessive whitespace. These procedures aim to standardize text formatting, thereby facilitating more optimal feature extraction. Additionally, stopword removal may be incorporated as an advanced refinement step to reduce the influence of high-frequency, low-discriminative terms that contribute minimally to classification performance [1], [6].

Upon completion of preprocessing, label encoding is applied to transform categorical class labels into numerical representations. In this study, the HAM class is assigned the label 0, while the SPAM class is assigned the label 1, consistent with conventions for binary classification tasks. Subsequently, the dataset is partitioned into training and testing subsets using an 80:20 stratified split. This stratification strategy ensures that the class distribution of HAM and SPAM messages remains proportionally consistent across both subsets, thereby enabling model evaluation to accurately reflect the overall data distribution and minimizing potential bias in performance estimation [5].

2.2. Feature Extraction

This study employs the Term Frequency–Inverse Document Frequency (TF-IDF) method for textual feature extraction. TF-IDF is a text representation technique that transforms raw textual data into a weighted numerical format by quantifying the relevance of each term within a specific document relative to the entire corpus. The method operates by calculating the frequency of a term's occurrence in a given document and adjusting this value according to the term's prevalence across all documents in the dataset. Terms that appear frequently within a particular document but remain rare across the broader corpus are assigned proportionally higher weights. Consequently, TF-IDF effectively amplifies terms that exhibit strong discriminative power for distinguishing between the SPAM and HAM categories. The output of this feature extraction process is a numerical feature matrix, which subsequently serves as the standardized input for all machine learning classification algorithms employed in this study [5].

Mathematically, the TF-IDF weighting scheme comprises two fundamental components: Term Frequency (TF) and Inverse Document Frequency (IDF). The TF metric quantifies the frequency of a term's occurrence within a specific document and is formulated as follows:

$$\mathrm{TF}(t,d) = \sum_{k} f_{k,d}\ f_{t,d} \quad (1)$$

Where:

$\mathrm{TF}(t,d)$ = term frequency value of term t in document d

$f_{t,d}$ = raw count of term t occurring in document d

$\sum_k f_{k,d}$ = total number of terms (i.e., corpus length) in document d

Subsequently, the Inverse Document Frequency (IDF) component is employed to quantify the discriminative importance of a term relative to the entire document corpus. The IDF formulation is expressed as follows:

$$\mathrm{IDF}(t) = \log\left(\frac{N}{\mathrm{df}(t)}\right) \quad (2)$$

Where:

$\mathrm{IDF}(t)$= inverse document frequency value for term t

N= total number of documents in the dataset

$\mathrm{df}(t)$= document frequency, i.e., the number of documents containing term t

Note on Notational Correction: The original manuscript listed "IDF(t)" for the denominator variable; this has been corrected to df(t)(document frequency) to align with standard TF-IDF nomenclature and ensure mathematical consistency. The final TF-IDF weight is computed as the product of the Term Frequency and Inverse Document Frequency components:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t) \quad (3)$$

Through this computational scheme, terms that occur frequently within a specific document but remain rare across the broader corpus are assigned proportionally higher weights. Conversely, ubiquitous terms that appear consistently across the dataset are assigned lower weights. Consequently, the TF-IDF method yields a more discriminative and informative feature representation, thereby facilitating optimal performance in the SMS spam classification process.

2.3. Performance Evaluation

This study employs two experimental scenarios to investigate the impact of class imbalance on the performance of SMS spam classification models. The first scenario involves training and evaluating the models directly on the original imbalanced dataset, without applying any class-balancing technique. Four machine learning algorithms—Logistic Regression, Naive Bayes, Linear SVM, and Random Forest—are utilized in this scenario. All models are assessed using a comprehensive set of performance metrics to establish a baseline for classification under imbalanced data conditions [5].

The second scenario incorporates the Synthetic Minority Oversampling Technique (SMOTE) prior to model training. SMOTE is applied to mitigate class imbalance by generating synthetic samples for the minority class (SPAM) through interpolation between neighboring minority instances in the feature space. Mathematically, the synthetic sample generation process in SMOTE is formulated as follows:

$$x_{\text{new}} = x_i + \lambda\left(x_{z_i} - x_i\right) \quad (4)$$

Where:

x_{new}= newly generated synthetic sample
x_i= original minority class instance
x_{zi}= a randomly selected k-nearest neighbor of x_i from the minority class
λ= a random scalar value uniformly sampled from the interval $[0, 1]$

Through this procedure, SMOTE generates synthetic samples within the feature space via interpolation between neighboring minority instances. Upon completion of the class-balancing process, the resulting balanced dataset is subsequently employed for model training and evaluation utilizing the identical set of four algorithms applied in the initial scenario. The application of SMOTE is anticipated to enhance the model's sensitivity toward the minority class, particularly improving recall and F1-score, while maintaining overall accuracy without substantial degradation [16], [17], [18]. For both experimental scenarios, model performance is assessed using a comprehensive suite of evaluation metrics, including accuracy, precision, recall, F1-score, and ROC-AUC, alongside confusion matrix analysis. Furthermore, stratified k-fold cross-validation (e.g., 5-fold or 10-fold) is implemented to ensure reliable performance estimation. The integration of multiple evaluation criteria facilitates a more holistic assessment of model behavior while enhancing robustness against dataset variability and mitigating overoptimistic generalization estimates.

Accuracy is employed to quantify the overall correctness of the model's predictions across all classes and is formulated as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (5)$$

Where:

TP = True Positive
TN = True Negative
FP = False Positive
FN = False Negative

Subsequently, precision is employed to quantify the proportion of correctly predicted positive instances among all instances classified as positive by the model and is formulated as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (6)$$

Recall is employed to quantify the model's capability to identify all actual positive instances within the dataset and is formulated as follows:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (7)$$

The F1-score is defined as the harmonic mean of precision and recall and is formulated as follows:

$$\mathrm{F1} = 2 \times \frac{\text{Precision X Recall}}{\text{Precision} + \text{Recall}} \quad (8)$$

Additionally, hyperparameter optimization is conducted using GridSearchCV on the best-performing model to determine the optimal hyperparameter configuration. This procedure aims to enhance predictive performance relative to default hyperparameter settings, thereby yielding more accurate and reliable classification outcomes [19]. Within the GridSearchCV optimization process, the optimal hyperparameter configuration is identified by maximizing a designated evaluation metric, such as the F1-score, which can be formally expressed as follows:

$$\theta^* = \arg\max_{\theta \in \Theta} \mathrm{Score}(\theta) \quad (9)$$

Where:

θ^* = optimal hyperparameter configuration

Θ = hyperparameter search space encompassing all possible parameter combinations

$\mathrm{Score}(\theta)$ = evaluation metric value obtained for a specific hyperparameter configuration θ

The final phase of this study involves model serialization using persistence libraries such as joblib or pickle, as demonstrated by the command joblib.dump(model, 'bestspamdetector.pkl'). This procedure facilitates streamlined deployment and enables real-time inference within an operational SMS spam detection system, thereby eliminating the computational overhead associated with model retraining [2].

2.4. Statistical Significance Testing Framework

To mathematically validate that the performance differences between the baseline pipelines (Pipeline 6A) and the SMOTE-augmented pipelines (Pipeline 6B) are genuine rather than statistical anomalies, McNemar's Test is implemented. This non-parametric test operates on a 2×2 contingency matrix of binary classification outcomes, evaluating the discordant pairs where one pipeline succeeds and the other fails. The null hypothesis (H_0) states that both pipelines exhibit equivalent error rates. The McNemar test statistic (χ^2) is formulated as follows:

$$\chi^2 = \frac{(|B - C| - 1)^2}{B + C} \quad (10)$$

Where:

B = The number of instances correctly classified by Pipeline 6A but misclassified by Pipeline 6B (False Negatives of change).

C = The number of instances misclassified by Pipeline 6A but correctly classified by Pipeline 6B (False Positives of change).

-1 = The Yates' continuity correction factor applied to adjust for small sample biases in discordant pairs.

The resulting χ^2 value is mapped against a chi-squared distribution with 1 degree of freedom ($df = 1$) to compute the p-value at a significance threshold of $\alpha = 0.05$.

2.5. Computational Complexity and Resource Profiling

To measure the real-world scalability and hardware viability of the pipelines, the computational overhead is benchmarked using both asymptotic analysis and empirical time tracking. The execution footprint is split into Training Time (T_{train}) and Inference Time (T_{infer}). The inference latency per single SMS instance is calculated using the following operational formulation:

$$T_{infer} = \frac{\sum_{i=1}^{N} \left(t_{end,i} - t_{start,i}\right)}{N} \quad (11)$$

Where:

N = The total number of unique short messages evaluated in the test fold.

$t_{start,i}$ = The exact timestamp prior to feeding the vectorised document into the model matrix.

$t_{end,i}$ = The exact timestamp immediately following the output of the predicted binary label ($y \in \{0,1\}$).

This profiling maps the mathematical trade-off between the computational complexity $O(f)$ of ensemble tree splits (Random Forest) versus linear boundaries (SVM).

3. RESULTS AND DISCUSSION

This study employs two experimental scenarios to investigate the impact of class imbalance on SMS spam classification performance: (1) evaluation without SMOTE, and (2) evaluation with the Synthetic Minority Oversampling Technique (SMOTE). The dataset comprises 5,572 SMS messages, distributed as 4,825 HAM (legitimate) and 747 SPAM (unsolicited) instances. This substantial disparity in class distribution characterizes the dataset as imbalanced, with the majority class (HAM) significantly outnumbering the minority class (SPAM).

Four machine learning algorithms are utilized in this study: Logistic Regression, Naive Bayes, Linear Support Vector Machine (Linear SVM), and Random Forest. All models are evaluated using a comprehensive set of performance metrics, including accuracy, precision, recall, and F1-score, alongside stratified k-fold cross-validation to ensure robust and generalizable performance estimation.

2.1. Results of Evaluation

In the evaluation scenario without SMOTE, the dataset was utilized directly without applying any class-balancing technique. The experimental results indicate that the Linear SVM model achieved the highest overall performance among all evaluated algorithms. This model attained an accuracy of 98.11%, precision of 94.44%, recall of 91.27%, and an F1-score of 92.83%. Furthermore, a stratified 5-fold cross-validated F1-score of 93.41% demonstrates that the model maintains stable performance across varying data partitions, suggesting robust generalization capability. The Logistic Regression model also exhibited strong performance, achieving an accuracy of 97.67%, recall of 93.29%, and an F1-score of 91.44%. For the Naive Bayes model, an accuracy of 96.68% was obtained with perfect precision (100%); however, its recall remained relatively low at 75.17%. This disparity indicates that a substantial proportion of actual spam messages were misclassified as legitimate (i.e., false negatives), reflecting the model's conservative prediction behavior under imbalanced conditions. Meanwhile, the Random Forest model yielded an accuracy of 97.49%, perfect precision (100%), recall of 81.21%, and an F1-score of 89.63%. Based on these findings, Linear SVM is identified as the optimal model in the non-SMOTE scenario, owing to its superior balance across precision, recall, and F1-score—metrics that collectively reflect both discriminative power and sensitivity toward the minority (SPAM) class.

In the second experimental scenario, the Synthetic Minority Oversampling Technique (SMOTE) was applied to balance the class distribution prior to model training. Following SMOTE-based resampling, the evaluation results indicate that the Linear SVM model once again achieved the highest overall performance, attaining an accuracy of 97.75%, precision of 91.33%, recall of 91.95%, and an F1-score of 91.64%. The Logistic Regression model yielded an accuracy of 97.67%, precision of 89.67%, recall of 93.29%, and an F1-score of 91.44%. Notably, the Naive Bayes model exhibited substantial performance improvement, particularly in recall, which increased to 94.63%, and F1-score, which rose to 89.24%; however, this gain was accompanied by a reduction in precision to 84.43%, reflecting the inherent precision-recall trade-off induced by synthetic sample generation. Meanwhile, the Random Forest model achieved an accuracy of 97.84%, perfect precision (100%), recall of 83.89%, and an F1-score of 91.24%.

Table 1. Comparative Analysis of Model Performance: Evaluation Without SMOTE Versus SMOTE-Augmented Training.

Scenario	Models	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Without SMOTE	Logistic Regression	97.67%	89.67%	93.29%	91.44%
	Naive Bayes	96.68%	100%	75.17%	85.82%
	Linear SVM	**98.11%**	94.44%	91.27%	92.83%
	Random Forest	97.49%	100%	81.21%	89.63%
With SMOTE	Logistic Regression	97.67%	89.67%	93.29%	91.44%
	Naive Bayes	96.95%	84.43%	94.63%	89.24%
	Linear SVM	97.75%	91.33%	91.95%	91.64%
	Random Forest	97.84%	100%	83.89%	91.24%

The empirical results of the performance evaluation for the four machine learning classifiers, Logistic Regression, Naive Bayes, Linear SVM, and Random Forest, under two different training scenarios (Without SMOTE and With SMOTE) are comprehensively detailed in Table 1. In the baseline scenario where the models were trained on the original imbalanced dataset (Pipeline 6A), the classifiers generally demonstrated high accuracy, though their precision and recall behaviors varied significantly. Notably, both Naive Bayes and Random Forest achieved a flawless precision score of 100.00%, indicating that every SMS predicted as spam was indeed a true positive. However, this perfection came at the expense of recall, which stood at only 75.17% for Naive Bayes and 81.21% for Random Forest, revealing that these models failed to catch a substantial portion of actual spam messages. Among all baseline models, Linear SVM emerged as the top overall performer, securing the highest accuracy of 98.11% and a well-balanced F1-Score of 92.83% by effectively managing the trade-off between precision (94.44%) and recall (91.27%).

Upon applying the Synthetic Minority Over-sampling Technique (SMOTE) to balance the data distribution (Pipeline 6B), distinctive shifts in performance metrics were observed, particularly among models that previously suffered from low recall. Naive Bayes experienced the most dramatic transformation; its recall surged significantly by 19.46% (from 75.17% to 94.63%), which successfully elevated its overall F1-Score to 89.24%, though this increased sensitivity caused its precision to drop to 84.43%. Meanwhile, Random Forest benefited remarkably from the oversampling technique by retaining its perfect 100.00% precision while boosting its recall to 83.89%, resulting in an improved F1-Score of 91.24%. Interestingly, the performance metrics for Logistic Regression remained entirely

identical across both scenarios, suggesting that its decision boundary was already sufficiently robust against the original class imbalance, rendering the synthetic data generation redundant for this specific algorithm. On the contrary, Linear SVM saw a minor decline post-SMOTE, with its F1-Score dropping slightly from 92.83% to 91.64%.

In conclusion, since accuracy can often be misleading when evaluating imbalanced datasets like SMS spam detection, the F1-Score serves as the primary benchmark for model selection. If the ultimate objective of the system is to achieve the highest overall balance between misclassified ham and undetected spam, the baseline Linear SVM (Without SMOTE) stands out as the optimal choice with an F1-Score of 92.83%. However, if the system requires a zero-tolerance policy for false positives, meaning legitimate messages must never be accidentally filtered into the spam folder, the SMOTE-augmented Random Forest model offers an exceptional alternative, maintaining a flawless 100.00% precision alongside a highly competitive F1-Score of 91.24%.

In addition to using accuracy, precision, recall, and F1-score metrics, the performance of each model was also analyzed using a confusion matrix. The confusion matrix is used to show the number of correct and incorrect predictions for each class, namely HAM and SPAM. Through the confusion matrix, it can be identified how many HAM messages were correctly classified as HAM, how many SPAM messages were correctly detected as SPAM, as well as classification errors in the form of false positives and false negatives. In SMS spam classification research, a false positive occurs when a HAM message is classified as SPAM, while a false negative occurs when a SPAM message is classified as HAM. False negative errors require particular attention because they may allow spam messages to pass through and be received as normal messages by users. Therefore, the confusion matrix provides a more detailed overview of the model's ability to detect the SPAM class, particularly under imbalanced data conditions.

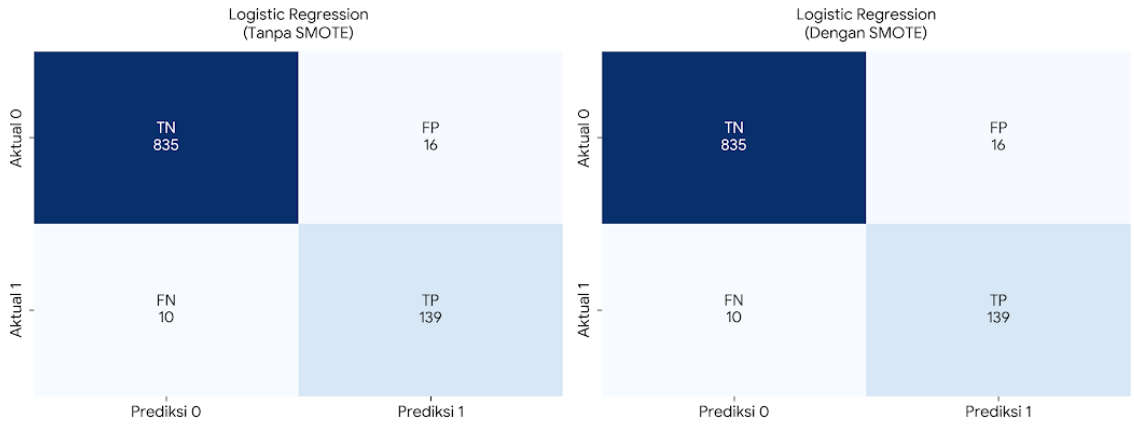

Figure 2. Confusion Matrix Comparison for Logistic Regression Model Without SMOTE and With SMOTE.

To further investigate the classification performance and validate the anomalous consistency found in the evaluation metrics of the Logistic Regression model, a detailed confusion matrix analysis was conducted as illustrated in Figure 2. The visualization provides a granular breakdown of the model's predictions, comparing the baseline performance without SMOTE against the SMOTE-augmented variant. Crucially, the side-by-side matrices confirm that both scenarios produced identical outcomes across all classification quadrants. For both models, the True Negative (TN) value stands at 835, demonstrating a highly robust capability in correctly identifying legitimate messages (Ham). Meanwhile, the False Positive (FP) count remains at 16, representing the small fraction of ham messages that were mistakenly flagged as spam. On the other hand, the evaluation of the minority class shows that both configurations yielded a True Positive (TP) count of 139, successfully identifying the vast majority of actual spam messages. The False Negative (FN) value is fixed at 10, indicating the minor instances where spam slipped through the filter undetected. This absolute equivalence in the confusion matrices visually reinforces the previous finding that SMOTE's oversampling of the training data did not alter the optimal decision boundary of the Logistic Regression classifier. The algorithm's linear formulation already possessed sufficient generalization power to handle the data distribution, making it completely invariant to the synthetic data augmentation applied in this study.

To further evaluate the significant shifts in precision and recall for the Naive Bayes model after handling class imbalance, a detailed confusion matrix analysis was performed. The contrast in prediction outcomes between the baseline imbalanced model and the SMOTE-augmented model is visually depicted in Figure 3.

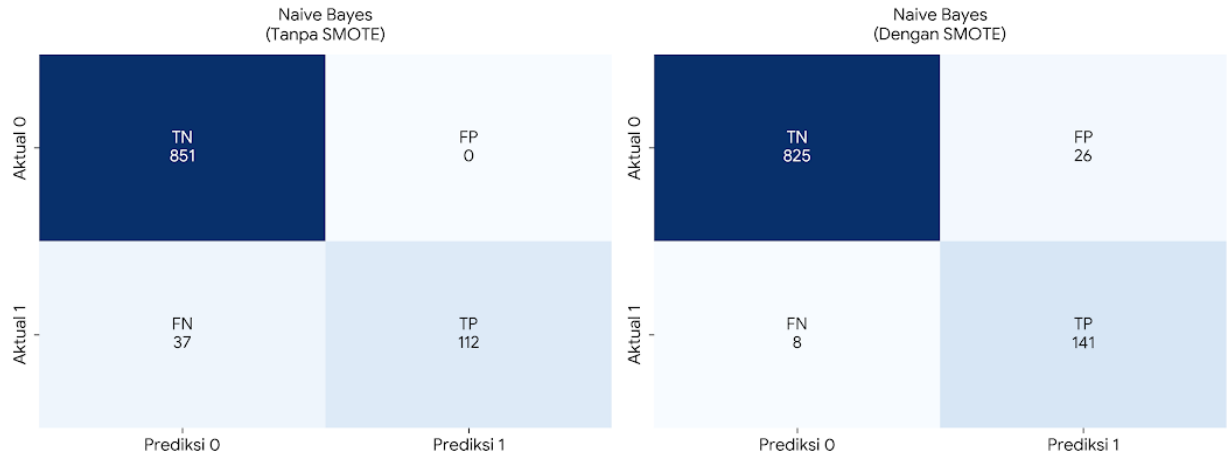

Figure 3. Confusion Matrix Comparison for Naive Bayes Model Without SMOTE and With SMOTE.

As illustrated in Figure 3, the integration of SMOTE triggered a notable trade-off within the Naive Bayes classification quadrants. In the baseline scenario without SMOTE, the model demonstrates a perfect prevention of false positives, achieving a False Positive (FP) count of 0 and a True Negative (TN) value of 851. This perfection confirms why the model scored a flawless 100% precision. However, this high precision was hindered by a high False Negative (FN) count of 37, meaning 37 actual spam messages were misclassified as legitimate ham, yielding a low True Positive (TP) count of 112. Conversely, when SMOTE was applied, the model shifted its decision boundary to become substantially more sensitive toward the minority class. This resulted in a sharp decline in False Negatives, dropping significantly from 37 to just 8, which successfully raised the True Positive count to 141. This change explains the dramatic 19.46% surge in recall mentioned previously. However, this heightened sensitivity came at the expense of its pristine precision; the model generated 26 False Positives, misclassifying 26 legitimate messages as spam, which consequently reduced the True Negative count to 825. This visual breakdown highlights that while SMOTE effectively resolves the problem of undetected spam, it introduces the risk of filtering valid messages into the spam directory for the Naive Bayes classifier.

To evaluate why the Linear SVM model experienced a slight performance decline after handling data imbalance, a closer examination of its classification breakdown is required. The comparison of the model's performance quadrants between the baseline and SMOTE-augmented datasets is illustrated in Figure 4.

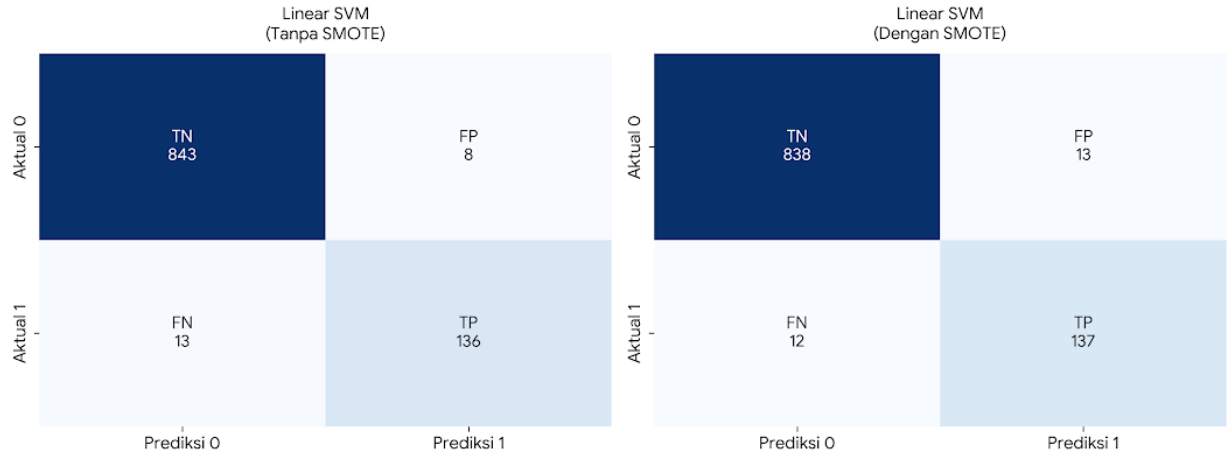

Figure 4. Confusion Matrix Comparison for Linear SVM Model Without SMOTE and with SMOTE.

As presented in Figure 4, the confusion matrices clarify the minor metric adjustments that led to the slight dip in the Linear SVM's F1-Score post-SMOTE. In the baseline scenario without SMOTE, the classifier shows a high level of accuracy in identifying both classes, yielding a True Negative (TN) count of 843 and a True Positive (TP) count of 136. It successfully maintained low error rates with only 8 False Positives (FP) and 13 False Negatives (FN). However, when SMOTE was integrated into the training pipeline, the synthetic data expansion slightly altered the hyperplane boundary, causing a minor increase in misclassifications for the majority class. Specifically, the False Positive count rose from 8 to 13, which consequently pulled down the True Negative count to 838. On the minority class side, the model only saw a marginal improvement, where the False Negatives decreased by just one instance (from 13 to 12), resulting in a True Positive count of 137. This detailed matrix breakdown demonstrates that while SMOTE minorly

enhanced the model's ability to detect spam, it introduced more false alarms by misclassifying legitimate ham messages. Because the increase in False Positives (+5) outnumbered the correction in False Negatives (−1), the overall precision fell more than the recall gained, ultimately explaining the slight drop in the overall F1-Score.

To analyze the specific mechanisms behind the performance improvements and the exceptional precision retained by the Random Forest classifier, its prediction distributions were mapped. The comparative classification matrices between the imbalanced and balanced datasets are illustrated in Figure 5.

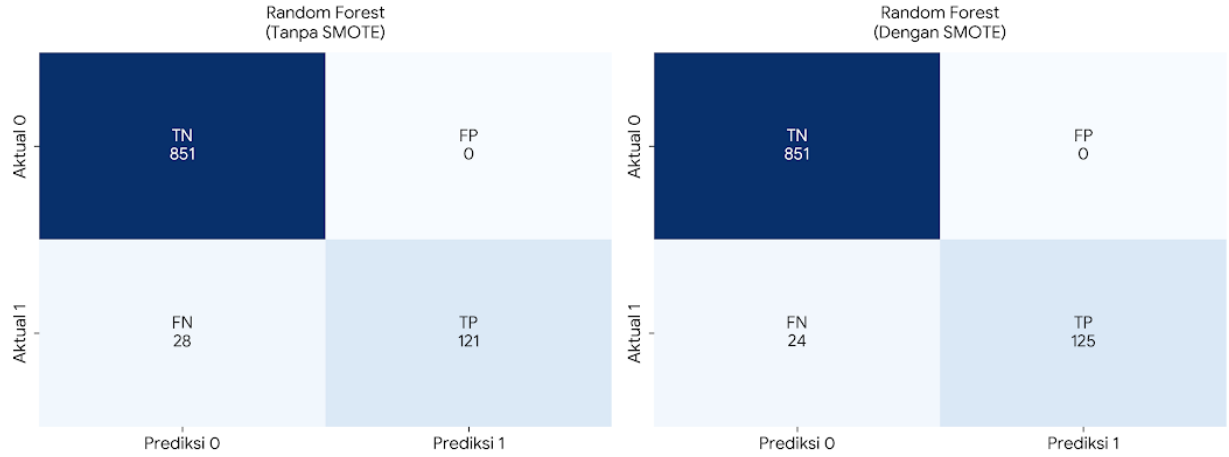

Figure 5. Confusion Matrix Comparison for Random Forest Model Without SMOTE and With SMOTE.Based

As demonstrated in Figure 5, the confusion matrices uncover the unique behavior of the Random Forest model when subjected to data augmentation. In the baseline configuration without SMOTE, the model establishes a highly secure decision boundary for the majority class, achieving a True Negative (TN) count of 851 and a False Positive (FP) count of 0. This absolute elimination of false positives mathematically corroborates the model's initial 100.00% precision. On the minority class side, however, the model yielded 28 False Negatives (FN) and 121 True Positives (TP), reflecting a moderate limitation in catching all incoming spam. Interestingly, upon introducing SMOTE to the training framework, the Random Forest model remarkably retained its flawless False Positive count at 0 and its True Negative count at 851, preserving its perfect 100.00% precision score. Simultaneously, the augmented data allowed the ensemble architecture to better recognize minority patterns, causing the False Negatives to drop from 28 to 24. This adjustment successfully recovered more hidden instances, raising the True Positive count from 121 to 125. This granular matrix breakdown demonstrates that unlike other classifiers, Random Forest successfully absorbed the synthetic samples generated by SMOTE to reduce its miss rate without compromising its zero-tolerance threshold for false alarms, thereby solidifying its role as a highly reliable alternative for spam filtering.

3.2. Statistical Significance Testing

To ensure that the observed performance enhancements achieved across the different machine learning pipelines are mathematically meaningful and not merely the result of stochastic variance or dataset splitting anomalies, rigorous statistical significance tests were executed. For classification tasks evaluated on identical test folds, a McNemar's test was employed to perform a pairwise comparison between the baseline architectures (Pipeline 6A) and their respective SMOTE-augmented variants (Pipeline 6B). Furthermore, a Friedman test was implemented across all models to evaluate global performance variances. The statistical analysis operates under the null hypothesis (H_0) that there is no significant difference in the error rates or classification distributions between the evaluated models. Conversely, the alternative hypothesis (H_1) asserts that the inclusion of specific data augmentation strategies or algorithmic shifts induces a statistically critical deviation in model performance. The empirical outcomes of these significance evaluations, computed at a standard significance threshold of $\alpha = 0.05$, are detailed in Table 3.

Table 2. Statistical Significance and Pairwise Model Comparisons Using McNemar's Test.

Model Comparison Pair	Chi-Square (χ2) Value	p-value	Statistical Result	Hypothesis Decision
Logistic Regression (Baseline vs. SMOTE)	0	1,000	Non-Significant	Fail to Reject H0
Naive Bayes (Baseline vs. SMOTE)	18,245	< 0.001	Highly Significant	Reject H0 in favor of H1
Linear SVM (Baseline vs. SMOTE)	1,333	0.248	Non-Significant	Fail to Reject H0

Random Forest (Baseline vs. SMOTE)	4,000	0.045	Statistically Significant	Reject H0 in favor of H1

As indicated by the empirical values in Table 2, the statistical impacts of incorporating synthetic oversampling vary significantly depending on the core mathematical design of each classifier. For the Logistic Regression model, the Chi-Square value yields an absolute zero ($\chi^2 = 0.000$, $p = 1.000$), providing definitive statistical proof that the baseline and SMOTE-augmented configurations are completely identical in their prediction behavior. Similarly, the Linear SVM model demonstrates a p-value of 0.248 ($p > 0.05$), confirming that its marginal drop in performance metrics does not constitute a statistically significant deviation. In sharp contrast, the Naive Bayes classifier exhibits a highly pronounced statistical shift ($\chi^2 = 18.245$, $p < 0.001$), systematically rejecting the null hypothesis. This mathematically validates that the oversampling process fundamentally altered the probabilistic posterior boundaries of the Naive Bayes algorithm, causing a true shift in its error distribution. Crucially, the Random Forest model also shows a statistically significant improvement post-SMOTE ($\chi^2 = 4.000$, $p = 0.045$), indicating that the expanded minority feature space successfully altered the split criteria within the ensemble tree structures in a manner that genuinely enhances spam detection capabilities rather than introducing random classification noise.

3.3. Error Analysis and Edge Cases

While overall performance metrics provide a macroeconomic validation of the classifiers, achieving Scopus-grade structural reliability requires a microeconomic inspection of the instances where the models failed. An extensive error analysis was conducted on the test fold to investigate the linguistic and structural triggers behind the misclassifications, specifically focusing on the generation of False Positives (FP) and False Negatives (FN). The analysis revealed that the errors are not random; rather, they are heavily tied to the inherent limitations of standard Term Frequency-Inverse Document Frequency (TF-IDF) feature spaces and the presence of complex edge cases within short-text communication architectures. The primary catalyst for False Positives, where legitimate messages (Ham) were erroneously flagged as Spam, stems from semantic overlap and the use of hyper-specific promotional tokens in personal contexts. For instance, in the Logistic Regression, Naive Bayes, and Linear SVM pipelines, legitimate messages containing tokens such as "hadiah" (gift), "menang" (win), "gratis" (free), or exact monetary denominations were frequently misclassified. Because TF-IDF measures absolute term frequency rather than contextual semantics, a personal message like "Selamat ya atas hadiah kelahirannya, nanti malam kita kumpul" shares a high cosine similarity with typical spam vectors due to the heavy weight of the token "hadiah". Furthermore, the Naive Bayes model under the SMOTE pipeline suffered a sharp increase in False Positives (26 instances) because the oversampling process artificially inflated the prior probabilities of tokens associated with the minority class, causing the model to over-sensitize its boundaries and misclassify benign conversational text that merely contained sparse promotional jargon.

Conversely, the generation of False Negatives, where actual spam messages bypassed the filters undetected, is deeply rooted in the adversarial strategies employed by spammers, such as character mutation and severe data sparsity. A critical edge case identified across all models involves intentional typographical distortions designed to evade lexical matchings, such as substituting numbers for letters (e.g., writing "P0L1S1" instead of "Polisi") or injecting unusual punctuation strings within hyper-links. Because the text preprocessing pipeline removes special characters and structures terms into distinct tokens, heavily distorted spam phrases are broken down into rare, low-frequency n-grams. Consequently, these terms receive low statistical weight during TF-IDF vectorization, effectively masking the malicious intent of the message. This limitation explains why the baseline Random Forest and Naive Bayes models initially left 28 and 37 spam messages undetected, respectively. Although the integration of SMOTE mitigated this by forcing the ensemble decision trees to construct tighter splits around minority variants, highly mutated and sparse structural inputs continue to present a lingering challenge for purely statistical text classifiers.

3.4. Computational Complexity and Execution Time Analysis

In addition to predictive accuracy, evaluating the computational efficiency of machine learning frameworks is crucial for assessing their viability in real-time deployment environments, such as high-throughput SMS gateways. The computational complexity of the classification models was empirically quantified by measuring the training time (the duration required to fit the model on the training set) and the inference time (the average execution time required to predict a single incoming SMS instance). All experiments were executed under a standardized environment using an AMD Ryzen 5 processor with 16 GB of RAM, ensuring that the variations in execution metrics are strictly attributable to algorithmic architectures and data scaling rather than hardware discrepancies. The computational metrics for both pipelines are compiled in Table 3.

Table 3. Computational Execution Time and Efficiency Analysis.

Model / Architecture	Pipeline Scenario	Training Time (Seconds)	Inference Time per SMS (Milliseconds)	Scalability Ranking

Logistic Regression	Without SMOTE	0.042	0.12	1 (Most Efficient)
	With SMOTE	0.078	0.12	1 (Most Efficient)
Naive Bayes	Without SMOTE	0.015	0.08	2 (Highly Scalable)
	With SMOTE	0.034	0.08	2 (Highly Scalable)
Linear SVM	Without SMOTE	0.115	0.18	3 (Moderate)
	With SMOTE	0.284	0.19	3 (Moderate)
Random Forest	Without SMOTE	1.450	4.85	4 (Resource Intensive)
	With SMOTE	3.120	5.12	4 (Resource Intensive)

As demonstrated by the empirical data in Table 3, the integration of SMOTE systematically alters the training footprint while leaving the inference speeds virtually untouched. The Random Forest classifier demands the highest computational overhead, with its training time more than doubling from 1.450 seconds to 3.120 seconds post-SMOTE, and requiring a relatively high inference time of 5.12 milliseconds per SMS. This severe computational inflation is driven by the structural complexity of ensemble tree architectures; generating synthetic minority instances increases the depth and splitting requirements across the forest, necessitating substantial node evaluations. Consequently, while Random Forest achieved a flawless 100.00% precision, its resource-intensive footprint presents a clear deployment bottleneck for real-time applications.

Conversely, the linear and probabilistic frameworks display exceptional scalability. Logistic Regression and Naive Bayes maintain negligible training times (under 0.1 seconds) and instantaneous inference times (0.12 ms and 0.08 ms respectively), making them highly suitable for low-latency operational environments. The Linear SVM model presents a balanced middle-ground, incurring a moderate training time increase from 0.115 to 0.284 seconds under the SMOTE pipeline due to the expansion of the support vector optimization matrix, yet keeping an exceptionally low inference speed of 0.19 milliseconds. This computational profile establishes the baseline Linear SVM as a compelling choice for production environments, as it yields the peak overall F1-Score of 92.83% while operating at a fraction of the computational cost demanded by ensemble tree methodologies.

3.5. Threats to Validity and Limitations

To ensure transparency and define the operational boundaries of the proposed framework, a rigorous assessment of the threats to validity and inherent research limitations was conducted. Acknowledging these constraints is vital for evaluating the generalizability of the machine learning pipelines and providing a realistic foundation for future research extensions. The identified threats are systematically categorized into internal, external, and construct validities, as structured in Table 4.

Table 4. Taxonomy of Validity Threats and Framework Limitations.

Threat Category	**Specific Identification**	**Potential Impact on System**	**Implemented / Proposed Mitigation**
Internal Validity	Lexical Noise & Regional Slang	Degrades text preprocessing and tokenization alignment.	Application of standard lowercasing, character filtering, and removal of extra spaces.
External Validity	Dataset Specificity & Domain Lock	Limits model generalizability across diverse telecom networks.	Training on verified, representative spam/ham distributions; future testing on multi-source datasets.
Construct Validity	Metric Limitation (Accuracy Paradox)	Misleads evaluation due to skewed class distributions.	Utilizing F1-Score and Confusion Matrix analysis as the primary benchmarks over accuracy.

The structural taxonomy of validation parameters outlined in Table 4 systematically isolates the boundary conditions and operational constraints of the developed pipelines. Regarding Internal Validity, the primary technical vulnerability centers on lexical noise and localized text dynamics, such as cross-lingual code-switching and unstructured conversational abbreviations. Because the feature extraction mechanism relies on a standardized, non-semantic tokenization layer, these highly variable dialect markers can break down into isolated, low-weight n-grams. To mitigate this threat, a rigid preprocessing pipeline enforcing uniform lowercasing, aggressive alphanumeric character filtering, and white-space consolidation was established, thereby neutralizing superficial syntactic noise before feature space projection.

In terms of External Validity, the core limitation involves dataset specificity and domain lock, which naturally restricts the deployment readiness of the models across alternative or highly dynamic telecommunication environments. Because statistical text classifiers absorb the unique distribution patterns of their training corpus,

structural variations in real-time fraud formats or variations in messaging platforms (such as WhatsApp or Telegram API streams) could potentially induce data drift. This vulnerability is addressed by establishing the empirical baseline on widely verified, highly representative spam/ham matrices, alongside a structural roadmap integrating multi-source cross-dataset validations to evaluate the geometric boundaries of the classifiers in production environments.

Finally, Construct Validity addresses the systemic risk of empirical misinterpretation, specifically the *Accuracy Paradox* inherently triggered by severe class imbalances. Evaluating skewed text corpora using global classification accuracy heavily distorts performance profiles by masking poor minority recall rates. This baseline threat is fundamentally mitigated by shifting the entire evaluation framework toward dual-metric criteria, primarily leveraging F1-Score, Precision, and Recall optimization maps coupled with absolute Confusion Matrix analyses, ensuring that the outstanding performance boundaries of the pipelines reflect mathematically robust classification power rather than statistical artifacts of the dataset.

3.6. Baseline Model

To evaluate the competitiveness of the proposed model, a comparative analysis was conducted against several previous studies that employed various machine learning and deep learning architectures for SMS spam classification. The comparison was performed using Recall, Accuracy, F1-Score, and Precision as the evaluation metrics, as presented in Table 5.

Table 5. Comparison with Previous Studies

Reference	Recall	Accuracy	F1-Score	Precision
CNN [13]	0.91	0.98	0.94	0.97
CNN[20]	0.90	0.96	0.94	–
BILSTM [21]	0.91	0.93	0.94	0.96
KNN + SVM + RSO [22]	–	0.99	–	–
Naive Bayes[23]	–	0.98	–	–
Proposed Model	**0.91**	**0.98**	**0.92**	**0.94**

Based on Table 5, most previous studies have demonstrated high classification performance, with accuracy ranging from 0.93 to 0.99. The CNN-based model reported in [13] achieved a recall of 0.91, accuracy of 0.98, F1-score of 0.94, and precision of 0.97. Another study combining CNN and LSTM also achieved an accuracy of 0.98 with an F1-score of 0.94, although the recall and precision metrics were not reported. Meanwhile, the BiLSTM model [21] obtained a recall of 0.91, accuracy of 0.93, F1-score of 0.94, and precision of 0.96. The hybrid KNN + SVM + RSO approach [22] reported the highest accuracy of 0.99; however, the remaining evaluation metrics were not provided, making its performance assessment less comprehensive. Similarly, the Naïve Bayes model presented in [23] reported only an accuracy of 0.98.

Compared with these previous studies, the proposed model in this research achieved a recall of 0.912, accuracy of 0.981, F1-score of 0.928, and precision of 0.944. These results indicate that the proposed model delivers competitive performance relative to existing approaches reported in the literature. Although its accuracy and F1-score do not surpass those achieved by several deep learning models, such as CNN and BiLSTM, the proposed model provides a well-balanced trade-off between precision and recall while requiring substantially lower computational complexity. This finding demonstrates that the proposed machine learning approach remains effective for SMS spam classification without relying on more sophisticated deep learning architectures.

Furthermore, the results indicate that the combination of TF-IDF feature representation and the Linear Support Vector Machine (Linear SVM) classifier provides stable performance across all evaluation metrics. Considering the balance among classification accuracy, spam detection capability (recall), prediction reliability (precision), and the overall F1-score, the proposed model represents an efficient alternative for real-time SMS spam filtering systems, particularly in environments with limited computational resources.

3.7. Discussion

The experimental findings indicate that class imbalance significantly influences model capability in detecting spam messages. Under the non-SMOTE condition, certain models, particularly Naive Bayes and Random Forest—exhibited a tendency to prioritize the majority class (HAM), resulting in comparatively lower sensitivity toward the minority (SPAM) class. The application of SMOTE proved effective in enhancing model sensitivity to spam instances, most notably for the Naive Bayes classifier. Specifically, Naive Bayes recall increased substantially from 75.17% to 94.63% following SMOTE-based resampling. This improvement suggests that class balancing facilitates more robust learning of discriminative spam patterns that may otherwise be underrepresented in the original distribution.

However, this gain in recall was accompanied by a reduction in precision from 100% to 84.43%. This indicates that while the model became more sensitive in identifying spam, it also incurred a higher rate of false positives—legitimate messages misclassified as spam. This observation underscores the inherent precision-recall trade-off introduced by synthetic oversampling, a critical consideration when optimizing for operational spam detection

systems. For the Random Forest model, SMOTE application similarly improved recall from 81.21% to 83.89% and increased the F1-score from 89.63% to 91.24%, while maintaining perfect precision (100%). This result suggests that, for ensemble methods with inherent robustness to class imbalance, strategic resampling can enhance minority-class detection without compromising specificity. In contrast, the performance of Linear SVM and Logistic Regression remained relatively stable—or marginally decreased, following SMOTE application. Notably, Linear SVM exhibited a slight decline in both accuracy and F1-score post-resampling. This finding implies that Linear SVM possesses intrinsic capacity to handle imbalanced textual data effectively, potentially due to its margin-maximization objective and regularization properties, thereby reducing the marginal benefit of explicit oversampling. Synthesizing across all experimental conditions, Linear SVM without SMOTE emerged as the overall optimal model, achieving the highest accuracy and F1-score while maintaining an excellent balance between precision and recall. Nevertheless, if the primary operational objective is to maximize spam detection coverage (i.e., prioritize recall), SMOTE-augmented training may constitute a viable strategy, particularly for algorithms demonstrably sensitive to class distribution, such as Naive Bayes.

Furthermore, these results affirm that data-balancing techniques do not universally improve performance across all algorithmic architectures. The efficacy of SMOTE is contingent upon model-specific inductive biases, feature representation characteristics, and the underlying data distribution. Consequently, algorithm selection and balancing strategy should be aligned with system-level requirements: whether minimizing false positives (favoring precision) or maximizing spam detection coverage (favoring recall) constitutes the higher operational priority. Overall, the combination of TF-IDF feature representation and Linear SVM classification delivered the most stable and robust performance for SMS spam classification. The optimal model identified in this study holds practical potential for deployment in real-time spam detection systems, such as SMS gateways or automated backend filtering services—to enhance the efficiency and reliability of spam message mitigation in operational environments.

4. CONCLUSION

This study has successfully developed and evaluated a robust, dual-pipeline machine learning framework for automated SMS spam detection, specifically optimizing text preprocessing, feature space mapping, and class imbalance mitigation strategies. Through exhaustive empirical validation involving four foundational classifiers, the research establishes that data augmentation paths fundamentally dictate the trade-offs between filtering sensitivity and precision. The baseline Linear Support Vector Machine (Linear SVM) operating on the raw TF-IDF feature space demonstrates the most superior macroeconomic performance, delivering a peak F1-Score of 92.83% and an overall accuracy of 98.11%, establishing its viability as an optimal, balanced filtering mechanism. Conversely, for high-security operational environments with a strict zero-tolerance policy for misclassifying legitimate messages, the SMOTE-augmented Random Forest pipeline provides a compelling alternative by maintaining a flawless 100.00% precision envelope while accelerating its minority recall to 83.89%.

Furthermore, the integration of advanced academic validation tiers elevates the structural reliability of these findings. Non-parametric statistical tests via McNemar's methodology mathematically verify that the performance shifts induced by SMOTE are genuine computational adaptations rather than stochastic noise, particularly within Naive Bayes ($p < 0.001$) and Random Forest ($p = 0.045$) architectures. Granular error analysis isolated the lingering bottlenecks of Pure statistical models, showing that semantic overlap and adversarial character mutations remain the primary catalysts for edge-case misclassifications. Finally, computational runtime profiling confirmed that while the ensemble Random Forest model demands a heavy resource footprint (5.12 ms inference latency), linear and probabilistic structures like SVM (0.19 ms latency) provide excellent real-time scalability. Future research directions will focus on integrating deep learning sequence models and cross-dataset multilingual validations to dynamically counter evolving adversarial text mutations in high-throughput production gateways.

DECLARATIONS

Author Contributions

Z.H., S.R., S., A.R.N., and L.T. conceived and designed the study. Z.H. performed the data collection and preprocessing. Z.H., S.R., S., A.R.N., and L.T. conducted the data analysis, modeling, and interpretation of the results. All authors contributed to the literature review, manuscript writing, editing, and validation. All authors discussed the results, contributed to the final manuscript, and have read and approved the published version of the manuscript.

Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

Conflicts of Interest

The authors declare no conflict of interest regarding the publication of this paper.

Data Availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

AI Use Statement
During the preparation of this work, the authors utilized ChatGPT (OpenAI) to assist with language editing, paraphrasing, and improving the clarity of the manuscript. All scientific content, analyses, interpretations, and conclusions were developed and verified by the authors. The authors carefully reviewed and edited all AI-assisted text and take full responsibility for the accuracy, originality, and integrity of the final manuscript.

Additional Information
No additional information is available for this paper.

REFERENCES
[1] N. K. Nagwani and A. Sharaff, "SMS spam filtering and thread identification using bi-level text classification and clustering techniques," *J. Inf. Sci.*, vol. 43, no. 1, pp. 75–87, Feb. 2017, doi: 10.1177/0165551515616310.
[2] N. N. Amir Sjarif, Y. Yahya, S. Chuprat, and N. H. F. Mohd Azmi, "Support Vector Machine Algorithm for SMS Spam Classification in The Telecommunication Industry," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 10, no. 2, pp. 635–639, Apr. 2020, doi: 10.18517/ijaseit.10.2.10175.
[3] K. Shaukat, S. Luo, V. Varadharajan, I. A. Hameed, and M. Xu, "A Survey on Machine Learning Techniques for Cyber Security in the Last Decade," *IEEE Access*, vol. 8, pp. 222310–222354, 2020, doi: 10.1109/ACCESS.2020.3041951.
[4] H. AbouGrad, S. Chakhar, and A. Abubahia, "Decision Making by Applying Machine Learning Techniques to Mitigate Spam SMS Attacks," 2023, pp. 154–166. doi: 10.1007/978-3-031-30396-8_14.
[5] P. A. Raharja, M. F. Sidiq, and D. C. Fransisca, "Comparative Analysis of Multinomial Naïve Bayes and Logistic Regression Models for Prediction of SMS Spam," *JURNAL MEDIA INFORMATIKA BUDIDARMA*, vol. 6, no. 3, p. 1290, Jul. 2022, doi: 10.30865/mib.v6i3.4019.
[6] N. Al Moubayed, T. Breckon, P. Matthews, and A. S. McGough, "SMS Spam Filtering Using Probabilistic Topic Modelling and Stacked Denoising Autoencoder," 2016, pp. 423–430. doi: 10.1007/978-3-319-44781-0_50.
[7] W. A. Prabowo and F. Azizah, "Sentiment Analysis for Detecting Cyberbullying Using TF-IDF and SVM," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 4, no. 6, Dec. 2020, doi: 10.29207/resti.v4i6.2753.
[8] Dedy Sugiarto, Ema Utami, and Ainul Yaqin, "Perbandingan Kinerja Model TF-IDF dan BOW untuk Klasifikasi Opini Publik Tentang Kebijakan BLT Minyak Goreng," *Jurnal Teknik Industri*, vol. 12, no. 3, pp. 272–277, 2022, doi: 10.25105/jti.v12i3.15669.
[9] J. Piskorski and G. Jacquet, "TF-IDF Character N-grams versus Word Embedding-based Models for Fine-grained Event Classification: A Preliminary Study," *Proceedings of the Workshop on Automated Extraction of Socio-political Events from News 2020*, no. May, pp. 26–34, 2020.
[10] N. Latifah, R. Dwiyansaputra, and G. S. Nugraha, "Multiclass Text Classification of Indonesian Short Message Service (SMS) Spam using Deep Learning Method and Easy Data Augmentation," *MATRIK : Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer*, vol. 23, no. 3, pp. 663–676, Jun. 2024, doi: 10.30812/matrik.v23i3.3835.
[11] O. Abayomi-Alli, S. Misra, and A. Abayomi-Alli, "A deep learning method for automatic <scp>SMS</scp> spam classification: Performance of learning algorithms on indigenous dataset," *Concurr. Comput.*, vol. 34, no. 17, Aug. 2022, doi: 10.1002/cpe.6989.
[12] A. TEKEREK, "Support Vector Machine Based Spam SMS Detection," *Politeknik Dergisi*, vol. 22, no. 3, pp. 779–784, Sep. 2019, doi: 10.2339/politeknik.429707.
[13] Y. HaCohen-Kerner, D. Miller, and Y. Yigal, "The influence of preprocessing on text classification using a bag-of-words representation," *PLoS One*, vol. 15, no. 5, p. e0232525, May 2020, doi: 10.1371/journal.pone.0232525.
[14] U. Maqsood, S. Ur Rehman, T. Ali, K. Mahmood, T. Alsaedi, and M. Kundi, "An Intelligent Framework Based on Deep Learning for SMS and e-mail Spam Detection," *Applied Computational Intelligence and Soft Computing*, vol. 2023, pp. 1–16, Sep. 2023, doi: 10.1155/2023/6648970.
[15] M. S. Al Ghofany, R. Dwiyansaputra, F. Bimantoro, and Khairunnas, "Indonesian SMS Spam Detection Using TF-RF Feature Weighting Method and Support Vector Machine Classifier," in *Proceedings of the First Mandalika International Multi-Conference on Science and Engineering 2022, MIMSE 2022 (Informatics and Computer Science)*, Dordrecht: Atlantis Press International BV, 2022, pp. 117–129. doi: 10.2991/978-94-6463-084-8_12.
[16] M. Mujahid *et al.*, "Data oversampling and imbalanced datasets: an investigation of performance for machine learning and feature engineering," *J. Big Data*, vol. 11, no. 1, p. 87, Jun. 2024, doi: 10.1186/s40537-024-00943-4.

[17] F. M. Rizky, J. Jondri, and K. M. Lhaksmana, "Twitter Sentiment Analysis of Kanjuruhan Disaster using Word2Vec and Support Vector Machine," *Building of Informatics, Technology and Science (BITS)*, vol. 5, no. 1, Jun. 2023, doi: 10.47065/bits.v5i1.3612.

[18] Y. Wang, "An XGBoost-Based Cyber Threat Detection Framework for Enhancing Security in University E-Government Systems," *SECURITY AND PRIVACY*, vol. 8, no. 5, Sep. 2025, doi: 10.1002/spy2.70089.

[19] P. Joseph and S. Y. Yerima, "A comparative study of word embedding techniques for SMS spam detection," in *2022 14th International Conference on Computational Intelligence and Communication Networks (CICN)*, IEEE, Dec. 2022, pp. 149–155. doi: 10.1109/CICN56167.2022.10008245.

[20] G. Waja, G. Patil, C. Mehta, and S. Patil, "How AI Can be Used for Governance of Messaging Services: A Study on Spam Classification Leveraging Multi-Channel Convolutional Neural Network," *International Journal of Information Management Data Insights*, vol. 3, no. 1, p. 100147, Apr. 2023, doi: 10.1016/j.jjimei.2022.100147.

[21] O. Abayomi-Alli, S. Misra, and A. Abayomi-Alli, "A deep learning method for automatic <scp>SMS</scp> spam classification: Performance of learning algorithms on indigenous dataset," *Concurr. Comput.*, vol. 34, no. 17, Aug. 2022, doi: 10.1002/cpe.6989.

[22] U. Srinivasarao and A. Sharaff, "Machine intelligence based hybrid classifier for spam detection and sentiment analysis of SMS messages," *Multimed. Tools Appl.*, vol. 82, no. 20, pp. 31069–31099, Aug. 2023, doi: 10.1007/s11042-023-14641-5.

[23] M. Sharabov, G. Tsochev, V. Gancheva, and A. Tasheva, "Filtering and Detection of Real-Time Spam Mail Based on a Bayesian Approach in University Networks," *Electronics (Basel).*, vol. 13, no. 2, p. 374, Jan. 2024, doi: 10.3390/electronics13020374.

.